



Panduan Developer

AWS Serverless Application Model



AWS Serverless Application Model: Panduan Developer

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang dapat menyebabkan kebingungan antara para pelanggan, atau dengan cara apa pun yang menghina atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan properti dari masing-masing pemilik, yang mungkin berafiliasi, terkait dengan, atau disponsori oleh Amazon, atau tidak.

Table of Contents

Apa itu AWS SAM?	1
Konsep Basic	2
Apa spesifikasi AWS SAM templatnya?	2
Apa AWS SAM templatnya?	3
Apa itu AWS SAMCLI?	10
Inisialisasi proyek baru	10
Membangun aplikasi Anda untuk penerapan	11
Lakukan debugging dan pengujian lokal	12
Men-deploy aplikasi Anda	13
Konfigurasi pipeline penyebaran CI/CD	14
Pantau dan pecahkan masalah aplikasi Anda di cloud	15
Sinkronkan perubahan lokal ke cloud saat Anda mengembangkan	16
Uji sumber daya yang didukung di cloud	17
Pelajari selengkapnya	17
Langkah selanjutnya	18
Konsep tanpa server	18
Konsep tanpa server	18
Langkah selanjutnya	19
Memulai	20
Prasyarat	20
Langkah 1: Mendaftar untuk AWS akun	20
Langkah 2: Buat akun pengguna IAM	21
Langkah 3: Buat ID kunci akses dan kunci akses rahasia	22
Langkah 4: Instal AWS CLI	24
Langkah 5: Gunakan AWS CLI untuk mengkonfigurasi AWS kredensial	25
Langkah selanjutnya	25
Instalasi AWS SAMCLI	26
Instalasi AWS SAMCLI	26
Memecahkan masalah	37
Langkah selanjutnya	37
Tutorial: Aplikasi Hello World	37
Prasyarat	39
Langkah 1: Inisialisasi contoh aplikasi Hello World	39
Langkah 2: Bangun aplikasi Anda	42

Langkah 3: Menyebarkan aplikasi Anda ke AWS Cloud	44
Langkah 4: Jalankan aplikasi Anda	49
Langkah 5: Berinteraksi dengan fungsi Anda di AWS Cloud	50
Langkah 6: Ubah dan sinkronkan aplikasi Anda ke AWS Cloud	51
Langkah 7: (Opsional) Uji aplikasi Anda secara lokal	55
Langkah 8: Hapus aplikasi Anda dari AWS Cloud	57
Memecahkan masalah	57
Pelajari selengkapnya	58
Menggunakan AWS SAM	59
Menggunakan AWS SAMCLI	59
Bagaimana AWS SAMCLI perintah didokumentasikan	60
Mengkonfigurasi AWS SAMCLI	61
sam build	66
sam deploy	80
sam init	101
sam local	109
sam remote invoke	130
sam remote test-event	157
sam sync	163
AWS SAMspesifikasi	177
Anatomi templat	177
YAML	178
Bagian templat	179
Langkah selanjutnya	181
Global	181
Sumber daya dan referensi properti	186
AWS::Serverless::Api	187
AWS::Serverless::Application	240
AWS::Serverless::Connector	245
AWS::Serverless::Function	259
AWS::Serverless::GraphQLApi	388
AWS::Serverless::HttpApi	424
AWS::Serverless::LayerVersion	455
AWS::Serverless::SimpleTable	461
AWS::Serverless::StateMachine	465
Atribut sumber daya	506

Pengecualian	507
Fungsi intrinsik	508
Sumber daya yang dibuat	509
Mereferensikan sumber daya AWS CloudFormation yang dibuat	509
Skenario sumber daya AWS CloudFormation yang dibuat	510
AWS::Serverless::Api	513
AWS::Serverless::Application	515
AWS::Serverless::Connector	515
AWS::Serverless::Function	516
AWS::Serverless::GraphQLApi	521
AWS::Serverless::HttpApi	523
AWS::Serverless::LayerVersion	525
AWS::Serverless::SimpleTable	525
AWS::Serverless::StateMachine	525
Ekstensi API Gateway	527
Mengotorisasi	529
Mengelola izin	529
AWS SAMkonektor	530
Templat kebijakan AWS SAM	530
AWS CloudFormationmekanisme	530
Praktik terbaik	530
AWS SAMkonektor	531
AWS SAMtemplat kebijakan	540
AWS CloudFormation	606
Menjadwalkan acara dengan EventBridge Scheduler	612
Apa itu EventBridge Scheduler?	612
EventBridge Dukungan penjadwal	612
Membuat acara Scheduler	612
Contoh-contoh	613
Pelajari selengkapnya	614
Memvalidasi file templat AWS SAM	615
Cara menggunakan lapisan	615
Termasuk lapisan dalam aplikasi Anda	615
Bagaimana lapisan di-cache secara lokal	616
Menggunakan aplikasi nest	618
Mendefinisikan aplikasi nest dari AWS Serverless Application Repository	619

Mendefinisikan aplikasi nest dari sistem file lokal	620
Men-deploy aplikasi nest	621
Pengendalian akses ke API	621
Memilih mekanisme untuk mengendalikan akses	624
Menyesuaikan respons kesalahan	624
Contoh-contoh	624
Contoh otorisasi Lambda	625
Contoh izin IAM	628
Contoh kolam pengguna Amazon Cognito	629
Contoh kunci API	630
Contoh kebijakan sumber daya	631
Contoh otorisasi OAuth 2.0/JWT	632
Contoh respons yang disesuaikan	633
Mengorkestrasi aplikasi	634
Contoh	635
Informasi lain	636
Penandatanganan kode	636
Contoh	637
Menyediakan profil penandatanganan dengan <code>sam deploy --guided</code>	639
Membangun	640
Aplikasi Pembangunan	640
Membangun arsip file .zip	641
Membangun citra kontainer	642
File variabel lingkungan kontainer	643
Mempercepat waktu pembuatan dengan membangun proyek Anda di folder sumber	644
Contoh-contoh	644
Membangun fungsi di luar AWS SAM	646
Membangun fungsi Node.js Lambda dengan esbuild	648
Membangun fungsi.NET 7 Native AOT	651
Membangun fungsi Lambda Karat dengan Cargo Lambda	654
Membangun lapisan	666
Contoh-contoh	644
Membangun waktu aktif kustom	669
Contoh-contoh	670
Pengujian dan Debugging	672
Menjalankan fungsi secara lokal	672

File variabel lingkungan	673
Lapisan	675
Pelajari selengkapnya	675
Menjalankan API Gateway lokal	675
File variabel lingkungan	676
Lapisan	678
Mengintegrasikan dengan uji otomatis	678
Membuat muatan peristiwa sampel	680
Langkah-langkah debugging fungsi Lambda secara lokal	680
Menggunakan AWS Kit alat	681
Menjalankan AWS SAM secara lokal dalam mode debug	683
Melewati argumen debug waktu aktif tambahan	683
Validasi dengan cfn-lint	684
Contoh-contoh	685
Pelajari selengkapnya	685
Men-deploy	686
Men-deploy menggunakan sistem CI/CD	686
Men-deploy menggunakan AWS SAM CLI	687
Memecahkan masalah penerapan menggunakan AWS SAMCLI	687
AWS SAMCLLError: “Kendala Keamanan Tidak Puas”	687
Deployment Gradual	688
Pelajari selengkapnya	675
Ubah alur yang sudah ada	688
AWS CodePipeline	689
Pipa Bitbucket	690
Jenkins	690
GitLab CI/CD	691
GitHub Tindakan	692
Menghasilkan alur awal	693
AWS CodePipeline	694
Jenkins, GitLab CI/CD, GitHub Tindakan, Pipa Bitbucket	696
Menyesuaikan saluran pipa starter	699
Contoh proyek	699
Contoh file	700
Menggunakan OIDC dengan pipa AWS SAM	701
Siapkan OIDC dengan pipa AWS SAM	701

Contoh	702
Pelajari selengkapnya	703
Unggah file lokal saat penerapan	703
Demo	704
Kasus penggunaan yang didukung	709
Pelajari selengkapnya	711
Pemantauan	713
Wawasan Aplikasi	713
Mengkonfigurasi Wawasan CloudWatch Aplikasi dengan AWS SAM	713
Langkah selanjutnya	717
Cara menggunakan log	717
Mengambil log dari tumpukan AWS CloudFormation	717
Mengambil log dari nama fungsi Lambda	717
Menunggu log	718
Melihat log pada rentang waktu tertentu	718
Memfilter log	718
Menyoroti error	718
Pencetakan indah JSON	718
Penerbitan	719
Prasyarat	719
Memublikasikan aplikasi baru	721
Langkah 1: Tambahkan bagian Metadata ke templat AWS SAM	721
Langkah 2: Kemas aplikasi	721
Langkah 3: Publikasikan aplikasi	722
Langkah 4: Bagikan aplikasi (opsional)	722
Memublikasikan versi baru aplikasi yang sudah ada	723
Topik Tambahan	723
Properti bagian Metadata	723
Properti	723
Kasus penggunaan	726
Contoh	727
Layanan pihak ketiga	728
Dukungan Terraform	728
AWS SAMCLITerraformdukungan	728
Memulai	730
Menggunakan AWS SAM CLI dengan Terraform	738

Menggunakan AWS SAM CLI dengan ServerLess.tf	742
Referensi Terraform	742
Contoh aplikasi	747
Proses peristiwa DynamoDB	747
Sebelum Anda memulai	747
Langkah 1: Inisialisasi aplikasi	747
Langkah 2: Uji aplikasi secara lokal	748
Langkah 3: Buat paket aplikasi	748
Langkah 4: Deploy aplikasi	749
Langkah selanjutnya	750
Proses peristiwa Amazon S3	750
Sebelum Anda memulai	750
Langkah 1: Inisialisasikan aplikasi	750
Langkah 2: Paketkan aplikasi	751
Langkah 3: Deploy aplikasi	752
Langkah 4: Uji aplikasi secara lokal	753
Langkah selanjutnya	753
AWS CDK	754
Memulai	754
Prasyarat	754
Membuat dan menguji aplikasi secara lokal AWS CDK	755
Pengujian lokal	757
Contoh	758
Membangun	759
Contoh	759
Men-deploy	760
Referensi AWS SAM	761
Spesifikasi AWS SAM	761
AWS SAMCLI referensi perintah	761
Templat kebijakan AWS SAM	762
Topik	762
AWS SAMCLI referensi	762
AWS SAMCLI referensi perintah	763
AWS SAMCLiberkas konfigurasi	838
Mengelola AWS SAMCLI versi	844
Verifikasi AWS SAM CLI penginstal	854

Menyiapkan kredensial AWS	867
Memecahkan Masalah	868
Referensi konektor	876
Jenis sumber daya konektor yang didukung	876
Kebijakan IAM yang dibuat oleh konektor	886
Menginstal Docker	909
Menginstal Docker	910
Langkah selanjutnya	913
Repositoryi citra	913
URI repositoryi citra	914
Contoh-contoh	915
Men-deploy secara bertahap	915
Secara bertahap menerapkan fungsi Lambda untuk pertama kalinya	919
Pelajari selengkapnya	920
AWS SAMCLImetri	920
Matikan telemetri untuk sesi	920
Menonaktifkan telemetri untuk profil Anda di semua sesi	920
Jenis informasi yang dikumpulkan	921
Pelajari selengkapnya	922
Catatan penting	922
2023	922
2020	923
Riwayat dokumen	924
.....	cmxlviii

Apa itu AWS Serverless Application Model (AWS SAM)?

The AWS Serverless Application Model (AWS SAM) adalah toolkit yang meningkatkan pengalaman pengembang dalam membangun dan menjalankan aplikasi tanpa server. AWS SAM menawarkan berbagai manfaat:

Tentukan kode infrastruktur aplikasi Anda dengan cepat, menggunakan lebih sedikit kode

AWS SAM Template penulis untuk menentukan kode infrastruktur aplikasi tanpa server Anda. Terapkan templat Anda secara langsung AWS CloudFormation untuk menyediakan sumber daya Anda.

Kelola aplikasi tanpa server Anda melalui seluruh siklus hidup pengembangannya

Gunakan AWS SAMCLI untuk mengelola aplikasi tanpa server Anda melalui fase penulisan, pembuatan, penerapan, pengujian, dan pemantauan siklus hidup pengembangan Anda. Untuk informasi selengkapnya, lihat [Menggunakan AWS SAMCLI](#).

Menyediakan izin dengan cepat antara sumber daya dengan konektor AWS SAM

Gunakan AWS SAM konektor di AWS SAM template Anda untuk menentukan izin antara AWS sumber daya Anda. AWS SAM mengubah kode Anda menjadi izin IAM yang diperlukan untuk memfasilitasi maksud Anda. Untuk informasi selengkapnya, lihat [Mengelola izin sumber daya dengan konektor AWS SAM](#).

Terus sinkronkan perubahan lokal ke cloud saat Anda mengembangkan

Gunakan AWS SAMCLI `aws sam sync` perintah untuk secara otomatis menyinkronkan perubahan lokal ke cloud, mempercepat alur kerja pengembangan dan pengujian cloud Anda. Untuk informasi selengkapnya, lihat [Menggunakan sam sync](#).

Kelola aplikasi Terraform tanpa server Anda

Gunakan AWS SAMCLI untuk melakukan debugging lokal dan pengujian fungsi dan lapisan Lambda Anda. Untuk informasi selengkapnya, lihat [AWS SAMCLITerraformdukungan](#).

Topik


- [Konsep Basic](#)
- [Apa spesifikasi AWS SAM templatnya?](#)
- [Apa AWS SAM templatnya?](#)

- [Apa itu AWS SAMCLI?](#)
- [Pelajari selengkapnya](#)
- [Langkah selanjutnya](#)
- [Konsep tanpa server](#)

Konsep Basic

AWS SAM terdiri dari tiga bagian utama:

1. AWS SAM spesifikasi template — Kerangka kerja sumber terbuka yang dapat Anda gunakan untuk menentukan infrastruktur aplikasi tanpa server Anda. AWS Kerangka kerja ini diakses melalui AWS SAM template.
2. AWS SAM template — apa yang Anda gunakan untuk mengakses spesifikasi AWS SAM template, kerangka kerja sumber terbuka yang dapat Anda gunakan untuk menentukan infrastruktur aplikasi tanpa server Anda. AWS SAM Template adalah ekstensi AWS CloudFormation template dengan beberapa komponen tambahan yang membuatnya lebih mudah untuk dikerjakan.
3. AWS SAM antarmuka baris perintah (AWS SAMCLI) - Alat baris perintah yang dapat Anda gunakan dengan AWS SAM template dan integrasi pihak ketiga yang didukung untuk membangun dan menjalankan aplikasi tanpa server Anda.

 Baru mengenal tanpa server?

Kami sarankan Anda meninjau [Konsep tanpa server](#).

Apa spesifikasi AWS SAM templatnya?

Spesifikasi AWS SAM template adalah kerangka kerja sumber terbuka yang dapat Anda gunakan untuk menentukan dan mengelola kode infrastruktur aplikasi tanpa server Anda. Spesifikasi AWS SAM template adalah:

- Dibangun di atas AWS CloudFormation - Gunakan AWS CloudFormation sintaks langsung dalam AWS SAM template Anda, mengambil keuntungan dari dukungan ekstensif sumber daya dan konfigurasi properti. Jika Anda sudah terbiasa AWS CloudFormation, Anda tidak perlu mempelajari layanan baru untuk mengelola kode infrastruktur aplikasi Anda.

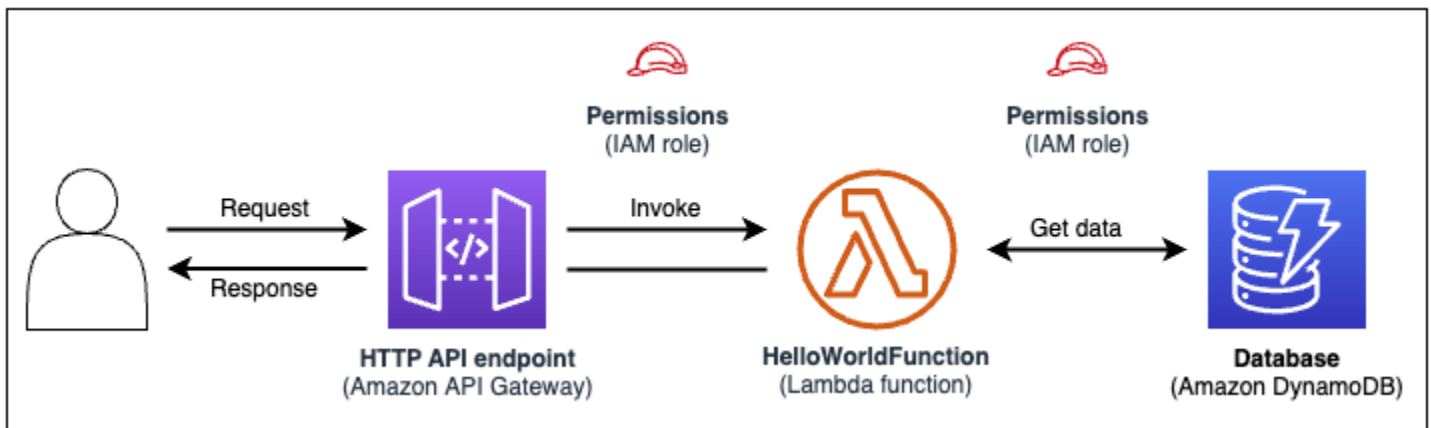
- Perpanjangan AWS CloudFormation - AWS SAM menawarkan sintaks uniknya sendiri yang berfokus secara khusus pada mempercepat pengembangan tanpa server. Anda dapat menggunakan kedua AWS CloudFormation dan AWS SAM sintaks dalam template yang sama.
- Sintaks abstrak dan pendek — Menggunakan AWS SAM sintaks, Anda dapat menentukan infrastruktur Anda dengan cepat, dalam baris kode yang lebih sedikit, dan dengan kemungkinan kesalahan yang lebih rendah. Sintaksnya secara khusus dikuratori untuk mengabstraksikan kompleksitas dalam mendefinisikan infrastruktur aplikasi tanpa server Anda.
- Transformasional — AWS SAM melakukan pekerjaan kompleks mengubah template Anda menjadi kode yang diperlukan untuk menyediakan infrastruktur Anda. AWS CloudFormation

Apa AWS SAM templatnya?

AWS SAM Template adalah apa yang Anda gunakan untuk mengakses spesifikasi AWS SAM template, kerangka kerja sumber terbuka yang dapat Anda gunakan untuk menentukan infrastruktur aplikasi tanpa server Anda AWS, dengan beberapa komponen tambahan yang membuatnya lebih mudah untuk dikerjakan. Dalam hal ini, AWS SAM template adalah perpanjangan dari AWS CloudFormation template.

Berikut adalah contoh aplikasi tanpa server dasar. Aplikasi ini memproses permintaan untuk mendapatkan semua item dari database melalui permintaan HTTP. Ini terdiri dari bagian-bagian berikut:

1. Fungsi yang berisi logika untuk memproses permintaan.
2. HTTP API untuk berfungsi sebagai komunikasi antara klien (requestor) dan aplikasi.
3. Database untuk menyimpan item.
4. Izin agar aplikasi berjalan dengan aman.



Kode infrastruktur aplikasi ini dapat didefinisikan dalam AWS SAM template berikut:

```

AWSTemplateFormatVersion: 2010-09-09
Transform: AWS::Serverless-2016-10-31
Resources:
  getAllItemsFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: src/get-all-items.getAllItemsHandler
      Runtime: nodejs12.x
      Events:
        Api:
          Type: HttpApi
          Properties:
            Path: /
            Method: GET
    Connectors:
      MyConn:
        Properties:
          Destination:
            Id: SampleTable
          Permissions:
            - Read
  SampleTable:
    Type: AWS::Serverless::SimpleTable
  
```

Dalam 23 baris kode, infrastruktur berikut didefinisikan:

- Fungsi yang menggunakan AWS Lambda layanan.
- API HTTP menggunakan layanan Amazon API Gateway.

- Database yang menggunakan layanan Amazon DynamoDB.
- Izin AWS Identity and Access Management (IAM) yang diperlukan untuk layanan ini untuk berinteraksi satu sama lain.

Untuk menyediakan infrastruktur ini, template dikerahkan ke AWS CloudFormation. Selama penyebaran, AWS SAM mengubah 23 baris kode menjadi AWS CloudFormation sintaks yang diperlukan untuk menghasilkan sumber daya ini di. AWS AWS CloudFormation Template yang diubah berisi lebih dari 200 baris kode!

AWS CloudFormation Template yang diubah

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
    "getAllItemsFunction": {
      "Type": "AWS::Lambda::Function",
      "Metadata": {
        "SamResourceId": "getAllItemsFunction"
      },
      "Properties": {
        "Code": {
          "S3Bucket": "aws-sam-cli-managed-default-samclisourcebucket-1a4x26zbcdkqr",
          "S3Key": "what-is-app/a6f856abf1b2c4f7488c09b367540b5b"
        },
        "Handler": "src/get-all-items.getAllItemsHandler",
        "Role": {
          "Fn::GetAtt": [
            "getAllItemsFunctionRole",
            "Arn"
          ]
        },
        "Runtime": "nodejs12.x",
        "Tags": [
          {
            "Key": "lambda:createdBy",
            "Value": "SAM"
          }
        ]
      }
    },
    "getAllItemsFunctionRole": {
```

```

    "Type": "AWS::IAM::Role",
    "Properties": {
      "AssumeRolePolicyDocument": {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Action": [
              "sts:AssumeRole"
            ],
            "Effect": "Allow",
            "Principal": {
              "Service": [
                "lambda.amazonaws.com"
              ]
            }
          }
        ]
      },
      "ManagedPolicyArns": [
        "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
      ],
      "Tags": [
        {
          "Key": "lambda:createdBy",
          "Value": "SAM"
        }
      ]
    }
  },
  "getAllItemsFunctionApiPermission": {
    "Type": "AWS::Lambda::Permission",
    "Properties": {
      "Action": "lambda:InvokeFunction",
      "FunctionName": {
        "Ref": "getAllItemsFunction"
      }
    },
    "Principal": "apigateway.amazonaws.com",
    "SourceArn": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:execute-api:${AWS::Region}:${AWS::AccountId}:
        ${__ApiId__}/${__Stage__}/GET/",
        {
          "__ApiId__": {
            "Ref": "ServerlessHttpApi"
          }
        }
      ]
    }
  }
}

```



```

        },
        "__Stage__": "*"
    }
]
}
},
"ServerlessHttpApi": {
  "Type": "AWS::ApiGatewayV2::Api",
  "Properties": {
    "Body": {
      "info": {
        "version": "1.0",
        "title": {
          "Ref": "AWS::StackName"
        }
      },
    },
    "paths": {
      "/": {
        "get": {
          "x-amazon-apigateway-integration": {
            "httpMethod": "POST",
            "type": "aws_proxy",
            "uri": {
              "Fn::Sub": "arn:${AWS::Partition}:apigateway:
${AWS::Region}:lambda:path/2015-03-31/functions/${getAllItemsFunction.Arn}/invocations"
            },
            "payloadFormatVersion": "2.0"
          },
          "responses": {}
        }
      }
    },
    "openapi": "3.0.1",
    "tags": [
      {
        "name": "httpapi:createdBy",
        "x-amazon-apigateway-tag-value": "SAM"
      }
    ]
  }
},
"ServerlessHttpApiApiGatewayDefaultStage": {

```

```
"Type": "AWS::ApiGatewayV2::Stage",
"Properties": {
  "ApiId": {
    "Ref": "ServerlessHttpApi"
  },
  "StageName": "$default",
  "Tags": {
    "httpapi:createdBy": "SAM"
  },
  "AutoDeploy": true
}
},
"SampleTable": {
  "Type": "AWS::DynamoDB::Table",
  "Metadata": {
    "SamResourceId": "SampleTable"
  },
  "Properties": {
    "AttributeDefinitions": [
      {
        "AttributeName": "id",
        "AttributeType": "S"
      }
    ],
    "KeySchema": [
      {
        "AttributeName": "id",
        "KeyType": "HASH"
      }
    ],
    "BillingMode": "PAY_PER_REQUEST"
  }
},
"getAllItemsFunctionMyConnPolicy": {
  "Type": "AWS::IAM::ManagedPolicy",
  "Metadata": {
    "aws:sam:connectors": {
      "getAllItemsFunctionMyConn": {
        "Source": {
          "Type": "AWS::Serverless::Function"
        },
        "Destination": {
          "Type": "AWS::Serverless::SimpleTable"
        }
      }
    }
  }
}
```

```
    }
  }
},
"Properties": {
  "PolicyDocument": {
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Action": [
          "dynamodb:GetItem",
          "dynamodb:Query",
          "dynamodb:Scan",
          "dynamodb:BatchGetItem",
          "dynamodb:ConditionCheckItem",
          "dynamodb: PartiQLSelect"
        ],
        "Resource": [
          {
            "Fn::GetAtt": [
              "SampleTable",
              "Arn"
            ]
          },
          {
            "Fn::Sub": [
              "${DestinationArn}/index/*",
              {
                "DestinationArn": {
                  "Fn::GetAtt": [
                    "SampleTable",
                    "Arn"
                  ]
                }
              ]
            ]
          }
        ]
      }
    ]
  },
  "Roles": [
    {
      "Ref": "getAllItemsFunctionRole"
    }
  ]
}
```

```
}  
  ]  
    }  
      }  
        }  
          }  
            }
```

Dengan menggunakan AWS SAM, Anda menentukan 23 baris kode infrastruktur. AWS SAM mengubah kode Anda menjadi 200+ baris AWS CloudFormation kode yang diperlukan untuk menyediakan aplikasi Anda.

Apa itu AWS SAMCLI?

AWS SAMCLI ini adalah alat baris perintah yang dapat Anda gunakan dengan AWS SAM templat dan integrasi pihak ketiga yang didukung untuk membangun dan menjalankan aplikasi tanpa server Anda. Gunakan AWS SAMCLI untuk:

- Menginisialisasi proyek aplikasi baru dengan cepat.
- Bangun aplikasi Anda untuk penerapan.
- Lakukan debugging dan pengujian lokal.
- Men-deploy aplikasi Anda.
- Konfigurasi pipeline penyebaran CI/CD.
- Pantau dan pecahkan masalah aplikasi Anda di cloud.
- Sinkronkan perubahan lokal ke cloud saat Anda mengembangkan.
- Dan banyak lagi!

Yang paling AWS SAMCLI baik digunakan saat digunakan dengan AWS SAM dan AWS CloudFormation templat. Ini juga bekerja dengan produk pihak ketiga seperti Terraform.

Inisialisasi proyek baru

Pilih dari template pemula atau pilih lokasi template kustom untuk memulai proyek baru.

Di sini, kita menggunakan `sam init` perintah untuk menginisialisasi proyek aplikasi baru. Kami memilih proyek Hello World Example untuk memulai. AWS SAMCLI Download template starter dan membuat struktur direktori folder proyek kami.

```
→ 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]: █
```

Membangun aplikasi Anda untuk penerapan

Package dependensi fungsi Anda dan atur kode proyek dan struktur folder Anda untuk mempersiapkan penerapan.

Di sini, kami menggunakan `sam build` perintah untuk mempersiapkan aplikasi kami untuk penyebaran. AWS SAM CLI membuat `.aws-sam` direktori dan mengatur dependensi aplikasi dan file kami di sana untuk penyebaran.

```
→ 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 █
```

Lakukan debugging dan pengujian lokal

Di mesin lokal Anda, simulasikan peristiwa, uji API, panggil fungsi, dan lainnya untuk men-debug dan menguji aplikasi Anda.

Di sini, kita menggunakan `sam local invoke` perintah untuk memanggil kita secara `HelloWorldFunction` lokal. Untuk mencapai hal ini, AWS SAMCLI membuat wadah lokal, membangun fungsi kita, memanggilnya, dan mengeluarkan hasilnya.

```
→ 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\"}"}█
```

Men-deploy aplikasi Anda

Konfigurasi setelah penerapan aplikasi Anda dan terapkan ke AWS Cloud untuk menyediakan sumber daya Anda.

Di sini, kami menggunakan `sam deploy --guided` perintah untuk menyebarkan aplikasi kami melalui aliran interaktif. AWS SAMCLIPanduan kami melalui konfigurasi pengaturan penerapan aplikasi kami, mengubah template kami menjadi AWS CloudFormation, dan menyebarkan AWS CloudFormation untuk membuat sumber daya kami.

```
→ 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
```

Konfigurasi pipeline penyebaran CI/CD

Buat pipeline integrasi dan pengiriman berkelanjutan (CI/CD) yang aman, menggunakan sistem CI/CD yang didukung.

Di sini, kita menggunakan `sam pipeline init --bootstrap` perintah untuk mengkonfigurasi pipeline penyebaran CI/CD untuk aplikasi kita. Ini AWS SAMCLI memandu kami melalui opsi kami dan menghasilkan AWS sumber daya dan file konfigurasi untuk digunakan dengan sistem CI/CD kami.

[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]:

Pantau dan pecahkan masalah aplikasi Anda di cloud

Lihat informasi penting tentang sumber daya yang Anda gunakan, kumpulkan log, dan gunakan alat pemantauan bawaan seperti. AWS X-Ray

Di sini, kami menggunakan sam list perintah untuk melihat sumber daya yang kami gunakan. Kami mendapatkan endpoint API kami dan memanggilnya, yang memicu fungsi kami. Kemudian, kita gunakan sam logs untuk melihat log fungsi kita.

```
→ sam-app sam logs --stack-name sam-app
2023/03/13/[$LATEST]0a433e844dd445bd82d0d78cd55e0cdc 2023-03-13T21:06:42.075000 INIT_START Runtime Version: python:3.9.v16 Runtime Version ARN: arn:aws:lambda:us-west-2::runtime:07a48df201798d627f2b950f03bb227aab4a655a1d019c3296406f95937e2525
2023/03/13/[$LATEST]0a433e844dd445bd82d0d78cd55e0cdc 2023-03-13T21:06:42.180000 START RequestId: 778e4226-0a80-435f-929b-5b19292ed9a7 Version: $LATEST
2023/03/13/[$LATEST]0a433e844dd445bd82d0d78cd55e0cdc 2023-03-13T21:06:42.181000 END RequestId: 778e4226-0a80-435f-929b-5b19292ed9a7
2023/03/13/[$LATEST]0a433e844dd445bd82d0d78cd55e0cdc 2023-03-13T21:06:42.182000 REPORT RequestId: 778e4226-0a80-435f-929b-5b19292ed9a7 Duration: 1.69 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 36 MB Init Duration: 104.13 ms
```

Sinkronkan perubahan lokal ke cloud saat Anda mengembangkan

Saat Anda mengembangkan di komputer lokal Anda, secara otomatis menyinkronkan perubahan ke cloud. Lihat perubahan Anda dengan cepat dan lakukan pengujian dan validasi di cloud.

Di sini, kami menggunakan `sam sync --watch` perintah untuk AWS SAMCLI mengawasi perubahan lokal. Kami memodifikasi `HelloWorldFunction` kode kami dan AWS SAMCLI secara otomatis mendeteksi perubahan dan menyebarkan pembaruan kami ke cloud.

```

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

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

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

```

```
Stack update succeeded. Sync infra completed.
```

```
Infra sync completed.
```

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

```
Syncing Lambda Function HelloWorldFunction...
```

```
Manifest is not changed for (HelloWorldFunction), running incremental build
```

```
Building codeuri: /Users/evzz/Demo/what-is/sam-app/hello_world runtime: python3.9 metadata: {} architecture: x86_64 functions: HelloWorldFunction
```

```
Running PythonPipBuilder:CopySource
```

```
Finished syncing Lambda Function HelloWorldFunction.
```

```
□
```

Uji sumber daya yang didukung di cloud

Memanggil dan meneruskan acara ke sumber daya yang didukung di cloud.

Di sini, kami menggunakan `sam remote invoke` perintah untuk menguji fungsi Lambda yang diterapkan di cloud. Kami menjalankan fungsi Lambda kami dan menerima log dan tanggapannya. Dengan fungsi Lambda kami yang dikonfigurasi untuk mengalirkan respons, responsnya AWS SAMCLI mengalir kembali secara real time.

Pelajari selengkapnya

Untuk terus mempelajarinya AWS SAM, lihat sumber daya berikut:

- [AWS SAM Lokakarya Lengkap](#) — Lokakarya yang dirancang untuk mengajari Anda banyak fitur utama yang AWS SAM menyediakan.
- [Sesi dengan SAM](#) - Seri video yang dibuat oleh tim Advokat Pengembang AWS Tanpa Server kami tentang penggunaan. AWS SAM

- [Serverless Land](#) — Situs yang menyatukan informasi terbaru, blog, video, kode, dan sumber belajar untuk AWS tanpa server.

Langkah selanjutnya

Jika ini adalah pertama kalinya Anda menggunakan AWS SAM, lihat [Memulai dengan AWS SAM](#).

Konsep tanpa server

Pelajari tentang konsep dasar tanpa server sebelum menggunakan AWS Serverless Application Model (AWS SAM).

Konsep tanpa server

Arsitektur berbasis peristiwa

Aplikasi tanpa server terdiri dari AWS layanan individual, seperti AWS Lambda untuk komputasi dan Amazon DynamoDB untuk manajemen basis data, yang masing-masing melakukan peran khusus. Layanan ini kemudian terintegrasi secara longgar satu sama lain melalui arsitektur berbasis peristiwa. Untuk mempelajari lebih lanjut tentang arsitektur berbasis peristiwa, lihat [Apa itu Arsitektur Berbasis Acara?](#) .

Infrastruktur sebagai Kode (IaC)

Infrastructure as Code (IaC) adalah cara memperlakukan infrastruktur dengan cara yang sama seperti pengembang memperlakukan kode, menerapkan ketelitian pengembangan kode aplikasi yang sama untuk penyediaan infrastruktur. Anda menentukan infrastruktur Anda dalam file template, menyebarkannya ke AWS, dan AWS membuat sumber daya untuk Anda. Dengan IaC, Anda menentukan dalam kode apa yang AWS ingin Anda berikan. Untuk informasi selengkapnya, lihat [Infrastruktur sebagai Kode](#) di Pengantar DevOps pada AWS AWS Whitepaper.

Teknologi tanpa server

Dengan teknologi AWS tanpa server, Anda dapat membangun dan menjalankan aplikasi tanpa harus mengelola server Anda sendiri. Semua manajemen server dilakukan oleh AWS, memberikan banyak manfaat seperti penskalaan otomatis dan ketersediaan tinggi bawaan, memungkinkan Anda membawa ide Anda ke produksi dengan cepat. Menggunakan teknologi tanpa server, Anda dapat fokus pada inti produk Anda tanpa harus khawatir tentang mengelola dan mengoperasikan server. Untuk mempelajari lebih lanjut tentang tanpa server, lihat yang berikut ini:

- [Tanpa server di AWS](#)
- [Panduan Pengembang Tanpa Server](#) - Memberikan gambaran konseptual pengembangan tanpa server di Cloud. AWS

Untuk pengenalan dasar tentang layanan tanpa AWS server inti, lihat Serverless [101: Memahami layanan tanpa server di Serverless Land](#).

Langkah selanjutnya

Untuk pengantar AWS SAM, lihat [Apa itu AWS Serverless Application Model \(AWS SAM\)?](#)

Memulai dengan AWS SAM

Mulailah dengan AWS Serverless Application Model (AWS SAM) dengan menginstal AWS SAM Command Line Interface (AWS SAMCLI).

Topik

- [Prasyarat AWS SAM](#)
- [Instalasi AWS SAMCLI](#)
- [Tutorial: Men-deploy aplikasi Hello World](#)

Prasyarat AWS SAM

Lengkapi prasyarat berikut sebelum menginstal dan menggunakan AWS Serverless Application Model Command Line Interface (). AWS SAM CLI

Untuk menggunakan AWS SAMCLI, Anda memerlukan yang berikut ini:

- AWS Akun, kredensial AWS Identity and Access Management (IAM), dan key pair akses IAM.
- The AWS Command Line Interface (AWS CLI) untuk mengkonfigurasi AWS kredensial.

Topik

- [Langkah 1: Mendaftar untuk AWS akun](#)
- [Langkah 2: Buat akun pengguna IAM](#)
- [Langkah 3: Buat ID kunci akses dan kunci akses rahasia](#)
- [Langkah 4: Instal AWS CLI](#)
- [Langkah 5: Gunakan AWS CLI untuk mengkonfigurasi AWS kredensial](#)
- [Langkah selanjutnya](#)

Langkah 1: Mendaftar untuk AWS akun

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar Akun AWS

1. Buka <https://portal.aws.amazon.com/billing/signup>.

2. Ikuti petunjuk secara online.

Anda akan diminta untuk menerima panggilan telepon dan memasukkan kode verifikasi pada keypad telepon sebagai bagian dari prosedur pendaftaran.

Saat Anda mendaftar Akun AWS, Pengguna root akun AWS akan dibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya dalam akun. Sebagai praktik terbaik keamanan, [tetapkan akses administratif ke pengguna administratif](#), dan hanya gunakan pengguna root untuk melakukan [tugas yang memerlukan akses pengguna root](#).

Langkah 2: Buat akun pengguna IAM

Untuk membuat pengguna administrator, pilih salah satu opsi berikut.

Pilih salah satu cara untuk mengelola administrator Anda	Untuk	Oleh	Anda juga bisa
Di Pusat Identitas IAM (Direkomendasikan)	Gunakan kredensi jangka pendek untuk mengakses. AWS Ini sejalan dengan praktik terbaik keamanan. Untuk informasi tentang praktik terbaik, lihat Praktik terbaik keamanan di IAM di Panduan Pengguna IAM.	Mengikuti petunjuk di Memulai di Panduan AWS IAM Identity Center Pengguna.	Konfigurasi akses terprogram dengan Mengonfigurasi AWS CLI yang akan digunakan AWS IAM Identity Center dalam AWS Command Line InterfacePanduan Pengguna.

Pilih salah satu cara untuk mengelola administrator Anda	Untuk	Oleh	Anda juga bisa
Di IAM (Tidak direkomendasikan)	Gunakan kredensi jangka panjang untuk mengakses. AWS	Mengikuti petunjuk dalam Membuat pengguna admin IAM pertama Anda dan grup pengguna di Panduan Pengguna IAM.	Konfigurasi akses terprogram dengan Mengelola kunci akses untuk pengguna IAM di Panduan Pengguna IAM .

Langkah 3: Buat ID kunci akses dan kunci akses rahasia

Untuk akses CLI, Anda memerlukan ID kunci akses dan kunci akses rahasia. Gunakan kredensi sementara alih-alih kunci akses jangka panjang jika memungkinkan. Kredensi sementara mencakup ID kunci akses, kunci akses rahasia, dan token keamanan yang menunjukkan kapan kredensialnya kedaluwarsa. Untuk informasi selengkapnya, lihat [Menggunakan kredensi sementara dengan AWS sumber daya](#) di Panduan Pengguna IAM.

Pengguna membutuhkan akses terprogram jika ingin berinteraksi dengan AWS di luar AWS Management Console. Cara memberikan akses terprogram bergantung pada jenis pengguna yang mengakses AWS.

Untuk memberi pengguna akses terprogram, pilih salah satu opsi berikut.

Pengguna mana yang membutuhkan akses terprogram?	Untuk	Oleh
Identitas tenaga kerja	Gunakan kredensial sementara untuk menandatangani permintaan terprogram	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan.

Pengguna mana yang membutuhkan akses terprogram?	Untuk	Oleh
(Pengguna yang dikelola di Pusat Identitas IAM)	ke AWS CLI, SDK AWS, atau API AWS.	<ul style="list-style-type: none">• Untuk AWS CLI, lihat Mengonfigurasi AWS CLI untuk menggunakan AWS IAM Identity Center di Panduan Pengguna AWS Command Line Interface.• Untuk SDK AWS, alat, dan API AWS, lihat Autentikasi Pusat Identitas IAM di Panduan Referensi SDK dan Alat AWS.
IAM	Gunakan kredensial sementara untuk menandatangani permintaan terprogram ke AWS CLI, SDK AWS, atau API AWS.	Mengikuti petunjuk dalam Menggunakan kredensial sementara dengan sumber daya AWS di Panduan Pengguna IAM.

Pengguna mana yang membutuhkan akses terprogram?	Untuk	Oleh
IAM	(Tidak direkomendasikan) Gunakan kredensial jangka panjang untuk menandatangani permintaan terprogram ke AWS CLI, SDK AWS, atau API AWS.	Mengikuti petunjuk untuk antarmuka yang ingin Anda gunakan. <ul style="list-style-type: none"> • Untuk AWS CLI, lihat Mengautentikasi menggunakan kredensial pengguna IAM di Panduan Pengguna AWS Command Line Interface. • Untuk SDK dan alat AWS, lihat Mengautentikasi menggunakan kredensial jangka panjang di Panduan Referensi SDK dan Alat AWS. • Untuk API AWS, lihat Mengelola kunci akses untuk pengguna IAM di Panduan Pengguna IAM.

Langkah 4: Instal AWS CLI

AWS CLI ini adalah alat open source yang memungkinkan Anda berinteraksi dengan Layanan AWS menggunakan perintah di shell baris perintah Anda. AWS SAM CLI memerlukan AWS CLI untuk kegiatan seperti mengkonfigurasi kredensial. Untuk mempelajari lebih lanjut tentang AWS CLI, lihat [Apa itu AWS Command Line Interface?](#) dalam AWS Command Line Interface User Guide.

Untuk menginstal AWS CLI, lihat [Menginstal atau memperbarui versi terbaru AWS CLI dari Panduan AWS Command Line Interface Pengguna](#).

Langkah 5: Gunakan AWS CLI untuk mengkonfigurasi AWS kredensial

Untuk mengonfigurasi kredensi dengan AWS CLI

1. Jalankan `aws configure` perintah dari baris perintah.
2. Konfigurasi berikut ini. Pilih setiap tautan untuk mempelajari lebih lanjut:
 - a. [ID kunci akses](#)
 - b. [Kunci akses rahasia](#)
 - c. [Wilayah AWS](#)
 - d. [Format keluaran](#)

Contoh berikut menunjukkan nilai sampel.

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

AWS CLI menyimpan informasi ini dalam profil (kumpulan pengaturan) yang disebutkan default dalam config file `credentials` dan. File-file ini terletak di `.aws` file di direktori home Anda. Secara default, informasi dalam profil ini digunakan saat Anda menjalankan AWS CLI perintah yang tidak secara eksplisit menentukan profil yang akan digunakan. Untuk informasi selengkapnya tentang `credentials` file, lihat [Pengaturan konfigurasi dan file kredensi](#) di Panduan AWS Command Line Interface Pengguna.

Untuk informasi selengkapnya tentang mengonfigurasi kredensial, seperti menggunakan konfigurasi dan file kredensial yang ada, lihat [Penyiapan cepat](#) di Panduan Pengguna. AWS Command Line Interface

Langkah selanjutnya

Anda sekarang siap untuk menginstal AWS SAM CLI dan mulai menggunakan AWS SAM. Untuk menginstal AWS SAM CLI, lihat [Instalasi AWS SAM CLI](#).

Instalasi AWS SAMCLI

Instal rilis terbaru dari AWS Serverless Application Model Command Line Interface (AWS SAMCLI) pada sistem operasi yang didukung.

Untuk informasi tentang mengelola versi yang saat ini diinstal AWS SAMCLI, termasuk cara memutakhirkan, menghapus instalasi, atau mengelola build malam hari, lihat. [Mengelola AWS SAMCLI versi](#)

i Apakah ini pertama kalinya Anda menginstal AWS SAM CLI?

Lengkapi semua [prasyarat](#) di bagian sebelumnya sebelum bergerak maju. Hal ini mencakup:

1. Mendaftar untuk sebuah AWS akun.
2. Membuat pengguna IAM administratif.
3. Membuat ID kunci akses dan kunci akses rahasia.
4. Instalasi AWS CLI.
5. Mengkonfigurasi AWS kredensial.

Topik

- [Instalasi AWS SAMCLI](#)
- [Memecahkan masalah](#)
- [Langkah selanjutnya](#)

Instalasi AWS SAMCLI

i Note

Mulai September 2023, tidak AWS akan lagi mempertahankan Homebrew installer AWS terkelola untuk AWS SAMCLI (`aws/tap/aws-sam-cli`). Jika Anda menggunakan Homebrew untuk menginstal dan mengelola AWS SAMCLI, lihat opsi berikut:

- Untuk terus menggunakan Homebrew, Anda dapat menggunakan penginstal yang dikelola komunitas. Untuk informasi selengkapnya, lihat [Mengelola AWS SAMCLI dengan Homebrew](#).

- Sebaiknya gunakan salah satu metode instalasi pihak pertama yang didokumentasikan di halaman ini. Sebelum menggunakan salah satu metode ini, lihat [Beralih dari Homebrew](#).

Untuk menginstal AWS SAMCLI, ikuti instruksi untuk sistem operasi Anda.

Linux

arm64 - command line installer

1. Unduh [AWS SAMCLIfile.zip](#) ke direktori pilihan Anda.
2. (Opsional) Anda dapat memverifikasi integritas penginstal sebelum instalasi. Untuk petunjuk, lihat [Verifikasi integritas AWS SAMCLI penginstal](#).
3. Buka zip file instalasi ke direktori pilihan Anda. Berikut ini adalah contoh, menggunakan `sam-installation` subdirektori.

Note

Gunakan ekuivalen jika sistem pengoperasian Anda tidak memiliki perintah unzip.

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

4. Instal AWS SAMCLI dengan menjalankan `install` executable. Executable ini terletak di direktori yang digunakan pada langkah sebelumnya. Berikut ini adalah contoh, menggunakan `sam-installation` subdirektori:

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

5. Verifikasi instalasi.

```
$ sam --version
```

Untuk mengonfirmasi instalasi yang berhasil, Anda akan melihat output seperti berikut ini tetapi itu menggantikan teks kurung dengan versi SAM CLI terbaru:

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

x86_64 - command line installer

1. Unduh [AWS SAMCLIfile.zip](#) ke direktori pilihan Anda.
2. (Opsional) Anda dapat memverifikasi integritas penginstal sebelum instalasi. Untuk petunjuk, lihat [Verifikasi integritas AWS SAMCLI penginstal](#).
3. Buka zip file instalasi ke direktori pilihan Anda. Berikut ini adalah contoh, menggunakan `sam-installation` subdirektori.

Note

Gunakan ekuivalen jika sistem pengoperasian Anda tidak memiliki perintah `unzip`.

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

4. Instal AWS SAMCLI dengan menjalankan `install` executable. Executable ini terletak di direktori yang digunakan pada langkah sebelumnya. Berikut ini adalah contoh, menggunakan `sam-installation` subdirektori:

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

5. Verifikasi instalasi.

```
$ sam --version
```

Untuk mengonfirmasi penginstalan yang berhasil, Anda akan melihat output yang menggantikan teks kurung berikut dengan versi terbaru yang tersedia:

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

macOS

Instal AWS SAMCLI menggunakan penginstal paketnya.

Note

Pemasang paket tersedia di `x86_64` dan `arm64` opsi. Pilih paket yang cocok dengan Mac perangkat keras Anda:

- `x86_64`— Gunakan untuk Macs menjalankan Intel prosesor.
- `arm64`- Gunakan untuk Macs menjalankan Apple silikon.

Untuk informasi lebih lanjut, lihat [Mackomputer dengan Apple silikon](#).

Menggunakan penginstal paket

Pemasang paket memiliki dua metode instalasi yang dapat Anda pilih:

1. GUI
2. Baris perintah

Anda dapat menginstal untuk semua pengguna atau hanya pengguna Anda saat ini. Untuk menginstal untuk semua pengguna, otorisasi superuser diperlukan.

Langkah-langkah instalasi

Instal AWS SAMCLI menggunakan salah satu opsi berikut.

GUI - All users

Untuk mengunduh penginstal paket

Note

Jika sebelumnya Anda menginstal AWS SAMCLI through Homebrew ataupun, Anda harus menghapusnya terlebih dahulu. Untuk petunjuk, lihat [Menghapus instalasi AWS SAMCLI](#).


- Untuk memulai instalasi, unduh macOS pkg ke direktori pilihan Anda:
 - [x86_64 - -x86_64.pkg aws-sam-cli-macos](#)
 - [arm64 - -arm64.pkg aws-sam-cli-macos](#)

Untuk memverifikasi penginstal paket (opsional)

- Anda dapat memverifikasi integritas penginstal sebelum instalasi. Untuk petunjuk, lihat [Verifikasi integritas AWS SAMCLI penginstal](#).

Untuk menginstal AWS SAMCLI

1. Jalankan file yang Anda unduh dan ikuti petunjuk di layar untuk melanjutkan melalui langkah-langkah Pendahuluan, Baca Saya, dan Lisensi.
2. Untuk Destination Select, pilih Instal untuk semua pengguna komputer ini.
3. Untuk Jenis Instalasi, pilih di mana AWS SAMCLI akan diinstal dan tekan Instal. Lokasi default yang disarankan adalah `/usr/local/aws-sam-cli`.

 Note

Untuk menjalankan sam perintah AWS SAMCLI with, installer secara otomatis membuat symlink antara `/usr/local/bin/sam` dan salah satu `/usr/local/aws-sam-cli/sam` atau folder instalasi yang Anda pilih.

4. AWS SAMCLIAkan menginstal dan Instalasi berhasil pesan akan ditampilkan. Tekan Tutup.

Untuk memverifikasi instalasi yang berhasil

- Verifikasi bahwa symlink AWS SAMCLI telah diinstal dengan benar dan symlink Anda dikonfigurasi dengan menjalankan:

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


GUI - Current user

Untuk mengunduh penginstal paket

Note

Jika sebelumnya Anda menginstal AWS SAMCLI through Homebrew ataupun, Anda harus menghapusnya terlebih dahulu. Untuk petunjuk, lihat [Menghapus instalasi AWS SAMCLI](#).

- Untuk memulai instalasi, unduh macOS pkg ke direktori pilihan Anda:
 - [x86_64 - -x86_64.pkg aws-sam-cli-macos](#)
 - [arm64 - -arm64.pkg aws-sam-cli-macos](#)

Untuk memverifikasi penginstal paket (opsional)

- Anda dapat memverifikasi integritas penginstal sebelum instalasi. Untuk petunjuk, lihat [Verifikasi integritas AWS SAMCLI penginstal](#).

Untuk menginstal AWS SAMCLI

1. Jalankan file yang Anda unduh dan ikuti petunjuk di layar untuk melanjutkan melalui langkah-langkah Pendahuluan, Baca Saya, dan Lisensi.
2. Untuk Destination Select, pilih Install for me only. Jika Anda tidak melihat opsi ini, lanjutkan ke langkah berikutnya.
3. Untuk Jenis Instalasi, lakukan hal berikut:
 1. Pilih di mana yang AWS SAMCLI akan diinstal. Lokasi default adalah `/usr/local/aws-sam-cli`. Pilih lokasi yang memiliki izin tulis untuk Anda. Untuk mengubah lokasi instalasi, pilih lokal dan pilih lokasi Anda. Tekan Lanjutkan setelah selesai.
 2. Jika Anda tidak mendapatkan opsi untuk memilih Install for me hanya pada langkah sebelumnya, pilih Change Install Location > Install for me only dan tekan Continue.
 3. Tekan Instal.
4. AWS SAMCLI akan menginstal dan Instalasi berhasil pesan akan ditampilkan. Tekan Tutup.

Untuk membuat symlink

- Untuk memanggil sam perintah AWS SAMCLI with the, Anda harus secara manual membuat symlink antara AWS SAMCLI program dan program Anda. \$PATH Buat symlink Anda dengan memodifikasi dan menjalankan perintah berikut:

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

- **sudo** — Jika pengguna Anda memiliki izin menulis \$PATH, tidak sudo diperlukan. Sebaliknya, sudo diperlukan.
- **Path-to** — Jalur ke tempat Anda menginstal program. AWS SAMCLI Misalnya, /Users/myUser/Desktop.
- **path-to-symlink-directory**- Variabel \$PATH lingkungan Anda. Lokasi default adalah /usr/local/bin.

Untuk memverifikasi instalasi yang berhasil

- Verifikasi bahwa symlink AWS SAMCLI telah diinstal dengan benar dan symlink Anda dikonfigurasi dengan menjalankan:

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

Command line - All users

Untuk mengunduh penginstal paket

Note

Jika sebelumnya Anda menginstal AWS SAMCLI through Homebrew ataupun, Anda harus menghapusnya terlebih dahulu. Untuk petunjuk, lihat [Menghapus instalasi AWS SAMCLI](#).

- Untuk memulai instalasi, unduh macOS pkg ke direktori pilihan Anda:

- [x86_64 - -x86_64.pkg aws-sam-cli-macos](#)
- [arm64 - -arm64.pkg aws-sam-cli-macos](#)

Untuk memverifikasi penginstal paket (opsional)

- Anda dapat memverifikasi integritas penginstal sebelum instalasi. Untuk petunjuk, lihat [Verifikasi integritas AWS SAMCLI penginstal](#).

Untuk menginstal AWS SAMCLI

- Ubah dan jalankan skrip instalasi:

```
$ 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

Untuk memanggil sam perintah AWS SAMCLI with, installer secara otomatis membuat symlink antara dan. `/usr/local/bin/sam` `/usr/local/aws-sam-cli/sam`

Untuk memverifikasi instalasi yang berhasil

- Verifikasi bahwa symlink AWS SAMCLI telah diinstal dengan benar dan symlink Anda dikonfigurasi dengan menjalankan:

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

Command line - Current user

Untuk mengunduh penginstal paket

Note

Jika sebelumnya Anda menginstal AWS SAMCLI through Homebrew ataupun, Anda harus menghapusnya terlebih dahulu. Untuk petunjuk, lihat [Menghapus instalasi AWS SAMCLI](#).

- Untuk memulai instalasi, unduh macOS pkg ke direktori pilihan Anda:
 - [x86_64 - -x86_64.pkg aws-sam-cli-macos](#)
 - [arm64 - -arm64.pkg aws-sam-cli-macos](#)

Untuk memverifikasi penginstal paket (opsional)

- Anda dapat memverifikasi integritas penginstal sebelum instalasi. Untuk petunjuk, lihat [Verifikasi integritas AWS SAMCLI penginstal](#).

Untuk menginstal AWS SAMCLI

1. Tentukan direktori instalasi yang Anda miliki izin menulis. Kemudian, buat xml file menggunakan template dan memodifikasinya untuk mencerminkan direktori instalasi Anda. Direktori harus sudah ada.

Misalnya, jika Anda mengganti *path-to-my-directory* dengan `/Users/myUser/Desktop`, folder `aws-sam-cli` program akan diinstal di sana.

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

```

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

```

2. Simpan xml file dan verifikasi bahwa itu valid dengan menjalankan yang berikut:

```

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

```

Output harus menampilkan preferensi yang akan diterapkan pada AWS SAMCLI program.

3. Jalankan yang berikut ini untuk menginstal AWS SAMCLI:

```

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

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

```

Untuk membuat symlink

- Untuk memanggil sam perintah AWS SAMCLI with the, Anda harus secara manual membuat symlink antara AWS SAMCLI program dan program Anda. \$PATH Buat symlink Anda dengan memodifikasi dan menjalankan perintah berikut:

```

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

```

- **sudo** — Jika pengguna Anda memiliki izin menulis \$PATH, tidak sudo diperlukan. Sebaliknya, sudo diperlukan.
- **Path-to** — Jalur ke tempat Anda menginstal program. AWS SAMCLI Misalnya, /Users/myUser/Desktop.
- **path-to-symlink-directory** - Variabel \$PATH lingkungan Anda. Lokasi default adalah /usr/local/bin.

Untuk memverifikasi instalasi yang berhasil

- Verifikasi bahwa symlink AWS SAMCLI telah diinstal dengan benar dan symlink Anda dikonfigurasi dengan menjalankan:

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

Windows

File Penginstal Windows (MSI) merupakan file penginstal paket untuk sistem pengoperasian Windows.

Ikuti langkah-langkah ini untuk menginstal AWS SAMCLI menggunakan file MSI.

1. Unduh AWS SAMCLI [64-bit](#).

Note

Jika Anda menggunakan versi Windows 32-bit, lihat [Menginstal AWS SAMCLI pada 32-bit Windows](#).

2. (Opsional) Anda dapat memverifikasi integritas penginstal sebelum instalasi. Untuk petunjuk, lihat [Verifikasi integritas AWS SAMCLI penginstal](#).
3. Verifikasi instalasi.

Setelah menyelesaikan instalasi, verifikasi dengan membuka command prompt atau PowerShell prompt baru. Anda harus dapat memanggil sam dari baris perintah.

```
sam --version
```

Setelah instalasi berhasil AWS SAMCLI, Anda akan melihat output seperti berikut:

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

4. Aktifkan jalur panjang (hanya Windows 10 dan yang lebih baru).

⚠ Important

AWS SAMCLIMungkin berinteraksi dengan jalur file yang melebihi batasan jalur maks Windows. Ini dapat menyebabkan kesalahan saat berjalan `sam init` karena `MAX_PATH` keterbatasan Windows 10. Untuk mengatasi masalah ini, perilaku jalur panjang yang baru harus dikonfigurasi.

Untuk mengaktifkan jalur panjang, lihat [Mengaktifkan Jalur Panjang di Windows 10, Versi 1607, dan Nanti](#) di Dokumentasi Pengembangan Aplikasi Microsoft Windows.

5. Instal Git.

Anda juga harus menginstal Git untuk mengunduh aplikasi contoh menggunakan perintah `sam init`. Untuk instruksi, lihat [Menginstal Git](#).

Memecahkan masalah

Jika Anda menemukan masalah saat menginstal AWS SAMCLI, lihat [Kesalahan penginstalan](#).

Langkah selanjutnya

Untuk mempelajari lebih lanjut tentang AWS SAMCLI dan untuk mulai membangun aplikasi tanpa server Anda sendiri, lihat berikut ini:

- [Tutorial: Men-deploy aplikasi Hello World](#)— S tep-by-step instruksi untuk mengunduh, membangun, dan menyebarkan aplikasi tanpa server dasar.
- [AWS SAM Lokakarya Lengkap](#) — Lokakarya yang dirancang untuk mengajari Anda banyak fitur utama yang AWS SAM menyediakan.
- [AWS SAM contoh aplikasi dan pola](#) - Contoh aplikasi dan pola dari penulis komunitas yang dapat Anda coba lebih lanjut.

Tutorial: Men-deploy aplikasi Hello World

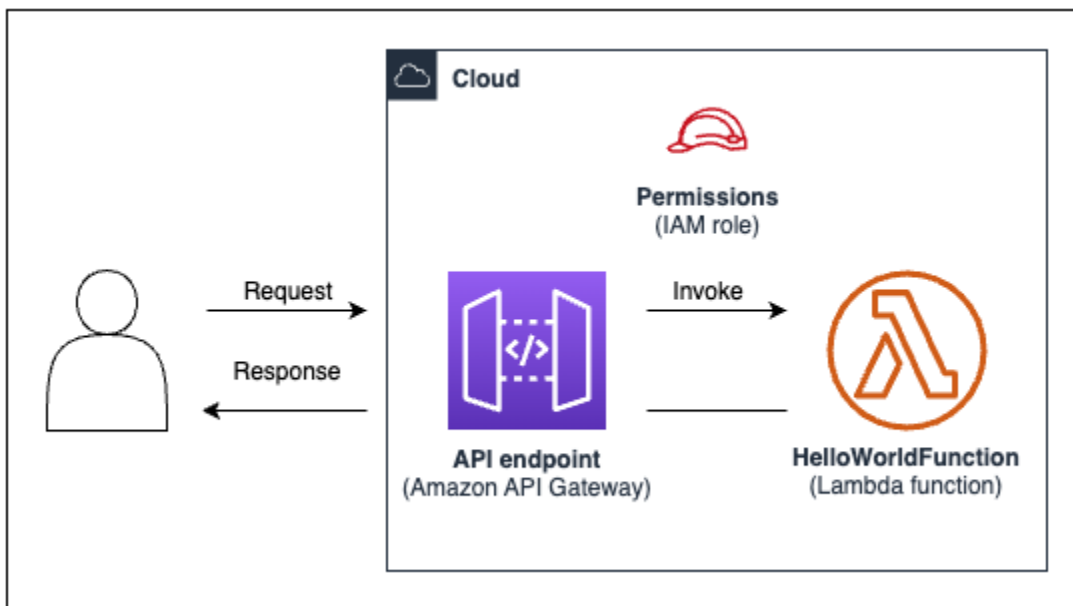
Dalam tutorial ini, Anda menggunakan AWS Serverless Application Model Command Line Interface (AWS SAMCLI) untuk menyelesaikan yang berikut:

- Menginisialisasi, membangun, dan menyebarkan contoh aplikasi Hello World.
- Buat perubahan lokal dan sinkronkan keAWS CloudFormation.
- Uji aplikasi Anda diAWS Cloud.
- Secara opsional, lakukan pengujian lokal pada host pengembangan Anda.
- Hapus aplikasi sampel dari fileAWS Cloud.

Contoh aplikasi Hello World mengimplementasikan backend API dasar. Ini terdiri dari sumber daya berikut:

- Amazon API Gateway — Titik akhir API yang akan Anda gunakan untuk menjalankan fungsi Anda.
- AWS Lambda— Fungsi yang memproses permintaan HTTP API GET dan mengembalikan `hello world` pesan.
- AWS Identity and Access ManagementPeran (IAM) — Memberikan izin bagi layanan untuk berinteraksi dengan aman.

Diagram berikut menunjukkan komponen untuk aplikasi ini:



Topik

- [Prasyarat](#)
- [Langkah 1: Inisialisasi contoh aplikasi Hello World](#)
- [Langkah 2: Bangun aplikasi Anda](#)

- [Langkah 3: Menyebarkan aplikasi Anda ke AWS Cloud](#)
- [Langkah 4: Jalankan aplikasi Anda](#)
- [Langkah 5: Berinteraksi dengan fungsi Anda di AWS Cloud](#)
- [Langkah 6: Ubah dan sinkronkan aplikasi Anda ke AWS Cloud](#)
- [Langkah 7: \(Opsional\) Uji aplikasi Anda secara lokal](#)
- [Langkah 8: Hapus aplikasi Anda dari AWS Cloud](#)
- [Memecahkan masalah](#)
- [Pelajari selengkapnya](#)

Prasyarat

Verifikasi bahwa Anda telah menyelesaikan yang berikut ini:

- [Prasyarat AWS SAM](#)
- [Instalasi AWS SAMCLI](#)

Langkah 1: Inisialisasi contoh aplikasi Hello World

Pada langkah ini, Anda akan menggunakan AWS SAM CLI untuk membuat contoh proyek aplikasi Hello World pada mesin lokal Anda.

Untuk menginisialisasi contoh aplikasi Hello World

1. Di baris perintah Anda, jalankan yang berikut dari direktori awal pilihan Anda:

```
$ sam init
```

2. Ini AWS SAM CLI akan memandu Anda melalui inisialisasi aplikasi baru. Konfigurasi berikut ini:
 1. Pilih AWSQuick Start Templates untuk memilih template awal.
 2. Pilih template Hello World Example dan unduh.
 3. Gunakan Python runtime dan tipe zip paket.
 4. Untuk tutorial ini, pilih keluar dari AWS X-Ray tracing. Untuk mempelajari lebih lanjut, lihat [Apa itu AWS X-Ray?](#) di Panduan AWS X-Ray Pengembang.

5. Untuk tutorial ini, pilih keluar dari pemantauan dengan Amazon CloudWatch Application Insights. Untuk mempelajari selengkapnya, lihat [Wawasan CloudWatch Aplikasi Amazon](#) di Panduan CloudWatch Pengguna Amazon.
6. Beri nama aplikasi Anda sebagai sam-app.

Untuk menggunakan aliran AWS SAM CLI interaktif:

- Kurung ([]) menunjukkan nilai default. Biarkan jawaban Anda kosong untuk memilih nilai default.
- Masukkan **y** untuk ya, dan **n** untuk tidak.

Berikut ini adalah contoh aliran sam `init` interaktif:

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

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

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

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

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

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

3. AWS SAMCLIDownload template awal Anda dan membuat struktur direktori proyek aplikasi pada mesin lokal Anda. Berikut ini adalah contoh dari AWS SAM CLI output:

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

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

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

```
Commands you can use next
```

```
=====
```

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

4. Dari baris perintah Anda, pindah ke sam-app direktori yang baru dibuat. Berikut ini adalah contoh dari apa yang AWS SAM CLI telah dibuat:

```
$ 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
```

Beberapa file penting untuk disorot:

- `hello_world/app.py`— Berisi kode fungsi Lambda Anda.
- `hello_world/requirements.txt`— Berisi Python dependensi apa pun yang dibutuhkan fungsi Lambda Anda.
- `samconfig.toml`— File konfigurasi untuk aplikasi Anda yang menyimpan parameter default yang digunakan oleh file AWS SAM CLI.
- `template.yaml`— AWS SAM Template yang berisi kode infrastruktur aplikasi Anda.

Anda sekarang memiliki aplikasi tanpa server yang sepenuhnya ditulis di mesin lokal Anda!

Langkah 2: Bangun aplikasi Anda

Pada langkah ini, Anda menggunakan AWS SAM CLI untuk membangun aplikasi Anda dan mempersiapkan penerapan. Saat Anda membangun, AWS SAM CLI membuat `.aws-sam` direktori dan mengatur dependensi fungsi, kode proyek, dan file proyek Anda di sana.

Untuk membangun aplikasi Anda

- Di baris perintah Anda, dari direktori `sam-app` proyek, jalankan yang berikut ini:

```
$ sam build
```

Note

Jika Anda tidak memiliki Python di mesin lokal Anda, gunakan `sam build --use-container` perintah sebagai gantinya. AWS SAM CLI akan membuat Docker wadah yang menyertakan runtime dan dependensi fungsi Anda. Perintah ini membutuhkan Docker pada mesin lokal Anda. Untuk menginstal Docker, lihat [Menginstal Docker](#).

Berikut ini adalah contoh dari AWS SAM CLI output:

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

Build Succeeded

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

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

Berikut ini adalah contoh singkat dari `.aws-sam` direktori yang dibuat oleh AWS SAM CLI:

```
.aws-sam
### build
```

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

Beberapa file penting untuk disorot:

- `build/HelloWorldFunction`— Berisi kode fungsi Lambda Anda dan dependensi. AWS SAM CLI membuat direktori untuk setiap fungsi dalam aplikasi Anda.
- `build/template.yaml`— Berisi salinan AWS SAM template Anda yang direferensikan oleh AWS CloudFormation saat penyebaran.
- `build.toml`— File konfigurasi yang menyimpan nilai parameter default yang direferensikan oleh AWS SAM CLI saat membangun dan menerapkan aplikasi Anda.

Anda sekarang siap untuk menyebarkan aplikasi Anda ke AWS Cloud

Langkah 3: Menyebarkan aplikasi Anda ke AWS Cloud

Note

Langkah ini membutuhkan konfigurasi AWS kredensial. Untuk informasi selengkapnya, lihat [Langkah 5: Gunakan AWS CLI untuk mengkonfigurasi AWS kredensial](#) di [Prasyarat AWS SAM](#).

Pada langkah ini, Anda menggunakan AWS SAM CLI untuk menyebarkan aplikasi Anda ke file. AWS Cloud Yang AWS SAM CLI akan melakukan hal berikut:

- Memandu Anda melalui konfigurasi pengaturan aplikasi Anda untuk penyebaran.
- Unggah file aplikasi Anda ke Amazon Simple Storage Service (Amazon S3).
- Ubah AWS SAM template Anda menjadi AWS CloudFormation template. Kemudian unggah template Anda ke AWS CloudFormation layanan untuk menyediakan AWS sumber daya Anda.

Untuk menyebarkan aplikasi Anda

1. Di baris perintah Anda, dari direktori `sam-app` proyek, jalankan yang berikut ini:

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

2. Ikuti alur AWS SAM CLI interaktif untuk mengonfigurasi pengaturan aplikasi Anda. Konfigurasi berikut ini:
 1. Nama AWS CloudFormation tumpukan — Tumpukan adalah kumpulan sumber AWS daya yang dapat Anda kelola sebagai satu unit. Untuk mempelajari lebih lanjut, lihat [Bekerja dengan tumpukan](#) di Panduan AWS CloudFormation Pengguna.
 2. Wilayah AWS Untuk menyebarkan AWS CloudFormation tumpukan Anda ke. Untuk informasi selengkapnya, lihat [AWS CloudFormation titik akhir](#) di Panduan AWS CloudFormation Pengguna.
 3. Untuk tutorial ini, pilih keluar dari konfirmasi perubahan sebelum menerapkan.
 4. Izinkan pembuatan peran IAM — Ini memungkinkan AWS SAM membuat peran IAM yang diperlukan untuk sumber daya API Gateway dan sumber daya fungsi Lambda Anda untuk berinteraksi.
 5. Untuk tutorial ini, pilih keluar dari menonaktifkan rollback.
 6. Izinkan HelloWorldFunction tanpa otorisasi yang ditentukan - Pesan ini ditampilkan karena titik akhir API Gateway Anda dikonfigurasi agar dapat diakses publik, tanpa otorisasi. Karena ini adalah konfigurasi yang dimaksudkan untuk aplikasi Hello World Anda, izinkan AWS SAM CLI untuk melanjutkan. Untuk informasi selengkapnya tentang mengonfigurasi otorisasi, lihat [Pengendalian akses ke API dari API Gateway](#)
 7. Simpan argumen ke file konfigurasi — Ini akan memperbarui `samconfig.toml` file aplikasi Anda dengan preferensi penerapan Anda.
 8. Pilih nama file konfigurasi default.
 9. Pilih lingkungan konfigurasi default.

Berikut ini adalah contoh output dari aliran `sam deploy --guided` interaktif:

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

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

```

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

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

```

3. AWS SAMCLIMenyebarkan aplikasi Anda dengan melakukan hal berikut:

- AWS SAMCLIni membuat bucket Amazon S3 dan mengunggah direktori Anda. `. aws - sam`
- Ini AWS SAM CLI mengubah AWS SAM template Anda menjadi AWS CloudFormation dan mengunggahnya ke layanan. AWS CloudFormation
- AWS CloudFormationmenyediakan sumber daya Anda.

Selama penerapan, AWS SAM CLI menampilkan kemajuan Anda. Berikut ini adalah contoh output:

```

Looking for resources needed for deployment:

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

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

```



```

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

```

```

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

```

```

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

```

```

Initiating deployment
=====

```

```

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

```

```

Waiting for changeset to be created..

```

```

CloudFormation stack changeset
-----

```

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

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

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

```
CloudFormation events from stack operations (refresh every 0.5 seconds)
```

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

```
CloudFormation outputs from deployed stack
```

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

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

Key          HelloWorldApi
Description  API Gateway endpoint URL for Prod stage for Hello World
function
```

```

Value          https://<restapiid>.execute-api.us-west-2.amazonaws.com/Prod/
hello/

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

```

```

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

```

Aplikasi Anda sekarang digunakan dan berjalan di! AWS Cloud

Langkah 4: Jalankan aplikasi Anda

Pada langkah ini, Anda akan mengirim permintaan GET ke titik akhir API Anda dan melihat output fungsi Lambda Anda.

Untuk mendapatkan nilai endpoint API Anda

1. Dari informasi yang ditampilkan oleh AWS SAM CLI pada langkah sebelumnya, cari Outputs bagian. Di bagian ini, cari HelloWorldApi sumber daya Anda untuk menemukan nilai titik akhir HTTP Anda. Berikut ini adalah contoh output:

```

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

```

2. Atau, Anda dapat menggunakan `sam list endpoints --output json` perintah untuk mendapatkan informasi ini. Berikut ini adalah contoh output:

```

$ sam list endpoints --output json

```

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

Untuk memanggil fungsi Anda

- Menggunakan browser atau baris perintah, kirim permintaan GET ke titik akhir API Anda. Berikut ini adalah contoh menggunakan perintah curl:

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

Langkah 5: Berinteraksi dengan fungsi Anda di AWS Cloud

Pada langkah ini, Anda menggunakan tombol AWS SAM CLI untuk menjalankan fungsi Lambda Anda di AWS Cloud

Untuk menjalankan fungsi Lambda Anda di cloud

1. Catat fungsi Anda `LogicalResourceId` dari langkah sebelumnya. Seharusnya `beginHelloWorldFunction`.
2. Di baris perintah Anda, dari direktori `sam-app` proyek, jalankan yang berikut ini:

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

3. AWS SAM CLI memanggil fungsi Anda di cloud dan mengembalikan respons. Berikut ini adalah contoh output:

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

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

Langkah 6: Ubah dan sinkronkan aplikasi Anda ke AWS Cloud

Pada langkah ini, Anda menggunakan AWS SAM CLI `sam sync --watch` perintah untuk menyinkronkan perubahan lokal ke file AWS Cloud.

Untuk menggunakan `sam sync`

1. Di baris perintah Anda, dari direktori `sam-app` proyek, jalankan yang berikut ini:

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

2. Ini AWS SAM CLI meminta Anda untuk mengonfirmasi bahwa Anda menyinkronkan tumpukan pengembangan. Karena `sam sync --watch` perintah secara otomatis menyebarkan perubahan lokal ke AWS Cloud dalam waktu nyata, kami merekomendasikannya hanya untuk lingkungan pengembangan.

AWS SAM CLI melakukan penerapan awal sebelum mulai memantau perubahan lokal. Berikut ini adalah contoh output:

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

```
The SAM CLI will use the AWS Lambda, Amazon API Gateway, and AWS StepFunctions APIs
to upload your code without
performing a CloudFormation deployment. This will cause drift in your
CloudFormation stack.
```

```
**The sync command should only be used against a development stack**.
```

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

```
Enter Y to proceed with the command, or enter N to cancel:
```

```
[Y/n]: y
```

```
Queued infra sync. Waiting for in progress code syncs to complete...
```

```
Starting infra sync.
```

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

```
Build Succeeded
```

```
Successfully packaged artifacts and wrote output template to file /var/
folders/45/5ct135bx3fn2551_pt15g6_80000gr/T/tmpq3x9vh63.
```

```
Execute the following command to deploy the packaged template
```

```
sam deploy --template-file /var/folders/45/5ct135bx3fn2551_pt15g6_80000gr/T/
tmpq3x9vh63 --stack-name <YOUR STACK NAME>
```

```
Deploying with following values
```

```
=====
```

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

```
Initiating deployment
```

```
=====
```

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

```
CloudFormation events from stack operations (refresh every 0.5 seconds)
```

```
-----
ResourceStatus           ResourceType
LogicalResourceId        ResourceStatusReason
```

```

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

```

CloudFormation outputs from deployed stack

Outputs

```

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

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

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

```

```
Stack update succeeded. Sync infra completed.
```

```
Infra sync completed.
```

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

Selanjutnya, Anda akan memodifikasi kode fungsi Lambda Anda. Secara otomatis AWS SAM CLI akan mendeteksi perubahan ini dan menyinkronkan aplikasi Anda ke file AWS Cloud.

Untuk memodifikasi dan menyinkronkan aplikasi Anda

1. Dalam IDE pilihan Anda, buka `sam-app/hello_world/app.py` file.
2. Ubah message dan simpan file Anda. Berikut ini adalah contohnya:

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

3. AWS SAM CLI Mendeteksi perubahan Anda dan menyinkronkan aplikasi Anda ke file. AWS Cloud Berikut ini adalah contoh output:

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

4. Untuk memverifikasi perubahan Anda, kirim permintaan GET ke titik akhir API Anda lagi.

```
$ curl https://ets1gv8lxi.execute-api.us-west-2.amazonaws.com/Prod/hello/
```



```
{"message": "hello everyone!"}
```

Langkah 7: (Opsional) Uji aplikasi Anda secara lokal

Note

Langkah ini opsional karena diperlukan Docker pada mesin lokal Anda.

Important

Untuk menggunakan AWS SAM CLI untuk pengujian lokal, Anda harus Docker menginstal dan mengkonfigurasi. Untuk informasi selengkapnya, lihat [Menginstal Docker](#).

Pada langkah ini, Anda menggunakan AWS SAM CLI `sam local` perintah untuk menguji aplikasi Anda secara lokal. Untuk mencapai hal ini, AWS SAM CLI menciptakan lingkungan lokal menggunakan Docker. Lingkungan lokal ini mengemulasi lingkungan eksekusi berbasis cloud dari fungsi Lambda Anda.

Anda akan melakukan hal berikut:

1. Buat lingkungan lokal untuk fungsi Lambda Anda dan panggil.
2. Host titik akhir HTTP API Anda secara lokal dan gunakan untuk menjalankan fungsi Lambda Anda.

Untuk menjalankan fungsi Lambda Anda secara lokal

1. Di baris perintah Anda, dari direktori `sam-app` proyek, jalankan yang berikut ini:

```
$ sam local invoke
```

2. AWS SAM CLI membuat Docker wadah lokal dan memanggil fungsi Anda. Berikut ini adalah contoh output:

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

```
Building image.....
Using local image: public.ecr.aws/lambda/python:3.9-rapid-x86_64.

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

Untuk meng-host API Anda secara lokal

1. Di baris perintah Anda, dari direktori sam-app proyek, jalankan yang berikut ini:

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

2. AWS SAMCLIni membuat Docker wadah lokal untuk fungsi Lambda Anda dan membuat server HTTP lokal untuk mensimulasikan titik akhir API Anda. Berikut ini adalah contoh output:

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

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

3. Menggunakan browser Anda atau baris perintah, kirim permintaan GET ke endpoint API lokal Anda. Berikut ini adalah contoh menggunakan perintah curl:

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

Langkah 8: Hapus aplikasi Anda dari AWS Cloud

Pada langkah ini, Anda menggunakan AWS SAM CLI `sam delete` perintah untuk menghapus aplikasi Anda dari fileAWS Cloud.

Untuk menghapus aplikasi Anda dari AWS Cloud

1. Di baris perintah Anda, dari direktori `sam-app` proyek, jalankan yang berikut ini:

```
$ sam delete
```

2. Mereka AWS SAM CLI akan meminta Anda untuk mengkonfirmasi. Kemudian, itu akan menghapus bucket dan AWS CloudFormation stack Amazon S3 aplikasi Anda. Berikut ini adalah contoh output:

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

Deleted successfully
```

Memecahkan masalah

Untuk memecahkan masalah AWS SAMCLI, lihat. [AWS SAMCLIpemecahan masalah](#)

Pelajari selengkapnya

Untuk terus mempelajarinya AWS SAM, lihat sumber daya berikut:

- [AWS SAM Lokakarya Lengkap](#) — Lokakarya yang dirancang untuk mengajari Anda banyak fitur utama yang AWS SAM menyediakan.
- [Sesi dengan SAM](#) - Seri video yang dibuat oleh tim Advokat Pengembang AWS Tanpa Server kami tentang penggunaan. AWS SAM
- [Serverless Land](#) — Situs yang menyatukan informasi terbaru, blog, video, kode, dan sumber belajar untuk AWS tanpa server.

Menggunakan AWS Serverless Application Model (AWS SAM)

Gunakan AWS Serverless Application Model (AWS SAM) untuk membangun dan menjalankan aplikasi tanpa server Anda.

Untuk pengantar AWS SAM, lihat [Apa itu AWS SAM?](#)

Untuk informasi tentang penggunaan AWS SAM template, lihat [AWS Serverless Application Model \(AWS SAM\) spesifikasi](#).

Topik

- [Menggunakan AWS SAM CLI](#)

Menggunakan AWS SAM CLI

Gunakan AWS Serverless Application Model Command Line Interface (AWS SAM CLI) dengan AWS SAM template dan integrasi pihak ketiga yang didukung untuk membangun dan menjalankan aplikasi tanpa server Anda.

Untuk pengantar AWS SAM, lihat [Apa itu AWS SAM?](#)

Topik

- [Bagaimana AWS SAM CLI perintah didokumentasikan](#)
- [Mengkonfigurasi AWS SAM CLI](#)
- [Menggunakan sam build](#)
- [Menggunakan sam deploy](#)
- [Menggunakan sam init](#)
- [Menggunakan sam local](#)
- [Menggunakan sam remote invoke](#)
- [Menggunakan sam remote test-event](#)
- [Menggunakan sam sync](#)

Bagaimana AWS SAMCLI perintah didokumentasikan

AWS SAMCLIperintah didokumentasikan menggunakan format berikut:

- Prompt - Linux Prompt didokumentasikan secara default dan ditampilkan sebagai (\$). Untuk perintah yang Windows spesifik, (>) digunakan sebagai prompt. Jangan sertakan prompt saat Anda mengetik perintah.
- Direktori — Ketika perintah harus dijalankan dari direktori tertentu, nama direktori ditampilkan sebelum simbol prompt.
- Input pengguna - Teks perintah yang Anda masukkan pada baris perintah diformat sebagai **user input**.
- Teks yang dapat diganti - Teks variabel, seperti nama file dan parameter diformat sebagai teks yang *dapat diganti*. Dalam perintah atau perintah multi-baris di mana input keyboard tertentu diperlukan, input keyboard juga dapat ditampilkan sebagai teks yang dapat diganti. Misalnya, *ENTER*.
- Output - Output yang dikembalikan sebagai respons terhadap perintah diformat sebagai **computer output**.

sam deployPerintah dan output berikut adalah contoh:

```
$ 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` adalah perintah yang Anda masukkan di baris perintah.
2. `sam deploy --guided --template` harus disediakan apa adanya.
3. `template.yaml` dapat diganti dengan nama file spesifik Anda.
4. Output dimulai pada `Configuring SAM deploy`.
5. Dalam output, `ENTER` dan `y` menunjukkan nilai yang dapat diganti yang Anda berikan.

Mengkonfigurasi AWS SAMCLI

Konfigurasi kredensial, pengaturan dasar, dan pengaturan proyek untuk Antarmuka Baris AWS Serverless Application Model Perintah () AWS SAMCLI.

Topik

- [Konfigurasi kredensial dan pengaturan dasar](#)
- [Konfigurasi pengaturan proyek](#)

Konfigurasi kredensial dan pengaturan dasar

Gunakan AWS Command Line Interface (AWS CLI) untuk mengonfigurasi pengaturan dasar seperti AWS kredensial, nama wilayah default, dan format output default. Setelah dikonfigurasi, Anda dapat menggunakan pengaturan ini dengan file AWS SAMCLI. Untuk mempelajari lebih lanjut, lihat berikut ini dari Panduan AWS Command Line Interface Pengguna:

- [Dasar-dasar konfigurasi](#)
- [Konfigurasi dan pengaturan file kredensi](#)
- [Profil bernama untuk AWS CLI](#)
- [Menggunakan profil bernama IAM Identity Center diaktifkan](#)

Untuk petunjuk penyiapan cepat, lihat [Langkah 5: Gunakan AWS CLI untuk mengkonfigurasi AWS kredensial](#).

Konfigurasi pengaturan proyek

Anda dapat menentukan pengaturan khusus proyek, seperti nilai parameter AWS SAM CLI perintah, dalam file konfigurasi yang akan digunakan dengan. AWS SAM CLI Untuk informasi selengkapnya tentang file konfigurasi ini, lihat [AWS SAM CLI berkas konfigurasi](#).

Menggunakan file konfigurasi

File konfigurasi disusun berdasarkan lingkungan, perintah, dan nilai parameter. Untuk informasi selengkapnya, lihat [Dasar-dasar file konfigurasi](#).

Untuk mengkonfigurasi lingkungan baru

1. Tentukan lingkungan baru Anda di file konfigurasi Anda.

Berikut ini adalah contoh menentukan prod lingkungan baru:

MAKAM

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

YAML

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

2. Tentukan nilai parameter sebagai pasangan nilai kunci di bagian parameter file konfigurasi.

Berikut ini adalah contoh menentukan nama stack aplikasi Anda untuk prod lingkungan.

MAKAM

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

YAML

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



```
stack_name: prod-app
```

- Gunakan `--config-env` opsi untuk menentukan lingkungan yang akan digunakan.

Berikut ini adalah contohnya:

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

Untuk mengkonfigurasi nilai parameter

- Tentukan AWS SAM CLI perintah yang ingin Anda konfigurasi nilai parameternya. Untuk mengkonfigurasi nilai parameter untuk semua AWS SAM CLI perintah, gunakan `global` pengenalan.

Berikut ini adalah contoh menentukan nilai parameter untuk `sam deploy` perintah default lingkungan:

MAKAM

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

YAML

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

Berikut ini adalah contoh menentukan nilai parameter untuk semua AWS SAM CLI perintah di default lingkungan:

MAKAM

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

YAML

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

2. Anda juga dapat menentukan nilai parameter dan memodifikasi file konfigurasi Anda melalui aliran AWS SAM CLI interaktif.

Berikut ini adalah contoh aliran `sam deploy --guided` interaktif:

```
$ 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
```

Untuk informasi selengkapnya, lihat [Membuat dan memodifikasi file konfigurasi](#).

Contoh-contoh

TOMLContoh dasar

Berikut ini adalah contoh file `samconfig.toml` konfigurasi:

```
...
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
```

YAMLContoh dasar

Berikut ini adalah contoh file `samconfig.yaml` konfigurasi:

```
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
```

Menggunakan sam build

Gunakan AWS Serverless Application Model perintah Command Line Interface (AWS SAMCLI) `sam build` untuk mempersiapkan aplikasi tanpa server Anda untuk langkah-langkah selanjutnya dalam alur kerja pengembangan Anda, seperti pengujian lokal atau penerapan ke file. AWS Cloud Perintah ini membuat `.aws-sam` direktori yang menyusun aplikasi Anda dalam format dan lokasi yang `sam local` `sam deploy` dibutuhkan.

- Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#).
- Untuk daftar opsi `sam build` perintah, lihat [sam build](#).
- Untuk contoh penggunaan `sam build` selama alur kerja pengembangan tipikal, lihat [Langkah 2: Bangun aplikasi Anda](#).

Note

Menggunakan `sam build` mengharuskan Anda memulai dengan komponen dasar aplikasi tanpa server pada mesin pengembangan Anda. Ini termasuk AWS SAM template, kode AWS Lambda fungsi, dan file dan dependensi khusus bahasa apa pun. Untuk mempelajari selengkapnya, lihat [Menggunakan sam init](#).

Topik

- [Membangun aplikasi dengan sam build](#)
- [Pengujian dan penyebaran lokal](#)
- [Praktik terbaik](#)
- [Opsi untuk sam build](#)
- [Memecahkan masalah](#)
- [Contoh-contoh](#)
- [Pelajari selengkapnya](#)

Membangun aplikasi dengan sam build

Sebelum menggunakan `sam build`, pertimbangkan dan konfigurasi yang berikut ini:

1. Fungsi dan lapisan Lambda — `sam build` Perintah dapat membangun fungsi dan lapisan Lambda. Untuk mempelajari lebih lanjut tentang lapisan Lambda, lihat. [Membangun lapisan](#)
2. Lambda runtime — Runtime menyediakan lingkungan khusus bahasa yang menjalankan fungsi Anda di lingkungan eksekusi saat dipanggil. Anda dapat mengonfigurasi runtime asli dan kustom.
 - a. Runtime asli - Buat fungsi Lambda Anda dalam runtime Lambda yang didukung dan buat fungsi Anda untuk menggunakan runtime Lambda asli di. AWS Cloud
 - b. Custom runtime - Buat fungsi Lambda Anda menggunakan bahasa pemrograman apa pun dan buat runtime Anda menggunakan proses kustom yang ditentukan dalam pembuat atau pihak ketiga makefile seperti. `esbuild` Untuk mempelajari selengkapnya, lihat [Membangun waktu aktif kustom](#).
3. Jenis paket Lambda - Fungsi Lambda dapat dikemas dalam jenis paket penyebaran Lambda berikut:
 - a. `.zip` file archive - Berisi kode aplikasi Anda dan dependensinya.
 - b. Gambar kontainer - Berisi sistem operasi dasar, runtime, ekstensi Lambda, kode aplikasi Anda dan dependensinya.

Pengaturan aplikasi ini dapat dikonfigurasi saat menginisialisasi aplikasi menggunakan `sam init`.

- Untuk mempelajari lebih lanjut tentang menggunakan `sam init`, lihat [Menggunakan sam init](#).
- Untuk mempelajari lebih lanjut tentang mengonfigurasi setelan ini di aplikasi Anda, lihat [Aplikasi Pembangunan](#).

Untuk membangun aplikasi

1. cd ke akar proyek Anda. Ini adalah lokasi yang sama dengan AWS SAM template Anda.

```
$ cd sam-app
```

2. Jalankan hal berikut:

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

Note

Opsi yang umum digunakan adalah `--use-container`. Untuk mempelajari selengkapnya, lihat [Membangun fungsi Lambda di dalam wadah yang disediakan](#).

Berikut ini adalah contoh dari AWS SAM CLI output:

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

Build Succeeded

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

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

3. AWS SAMCLI Membuat direktori `.aws-sam build`. Berikut ini adalah contohnya:

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

Tergantung pada bagaimana aplikasi Anda dikonfigurasi, AWS SAM CLI melakukan hal berikut:

1. Mengunduh, menginstal, dan mengatur dependensi di direktori `.aws-sam/build`
2. Siapkan kode Lambda Anda. Ini dapat mencakup kompilasi kode Anda, membuat binari yang dapat dieksekusi, dan membangun gambar kontainer.
3. Salinan membangun artefak ke `.aws-sam` direktori. Formatnya akan bervariasi berdasarkan jenis paket aplikasi Anda.
 - a. Untuk jenis paket.zip, artefak belum di-zip sehingga dapat digunakan untuk pengujian lokal. AWS SAMCLIRitsleting aplikasi Anda saat menggunakan `sam deploy`.
 - b. Untuk jenis paket gambar kontainer, gambar kontainer dibuat secara lokal dan direferensikan dalam file `.aws-sam/build.toml`
4. Salin AWS SAM template ke `.aws-sam` direktori dan memodifikasinya dengan jalur file baru bila diperlukan.

Berikut ini adalah komponen utama yang membentuk artefak build Anda di `.aws-sam` direktori:

- Direktori build — Berisi fungsi dan lapisan Lambda Anda yang terstruktur secara independen satu sama lain. Ini menghasilkan struktur unik untuk setiap fungsi atau lapisan dalam `.aws-sam/build` direktori.
- AWS SAMTemplate — Dimodifikasi dengan nilai yang diperbarui berdasarkan perubahan selama proses pembuatan.
- File BUILD.TOLL — File konfigurasi yang berisi pengaturan build yang digunakan oleh file. AWS SAM CLI

Pengujian dan penyebaran lokal

Saat melakukan pengujian lokal dengan `sam local` atau penerapan dengan `sam deploy`, AWS SAM CLI melakukan hal berikut:

1. Ini pertama memeriksa untuk melihat apakah `.aws-sam` direktori ada dan apakah AWS SAM template terletak di dalam direktori itu. Jika kondisi ini terpenuhi, AWS SAM CLI menganggap ini sebagai direktori root aplikasi Anda.
2. Jika kondisi ini tidak terpenuhi, AWS SAM CLI menganggap lokasi asli AWS SAM template Anda sebagai direktori root aplikasi Anda.

Saat mengembangkan, jika perubahan dilakukan pada file aplikasi asli Anda, jalankan `sam build` untuk memperbarui `.aws-sam` direktori sebelum menguji secara lokal.

Praktik terbaik

- Jangan mengedit kode apa pun di bawah `.aws-sam/build` direktori. Sebagai gantinya, perbarui kode sumber asli Anda di folder proyek Anda dan jalankan `sam build` untuk memperbarui `.aws-sam/build` direktori.
- Saat Anda memodifikasi file asli Anda, jalankan `sam build` untuk memperbarui `.aws-sam/build` direktori.
- Anda mungkin AWS SAM CLI ingin mereferensikan direktori root asli proyek Anda alih-alih `.aws-sam` direktori, seperti saat mengembangkan dan menguji dengan `sam local`. Hapus `.aws-sam` direktori atau AWS SAM template dalam `.aws-sam` direktori untuk AWS SAM CLI mengenali direktori proyek asli Anda sebagai direktori proyek root. Saat siap, jalankan `sam build` lagi untuk membuat `.aws-sam` direktori.
- Ketika Anda menjalankan `sam build`, `.aws-sam/build` direktori akan ditimpa setiap kali. `.aws-sam` Direktori tidak. Jika Anda ingin menyimpan file, seperti log, simpan `.aws-sam` untuk mencegahnya ditimpa.

Opsi untuk `sam build`

Membangun sumber daya tunggal

Berikan ID logis sumber daya untuk hanya membangun sumber daya itu. Berikut ini adalah contohnya:


```
$ sam build HelloWorldFunction
```

Untuk membangun sumber daya aplikasi atau tumpukan bersarang, berikan ID logis aplikasi atau tumpukan bersama dengan ID logis sumber daya menggunakan format `<stack-logical-id>/<resource-logical-id>`:

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

Membangun fungsi Lambda di dalam wadah yang disediakan

`--use-container` Opsi mengunduh gambar kontainer dan menggunakannya untuk membangun fungsi Lambda Anda. Wadah lokal kemudian direferensikan dalam `.aws-sam/build.toml` file Anda.

Opsi ini Docker harus diinstal. Untuk petunjuk, lihat [Menginstal Docker](#).

Berikut ini adalah contoh dari perintah ini:

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

Anda dapat menentukan gambar kontainer yang akan digunakan dengan `--build-image` opsi. Berikut ini adalah contohnya:

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

Untuk menentukan gambar kontainer yang akan digunakan untuk satu fungsi, berikan fungsi ID logis. Berikut ini adalah contohnya:

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

Teruskan variabel lingkungan ke wadah build

Gunakan `--container-env-var` untuk meneruskan variabel lingkungan ke wadah build. Berikut ini adalah contohnya:

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

Untuk meneruskan variabel lingkungan dari file, gunakan `--container-env-var-file` opsi. Berikut ini adalah contohnya:

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

Contoh `env.json` file:

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

Mempercepat pembangunan aplikasi yang berisi banyak fungsi

Ketika Anda menjalankan `sam build` aplikasi dengan beberapa fungsi, AWS SAM CLI membangun setiap fungsi satu per satu. Untuk mempercepat proses pembuatan, gunakan `--parallel` opsi. Ini membangun semua fungsi dan lapisan Anda secara bersamaan.

Berikut ini adalah contoh dari perintah ini:

```
$ sam build --parallel
```

Mempercepat waktu pembuatan dengan membangun proyek Anda di folder sumber

Untuk runtime dan metode build yang didukung, Anda dapat menggunakan `--build-in-source` opsi untuk membangun proyek secara langsung di folder sumber. Secara default, AWS SAM CLI build dalam direktori sementara, yang melibatkan penyalinan kode sumber dan file proyek. Dengan `--build-in-source`, AWS SAM CLI build langsung di folder sumber Anda, yang mempercepat proses pembuatan dengan menghapus kebutuhan untuk menyalin file ke direktori sementara.

Untuk daftar runtime dan metode build yang didukung, lihat [--build-in-source](#).

Memecahkan masalah

Untuk memecahkan masalah AWS SAMCLI, lihat [AWS SAMCLIpemecahan masalah](#)

Contoh-contoh

Membangun aplikasi yang menggunakan runtime asli dan tipe paket.zip

Untuk contoh ini, lihat [Tutorial: Men-deploy aplikasi Hello World](#).

Membangun aplikasi yang menggunakan runtime asli dan tipe paket gambar

Pertama, kita jalankan `sam init` untuk menginisialisasi aplikasi baru. Selama aliran interaktif, kami memilih jenis Image paket. Berikut ini adalah contohnya:

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

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

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

Which runtime would you like to use?
...
 11 - java8
 12 - nodejs18.x
 13 - nodejs16.x
 14 - nodejs14.x
...
Runtime: 12
```

```
What package type would you like to use?
```

```
1 - Zip
```

```
2 - Image
```

```
Package type: 2
```

```
Based on your selections, the only dependency manager available is npm.
```

```
We will proceed copying the template using npm.
```

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

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

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

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

```
-----  
Generating application:  
-----
```

```
Name: sam-app
```

```
Base Image: amazon/nodejs18.x-base
```

```
Architectures: x86_64
```

```
Dependency Manager: npm
```

```
Output Directory: .
```

```
Configuration file: sam-app/samconfig.toml
```

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

```
...
```

AWS SAMCLIMenginisialisasi aplikasi dan membuat direktori proyek berikut:

```
sam-app  
### README.md  
### events  
#   ### event.json  
### hello-world  
#   ### Dockerfile
```

```
#   ### app.mjs
#   ### package.json
#   ### tests
#       ### unit
#           ### test-handler.mjs
### samconfig.toml
### template.yaml
```

Selanjutnya, kami menjalankan `sam build` untuk membangun aplikasi kami:

```
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
```

Membangun aplikasi yang mencakup bahasa pemrograman yang dikompilasi

Dalam contoh ini, kita membangun sebuah aplikasi yang berisi fungsi Lambda menggunakan runtime. Go

Pertama, kami menginisialisasi aplikasi baru menggunakan `sam init` dan mengonfigurasi aplikasi kami untuk digunakanGo:

```
$ sam init

...

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

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

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

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

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

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

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

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

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

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

```
-----  
Generating application:  
-----
```

```
Name: sam-app  
Runtime: go1.x  
Architectures: x86_64  
Dependency Manager: mod  
Application Template: hello-world  
Output Directory: .  
Configuration file: sam-app/samconfig.toml
```

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

```
...
```

AWS SAMCLIMenginisialisasi aplikasi. Berikut ini adalah contoh struktur direktori aplikasi:

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

Kami mereferensikan README .md file untuk persyaratan aplikasi ini.

```

...
## 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)
...

```

Selanjutnya, kita jalankan `sam local invoke` untuk menguji fungsi kita. Kesalahan perintah ini sejak Go tidak diinstal pada mesin lokal kami:

```

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"}%

```

Selanjutnya, kita jalankan `sam build` untuk membangun aplikasi kita. Kami menemukan kesalahan karena Go tidak diinstal pada mesin lokal kami:

```

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
```

Meskipun kami dapat mengonfigurasi mesin lokal kami untuk membangun fungsi kami dengan benar, kami malah menggunakan `--use-container` opsi dengansam `build`. AWS SAMCLIDownload gambar kontainer, membangun fungsi kita menggunakan native `GoModulesBuilder`, dan menyalin biner yang dihasilkan ke `.aws-sam/build/HelloWorldFunction` direktori kita.

```
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
```

Berikut ini adalah contoh `.aws-sam` direktori:

```
.aws-sam
### build
#   ### HelloWorldFunction
#   #   ### hello-world
#   ### template.yaml
```

```
### build.toml
### cache
#   ### c860d011-4147-4010-addb-2eaa289f4d95
#       ### hello-world
### deps
```

Selanjutnya, kita larisam `local invoke`. Fungsi kami berhasil dipanggil:

```
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"}%
```

Pelajari selengkapnya

Untuk mempelajari lebih lanjut tentang menggunakan `sam build` perintah, lihat berikut ini:

- [PembelajaranAWS SAM: sam build](#) — Seri “PembelajaranAWS SAM” Tanah Tanpa Server aktif. YouTube
- [Belajar AWS SAM | sam build | E3](#) — Seri “PembelajaranAWS SAM” Tanah Tanpa Server di. YouTube
- [AWS SAMbuild: bagaimana menyediakan artefak untuk penerapan \(Sesi Dengan SAM S2E8\)](#) — Sesi dengan seri aktif. AWS SAM YouTube
- [AWS SAMbuild kustom: Cara menggunakan Makefiles untuk menyesuaikan build di SAM \(S2E9\)](#) - Sesi dengan seri aktif. AWS SAM YouTube

Menggunakan sam deploy

Gunakan AWS Serverless Application Model perintah Command Line Interface (AWS SAMCLI) `sam deploy` untuk menyebarkan aplikasi tanpa server Anda ke file. AWS Cloud

- Untuk pengantar AWS SAM CLI, lihat [Apa itu AWS SAM CLI?](#).
- Untuk daftar opsi `sam deploy` perintah, lihat [sam deploy](#).
- Untuk contoh penggunaan `sam deploy` selama alur kerja pengembangan tipikal, lihat [Langkah 3: Menyebarkan aplikasi Anda ke AWS Cloud](#).

Topik

- [Prasyarat](#)
- [Menyebarkan aplikasi menggunakan sam deploy](#)
- [Praktik terbaik](#)
- [Opsi untuk penyebaran sam](#)
- [Memecahkan masalah](#)
- [Contoh-contoh](#)
- [Pelajari selengkapnya](#)

Prasyarat

Untuk menggunakan `sam deploy`, instal AWS SAM CLI dengan menyelesaikan yang berikut ini:

- [Prasyarat AWS SAM](#).
- [Instalasi AWS SAM CLI](#).

Sebelum menggunakan `sam deploy`, kami merekomendasikan pemahaman dasar tentang hal-hal berikut:

- [Mengkonfigurasi AWS SAM CLI](#).
- [Menggunakan sam init](#).
- [Menggunakan sam build](#).

Menyebarkan aplikasi menggunakan sam deploy

Saat Anda menerapkan aplikasi tanpa server untuk pertama kalinya, gunakan opsi tersebut. --guided Ini AWS SAM CLI akan memandu Anda melalui alur interaktif untuk mengonfigurasi pengaturan penerapan aplikasi Anda.

Untuk menyebarkan aplikasi menggunakan aliran interaktif

1. Pergi ke direktori root proyek Anda. Ini adalah lokasi yang sama dengan AWS SAM template Anda.

```
$ cd sam-app
```

2. Jalankan perintah berikut:

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

3. Selama alur interaktif, Anda AWS SAM CLI akan meminta opsi untuk mengonfigurasi pengaturan penerapan aplikasi Anda.

Kurung ([]) menunjukkan nilai default. Biarkan jawaban Anda kosong untuk memilih nilai default. Nilai default diperoleh dari file konfigurasi berikut:

- `~/.aws/config`— Pengaturan AWS akun umum Anda.
- `~/.aws/credentials`— Kredensi AWS akun Anda.
- `<project>/samconfig.toml`— File konfigurasi proyek Anda.

Berikan nilai dengan menjawab AWS SAM CLI petunjuknya. Misalnya, Anda dapat memasukkan `y` nilai `ya`, `n` untuk tidak, atau string.

AWS SAM CLI Menulis tanggapan Anda ke `samconfig.toml` file proyek Anda. Untuk penerapan berikutnya, Anda dapat menggunakan `sam deploy` untuk menerapkan menggunakan nilai yang dikonfigurasi ini. Untuk mengkonfigurasi ulang nilai-nilai ini, gunakan `sam deploy --guided` lagi atau langsung memodifikasi file konfigurasi Anda.

Berikut ini adalah contoh output:

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

Configuring SAM deploy
=====

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

    Setting default arguments for 'sam deploy'
```

```

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

```

4. Selanjutnya, AWS SAM CLI menyebarkan aplikasi Anda ke file. AWS Cloud Selama penerapan, kemajuan ditampilkan di prompt perintah Anda. Berikut ini adalah tahapan utama dalam penyebaran:
- Untuk aplikasi dengan AWS Lambda fungsi yang dikemas sebagai arsip file.zip, AWS SAM CLI ritsleting dan upload paket ke bucket Amazon Simple Storage Service (Amazon S3). Jika perlu, AWS SAM CLI akan membuat ember baru.
 - Untuk aplikasi dengan fungsi Lambda paket sebagai image kontainer, AWS SAM CLI upload gambar ke Amazon Elastic Container Registry (Amazon ECR). Jika perlu, AWS SAM CLI akan membuat repositori baru.
 - AWS SAMCLIMembuat set AWS CloudFormation perubahan dan menyebarkan aplikasi Anda AWS CloudFormation sebagai tumpukan.
 - AWS SAMCLIni memodifikasi AWS SAM template yang Anda gunakan dengan CodeUri nilai baru untuk fungsi Lambda Anda.

Berikut ini adalah contoh dari output AWS SAM CLI deployment:

```

Looking for resources needed for deployment:

Managed S3 bucket: aws-sam-cli-managed-default-
samclisourcebucket-1a4x26zbcdkqr

```

A different default S3 bucket can be set in `samconfig.toml` and auto resolution of buckets turned off by setting `resolve_s3=False`

Parameter `"stack_name=sam-app"` in `[default.deploy.parameters]` is defined as a global parameter `[default.global.parameters]`.

This parameter will be only saved under `[default.global.parameters]` in `/Users/.../sam-app/samconfig.toml`.

Saved arguments to config file

Running `'sam deploy'` for future deployments will use the parameters saved above.

The above parameters can be changed by modifying `samconfig.toml`

Learn more about `samconfig.toml` syntax at

<https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-sam-cli-config.html>

```

Uploading to sam-app-zip/da3c598813f1c2151579b73ad788cac8 262144 / 619839
(42.29%)Uploading to sam-app-zip/da3c598813f1c2151579b73ad788cac8 524288 / 619839
(84.58%)Uploading to sam-app-zip/da3c598813f1c2151579b73ad788cac8 619839 /
619839 (100.00%)

```

Deploying with following values

=====

```

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

```

Initiating deployment

=====

```

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

```

Waiting for changeset to be created..

CloudFormation stack changeset

```

Operation
Replacement
-----
+ Add          HelloWorldFunctionHell  AWS::Lambda::Permissio  N/A
                oWorldPermissionProd      n
+ Add          HelloWorldFunctionRole  AWS::IAM::Role           N/A
+ Add          HelloWorldFunction       AWS::Lambda::Function    N/A
+ Add          ServerlessRestApiDeplo  AWS::ApiGateway::Deplo  N/A
                yment47fc2d5f9d          yment
+ Add          ServerlessRestApiProdS  AWS::ApiGateway::Stage  N/A
                tage
+ Add          ServerlessRestApi       AWS::ApiGateway::RestA  N/A
                pi
-----

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

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

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

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

CREATE_IN_PROGRESS 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 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	ServerlessRestApiDeplo	-

	yment	yment47fc2d5f9d	
CREATE_IN_PROGRESS	AWS::ApiGateway::Stage	ServerlessRestApiProdS	-
		tage	
CREATE_IN_PROGRESS creation	AWS::ApiGateway::Stage	ServerlessRestApiProdS	Resource
Initiated		tage	
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. Untuk melihat aplikasi yang Anda gunakan, lakukan hal berikut:

1. Buka AWS CloudFormation konsol langsung dengan URL <https://console.aws.amazon.com/cloudformation>.
2. Pilih Tumpukan.
3. Identifikasi tumpukan Anda dengan nama aplikasi dan pilih.

Verifikasi perubahan sebelum penerapan

Anda dapat mengonfigurasi AWS SAM CLI untuk menampilkan set AWS CloudFormation perubahan Anda dan meminta konfirmasi sebelum menerapkan.

Untuk mengonfirmasi perubahan sebelum penerapan

1. Selamasam `deploy --guided`, masukkan **Y** untuk mengonfirmasi perubahan sebelum penerapan.

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

Atau, Anda dapat memodifikasi `samconfig.toml` file Anda dengan yang berikut:

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

2. Selama penyebaran, AWS SAM CLI akan meminta Anda untuk mengonfirmasi perubahan sebelum penerapan. Berikut ini adalah contohnya:

```
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
```

Tentukan parameter tambahan selama penerapan

Anda dapat menentukan nilai parameter tambahan untuk dikonfigurasi saat menerapkan. Anda melakukan ini dengan memodifikasi AWS SAM template Anda dan mengonfigurasi nilai parameter Anda selama penerapan.

Untuk menentukan parameter tambahan

1. Ubah Parameters bagian AWS SAM template Anda. Berikut ini adalah contohnya:

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

2. Jalankan `sam deploy --guided`. Berikut ini adalah contoh output:

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

Configuring SAM deploy
=====

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

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

Konfigurasi penandatanganan kode untuk fungsi Lambda Anda

Anda dapat mengonfigurasi penandatanganan kode untuk fungsi Lambda Anda saat penerapan. Anda melakukan ini dengan memodifikasi AWS SAM template Anda dan mengonfigurasi penandatanganan kode selama penerapan.

Untuk mengonfigurasi penandatanganan kode

1. Tentukan `CodeSigningConfigArn` dalam AWS SAM template Anda. Berikut ini adalah contohnya:

```
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. Jalankan `sam deploy --guided`. Ini AWS SAM CLI akan meminta Anda untuk mengonfigurasi penandatanganan kode. Berikut ini adalah contoh output:

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

Praktik terbaik

- Saat menggunakan `sam deploy`, AWS SAM CLI menerapkan artefak build aplikasi Anda yang terletak di direktori `.aws-sam`. Saat Anda membuat perubahan pada file asli aplikasi Anda, jalankan `sam build` untuk memperbarui `.aws-sam` direktori sebelum menerapkan.
- Saat menerapkan aplikasi untuk pertama kalinya, gunakan `sam deploy --guided` untuk mengonfigurasi pengaturan penerapan. Untuk penerapan berikutnya, Anda dapat menggunakan `sam deploy` untuk menerapkan dengan pengaturan yang dikonfigurasi.

Opsi untuk penyebaran sam

Berikut ini adalah opsi yang umum digunakan untuk `sam deploy`. Untuk daftar semua opsi, lihat [sam deploy](#).

Gunakan alur interaktif terpandu untuk menerapkan aplikasi Anda

Gunakan `--guided` opsi untuk mengonfigurasi pengaturan penerapan aplikasi Anda melalui alur interaktif. Berikut ini adalah contohnya:

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

Pengaturan penerapan aplikasi Anda disimpan dalam `samconfig.toml` file proyek Anda. Untuk mempelajari selengkapnya, lihat [Konfigurasi pengaturan proyek](#).

Memecahkan masalah

Untuk memecahkan masalah AWS SAM CLI, lihat [AWS SAM CLI pemecahan masalah](#)

Contoh-contoh

Menyebarkan aplikasi Hello World yang berisi fungsi Lambda yang dikemas sebagai arsip file.zip

Sebagai contoh, lihat [Langkah 3: Menyebarkan aplikasi Anda ke AWS Cloud](#) di tutorial aplikasi Hello World.

Menyebarkan aplikasi Hello World yang berisi fungsi Lambda yang dikemas sebagai gambar kontainer

Pertama, kita gunakan `sam init` untuk membuat aplikasi Hello World kita. Selama aliran interaktif, kami memilih Python3.9 runtime dan jenis Image paket.

```
$ 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`

...

Selanjutnya, kita `cd` ke direktori root proyek kita dan jalankan `sam build`. AWS SAM CLI membangun fungsi Lambda kami secara lokal menggunakan Docker

```
sam-app $ sam build
Building codeuri: /Users/.../sam-app runtime: None metadata: {'Dockerfile':
'Dockerfile', 'DockerContext': '/Users/.../sam-app/hello_world', 'DockerTag':
'python3.9-v1'} architecture: x86_64 functions: HelloWorldFunction
Building image for HelloWorldFunction function
Setting DockerBuildArgs: {} for HelloWorldFunction function
Step 1/5 : FROM public.ecr.aws/lambda/python:3.9
----> 0a5e3da309aa
Step 2/5 : COPY requirements.txt ./
----> abc4e82e85f9
Step 3/5 : RUN python3.9 -m pip install -r requirements.txt -t .
----> [Warning] The requested image's platform (linux/amd64) does not match the
detected host platform (linux/arm64/v8) and no specific platform was requested
----> Running in 43845e7aa22d
Collecting requests
  Downloading requests-2.28.2-py3-none-any.whl (62 kB)
##### 62.8/62.8 KB 829.5 kB/s eta 0:00:00
Collecting idna<4,>=2.5
  Downloading idna-3.4-py3-none-any.whl (61 kB)
##### 61.5/61.5 KB 2.4 MB/s eta 0:00:00
Collecting charset-normalizer<4,>=2
  Downloading charset_normalizer-3.1.0-cp39-cp39-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (199 kB)
##### 199.2/199.2 KB 2.1 MB/s eta 0:00:00
Collecting certifi>=2017.4.17
  Downloading certifi-2022.12.7-py3-none-any.whl (155 kB)
##### 155.3/155.3 KB 10.2 MB/s eta 0:00:00
Collecting urllib3<1.27,>=1.21.1
  Downloading urllib3-1.26.15-py2.py3-none-any.whl (140 kB)
##### 140.9/140.9 KB 9.1 MB/s eta 0:00:00
Installing collected packages: urllib3, idna, charset-normalizer, certifi, requests
Successfully installed certifi-2022.12.7 charset-normalizer-3.1.0 idna-3.4
requests-2.28.2 urllib3-1.26.15
Removing intermediate container 43845e7aa22d
----> cab8ace899ce
Step 4/5 : COPY app.py ./
```



```
---> 4146f3cd69f2
Step 5/5 : CMD ["app.lambda_handler"]
---> [Warning] The requested image's platform (linux/amd64) does not match the
detected host platform (linux/arm64/v8) and no specific platform was requested
---> Running in f4131ddffb31
Removing intermediate container f4131ddffb31
---> d2f5180b2154
Successfully built d2f5180b2154
Successfully tagged helloworldfunction:python3.9-v1

Build Succeeded

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

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

Selanjutnya, kami menjalankan `sam deploy --guided` untuk menyebarkan aplikasi kami. Ini AWS SAM CLI memandu kami melalui konfigurasi pengaturan penerapan kami. Kemudian, AWS SAM CLI menyebarkan aplikasi kita ke file. AWS Cloud

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

Configuring SAM deploy
=====

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

Setting default arguments for 'sam deploy'
=====
Stack Name [sam-app]: ENTER
AWS Region [us-west-2]: ENTER
#Shows you resources changes to be deployed and require a 'Y' to initiate
deploy
Confirm changes before deploy [Y/n]: ENTER
```

```
#SAM needs permission to be able to create roles to connect to the resources in
your template
Allow SAM CLI IAM role creation [Y/n]: ENTER
#Preserves the state of previously provisioned resources when an operation
fails
Disable rollback [y/N]: ENTER
HelloWorldFunction may not have authorization defined, Is this okay? [y/N]: y
Save arguments to configuration file [Y/n]: ENTER
SAM configuration file [samconfig.toml]: ENTER
SAM configuration environment [default]: ENTER

Looking for resources needed for deployment:

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

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

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

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

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

Deploying with following values
=====
```

```

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

Initiating deployment
 =====

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

Waiting for changeset to be created..

CloudFormation stack changeset

Operation	LogicalResourceId	ResourceType	Replacement
+ Add	HelloWorldFunctionHell 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              User
Initiated
                                tack

CREATE_IN_PROGRESS  AWS::IAM::Role        HelloWorldFunctionRole -
CREATE_IN_PROGRESS  AWS::IAM::Role        HelloWorldFunctionRole Resource
creation                                                    Initiated

CREATE_COMPLETE     AWS::IAM::Role        HelloWorldFunctionRole -
CREATE_IN_PROGRESS  AWS::Lambda::Function  HelloWorldFunction   -
CREATE_IN_PROGRESS  AWS::Lambda::Function  HelloWorldFunction   Resource
creation
    
```

				Initiated
CREATE_COMPLETE	AWS::Lambda::Function	HelloWorldFunction	-	
CREATE_IN_PROGRESS	AWS::ApiGateway::RestA 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
```

Pelajari selengkapnya

Untuk mempelajari lebih lanjut tentang menggunakan AWS SAM CLI `sam deploy` perintah, lihat yang berikut ini:

- [AWS SAM Lokakarya Lengkap: Modul 3 - Terapkan secara manual](#) - Pelajari cara membuat, mengemas, dan menerapkan aplikasi tanpa server menggunakan aplikasi. AWS SAM CLI

Menggunakan `sam init`

AWS Serverless Application Model Perintah Command Line Interface (AWS SAM CLI) `sam init` menyediakan opsi untuk menginisialisasi aplikasi tanpa server baru yang terdiri dari:

- AWS SAM Template untuk menentukan kode infrastruktur Anda.
- Struktur folder yang mengatur aplikasi Anda.
- Konfigurasi untuk AWS Lambda fungsi Anda.

Untuk pengantar AWS SAM CLI, lihat [Apa itu AWS SAM CLI?](#).

Untuk daftar opsi AWS SAM CLI `sam init` perintah, lihat [sam init](#).

Topik

- [Menginisialisasi aplikasi tanpa server baru](#)
- [Opsi untuk `sam init`](#)
- [Memecahkan masalah](#)
- [Contoh-contoh](#)
- [Pelajari selengkapnya](#)

Menginisialisasi aplikasi tanpa server baru

Untuk menginisialisasi aplikasi tanpa server baru menggunakan AWS SAM CLI

1. `cd` ke direktori awal.
2. Jalankan yang berikut ini di baris perintah:

```
$ sam init
```

3. Ini AWS SAM CLI akan memandu Anda melalui aliran interaktif untuk membuat aplikasi tanpa server baru.

Pilih template awal

Template terdiri dari yang berikut:

1. AWS SAMTemplate untuk kode infrastruktur Anda.
2. Direktori proyek awal yang mengatur file proyek Anda. Misalnya, ini mungkin termasuk:
 - a. Struktur untuk kode fungsi Lambda Anda dan dependensinya.
 - b. eventsFolder yang berisi peristiwa pengujian untuk pengujian lokal.
 - c. testsFolder untuk mendukung pengujian unit.
 - d. samconfig.tomlFile untuk mengkonfigurasi pengaturan proyek.
 - e. ReadMeFile dan file proyek awal dasar lainnya.

Berikut ini adalah contoh direktori proyek awal:

```
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
```

Anda dapat memilih dari daftar Template Mulai AWS Cepat yang tersedia atau menyediakan Lokasi Template Kustom Anda sendiri.

Untuk memilih Template Mulai AWS Cepat

1. Saat diminta, pilih Templat Mulai AWS Cepat.
2. Pilih template Mulai AWS Cepat untuk memulai. Berikut ini adalah contohnya:

```
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
```

Untuk memilih lokasi template kustom Anda sendiri

1. Saat diminta, pilih Lokasi Template Kustom.

```
Which template source would you like to use?
```

- 1 - AWS Quick Start Templates
- 2 - Custom Template Location

```
Choice: 2
```

2. Ini AWS SAM CLI akan meminta Anda untuk memberikan lokasi template.

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

Berikan salah satu lokasi berikut ke arsip file.zip template Anda:

- GitHubrepositori — Jalur ke file.zip di repositori Anda. GitHub File harus berada di root repositori Anda.
 - Mercurialrepositori — Jalur ke file.zip di repositori Anda. Mercurial File harus berada di root repositori Anda.
 - .zip path — HTTPS atau jalur lokal ke file.zip Anda.
3. Ini AWS SAM CLI akan menginisialisasi aplikasi tanpa server Anda menggunakan template kustom Anda.

Pilih runtime

Ketika Anda memilih Template Mulai AWS Cepat, AWS SAM CLI meminta Anda untuk memilih runtime untuk fungsi Lambda Anda. Daftar opsi yang ditampilkan oleh adalah runtime AWS SAM CLI yang didukung secara native oleh Lambda.

- [Runtime](#) menyediakan lingkungan khusus bahasa yang berjalan di lingkungan eksekusi.
- [Saat di-deploy keAWS Cloud, layanan Lambda akan memanggil fungsi Anda di lingkungan eksekusi.](#)

Anda dapat menggunakan bahasa pemrograman lain dengan runtime khusus. Untuk melakukan ini, Anda perlu membuat struktur aplikasi awal secara manual. Anda kemudian dapat menggunakan `sam init` untuk menginisialisasi aplikasi dengan cepat dengan mengonfigurasi lokasi template kustom.

Dari pilihan Anda, AWS SAM CLI membuat direktori awal untuk kode fungsi dan dependensi Lambda Anda.

Jika Lambda mendukung beberapa manajer ketergantungan untuk runtime Anda, Anda akan diminta untuk memilih manajer ketergantungan pilihan Anda.

Pilih jenis paket

Bila Anda memilih AWSQuick Start Template dan runtime, AWS SAM CLI meminta Anda untuk memilih jenis paket. Jenis paket menentukan bagaimana fungsi Lambda Anda digunakan untuk digunakan dengan layanan Lambda. Dua jenis paket yang didukung adalah:

1. Container image - Berisi sistem operasi dasar, runtime, ekstensi Lambda, kode aplikasi Anda, dan dependensinya.
2. .zip file archive - Berisi kode aplikasi Anda dan dependensinya.

Untuk mempelajari lebih lanjut tentang jenis paket penerapan, lihat Paket penerapan [Lambda di AWS Lambda Panduan Pengembang](#).

Berikut ini adalah contoh struktur direktori aplikasi dengan fungsi Lambda dikemas sebagai gambar kontainer. AWS SAMCLIDownload gambar dan membuat Dockerfile dalam direktori fungsi untuk menentukan gambar.

```
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
```

Berikut ini adalah contoh struktur direktori aplikasi dengan fungsi yang dikemas sebagai arsip file.zip.

```
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
```

Konfigurasi AWS X-Ray penelusuran

Anda dapat memilih untuk mengaktifkan AWS X-Ray tracing. Untuk mempelajari lebih lanjut, lihat [Apa itu AWS X-Ray?](#) di Panduan AWS X-Ray Pengembang.

Jika Anda mengaktifkan, AWS SAM CLI mengkonfigurasi AWS SAM template Anda. Berikut ini adalah contohnya:

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

Konfigurasi pemantauan dengan Amazon CloudWatch Application Insights

Anda dapat memilih untuk mengaktifkan pemantauan menggunakan Amazon CloudWatch Application Insights. Untuk mempelajari selengkapnya, lihat [Wawasan CloudWatch Aplikasi Amazon](#) di Panduan CloudWatch Pengguna Amazon.

Jika Anda mengaktifkan, AWS SAM CLI mengkonfigurasi AWS SAM template Anda. Berikut ini adalah contohnya:

```
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

```

Beri nama aplikasi Anda

Berikan nama untuk aplikasi Anda. AWS SAMCLI membuat folder tingkat atas untuk aplikasi Anda menggunakan nama ini.

Opsi untuk sam init

Berikut ini adalah beberapa opsi utama yang dapat Anda gunakan dengan `sam init` perintah. Untuk daftar semua opsi, lihat [sam init](#).

Menginisialisasi aplikasi menggunakan lokasi template kustom

Gunakan `--location` opsi dan berikan lokasi template kustom yang didukung. Berikut ini adalah contohnya:

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

Menginisialisasi aplikasi tanpa aliran interaktif

Gunakan `--no-interactive` opsi dan berikan pilihan konfigurasi Anda di baris perintah untuk melewati alur interaktif. Berikut ini adalah contohnya:

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

Memecahkan masalah

Untuk memecahkan masalah AWS SAMCLI, lihat [AWS SAMCLIpemecahan masalah](#)

Contoh-contoh

Menginisialisasi aplikasi tanpa server baru menggunakan Hello World Starter Template AWS

Untuk contoh ini, lihat [Langkah 1: Inisialisasi contoh aplikasi Hello World](#) di Tutorial: Menyebarkan aplikasi Hello World.

Inisialisasi aplikasi tanpa server baru dengan lokasi template khusus

Berikut ini adalah contoh penyediaan GitHub lokasi ke template kustom Anda:

```
$ 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
```

Berikut ini adalah contoh dari path file lokal:

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

Berikut ini adalah contoh jalur yang dapat dijangkau oleh HTTPS:

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

Pelajari selengkapnya

Untuk mempelajari lebih lanjut tentang menggunakan `sam init` perintah, lihat berikut ini:

- [PembelajaranAWS SAM: sam init](#) — seri Serverless Land AWS SAM “Belajar” aktifYouTube.
- [Penataan aplikasi tanpa server untuk digunakan dengan AWS SAMCLI \(Sesi dengan SAM S2E7\) - Sesi dengan](#) seri aktif. AWS SAM YouTube

Menggunakan sam local

Gunakan AWS Serverless Application Model perintah Command Line Interface (AWS SAMCLI) `sam local` untuk menguji aplikasi tanpa server Anda secara lokal.

Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#).

Prasyarat

Untuk menggunakan `sam local`, instal AWS SAM CLI dengan menyelesaikan yang berikut ini:

- [Prasyarat AWS SAM](#).
- [Instalasi AWS SAMCLI](#).

Sebelum menggunakan `sam local`, kami merekomendasikan pemahaman dasar tentang hal-hal berikut:

- [Mengkonfigurasi AWS SAMCLI](#).
- [Menggunakan sam init](#).
- [Menggunakan sam build](#).
- [Menggunakan sam deploy](#).

Menggunakan sam local perintah

Gunakan `sam local` perintah dengan subperintahnya untuk melakukan berbagai jenis pengujian lokal untuk aplikasi Anda.

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

Untuk mempelajari lebih lanjut tentang setiap subperintah, lihat berikut ini:

- [sam local generate-event](#)— Hasilkan Layanan AWS acara untuk pengujian lokal.
- [sam local invoke](#)— Memulai pemanggilan satu kali fungsi secara lokal. AWS Lambda
- [sam local start-api](#)— Jalankan fungsi Lambda Anda menggunakan server HTTP lokal.
- [sam local start-lambda](#)— Jalankan fungsi Lambda Anda menggunakan server HTTP lokal untuk digunakan dengan AWS CLI atau SDK.

Menggunakan `sam local generate-event`

Gunakan AWS Serverless Application Model `sam local generate-event` subperintah Command Line Interface (AWS SAMCLI) untuk menghasilkan sampel payload peristiwa untuk didukung. Layanan AWS Anda kemudian dapat memodifikasi dan meneruskan peristiwa ini ke sumber daya lokal untuk pengujian.

- Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#).
- Untuk daftar opsi `sam local generate-event` perintah, lihat [sam local generate-event](#).

Peristiwa adalah objek JSON yang dihasilkan ketika Layanan AWS melakukan tindakan atau tugas. Peristiwa ini berisi informasi spesifik, seperti data yang diproses atau stempel waktu acara. Sebagian besar Layanan AWS menghasilkan acara dan setiap acara layanan diformat secara unik untuk layanannya.

Peristiwa yang dihasilkan oleh satu layanan diteruskan ke layanan lain sebagai sumber acara. Misalnya, item yang ditempatkan di bucket Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3) dapat menghasilkan peristiwa. Acara ini kemudian dapat digunakan sebagai sumber peristiwa untuk suatu AWS Lambda fungsi untuk memproses data lebih lanjut.

Peristiwa yang Anda hasilkan `sam local generate-event` diformat dalam struktur yang sama dengan peristiwa aktual yang dibuat oleh AWS layanan. Anda dapat memodifikasi konten acara ini dan menggunakannya untuk menguji sumber daya dalam aplikasi Anda.

Prasyarat

Untuk menggunakan `sam local generate-event`, instal AWS SAM CLI dengan menyelesaikan yang berikut ini:

- [Prasyarat AWS SAM](#).
- [Instalasi AWS SAMCLI](#).

Sebelum menggunakan `sam local generate-event`, kami merekomendasikan pemahaman dasar tentang hal-hal berikut:

- [Mengkonfigurasi AWS SAMCLI](#).
- [Menggunakan `sam init`](#).
- [Menggunakan `sam build`](#).

- [Menggunakan sam deploy](#).

Hasilkan contoh acara

Gunakan AWS SAM CLI `sam local generate-event` subperintah untuk menghasilkan peristiwa yang didukung Layanan AWS.

Untuk melihat daftar yang didukung Layanan AWS

1. Jalankan hal berikut:

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

2. Daftar yang didukung Layanan AWS akan ditampilkan. Berikut ini adalah contohnya:

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

Untuk menghasilkan acara lokal

1. Jalankan `sam local generate-event` dan berikan nama layanan yang didukung. Ini akan menampilkan daftar jenis acara yang dapat Anda hasilkan. Berikut ini adalah contohnya:

```
$ 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

- Untuk menghasilkan contoh acara, jalankan `sam local generate-event`, berikan layanan dan jenis acara.

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

Berikut ini adalah contohnya:

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

```
        "eTag": "0123456789abcdef0123456789abcdef",
        "sequencer": "0A1B2C3D4E5F678901"
    }
}
]
}
```

Contoh peristiwa ini berisi nilai placeholder. Anda dapat memodifikasi nilai-nilai ini untuk mereferensikan sumber daya aktual dalam aplikasi atau nilai Anda untuk membantu pengujian lokal.

Untuk memodifikasi contoh peristiwa

1. Anda dapat memodifikasi contoh peristiwa di command prompt. Untuk melihat opsi Anda, jalankan yang berikut ini:

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

Berikut ini adalah contohnya:

```
$ 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.

- Gunakan salah satu opsi ini di prompt perintah untuk memodifikasi payload acara sampel Anda. Berikut ini adalah contohnya:

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

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

```
}
```

Gunakan peristiwa yang dihasilkan untuk pengujian lokal

Simpan peristiwa yang Anda hasilkan secara lokal dan gunakan `sam local` subperintah lain untuk menguji.

Untuk menyimpan acara yang Anda hasilkan secara lokal

- Jalankan hal berikut:

```
$ sam local generate-event <service> <event> <event-option> > <filename.json>
```

Berikut ini adalah contoh acara yang disimpan sebagai `s3.json` file di `events` folder proyek kami.

```
sam-app$ sam local generate-event s3 put --bucket MyBucket > events/s3.json
```

Untuk menggunakan peristiwa yang dihasilkan untuk pengujian lokal

- Lewati acara dengan `sam local` subperintah lain dengan menggunakan `--event` opsi.

Berikut ini adalah contoh penggunaan `s3.json` acara untuk memanggil fungsi Lambda kami secara lokal:

```
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
```

Pelajari selengkapnya

Untuk daftar semua `sam local generate-event` opsi, lihat [sam local generate-event](#).

Untuk demo penggunaannya `sam local`, lihat [AWS SAM untuk pengembangan lokal. Menguji AWS Cloud sumber daya dari lingkungan pengembangan lokal](#) di Sesi Tanah Tanpa Server dengan seri SAM aktif. YouTube

Menggunakan `sam local invoke`

Gunakan AWS Serverless Application Model `sam local invoke` subperintah Command Line Interface (AWS SAM CLI) untuk memulai pemanggilan satu kali fungsi secara lokal. AWS Lambda

- Untuk pengantar AWS SAM CLI, lihat [Apa itu AWS SAM CLI?](#).
- Untuk daftar opsi `sam local invoke` perintah, lihat [sam local invoke](#).
- Untuk contoh penggunaan `sam local invoke` selama alur kerja pengembangan tipikal, lihat [Langkah 7: \(Opsional\) Uji aplikasi Anda secara lokal](#).

Prasyarat

Untuk menggunakan `sam local invoke`, instal AWS SAM CLI dengan menyelesaikan yang berikut ini:

- [Prasyarat AWS SAM](#).
- [Instalasi AWS SAM CLI](#).

Sebelum menggunakan `sam local invoke`, kami merekomendasikan pemahaman dasar tentang hal-hal berikut:

- [Mengkonfigurasi AWS SAM CLI](#).
- [Menggunakan `sam init`](#).
- [Menggunakan `sam build`](#).
- [Menggunakan `sam deploy`](#).

Memanggil fungsi Lambda secara lokal

Ketika Anda menjalankan `sam local invoke`, AWS SAM CLI mengasumsikan bahwa direktori kerja Anda saat ini adalah direktori root proyek Anda. Yang pertama AWS SAM CLI akan mencari

template.[yaml|yml] file dalam .aws-sam subfolder. Jika tidak ditemukan, AWS SAM CLI akan mencari template.[yaml|yml] file dalam direktori kerja Anda saat ini.

Untuk menjalankan fungsi Lambda secara lokal

1. Dari direktori root proyek Anda, jalankan yang berikut ini:

```
$ sam local invoke <options>
```

2. Jika aplikasi Anda berisi lebih dari satu fungsi, berikan ID logis fungsi tersebut. Berikut ini adalah contohnya:

```
$ sam local invoke HelloWorldFunction
```

3. AWS SAM CLI membangun fungsi Anda dalam wadah lokal menggunakan Docker. Kemudian memanggil fungsi Anda dan mengeluarkan respons fungsi Anda.

Berikut ini adalah contohnya:

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

Mengelola log

Saat menggunakan `sam local invoke`, output runtime fungsi Lambda (misalnya, log) adalah output ke `stderr`, dan hasil fungsi Lambda adalah output ke `stdout`

Berikut ini adalah contoh fungsi Lambda dasar:

```
def handler(event, context):
    print("some log") # this goes to stderr
    return "hello world" # this goes to stdout
```

Anda dapat menyimpan output standar ini. Berikut ini adalah contohnya:

```
$ 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
```

Anda dapat menggunakan output standar ini untuk lebih mengotomatiskan proses pengembangan lokal Anda.

Opsi

Lulus acara khusus untuk menjalankan fungsi Lambda

Untuk meneruskan acara ke fungsi Lambda, gunakan opsi. `--event` Berikut ini adalah contohnya:

```
$ sam local invoke --event events/s3.json S3JsonLoggerFunction
```


Anda dapat membuat acara dengan `sam local generate-event` subperintah. Untuk mempelajari selengkapnya, lihat [Menggunakan sam local generate-event](#).

Lulus variabel lingkungan saat menjalankan fungsi Lambda Anda

Jika fungsi Lambda Anda menggunakan variabel lingkungan, Anda dapat meneruskannya selama pengujian lokal dengan opsi `--env-vars`. Ini adalah cara yang bagus untuk menguji fungsi Lambda secara lokal dengan layanan di aplikasi Anda yang sudah digunakan cloud. Berikut ini adalah contohnya:

```
$ sam local invoke --env-vars locals.json
```

Tentukan templat atau fungsi

Untuk menentukan template untuk referensi AWS SAM CLI ke, gunakan `--template` opsi. AWS SAMCLI akan memuat hanya AWS SAM template itu dan sumber daya yang ditunjukkannya.

Untuk memanggil fungsi aplikasi atau tumpukan bersarang, berikan aplikasi atau tumpukan ID logis bersama dengan ID logika fungsi. Berikut ini adalah contohnya:

```
$ sam local invoke StackLogicalId/FunctionLogicalId
```

Uji fungsi Lambda dari proyek Anda Terraform

Gunakan `--hook-name` opsi untuk menguji fungsi Lambda secara lokal dari Terraform proyek Anda. Untuk mempelajari selengkapnya, lihat [Menggunakan AWS SAMCLI with Terraform untuk debugging dan pengujian lokal](#).

Berikut ini adalah contohnya:

```
$ sam local invoke --hook-name terraform --beta-features
```

Praktik terbaik

Jika aplikasi Anda memiliki `.aws-sam` direktori dari `berjalansam build`, pastikan untuk menjalankan `sam build` setiap kali Anda memperbarui kode fungsi Anda. Kemudian, jalankan `sam local invoke` untuk menguji kode fungsi Anda yang diperbarui secara lokal.

Pengujian lokal adalah solusi hebat untuk pengembangan dan pengujian cepat sebelum menerapkan ke cloud. Namun, pengujian lokal tidak memvalidasi semuanya, seperti izin antar sumber daya

Anda di cloud. Sebisa mungkin, uji aplikasi Anda di cloud. Sebaiknya [gunakan sam sync](#) untuk mempercepat alur kerja pengujian cloud Anda.

Contoh-contoh

Buat contoh peristiwa Amazon API Gateway dan gunakan untuk menjalankan fungsi Lambda secara lokal

Pertama, kami menghasilkan payload peristiwa API Gateway API API API dan menyimpannya ke events folder kami.

```
$ sam local generate-event apigateway http-api-proxy > events/apigateway_event.json
```

Selanjutnya, kami memodifikasi fungsi Lambda kami untuk mengembalikan nilai parameter dari acara tersebut.

```
def lambda_handler(event, context):
    print("HelloWorldFunction invoked")
    return {
        "statusCode": 200,
        "body": json.dumps({
            "message": event['queryStringParameters']['parameter2'],
        }),
    }
```

Selanjutnya, kami secara lokal memanggil fungsi Lambda kami dan menyediakan acara khusus kami.

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

Lulus variabel lingkungan saat menjalankan fungsi Lambda secara lokal

Aplikasi ini memiliki fungsi Lambda yang menggunakan variabel lingkungan untuk nama tabel Amazon DynamoDB. Berikut ini adalah contoh fungsi yang didefinisikan dalam AWS SAM template:

```
AWSTemplateFormatVersion: 2010-09-09
Transform: AWS::Serverless-2016-10-31
...
Resources:
  getAllItemsFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: src/get-all-items.getAllItemsHandler
      Description: get all items
      Policies:
        - DynamoDBReadPolicy:
            TableName: !Ref SampleTable
    Environment:
      Variables:
        SAMPLE_TABLE: !Ref SampleTable
    ...
```

Kami ingin menguji fungsi Lambda kami secara lokal sambil berinteraksi dengan tabel DynamoDB kami di cloud. Untuk melakukan ini, kami membuat file variabel lingkungan kami dan menyimpannya di direktori root proyek kami sebagai `locals.json`. Nilai yang disediakan di sini untuk `SAMPLE_TABLE` referensi tabel DynamoDB kami di cloud.

```
{
  "getAllItemsFunction": {
    "SAMPLE_TABLE": "dev-demo-SampleTable-1U991234LD5UM98"
  }
}
```

Selanjutnya, kita menjalankan `sam local invoke` dan meneruskan variabel lingkungan kita dengan `--env-vars` opsi.

```
$ 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": "{}"}
```

Pelajari selengkapnya

Untuk daftar semua `sam local invoke` opsi, lihat [sam local invoke](#).

Untuk demo penggunaan `sam local`, lihat [AWS SAM untuk pengembangan lokal. Menguji AWS Cloud sumber daya dari lingkungan pengembangan lokal](#) di Sesi Tanah Tanpa Server dengan seri SAM aktif. YouTube

Menggunakan `sam local start-api`

Gunakan AWS Serverless Application Model `sam local start-api` subperintah Command Line Interface (AWS SAM CLI) untuk menjalankan AWS Lambda fungsi Anda secara lokal dan menguji melalui host server HTTP lokal. Jenis pengujian ini berguna untuk fungsi Lambda yang dipanggil oleh titik akhir Amazon API Gateway.

- Untuk pengantar AWS SAM CLI, lihat [Apa itu AWS SAM CLI?](#).
- Untuk daftar opsi `sam local start-api` perintah, lihat [sam local start-api](#).
- Untuk contoh penggunaan `sam local start-api` selama alur kerja pengembangan tipikal, lihat [Langkah 7: \(Opsional\) Uji aplikasi Anda secara lokal](#).

Prasyarat

Untuk menggunakan `sam local start-api`, instal AWS SAM CLI dengan menyelesaikan yang berikut ini:

- [Prasyarat AWS SAM](#).
- [Instalasi AWS SAM CLI](#).

Sebelum menggunakan `sam local start-api`, kami merekomendasikan pemahaman dasar tentang hal-hal berikut:

- [Mengkonfigurasi AWS SAM CLI](#).

- [Menggunakan sam init.](#)
- [Menggunakan sam build.](#)
- [Menggunakan sam deploy.](#)

Menggunakan sam local start-api

Ketika Anda menjalankansam local start-api, AWS SAM CLI mengasumsikan bahwa direktori kerja Anda saat ini adalah direktori root proyek Anda. Yang pertama AWS SAM CLI akan mencari template.[yaml|yml] file dalam .aws-sam subfolder. Jika tidak ditemukan, AWS SAM CLI akan mencari template.[yaml|yml] file dalam direktori kerja Anda saat ini.

Untuk memulai server HTTP lokal

1. Dari direktori root proyek Anda, jalankan yang berikut ini:

```
$ sam local start-api <options>
```

2. AWS SAMCLIMembangun fungsi Lambda Anda dalam wadah Docker lokal. Kemudian output alamat lokal dari endpoint server HTTP Anda. Berikut ini adalah contohnya:

```
$ 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. Anda dapat menjalankan fungsi Lambda Anda melalui browser atau command prompt. Berikut ini adalah contohnya:

```
sam-app$ curl http://127.0.0.1:3000/hello
{"message": "Hello world!"}%
```

4. Saat Anda membuat perubahan pada kode fungsi Lambda Anda, pertimbangkan hal berikut untuk menyegarkan server HTTP lokal Anda:
- Jika aplikasi Anda tidak memiliki `.aws-sam` direktori dan fungsi Anda menggunakan bahasa yang ditafsirkan, AWS SAM CLI maka akan secara otomatis memperbarui fungsi Anda dengan membuat wadah baru dan menghostingnya.
 - Jika aplikasi Anda memiliki `.aws-sam` direktori, Anda perlu menjalankan `sam build` untuk memperbarui fungsi Anda. Kemudian jalankan `sam local start-api` lagi untuk meng-host fungsi.
 - Jika fungsi Anda menggunakan bahasa yang dikompilasi atau jika proyek Anda memerlukan dukungan pengemasan yang kompleks, jalankan solusi build Anda sendiri untuk memperbarui fungsi Anda. Kemudian jalankan `sam local start-api` lagi untuk meng-host fungsi.

Fungsi Lambda yang menggunakan otorisasi Lambda

Note

Fitur ini baru di AWS SAM CLI versi 1.80.0. Untuk meningkatkan, lihat [Memutakhirkan AWS SAMCLI](#).

Untuk fungsi Lambda yang menggunakan otorisasi Lambda, Lambda akan AWS SAM CLI secara otomatis memanggil otorisasi Lambda Anda sebelum menjalankan titik akhir fungsi Lambda Anda.

Berikut ini adalah contoh memulai server HTTP lokal untuk fungsi yang menggunakan otorisasi Lambda:

```
$ 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

Saat Anda memanggil titik akhir fungsi Lambda Anda melalui server HTTP lokal, AWS SAM CLI yang pertama memanggil otorisasi Lambda Anda. Jika otorisasi berhasil, AWS SAM CLI akan memanggil titik akhir fungsi Lambda Anda. Berikut ini adalah contohnya:

```
$ 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 ms`Billed 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 ms`Billed 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 -
```

Opsi

Terus menggunakan kembali kontainer untuk mempercepat pemanggilan fungsi lokal

Secara default, AWS SAM CLI membuat wadah baru setiap kali fungsi Anda dipanggil melalui server HTTP lokal. Gunakan `--warm-containers` opsi untuk menggunakan kembali wadah Anda secara otomatis untuk pemanggilan fungsi. Ini mempercepat waktu yang dibutuhkan untuk mempersiapkan fungsi Lambda Anda AWS SAM CLI untuk pemanggilan lokal. Anda dapat menyesuaikan opsi ini lebih lanjut dengan memberikan `lazy` argumen `eager` atau.

- `eager`— Wadah untuk semua fungsi dimuat saat startup dan bertahan di antara pemanggilan.
- `lazy`— Kontainer hanya dimuat ketika setiap fungsi pertama kali dipanggil. Mereka kemudian bertahan untuk doa tambahan.

Berikut ini adalah contohnya:

```
$ sam local start-api --warm-containers eager
```

Saat menggunakan `--warm-containers` dan memodifikasi kode fungsi Lambda Anda:

- Jika aplikasi Anda memiliki `.aws-sam` direktori, jalankan `sam build` untuk memperbarui kode fungsi Anda di artefak build aplikasi Anda.
- Ketika perubahan kode terdeteksi, secara AWS SAM CLI otomatis menutup wadah fungsi Lambda.
- Ketika Anda memanggil fungsi lagi, secara AWS SAM CLI otomatis membuat wadah baru.

Tentukan gambar kontainer yang akan digunakan untuk fungsi Lambda Anda

Secara default, AWS SAM CLI menggunakan gambar dasar Lambda dari Amazon Elastic Container Registry (Amazon ECR) Registry (Amazon ECR) untuk menjalankan fungsi Anda secara lokal. Gunakan `--invoke-image` opsi untuk mereferensikan gambar kontainer kustom. Berikut ini adalah contohnya:

```
$ sam local start-api --invoke-image public.ecr.aws/sam/emu-python3.8
```


Anda dapat menentukan fungsi yang akan digunakan dengan gambar wadah khusus. Berikut ini adalah contohnya:

```
$ sam local start-api --invoke-image Function1=amazon/aws/sam-cli-emulation-image-python3.8
```

Tentukan templat untuk diuji secara lokal

Untuk menentukan template untuk referensi AWS SAM CLI ke, gunakan `--template` opsi. AWS SAMCLI akan memuat hanya AWS SAM template itu dan sumber daya yang ditunjukkannya. Berikut ini adalah contohnya:

```
$ sam local start-api --template myTemplate.yaml
```

Tentukan lingkungan pengembangan host dari fungsi Lambda Anda

Secara default, `sam local start-api` subperintah membuat server HTTP menggunakan `localhost` dengan alamat `127.0.0.1` IP. Anda dapat menyesuaikan nilai-nilai ini jika lingkungan pengembangan lokal Anda terisolasi dari mesin lokal Anda.

Gunakan `--container-host` opsi untuk menentukan host. Berikut ini adalah contohnya:

```
$ sam local start-api --container-host host.docker.internal
```

Gunakan `--container-host-interface` opsi untuk menentukan alamat IP jaringan host yang harus diikat oleh port kontainer. Berikut ini adalah contohnya:

```
$ sam local start-api --container-host-interface 0.0.0.0
```

Praktik terbaik

Jika aplikasi Anda memiliki `.aws-sam` direktori dari `berjalansam build`, pastikan untuk menjalankan `sam build` setiap kali Anda memperbarui kode fungsi Anda. Kemudian, jalankan `sam local start-api` untuk menguji kode fungsi Anda yang diperbarui secara lokal.

Pengujian lokal adalah solusi hebat untuk pengembangan dan pengujian cepat sebelum menerapkan ke cloud. Namun, pengujian lokal tidak memvalidasi semuanya, seperti izin antar sumber daya Anda di cloud. Sebisa mungkin, uji aplikasi Anda di cloud. Sebaiknya [gunakan `sam sync`](#) untuk mempercepat alur kerja pengujian cloud Anda.

Pelajari selengkapnya

Untuk daftar semua `sam local start-api` opsi, lihat [sam local start-api](#).

Menggunakan `sam local start-lambda`

Gunakan AWS Serverless Application Model `sam local start-lambda` subperintah Command Line Interface (AWS SAM CLI) untuk menjalankan AWS Lambda fungsi Anda melalui AWS Command Line Interface (AWS CLI) atau SDK. Perintah ini memulai titik akhir lokal yang mengemulasi AWS Lambda.

- Untuk pengantar AWS SAM CLI, lihat [Apa itu AWS SAM CLI?](#).
- Untuk daftar opsi `sam local start-lambda` perintah, lihat [sam local start-lambda](#).

Prasyarat

Untuk menggunakan `sam local start-lambda`, instal AWS SAM CLI dengan menyelesaikan yang berikut ini:

- [Prasyarat AWS SAM](#).
- [Instalasi AWS SAM CLI](#).

Sebelum menggunakan `sam local start-lambda`, kami merekomendasikan pemahaman dasar tentang hal-hal berikut:

- [Mengkonfigurasi AWS SAM CLI](#).
- [Menggunakan `sam init`](#).
- [Menggunakan `sam build`](#).
- [Menggunakan `sam deploy`](#).

Menggunakan `sam local start-lambda`

Ketika Anda menjalankan `sam local start-lambda`, AWS SAM CLI mengasumsikan bahwa direktori kerja Anda saat ini adalah direktori root proyek Anda. Yang pertama AWS SAM CLI akan mencari `template.[yaml|yml]` file dalam `.aws-sam` subfolder. Jika tidak ditemukan, AWS SAM CLI akan mencari `template.[yaml|yml]` file dalam direktori kerja Anda saat ini.

Untuk menggunakan sam local start-lambda

1. Dari direktori root proyek Anda, jalankan yang berikut ini:

```
$ sam local start-lambda <options>
```

2. AWS SAMCLIMembangun fungsi Lambda Anda dalam wadah Docker lokal. Kemudian output alamat lokal ke endpoint server HTTP Anda. Berikut ini adalah contohnya:

```
$ 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. Gunakan AWS CLI atau SDK untuk menjalankan fungsi Lambda Anda secara lokal.

Berikut ini adalah contoh menggunakanAWS CLI:

```
$ aws lambda invoke --function-name "HelloWorldFunction" --endpoint-
url "http://127.0.0.1:3001" --no-verify-ssl out.txt
```

```
StatusCode: 200
(END)
```

Berikut ini adalah contoh menggunakan AWS SDK forPython:

```
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")
```

Opsi

Tentukan templat

Untuk menentukan template untuk referensi AWS SAM CLI ke, gunakan `--template` opsi. AWS SAMCLI akan memuat hanya AWS SAM template itu dan sumber daya yang ditunjukkannya. Berikut ini adalah contohnya:

```
$ sam local start-lambda --template myTemplate.yaml
```

Praktik terbaik

Jika aplikasi Anda memiliki `.aws-sam` direktori dari `berjalansam build`, pastikan untuk menjalankan `sam build` setiap kali Anda memperbarui kode fungsi Anda. Kemudian, jalankan `sam local start-lambda` untuk menguji kode fungsi Anda yang diperbarui secara lokal.

Pengujian lokal adalah solusi hebat untuk pengembangan dan pengujian cepat sebelum menerapkan ke cloud. Namun, pengujian lokal tidak memvalidasi semuanya, seperti izin antar sumber daya Anda di cloud. Sebisa mungkin, uji aplikasi Anda di cloud. Sebaiknya [gunakan sam sync](#) untuk mempercepat alur kerja pengujian cloud Anda.

Pelajari selengkapnya

Untuk daftar semua `sam local start-lambda` opsi, lihat [sam local start-lambda](#).

Menggunakan sam remote invoke

Gunakan AWS Serverless Application Model perintah Command Line Interface (AWS SAMCLI) `sam remote invoke` untuk berinteraksi dengan AWS sumber daya yang didukung di AWS Cloud. Anda dapat menggunakan `sam remote invoke` untuk memanggil sumber daya berikut:

- Amazon Kinesis Data Streams — Mengirim catatan data ke aplikasi Kinesis Data Streams.
- AWS Lambda— Memanggil dan meneruskan acara ke fungsi Lambda Anda.
- Amazon Simple Queue Service (Amazon Simple Service) — Kirim pesan ke antrian Amazon SQS.
- AWS Step Functions— Memanggil mesin status Step Functions untuk memulai eksekusi.

Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#).

Untuk contoh penggunaan `sam remote invoke` selama alur kerja pengembangan tipikal, lihat [Langkah 5: Berinteraksi dengan fungsi Anda di AWS Cloud](#).

Topik

- [Menggunakan perintah `sam remote invoke`](#)
- [Menggunakan opsi perintah panggilan jarak jauh `sam`](#)
- [Konfigurasi file konfigurasi proyek Anda](#)
- [Contoh-contoh](#)
- [Tautan terkait](#)

Prasyarat

Untuk menggunakan `sam remote invoke`, instal AWS SAM CLI dengan menyelesaikan yang berikut:

- [Prasyarat AWS SAM](#).
- [Instalasi AWS SAMCLI](#).

Kami juga merekomendasikan untuk meningkatkan ke versi terbaru dari AWS SAM CLI Untuk mempelajari selengkapnya, lihat [Memutakhirkan AWS SAMCLI](#).

Sebelum menggunakan `sam remote invoke`, kami merekomendasikan pemahaman dasar tentang hal-hal berikut:

- [Mengkonfigurasi AWS SAMCLI](#).
- [Menggunakan `sam init`](#).
- [Menggunakan `sam build`](#).

- [Menggunakan sam deploy](#).
- [Menggunakan sam sync](#).

Menggunakan perintah sam remote invoke

Sebelum menggunakan perintah ini, sumber daya Anda harus dikerahkan ke file. AWS Cloud

Gunakan struktur perintah berikut dan jalankan dari direktori root proyek Anda:

```
$ sam remote invoke <arguments> <options>
```

Note

Halaman ini akan menampilkan opsi yang disediakan di command prompt. Anda juga dapat mengonfigurasi opsi dalam file konfigurasi proyek Anda alih-alih meneruskannya di prompt perintah. Untuk mempelajari lebih lanjut, [Konfigurasi pengaturan proyek](#).

Untuk deskripsi sam remote invoke argumen dan opsi, lihat [sam remote invoke](#).

Menggunakan dengan Kinesis Data Streams

Anda dapat mengirim catatan data ke aplikasi Kinesis Data Streams. Ini AWS SAM CLI akan mengirim catatan data Anda dan mengembalikan ID pecahan dan nomor urut. Berikut ini adalah contohnya:

```
$ 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"  
}%
```

Untuk mengirim catatan data

1. Berikan nilai ID sumber daya sebagai argumen untuk aplikasi Kinesis Data Streams Anda. Untuk informasi tentang ID sumber daya yang valid, lihat [ID Sumber Daya](#).
2. Berikan catatan data sebagai acara untuk dikirim ke aplikasi Kinesis Data Streams Anda. Anda dapat memberikan acara di baris perintah menggunakan `--event` opsi, atau dari file yang menggunakan `--event-file`. Jika Anda tidak memberikan acara, AWS SAM CLI mengirimkan acara kosong.

Menggunakan dengan fungsi Lambda

Anda dapat menjalankan fungsi Lambda di cloud dan meneruskan acara kosong atau memberikan acara di baris perintah atau dari file. AWS SAM CLI akan memanggil fungsi Lambda Anda dan mengembalikan responsnya. Berikut ini adalah contohnya:

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

Untuk menjalankan fungsi Lambda

1. Berikan nilai ID sumber daya sebagai argumen untuk fungsi Lambda Anda. Untuk informasi tentang ID sumber daya yang valid, lihat [ID Sumber Daya](#).
2. Berikan acara untuk dikirim ke fungsi Lambda Anda. Anda dapat memberikan acara di baris perintah menggunakan `--event` opsi, atau dari file yang menggunakan `--event-file`. Jika Anda tidak memberikan acara, AWS SAM CLI mengirimkan acara kosong.

Fungsi Lambda dikonfigurasi dengan streaming respons

`sam remote invoke` Perintah ini mendukung fungsi Lambda yang dikonfigurasi untuk mengalirkan respons. Anda dapat mengonfigurasi fungsi Lambda untuk mengalirkan respons menggunakan [FunctionUrlConfig](#) properti di templat Anda AWS SAM. Saat Anda menggunakan `sam remote`

`invoke`, AWS SAM CLI akan secara otomatis mendeteksi konfigurasi Lambda Anda dan memanggil dengan streaming respons.

Lihat contohnya di [Memanggil fungsi Lambda yang dikonfigurasi untuk mengalirkan respons](#).

Lulus peristiwa pengujian yang dapat dibagikan ke fungsi Lambda di cloud

Acara uji yang dapat dibagikan adalah acara uji yang dapat Anda bagikan dengan orang lain dalam hal yang sama di AWS. Untuk mempelajari selengkapnya, lihat [Acara pengujian yang dapat dibagikan](#) di Panduan AWS Lambda Pengembang.

Mengakses dan mengelola acara pengujian yang dapat dibagikan

Anda dapat menggunakan AWS SAM CLI `sam remote test-event` perintah untuk mengakses dan mengelola peristiwa pengujian yang dapat dibagikan. Misalnya, Anda dapat menggunakan `sam remote test-event` untuk melakukan hal berikut:

- Ambil peristiwa pengujian yang dapat dibagikan dari registri EventBridge skema Amazon.
- Ubah peristiwa pengujian yang dapat dibagikan secara lokal dan unggah ke registri EventBridge skema.
- Hapus peristiwa pengujian yang dapat dibagikan dari registri EventBridge skema.

Untuk mempelajari selengkapnya, lihat [Menggunakan sam remote test-event](#).

Lulus acara pengujian yang dapat dibagikan ke fungsi Lambda di cloud

Untuk meneruskan peristiwa pengujian yang dapat dibagikan dari registri EventBridge skema ke fungsi Lambda Anda di cloud, gunakan `--test-event-name` opsi dan berikan nama acara pengujian yang dapat dibagikan. Berikut ini adalah contohnya:

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --test-event-name demo-event
```

Jika Anda menyimpan peristiwa pengujian yang dapat dibagikan secara lokal, Anda dapat menggunakan `--event-file` opsi dan memberikan jalur file dan nama acara pengujian lokal. Berikut ini adalah contohnya:

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --event-file demo-event.json
```


Menggunakan dengan Amazon SQS

Anda dapat mengirim pesan ke antrian Amazon SQS. AWS SAM CLI Pengembalian berikut ini:

- ID Pesan
- MD5 dari badan pesan
- Metadata respons

Berikut ini adalah contohnya:

```
$ sam remote invoke MySqsQueue --stack-name sqs-example -event hello
```

```
Sending message to SQS queue MySqsQueue
```

```
{  
  "MD5ofMessageBody": "5d41402abc4b2a76b9719d911017c592",  
  "MessageId": "05c7af65-9ae8-4014-ae28-809d6d8ec652"  
}%
```

Untuk mengirim pesan

1. Berikan nilai ID sumber daya sebagai argumen untuk antrian Amazon SQS. Untuk informasi tentang ID sumber daya yang valid, lihat [ID Sumber Daya](#).
2. Berikan acara untuk dikirim ke antrian Amazon SQS Anda. Anda dapat memberikan acara di baris perintah menggunakan `--event` opsi, atau dari file yang menggunakan `--event-file`. Jika Anda tidak memberikan acara, AWS SAM CLI mengirimkan acara kosong.

Menggunakan dengan Step Functions

Anda dapat memanggil mesin status Step Functions untuk memulai eksekusi. AWS SAM CLI akan menunggu alur kerja mesin negara selesai dan mengembalikan output dari langkah terakhir dalam eksekusi. Berikut ini adalah contohnya:

```
$ sam remote invoke HelloWorldStateMachine --stack-name state-machine-example --  
event '{"is_developer": true}'
```

```
Invoking Step Function HelloWorldStateMachine
```

```
"Hello Developer World"%
```

Untuk memanggil mesin negara

1. Berikan nilai ID sumber daya sebagai argumen untuk mesin status Step Functions. Untuk informasi tentang ID sumber daya yang valid, lihat [ID Sumber Daya](#).
2. Berikan acara untuk dikirim ke mesin negara Anda. Anda dapat memberikan acara di baris perintah menggunakan `--event` opsi, atau dari file yang menggunakan `--event-file`. Jika Anda tidak memberikan acara, AWS SAM CLI mengirimkan acara kosong.

Menggunakan opsi perintah panggilan jarak jauh sam

Bagian ini mencakup beberapa opsi utama yang dapat Anda gunakan dengan `sam remote invoke` perintah. Untuk daftar lengkap opsi, lihat [sam remote invoke](#).

Lulus acara ke sumber daya Anda

Gunakan opsi berikut untuk meneruskan acara ke sumber daya Anda di cloud:

- `--event`— Lewati acara di baris perintah.
- `--event-file`— Lulus acara dari file.

Contoh Lambda

Gunakan `--event` untuk meneruskan acara di baris perintah sebagai nilai string:

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --event '{"message": "hello!"}'
```

```
Invoking Lambda Function HelloWorldFunction
```

```
START RequestId: b992292d-1fac-4aa2-922a-c9dc5c6fceab Version: $LATEST
END RequestId: b992292d-1fac-4aa2-922a-c9dc5c6fceab
REPORT RequestId: b992292d-1fac-4aa2-922a-c9dc5c6fceab Duration: 16.41 ms Billed
Duration: 17 ms Memory Size: 128 MB Max Memory Used: 67 MB Init Duration: 185.96
ms
{"statusCode":200,"body":{"\"message\": \"hello!\"}}%
```

Gunakan `--event-file` untuk meneruskan acara dari file dan memberikan path ke file:

```
$ cat event.json
```

```
{"message": "hello from file"}%
```

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --event-file event.json
```

Invoking Lambda Function HelloWorldFunction

```
START RequestId: 3bc71f7d-153a-4b1e-8c9a-901d91b1bec9 Version: $LATEST
```

```
END RequestId: 3bc71f7d-153a-4b1e-8c9a-901d91b1bec9
```

```
REPORT RequestId: 3bc71f7d-153a-4b1e-8c9a-901d91b1bec9 Duration: 21.15 ms Billed
```

```
Duration: 22 ms Memory Size: 128 MB Max Memory Used: 67 MB
```

```
{"statusCode":200,"body":{"\"message\": \"hello from file\"}}%
```

Lulus acara menggunakan **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\"}}%
```

Konfigurasi output AWS SAMCLI respons

Saat Anda memanggil sumber daya yang didukung `sam remote invoke`, akan AWS SAM CLI mengembalikan respons yang berisi berikut ini:

- Minta metadata — Metadata yang terkait dengan permintaan. Ini termasuk ID permintaan dan waktu mulai permintaan.
- Respons sumber daya — Respons dari sumber daya Anda setelah dipanggil di cloud.

Anda dapat menggunakan `--output` opsi untuk mengkonfigurasi respons AWS SAM CLI output.

Nilai opsi berikut tersedia:

- `json`— Metadata dan respon sumber daya dikembalikan dalam struktur. JSON Respons berisi SDK output penuh.
- `text`— Metadata dikembalikan dalam struktur teks. Respons sumber daya dikembalikan dalam format output sumber daya.

Berikut ini adalah contoh json output:

```
$ sam remote invoke --stack-name sam-app --output json
```

```
Invoking Lambda Function HelloWorldFunction
```

```
{
  "ResponseMetadata": {
    "RequestId": "3bdf9a30-776d-4a90-94a6-4cccc0fc7b41",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "date": "Mon, 19 Jun 2023 17:15:46 GMT",
      "content-type": "application/json",
      "content-length": "57",
      "connection": "keep-alive",
      "x-amzn-requestid": "3bdf9a30-776d-4a90-94a6-4cccc0fc7b41",
      "x-amzn-remapped-content-length": "0",
      "x-amz-executed-version": "$LATEST",
      "x-amz-log-result":
"U1RBULQgUmVxdWVzdElk0iAzYmRmOWEzMC03NzZkLTRhOTAtOTRhNi00Y2NjYzBmYzdiNDEgVmVyc2lvbjogJExBVEVTV
      "x-amzn-trace-id":
"root=1-64908d42-17dab270273fcc6b527dd6b8;sampld=0;lineage=2301f8dc:0"
    },
    "RetryAttempts": 0
  },
  "StatusCode": 200,
  "LogResult":
"U1RBULQgUmVxdWVzdElk0iAzYmRmOWEzMC03NzZkLTRhOTAtOTRhNi00Y2NjYzBmYzdiNDEgVmVyc2lvbjogJExBVEVTV
  "ExecutedVersion": "$LATEST",
  "Payload": "{\"statusCode\":200,\"body\": \"{\\\"message\\\":\\\"hello world\\\"}\"}"
}%
```

Saat Anda menentukan json output, seluruh respons dikembalikan ke stdout. Berikut ini adalah contohnya:

```
$ 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": "$LATEST",
      "x-amz-log-result":
"U1RBULQgUmVxdWVzdElk0iBkMzBkMjgwZi04MTg4LTQzNzItYmM5NC1jZTBmMTYwM2I2YmIgVmVyc2lvbjoyJExBVEVTV
      "x-amzn-trace-id":
"root=1-649091fc-771473c7778689627a6122b7;sampld=0;lineage=2301f8dc:0"
    },
    "RetryAttempts": 0
  },
  "StatusCode": 200,
  "LogResult":
"U1RBULQgUmVxdWVzdElk0iBkMzBkMjgwZi04MTg4LTQzNzItYmM5NC1jZTBmMTYwM2I2YmIgVmVyc2lvbjoyJExBVEVTV
  "ExecutedVersion": "$LATEST",
  "Payload": "{\"statusCode\":200,\"body\": \"{\\\"message\\\":\\\"hello world\\\"}\"}"
}%
```

Berikut ini adalah contoh text output:

```
$ sam remote invoke --stack-name sam-app --output text
```

```
Invoking Lambda Function HelloWorldFunction
```

```
START RequestId: 4dbacc43-1ec6-47c2-982b-9dc4620144d6 Version: $LATEST
END RequestId: 4dbacc43-1ec6-47c2-982b-9dc4620144d6
REPORT RequestId: 4dbacc43-1ec6-47c2-982b-9dc4620144d6 Duration: 9.13 ms Billed
Duration: 10 ms Memory Size: 128 MB Max Memory Used: 67 MB Init Duration: 165.50
ms
{"statusCode":200,"body":{"\"message\": \"hello world\"}}%
```

Saat Anda menentukan text output, output runtime fungsi Lambda (misalnya, log) dikembalikan ke. `stderr` Payload fungsi Lambda dikembalikan ke. `stdout` Berikut ini adalah contohnya:

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

Sesuaikan Boto3 parameter

Untuk `remote invoke`, AWS SAM CLI menggunakan AWS SDK for Python (Boto3) untuk berinteraksi dengan sumber daya Anda di cloud. Anda dapat menggunakan `--parameter` opsi untuk menyesuaikan Boto3 parameter. Untuk daftar parameter yang didukung yang dapat Anda sesuaikan, lihat [--parameter](#).

Contoh-contoh

Memanggil fungsi Lambda untuk memvalidasi nilai parameter dan memverifikasi izin:

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --  
parameter InvocationType="DryRun"
```

Gunakan `--parameter` opsi beberapa kali dalam satu perintah untuk menyediakan beberapa parameter:

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --  
parameter InvocationType="Event" --parameter LogType="None"
```

Pilihan lain

Untuk daftar lengkap `sam remote invoke` opsi, lihat [sam remote invoke](#).

Konfigurasi file konfigurasi proyek Anda

Untuk mengkonfigurasi `sam remote invoke` dalam file konfigurasi Anda, gunakan `remote_invoke` dalam tabel Anda. Berikut ini adalah contoh `samconfig.toml` file yang mengkonfigurasi nilai default untuk `sam remote invoke` perintah.

```
...  
version =0.1  
  
[default]  
...  
[default.remote_invoke.parameters]  
stack_name = "cloud-app"  
event = '{"message": "Hello!"}'
```

Contoh-contoh

Untuk contoh dasar penggunaannya `remote invoke`, lihat [Menguji AWS Lambda fungsi dengan AWS SAM remote](#) di AWSCompute Blog.

Contoh Kinesis Data Streams

Contoh dasar

Kirim catatan data ke aplikasi Kinesis Data Streams dari file. Aplikasi Kinesis Data Streams diidentifikasi dengan menyediakan ARN untuk ID sumber daya:

```
$ sam remote invoke arn:aws:kinesis:us-west-2:01234567890:stream/kinesis-example-KinesisStream-BgnLcAey4xUQ --event-file event.json
```

Kirim acara yang disediakan di baris perintah ke aplikasi Kinesis Data Streams:

```
$ 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"  
}%
```

Dapatkan ID fisik aplikasi Kinesis Data Streams. Kemudian, berikan acara di baris perintah:

```
$ 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"
}%
```

Berikan string JSON di baris perintah sebagai acara:

```
$ sam remote invoke KinesisStream --stack-name kinesis-example --event '{"method": "GET", "body": ""}'
```

```
Putting record to Kinesis data stream KinesisStream
```

```
{
  "ShardId": "shardId-000000000000",
  "SequenceNumber": "49646251411914806775980904492868617924990209230536441858"
}%
```

Kirim acara kosong ke aplikasi Kinesis Data Streams:

```
$ sam remote invoke KinesisStream --stack-name kinesis-example
```

```
Putting record to Kinesis data stream KinesisStream
```

```
{
  "ShardId": "shardId-000000000000",
  "SequenceNumber": "49646251411914806775980904866469008589597168190416224258"
}%
```

Kembalikan AWS SAM CLI respons dalam format JSON:

```
$ 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
}
}%

```

Kembalikan output JSON ke 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": "4964625141191480677598090639777867595039988349006774274",
  "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
  }
}

```

```
}%
```

Contoh Lambda

Contoh dasar

Memanggil fungsi Lambda dengan menyediakan ARN sebagai ID sumber daya:

```
$ sam remote invoke arn:aws:lambda:us-west-2:012345678910:function:sam-app-HelloWorldFunction-ohRFEn2RuAvp
```

Memanggil fungsi Lambda dengan memberikan ID logis sebagai ID sumber daya:

Anda juga harus memberikan nama AWS CloudFormation tumpukan menggunakan `--stack-name` opsi. Berikut ini adalah contohnya:

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

Jika aplikasi Anda berisi satu fungsi Lambda, Anda tidak perlu menentukan ID logisnya. Anda hanya dapat memberikan `--stack-name` opsi. Berikut ini adalah contohnya:

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

Memanggil fungsi Lambda dengan memberikan ID fisik sebagai ID sumber daya:

ID fisik akan dibuat saat Anda menerapkan menggunakan AWS CloudFormation.

```
$ sam remote invoke sam-app-HelloWorldFunction-TZvxQRFNv0k4
```

Memanggil fungsi Lambda dari tumpukan anak:

Untuk contoh ini, aplikasi kami berisi struktur direktori berikut:

```
lambda-example
### childstack
#   ### function
#   #   ### __init__.py
#   #   ### app.py
#   #   ### requirements.txt
#   ### template.yaml
### events
#   ### event.json
```

```
### samconfig.toml
### template.yaml
```

Untuk memanggil fungsi Lambda `childstack` kami, kami menjalankan yang berikut:

```
$ 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\": {}}"}%
```

Memanggil fungsi Lambda yang dikonfigurasi untuk mengalirkan respons

Dalam contoh ini, kita menggunakan AWS SAM CLI untuk menginisialisasi aplikasi tanpa server baru yang berisi fungsi Lambda yang dikonfigurasi untuk mengalirkan responsnya. Kami menyebarkan aplikasi kami ke AWS Cloud dan menggunakan `sam remote invoke` untuk berinteraksi dengan fungsi kami di cloud.

Kita mulai dengan menjalankan `sam init` perintah untuk membuat aplikasi tanpa server baru. Kami memilih template mulai cepat Lambda Response Streaming dan memberi nama aplikasi kami `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
```

AWS SAMCLIni menciptakan proyek kami dengan struktur berikut:

```
lambda-streaming-nodejs-app
### README.md
### __tests__
#   ### unit
#       ### index.test.js
### package.json
### samconfig.toml
### src
#   ### index.js
### template.yaml
```

Berikut ini adalah contoh kode fungsi Lambda kami:

```
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(`

${loremIpsum1}

`);
    await new Promise(r => setTimeout(r, 1000));

    responseStream.write("<p>DONE!</p>");
    responseStream.write("</html>");
    responseStream.end();
  }
);

```

Berikut ini adalah contoh `template.yaml` file kami. Streaming respons untuk fungsi Lambda kami dikonfigurasi menggunakan properti `FunctionUrlConfig`

```

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

Biasanya, Anda dapat menggunakan `sam build` dan `sam deploy --guided` membangun dan menyebarkan aplikasi produksi. Dalam contoh ini, kita akan mengasumsikan lingkungan pengembangan dan menggunakan `sam sync` perintah untuk membangun dan menyebarkan aplikasi kita.

 **Note**

`sam sync`Perintah ini direkomendasikan untuk lingkungan pengembangan. Untuk mempelajari selengkapnya, lihat [Menggunakan sam sync](#).

Sebelum menjalankan `sam sync`, kami memverifikasi bahwa proyek kami dikonfigurasi dengan benar di `samconfig.toml` file kami. Yang terpenting, kami memverifikasi nilai untuk `stack_name` dan `watch`. Dengan nilai-nilai ini ditentukan dalam file konfigurasi kami, kami tidak harus menyediakannya di baris perintah.

```
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"
```

Selanjutnya, kita jalankan `sam sync` untuk membangun dan menyebarkan aplikasi kita. Karena `--watch` opsi dikonfigurasi dalam file konfigurasi kami, AWS SAM CLI akan membangun aplikasi kami, menyebarkan aplikasi kami, dan melihat perubahan.

```
$ 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-
samclisourcebucket-1a4x26zbcdkqr
Capabilities        : ["CAPABILITY_NAMED_IAM",
"CAPABILITY_AUTO_EXPAND"]
Parameter overrides : {}
Signing Profiles    : null
```

Initiating deployment

```
=====
```

2023-06-20 12:11:16 - Waiting for stack create/update to complete

CloudFormation events from stack operations (refresh every 0.5 seconds)

```
-----
```

ResourceStatus	ResourceType	LogicalResourceId	ResourceStatusReason
CREATE_IN_PROGRESS	AWS::CloudFormation::St	lambda-streaming-	
Transformation	ack	nodejs-app	
succeeded			
CREATE_IN_PROGRESS	AWS::IAM::Role	StreamingFunctionRole	-
CREATE_IN_PROGRESS	AWS::CloudFormation::St	AwsSamAutoDependencyLay	-

```
-----
```

	ack	erNestedStack	
CREATE_IN_PROGRESS creation	AWS::IAM::Role	StreamingFunctionRole	Resource
Initiated			
CREATE_IN_PROGRESS creation	AWS::CloudFormation::St	AwsSamAutoDependencyLay	Resource
Initiated			
CREATE_COMPLETE	AWS::IAM::Role	StreamingFunctionRole	-
CREATE_COMPLETE	AWS::CloudFormation::St	AwsSamAutoDependencyLay	-
	ack	erNestedStack	
CREATE_IN_PROGRESS	AWS::Lambda::Function	StreamingFunction	-
CREATE_IN_PROGRESS creation	AWS::Lambda::Function	StreamingFunction	Resource
Initiated			
CREATE_COMPLETE	AWS::Lambda::Function	StreamingFunction	-
CREATE_IN_PROGRESS	AWS::Lambda::Url	StreamingFunctionUrl	-
CREATE_IN_PROGRESS creation	AWS::Lambda::Url	StreamingFunctionUrl	Resource
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://wxgkcc2dyntgtrwhf2dgdcvylu0rnnof.lambda-url.us-west-2.on.aws/

Stack creation succeeded. Sync infra completed.

Infra sync completed.

Sekarang fungsi kita diterapkan ke cloud, kita dapat menggunakan `sam remote invoke` untuk berinteraksi dengan fungsi kita. AWS SAM CLI secara otomatis mendeteksi bahwa fungsi kami dikonfigurasi untuk streaming respons dan segera mulai mengeluarkan respons streaming dari fungsi kami secara real time.

```
$ sam remote invoke StreamingFunction
```

```
Invoking Lambda Function StreamingFunction
```

```
{"statusCode":200,"headers":{"Content-Type":"text/html","X-Custom-Header":"Example-Custom-Header"}}<html><p>First write!</p><h1>Streaming h1</h1><h2>Streaming h2</h2><h3>Streaming h3</h3><p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque vitae mi tincidunt tellus ultricies dignissim id et diam. Morbi pharetra eu nisi et finibus. Vivamus diam nulla, vulputate et nisl cursus, pellentesque vehicula libero. Cras imperdiet lorem ante, non posuere dolor sollicitudin a. Vestibulum ipsum lacus, blandit nec augue id, lobortis dictum urna. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia curae; Morbi auctor orci eget tellus aliquam, non maximus massa porta. In diam ante, pulvinar aliquam nisl non, elementum
```

```
hendrerit sapien. Vestibulum massa nunc, mattis non congue vitae, placerat in quam.
Nam vulputate lectus metus, et dignissim erat varius a.</p><p>DONE!</p></html>START
RequestId: 1e4cdf04-60de-4769-b3a2-c1481982deb4 Version: $LATEST
END RequestId: 1e4cdf04-60de-4769-b3a2-c1481982deb4
REPORT RequestId: 1e4cdf04-60de-4769-b3a2-c1481982deb4 Duration: 4088.66 ms
Billed Duration: 4089 ms Memory Size: 128 MB Max Memory Used: 68 MB Init
Duration: 168.45 ms
```

Ketika kita memodifikasi kode fungsi kita, AWS SAM CLI langsung mendeteksi dan segera menyebarkan perubahan kita. Berikut adalah contoh AWS SAM CLI output setelah perubahan dilakukan pada kode fungsi kami:

```
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.
```

Kita sekarang dapat menggunakan `sam remote invoke` lagi untuk berinteraksi dengan fungsi kita di cloud dan menguji perubahan kita.

Contoh SQS

Contoh dasar

Memanggil antrian Amazon SQS dengan memberikan ARN sebagai ID sumber daya:

```
$ 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  
  }  
}%
```

Contoh Step Functions

Contoh dasar

Memanggil mesin status dengan memberikan ID fisiknya sebagai ID sumber daya:

Pertama, kami gunakan `sam list resources` untuk mendapatkan ID fisik kami:

```
$ 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"  
  }  
]
```

```
]
```

Selanjutnya, kami memanggil mesin negara kami menggunakan ID fisik sebagai ID sumber daya. Kami melewati acara di baris perintah dengan `--event` opsi:

```
$ 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"%
```

Memanggil mesin status dengan melewatkan acara kosong:

```
$ sam remote invoke HelloWorldStateMachine --stack-name state-machine-example
```

```
Invoking Step Function HelloWorldStateMachine  
"Hello World"%
```

Tautan terkait

Untuk dokumentasi yang terkait dengan `sam remote invoke` dan menggunakan AWS SAMCLI, lihat berikut ini:

- [sam remote invoke](#)
- [AWS SAMCLIpemecahan masalah](#)

Menggunakan sam remote test-event

Gunakan AWS Serverless Application Model perintah Command Line Interface (AWS SAMCLI) `sam remote test-event` untuk mengakses dan mengelola peristiwa pengujian yang dapat dibagikan untuk AWS Lambda fungsi Anda.

Untuk mempelajari lebih lanjut tentang peristiwa pengujian yang dapat dibagikan, lihat [Acara pengujian yang dapat dibagikan](#) di Panduan AWS LambdaPengembang.

Topik

- [Siapkan AWS SAMCLI untuk digunakan sam remote test-event](#)
- [Menggunakan sam remote test-event perintah](#)
- [Menggunakan acara uji yang dapat dibagikan](#)
- [Mengelola acara uji yang dapat dibagikan](#)

Prasyarat

Untuk menggunakan `sam remote test-event`, instal AWS SAM CLI dengan menyelesaikan yang berikut ini:

- [Prasyarat AWS SAM.](#)
- [Instalasi AWS SAMCLI.](#)

Jika Anda sudah AWS SAM CLI menginstal, kami sarankan untuk memutakhirkan ke versi terbaru dari AWS SAM CLI versi. Untuk mempelajari selengkapnya, lihat [Memutakhirkan AWS SAMCLI](#).

Sebelum menggunakan `sam remote test-event`, kami merekomendasikan pemahaman dasar tentang hal-hal berikut:

- [Mengkonfigurasi AWS SAMCLI.](#)
- [Menggunakan sam init.](#)
- [Menggunakan sam build.](#)
- [Menggunakan sam deploy.](#)
- [Menggunakan sam sync.](#)

Siapkan AWS SAMCLI untuk digunakan sam remote test-event

Selesaikan langkah-langkah pengaturan berikut untuk menggunakan AWS SAM CLI `sam remote test-event` perintah:

1. Konfigurasi AWS SAM CLI untuk menggunakan Akun AWS - Acara pengujian yang dapat dibagikan untuk Lambda dapat diakses dan dikelola oleh pengguna dalam hal yang sama. Akun AWS Untuk mengkonfigurasi AWS SAM CLI untuk menggunakan Anda Akun AWS, lihat [Mengkonfigurasi AWS SAMCLI](#).
2. Konfigurasi izin untuk acara pengujian yang dapat dibagikan — Untuk mengakses dan mengelola peristiwa pengujian yang dapat dibagikan, Anda harus memiliki izin yang tepat. Untuk

mempelajari selengkapnya, lihat [Acara pengujian yang dapat dibagikan](#) di Panduan AWS Lambda Pengembang.

Menggunakan sam remote test-event perintah

AWS SAMCLIsam remote test-eventPerintah ini menyediakan subperintah berikut yang dapat Anda gunakan untuk mengakses dan mengelola peristiwa pengujian yang dapat dibagikan:

- `delete`— Hapus acara pengujian yang dapat dibagikan dari registri EventBridge skema Amazon.
- `get`— Dapatkan acara uji yang dapat dibagikan dari registri EventBridge skema.
- `list`— Buat daftar peristiwa pengujian yang dapat dibagikan yang ada untuk fungsi dari registri EventBridge skema.
- `put`— Simpan acara dari file lokal ke registri EventBridge skema.

Untuk membuat daftar subperintah ini menggunakan AWS SAMCLI, jalankan yang berikut ini:

```
$ sam remote test-event --help
```

Menghapus acara pengujian yang dapat dibagikan

Anda dapat menghapus peristiwa pengujian yang dapat dibagikan dengan menggunakan `delete` subperintah bersama dengan yang berikut:

- Berikan nama acara pengujian yang dapat dibagikan untuk dihapus.
- Berikan ID yang dapat diterima dari fungsi Lambda yang terkait dengan acara tersebut.
- Jika Anda memberikan ID logis fungsi Lambda, Anda juga harus memberikan nama AWS CloudFormation tumpukan yang terkait dengan fungsi Lambda.

Berikut ini adalah contoh:

```
$ sam remote test-event delete HelloWorldFunction --stack-name sam-app --name demo-event
```

Untuk daftar opsi yang akan digunakan dengan `delete` subperintah, lihat [sam remote test-event delete](#). Anda juga dapat menjalankan yang berikut ini dari AWS SAMCLI:

```
$ sam remote test-event delete --help
```

Mendapatkan acara uji yang dapat dibagikan

Anda bisa mendapatkan acara pengujian yang dapat dibagikan dari registri EventBridge skema dengan menggunakan `get` subperintah bersama dengan yang berikut:

- Berikan nama acara uji yang dapat dibagikan untuk didapatkan.
- Berikan ID yang dapat diterima dari fungsi Lambda yang terkait dengan acara tersebut.
- Jika Anda memberikan ID logis fungsi Lambda, Anda juga harus memberikan nama AWS CloudFormation tumpukan yang terkait dengan fungsi Lambda.

Berikut ini adalah contoh yang mendapatkan acara pengujian yang dapat dibagikan bernama `demo-event` yang terkait dengan fungsi `HelloWorldFunction` Lambda dari `sam-app` tumpukan. Perintah ini akan mencetak acara ke konsol Anda.

```
$ sam remote test-event get HelloWorldFunction --stack-name sam-app --name demo-event
```

Untuk mendapatkan acara pengujian yang dapat dibagikan dan menyimpannya ke mesin lokal Anda, gunakan `--output-file` opsi dan berikan jalur dan nama file. Berikut ini adalah contoh yang menyimpan `demo-event` seperti `demo-event.json` pada direktori kerja saat ini:

```
$ sam remote test-event get HelloWorldFunction --stack-name sam-app --name demo-event --output-file demo-event.json
```

Untuk daftar opsi yang akan digunakan dengan `get` subperintah, lihat [sam remote test-event get](#). Anda juga dapat menjalankan yang berikut ini dari AWS SAMCLI:

```
$ sam remote test-event get --help
```

Daftar acara uji yang dapat dibagikan

Anda dapat mencantumkan semua peristiwa pengujian yang dapat dibagikan untuk fungsi Lambda tertentu dari registri skema. Gunakan `list` subperintah bersama dengan yang berikut ini:

- Berikan ID yang dapat diterima dari fungsi Lambda yang terkait dengan peristiwa.

- Jika Anda memberikan ID logis fungsi Lambda, Anda juga harus memberikan nama AWS CloudFormation tumpukan yang terkait dengan fungsi Lambda.

Berikut ini adalah contoh yang memperoleh daftar semua peristiwa pengujian yang dapat dibagikan yang terkait dengan fungsi HelloWorldFunction Lambda dari tumpukan: sam-app

```
$ sam remote test-event list HelloWorldFunction --stack-name sam-app
```

Untuk daftar opsi yang akan digunakan dengan list subperintah, lihat [sam remote test-event list](#). Anda juga dapat menjalankan yang berikut ini dari AWS SAMCLI:

```
$ sam remote test-event list --help
```

Menyimpan acara uji yang dapat dibagikan

Anda dapat menyimpan peristiwa pengujian yang dapat dibagikan ke registri EventBridge skema. Gunakan put subperintah bersama dengan yang berikut ini:

- Berikan ID yang dapat diterima dari fungsi Lambda yang terkait dengan peristiwa pengujian yang dapat dibagikan.
- Berikan nama untuk acara pengujian yang dapat dibagikan.
- Berikan jalur file dan nama ke acara lokal untuk diunggah.

Berikut ini adalah contoh yang menyimpan demo-event.json acara lokal sebagai demo-event dan mengaitkannya dengan fungsi HelloWorldFunction Lambda dari sam-app tumpukan:

```
$ sam remote test-event put HelloWorldFunction --stack-name sam-app --name demo-event --file demo-event.json
```

Jika peristiwa pengujian yang dapat dibagikan dengan nama yang sama ada di registri EventBridge skema, AWS SAM CLI maka tidak akan menimpa. Untuk menimpa, tambahkan --force opsi ke perintah Anda.

Untuk daftar opsi yang akan digunakan dengan put subperintah, lihat [sam remote test-event put](#). Anda juga dapat menjalankan yang berikut ini dari AWS SAMCLI:

```
$ sam remote test-event put --help
```

Menggunakan acara uji yang dapat dibagikan

Gunakan peristiwa pengujian yang dapat dibagikan untuk menguji fungsi Lambda Anda di dengan AWS Cloud `sam remote invoke` perintah. Untuk mempelajari selengkapnya, lihat [Lulus peristiwa pengujian yang dapat dibagikan ke fungsi Lambda di cloud](#).

Mengelola acara uji yang dapat dibagikan

Topik ini berisi contoh bagaimana Anda dapat mengelola dan menggunakan peristiwa pengujian yang dapat dibagikan.

Dapatkan acara pengujian yang dapat dibagikan, modifikasi, dan gunakan

Anda bisa mendapatkan acara pengujian yang dapat dibagikan dari registri EventBridge skema, memodifikasinya secara lokal, dan menggunakan acara pengujian lokal dengan fungsi Lambda Anda di. AWS Cloud Berikut ini adalah contoh:

1. Mengambil peristiwa pengujian yang dapat dibagikan - Gunakan `sam remote test-event get` subperintah untuk mengambil peristiwa pengujian yang dapat dibagikan untuk fungsi Lambda tertentu dan menyimpannya secara lokal:

```
$ sam remote test-event get HelloWorldFunction --stack-name sam-app --name demo-event --output-file demo-event.json
```

2. Ubah acara pengujian yang dapat dibagikan - Gunakan editor teks pilihan Anda untuk memodifikasi acara pengujian yang dapat dibagikan.
3. Gunakan acara pengujian yang dapat dibagikan - Gunakan `sam remote invoke` perintah dan berikan jalur file dan nama acara dengan `--event-file`:

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --event-file demo-event.json
```

Dapatkan acara pengujian yang dapat dibagikan, modifikasi, unggah, dan gunakan

Anda bisa mendapatkan acara pengujian yang dapat dibagikan dari registri EventBridge skema, memodifikasinya secara lokal, dan mengunggahnya. Kemudian, Anda dapat lulus acara pengujian yang dapat dibagikan langsung ke fungsi Lambda Anda di. AWS Cloud Berikut ini adalah contoh:

1. Mengambil peristiwa pengujian yang dapat dibagikan - Gunakan **sam remote test-event get** subperintah untuk mengambil peristiwa pengujian yang dapat dibagikan untuk fungsi Lambda tertentu dan menyimpannya secara lokal:

```
$ sam remote test-event get HelloWorldFunction --stack-name sam-app --name demo-event --output-file demo-event.json
```

2. Ubah acara pengujian yang dapat dibagikan - Gunakan editor teks pilihan Anda untuk memodifikasi acara pengujian yang dapat dibagikan.
3. Unggah peristiwa pengujian yang dapat dibagikan — Gunakan **sam remote test-event put** subperintah untuk mengunggah dan menyimpan peristiwa pengujian yang dapat dibagikan ke registri skema. EventBridge Dalam contoh ini, kami menggunakan **--force** opsi untuk menimpa versi lama dari pengujian kami yang dapat dibagikan:

```
$ sam remote test-event put HelloWorldFunction --stack-name sam-app --name demo-event --file demo-event.json --force
```

4. Lulus peristiwa pengujian yang dapat dibagikan ke fungsi Lambda Anda — Gunakan **sam remote invoke** perintah untuk meneruskan peristiwa pengujian yang dapat dibagikan langsung ke fungsi Lambda Anda di: AWS Cloud

```
$ sam remote invoke HelloWorldFunction --stack-name sam-app --test-event-name demo-event
```

Menggunakan sam sync

AWS Serverless Application Model Perintah Command Line Interface (AWS SAMCLI) **sam sync** menyediakan opsi untuk menyinkronkan perubahan aplikasi lokal dengan cepat ke file AWS Cloud. Gunakan **sam sync** saat mengembangkan aplikasi Anda untuk:

1. Secara otomatis mendeteksi dan menyinkronkan perubahan lokal ke file AWS Cloud.
2. Sesuaikan perubahan lokal apa yang disinkronkan ke file. AWS Cloud
3. Siapkan aplikasi Anda di cloud untuk pengujian dan validasi.

Dengan **sam sync**, Anda dapat membuat alur kerja pengembangan cepat yang mempersingkat waktu yang diperlukan untuk menyinkronkan perubahan lokal Anda ke cloud untuk pengujian dan validasi.

Note

`aws sam sync` Perintah ini direkomendasikan untuk lingkungan pengembangan. Untuk lingkungan produksi, sebaiknya gunakan `aws sam deploy` atau konfigurasi pipeline continuous integration and delivery (CI/CD). Untuk mempelajari selengkapnya, lihat [Men-deploy aplikasi nirserver](#).

`aws sam sync` Perintah adalah bagian dari AWS SAM Accelerate. AWS SAM Accelerate menyediakan alat yang dapat Anda gunakan untuk mempercepat pengalaman mengembangkan dan menguji aplikasi tanpa server di AWS Cloud.

Topik

- [Secara otomatis mendeteksi dan menyinkronkan perubahan lokal ke AWS Cloud](#)
- [Kustomisasi perubahan lokal apa yang disinkronkan ke AWS Cloud](#)
- [Siapkan aplikasi Anda di cloud untuk pengujian dan validasi](#)
- [Opsi untuk perintah sinkronisasi `aws sam sync`](#)
- [Memecahkan masalah](#)
- [Contoh-contoh](#)
- [Pelajari selengkapnya](#)

Secara otomatis mendeteksi dan menyinkronkan perubahan lokal ke AWS Cloud

Jalankan `aws sam sync` dengan `--watch` opsi untuk mulai menyinkronkan aplikasi Anda ke file. AWS Cloud Ini melakukan hal berikut:

1. Membangun aplikasi Anda — Proses ini mirip dengan menggunakan `aws sam build` perintah.
2. Menerapkan aplikasi Anda — AWS SAM CLI Menyebarkan aplikasi Anda untuk AWS CloudFormation menggunakan pengaturan default Anda. Nilai default berikut digunakan:
 - a. AWS kredensi dan pengaturan konfigurasi umum ditemukan di folder `.aws` pengguna Anda.
 - b. Pengaturan penerapan aplikasi ditemukan di `samconfig.toml` file aplikasi Anda.

Jika nilai default tidak dapat ditemukan, AWS SAM CLI akan memberi tahu Anda dan keluar dari proses sinkronisasi.

3. Perhatikan perubahan lokal — AWS SAM CLI Tetap berjalan dan perhatikan perubahan lokal pada aplikasi Anda. Inilah yang disediakan `--watch` opsi.

Opsi ini dapat dihidupkan secara default. Untuk nilai default, lihat `samconfig.toml` file aplikasi Anda. Berikut ini adalah file contoh:

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

4. Sinkronkan perubahan lokal ke AWS Cloud — Saat Anda membuat perubahan lokal, AWS SAM CLI mendeteksi dan menyinkronkan perubahan tersebut AWS Cloud melalui metode tercepat yang tersedia. Tergantung pada jenis perubahan, berikut ini dapat terjadi:
 - a. Jika sumber daya Anda yang diperbarui mendukung API AWS layanan, AWS SAM CLI akan menggunakannya untuk menyebarkan perubahan Anda. Ini menghasilkan sinkronisasi cepat untuk memperbarui sumber daya Anda diAWS Cloud.
 - b. Jika sumber daya yang diperbarui tidak mendukung API AWS layanan, AWS SAM CLI maka akan melakukan AWS CloudFormation penerapan. Ini memperbarui seluruh aplikasi Anda diAWS Cloud. Meskipun tidak secepat itu, itu mencegah Anda dari keharusan memulai penerapan secara manual.

Karena `sam sync` perintah secara otomatis memperbarui aplikasi Anda diAWS Cloud, disarankan untuk lingkungan pengembangan saja. Ketika Anda berlaris `sync`, Anda akan diminta untuk mengkonfirmasi:

```
**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
```

Kustomisasi perubahan lokal apa yang disinkronkan ke AWS Cloud

Berikan opsi untuk menyesuaikan perubahan lokal apa yang disinkronkan ke AWS Cloud. Ini dapat mempercepat waktu yang diperlukan untuk melihat perubahan lokal Anda di cloud untuk pengujian dan validasi.

Misalnya, berikan `--code` opsi untuk menyinkronkan hanya perubahan kode, seperti kode AWS Lambda fungsi. Selama pengembangan, jika Anda berfokus secara khusus pada kode Lambda, ini akan membuat perubahan Anda ke cloud dengan cepat untuk pengujian dan validasi. Berikut ini adalah contohnya:

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

Untuk menyinkronkan hanya perubahan kode untuk fungsi atau lapisan Lambda tertentu, gunakan opsi. `--resource-id` Berikut ini adalah contohnya:

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

Siapkan aplikasi Anda di cloud untuk pengujian dan validasi

`sam sync` Perintah secara otomatis menemukan metode tercepat yang tersedia untuk memperbarui aplikasi Anda di file. AWS Cloud Ini dapat mempercepat alur kerja pengembangan dan pengujian cloud Anda. Dengan memanfaatkan API AWS layanan, Anda dapat dengan cepat mengembangkan, menyinkronkan, dan menguji sumber daya yang didukung. Untuk contoh langsung, lihat [Modul 6 - AWS SAM Mempercepat](#) di Lokakarya Lengkap AWS SAM.

Opsi untuk perintah sinkronisasi sam

Berikut ini adalah beberapa opsi utama yang dapat Anda gunakan untuk memodifikasi `sam sync` perintah. Untuk daftar semua opsi, lihat [sam sync](#).

Lakukan penerapan satu kali AWS CloudFormation

Gunakan `--no-watch` opsi untuk mematikan sinkronisasi otomatis. Berikut ini adalah contohnya:

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

AWS SAM CLI akan melakukan AWS CloudFormation penyebaran satu kali. Perintah ini mengelompokkan tindakan yang dilakukan oleh `sam build` dan `sam deploy` perintah.

Lewati AWS CloudFormation penerapan awal

Anda dapat menyesuaikan apakah AWS CloudFormation penerapan diperlukan setiap kali `sam sync` dijalankan.

- Menyediakan `--no-skip-deploy-sync` untuk memerlukan AWS CloudFormation penerapan setiap kali `sync` dijalankan. Ini memastikan bahwa infrastruktur lokal Anda disinkronkan AWS CloudFormation, mencegah penyimpangan. Menggunakan opsi ini memang menambah waktu tambahan untuk alur kerja pengembangan dan pengujian Anda.
- Menyediakan `--skip-deploy-sync` untuk membuat AWS CloudFormation penyebaran opsional. Ini AWS SAM CLI akan membandingkan AWS SAM template lokal Anda dengan AWS CloudFormation template yang Anda gunakan dan akan melewati AWS CloudFormation penerapan awal jika perubahan tidak terdeteksi. Melewatkan AWS CloudFormation penerapan dapat menghemat waktu Anda saat menyinkronkan perubahan lokal ke file. AWS Cloud

Jika tidak ada perubahan yang terdeteksi, AWS SAM CLI akan tetap melakukan AWS CloudFormation penerapan dalam skenario berikut:

- Jika sudah 7 hari atau lebih sejak AWS CloudFormation penerapan terakhir Anda.
- Jika sejumlah besar perubahan kode fungsi Lambda terdeteksi, menjadikan AWS CloudFormation penerapan metode tercepat untuk memperbarui aplikasi Anda.

Berikut ini adalah contohnya:

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

Sinkronkan sumber daya dari tumpukan bersarang

Untuk menyinkronkan sumber daya dari tumpukan bersarang

1. Berikan tumpukan root menggunakan `--stack-name`.
2. Identifikasi sumber daya dalam tumpukan bersarang menggunakan format berikut: *nestedStackId/resourceId*.
3. Berikan sumber daya di tumpukan bersarang menggunakan `--resource-id`.

Berikut ini adalah contohnya:

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

Untuk informasi selengkapnya tentang membuat aplikasi bersarang, lihat [Menggunakan aplikasi nest](#).

Tentukan AWS CloudFormation tumpukan tertentu untuk diperbarui

Untuk menentukan AWS CloudFormation tumpukan tertentu untuk diperbarui, berikan `--stack-name` opsi. Berikut ini adalah contohnya:

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

Mempercepat waktu pembuatan dengan membangun proyek Anda di folder sumber

Untuk runtime dan metode build yang didukung, Anda dapat menggunakan `--build-in-source` opsi untuk membangun proyek secara langsung di folder sumber. Secara default, AWS SAM CLI build dalam direktori sementara, yang melibatkan penyalinan kode sumber dan file proyek. Dengan `--build-in-source`, AWS SAM CLI build langsung di folder sumber Anda, yang mempercepat proses pembuatan dengan menghapus kebutuhan untuk menyalin file ke direktori sementara.

Untuk daftar runtime dan metode build yang didukung, lihat [--build-in-source](#).

Tentukan file dan folder yang tidak akan memulai sinkronisasi

Gunakan `--watch-exclude` opsi untuk menentukan file atau folder apa pun yang tidak akan memulai sinkronisasi saat diperbarui. Untuk informasi selengkapnya tentang metrik ini, lihat [--watch-exclude](#).

Berikut ini adalah contoh yang mengecualikan `package-lock.json` file yang terkait dengan `HelloWorldFunction` fungsi kami:

```
$ sam sync --watch --watch-exclude HelloWorldFunction=package-lock.json
```

Ketika perintah ini dijalankan, AWS SAM CLI akan memulai proses sinkronisasi. Ini termasuk yang berikut:

- Jalankan `sam build` untuk membangun fungsi Anda dan mempersiapkan aplikasi Anda untuk penerapan.
- Jalankan `sam deploy` untuk menyebarkan aplikasi Anda.
- Perhatikan perubahan pada aplikasi Anda.

Saat kami memodifikasi `package-lock.json` file, file AWS SAM CLI tidak akan memulai sinkronisasi. Ketika file lain diperbarui, AWS SAM CLI akan memulai sinkronisasi, yang akan menyertakan `package-lock.json` file.

Berikut ini adalah contoh menentukan fungsi Lambda dari tumpukan anak:

```
$ sam sync --watch --watch-exclude ChildStackA/MyFunction=database.sqlite3
```

Memecahkan masalah

Untuk memecahkan masalah AWS SAMCLI, lihat. [AWS SAMCLIpemecahan masalah](#)

Contoh-contoh

Menggunakan sam sync untuk memperbarui aplikasi Hello World

Dalam contoh ini, kita mulai dengan menginisialisasi contoh aplikasi Hello World. Untuk mempelajari lebih lanjut tentang aplikasi ini, lihat [Tutorial: Men-deploy aplikasi Hello World](#).

Menjalankan sam sync memulai proses build dan deployment.

```
$ 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_pt15g6_80000gr/T/tmpx_5t4u3f.
Execute the following command to deploy the packaged template
sam deploy --template-file /var/folders/45/5ct135bx3fn2551_pt15g6_80000gr/T/tmpx_5t4u3f
--stack-name <YOUR STACK NAME>
```

```
Deploying with following values
```

```
=====
```

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

```
Initiating deployment
```

```
=====
```

```
2023-03-17 11:17:19 - Waiting for stack create/update to complete
```

```
CloudFormation events from stack operations (refresh every 0.5 seconds)
```

```
-----
```

ResourceStatus	LogicalResourceId	ResourceType	ResourceStatusReason
CREATE_IN_PROGRESS	Transformation	AWS::CloudFormation::Stack	Transformation succeeded
CREATE_IN_PROGRESS	AwsSamAutoDependencyLayerNestedSt	AWS::CloudFormation::Stack	-
CREATE_IN_PROGRESS	HelloWorldFunctionRole	AWS::IAM::Role	-
CREATE_IN_PROGRESS	HelloWorldFunctionRole	AWS::IAM::Role	Resource creation Initiated
CREATE_IN_PROGRESS	AwsSamAutoDependencyLayerNestedSt	AWS::CloudFormation::Stack	Resource creation Initiated
CREATE_COMPLETE	HelloWorldFunctionRole	AWS::IAM::Role	-

```
-----
ack
ack
```

```

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.
```

Setelah penerapan selesai, kami memodifikasi HelloWorldFunction kode. The AWS SAM CLI mendeteksi perubahan ini dan menyinkronkan aplikasi kita ke file. AWS Cloud Karena AWS Lambda mendukung API AWS layanan, sinkronisasi cepat dilakukan.

```
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.
```

Selanjutnya, kita memodifikasi endpoint API kita di AWS SAM template aplikasi. Kami berubah /hello menjadi/helloworld.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  HelloWorldFunction:
    ...
```

Properties:

...

Events:

HelloWorld:

Type: Api

Properties:

Path: */helloworld*

Method: get

Karena sumber daya Amazon API Gateway tidak mendukung API AWS layanan, maka secara AWS SAM CLI otomatis melakukan AWS CloudFormation penerapan. Berikut ini adalah contoh output:

```
Queued infra sync. Waiting for in progress code syncs to complete...
Starting infra sync.
Manifest is not changed for (HelloWorldFunction), running incremental build
Building codeuri: /Users/.../Demo/sync/sam-app/hello_world runtime: python3.9 metadata:
  {} architecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:CopySource
```

Build Succeeded

```
Successfully packaged artifacts and wrote output template to file /var/
folders/45/5ct135bx3fn2551_pt15g6_80000gr/T/tmpuabo0jb9.
Execute the following command to deploy the packaged template
sam deploy --template-file /var/folders/45/5ct135bx3fn2551_pt15g6_80000gr/T/tmpuabo0jb9
--stack-name <YOUR STACK NAME>
```

Deploying with following values

=====

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

Initiating deployment

=====

2023-03-17 14:41:18 - Waiting for stack create/update to complete

CloudFormation events from stack operations (refresh every 0.5 seconds)

ResourceStatus	ResourceType	LogicalResourceId	ResourceStatusReason
UPDATE_IN_PROGRESS	AWS::CloudFormation::Stack	Transformation	succeeded
UPDATE_IN_PROGRESS	AWS::CloudFormation::Stack	AwsSamAutoDependencyLayerNestedSt	-
UPDATE_COMPLETE	AWS::CloudFormation::Stack	AwsSamAutoDependencyLayerNestedSt	-
UPDATE_IN_PROGRESS	AWS::ApiGateway::RestApi	ServerlessRestApi	-
UPDATE_COMPLETE	AWS::ApiGateway::RestApi	ServerlessRestApi	-
CREATE_IN_PROGRESS	AWS::ApiGateway::Deployment	ServerlessRestApiDeployment8cf30e	-
UPDATE_IN_PROGRESS	AWS::Lambda::Permission	HelloWorldFunctionHelloWorldPermi	Requested update requires the creation of a new physical resource; hence creating one.
UPDATE_IN_PROGRESS	AWS::Lambda::Permission	HelloWorldFunctionHelloWorldPermi	Resource creation Initiated
CREATE_IN_PROGRESS	AWS::ApiGateway::Deployment	ServerlessRestApiDeployment8cf30e	Resource creation Initiated
CREATE_COMPLETE	AWS::ApiGateway::Deployment	ServerlessRestApiDeployment8cf30e	-
UPDATE_IN_PROGRESS	AWS::ApiGateway::Stage	ServerlessRestApiProdStage	-
UPDATE_COMPLETE	AWS::ApiGateway::Stage	ServerlessRestApiProdStage	-
UPDATE_COMPLETE	AWS::Lambda::Permission	HelloWorldFunctionHelloWorldPermi	-


```

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-
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-2P1N6TPTQoco

```

Stack update succeeded. Sync infra completed.

Infra sync completed.

Pelajari selengkapnya

Untuk deskripsi semua sam sync opsi, lihat [sam sync](#).

AWS Serverless Application Model(AWS SAM) spesifikasi

Anda menggunakan spesifikasi AWS SAM untuk menentukan aplikasi nirserver Anda. Bagian ini menyediakan detail untuk bagian templat, tipe sumber daya, properti sumber daya, tipe data, atribut sumber daya, fungsi intrinsik, dan ekstensi API Gateway AWS SAM yang dapat Anda gunakan di templat AWS SAM.

Templat AWS SAM adalah ekstensi dari templat AWS CloudFormation, dengan beberapa komponen tambahan yang membuat templat tersebut lebih mudah untuk digunakan. Untuk referensi lengkap templat AWS CloudFormation, lihat [referensi templat AWS CloudFormation](#) di Panduan Pengguna AWS CloudFormation.

Topik

- [AWS SAM template anatomi](#)
- [Sumber daya dan referensi properti AWS SAM](#)
- [Atribut sumber daya](#)
- [Fungsi intrinsik](#)
- [Sumber daya AWS CloudFormation yang dibuat](#)
- [Ekstensi API Gateway](#)

AWS SAM template anatomi

File AWS SAM templat mengikuti format file AWS CloudFormation templat, yang dijelaskan dalam [Anatomi templat](#) di Panduan AWS CloudFormation Pengguna. Perbedaan utama antara file AWS SAM template dan file AWS CloudFormation template adalah sebagai berikut:

- Transformasi deklarasi. Deklarasi Transform: `AWS::Serverless-2016-10-31` diperlukan untuk file templat AWS SAM . Deklarasi ini mengidentifikasi file AWS CloudFormation template sebagai file AWS SAM template. Untuk informasi selengkapnya tentang transformasi, lihat [Transformasi](#) di Panduan Pengguna AWS CloudFormation .
- Bagian global. `Globals` Bagian ini unik untuk AWS SAM. Bagian ini mendefinisikan properti yang umum untuk semua fungsi nirserver dan API Anda. Semua sumber daya `AWS::Serverless::Function`, `AWS::Serverless::Api`, dan `AWS::Serverless::SimpleTable` mewarisi properti yang didefinisikan dalam bagian `Globals`. Untuk informasi lebih lanjut tentang bagian ini, lihat [Bagian global dari templat AWS SAM](#).

- Bagian sumber daya. Dalam AWS SAM template Resources bagian ini dapat berisi kombinasi AWS CloudFormation sumber daya dan sumber AWS SAM daya. Untuk informasi selengkapnya tentang AWS CloudFormation sumber daya, lihat [referensi jenisAWS sumber daya dan properti](#) di PanduanAWS CloudFormation Pengguna. Untuk informasi selengkapnya tentang AWS SAM sumber daya, lihat [Sumber daya dan referensi properti AWS SAM](#).
- Bagian parameter. Objek yang dinyatakan dalam bagian Parameters menyebabkan perintah `sam deploy --guided` untuk memberikan petunjuk tambahan kepada pengguna. Untuk contoh objek yang dideklarasikan dan petunjuk yang sesuai, lihat [sam deploy](#) di referensi AWS SAMCLI perintah.

Semua bagian lain dari file AWS SAM template sesuai dengan bagian file AWS CloudFormation template dengan nama yang sama.

YAML

Contoh berikut menunjukkan fragmen templat dalam format YMAL.

```
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

Bagian templat

AWS SAM template dapat mencakup beberapa bagian utama. Hanya bagian Transform dan Resources yang wajib diisi.

Anda dapat menyertakan bagian templat dengan urutan apa pun. Namun, jika menggunakan ekstensi bahasa, Anda harus menambahkan `AWS::LanguageExtensions` sebelum transformasi tanpa server (yaitu, sebelumnya `AWS::Serverless-2016-10-31`) seperti yang ditunjukkan pada contoh berikut:

Transform:

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

Saat Anda membangun template Anda, akan sangat membantu untuk menggunakan urutan logis yang ditampilkan dalam daftar berikut. Hal ini karena nilai-nilai dalam satu bagian mungkin merujuk ke nilai-nilai dari bagian sebelumnya.

[Transformasi \(wajib\)](#)

Untuk AWS SAM template, Anda harus menyertakan bagian ini dengan nilai `AWS::Serverless-2016-10-31`.

Transformasi tambahan termasuk hal opsional. Untuk informasi selengkapnya tentang transformasi, lihat [Transformasi](#) di Panduan Pengguna AWS CloudFormation .

[Globals \(opsional\)](#)

Properti yang umum untuk semua fungsi nirserver, API, dan tabel sederhana. Semua sumber daya `AWS::Serverless::Function`, `AWS::Serverless::Api`, dan `AWS::Serverless::SimpleTable` mewarisi properti yang didefinisikan dalam bagian `Globals`.

Bagian ini unik untuk AWS SAM. Tidak ada bagian yang sesuai di templat AWS CloudFormation .

[Deskripsi \(opsional\)](#)

Sebuah string teks yang menggambarkan templat.

Bagian ini sesuai langsung dengan `Description` bagian AWS CloudFormation templat.

Metadata (opsional)

Objek yang memberikan informasi tambahan tentang templat.

Bagian ini sesuai langsung dengan `Metadata` bagian AWS CloudFormation templat.

Parameter (opsional)

Nilai-nilai untuk diteruskan ke templat Anda pada saat waktu aktif (ketika Anda membuat atau memperbarui tumpukan). Anda dapat merujuk ke parameter dari bagian `Resources` dan `Outputs` dari templat.

Nilai-nilai yang diteruskan dalam menggunakan parameter `--parameter-overrides` dari perintah `sam deploy`—dan entri dalam file konfigurasi—mengutamakan mengambil entri pada file templat AWS SAM. Untuk informasi selengkapnya tentang `sam deploy` perintah, lihat [sam deploy](#) di referensi AWS SAMCLI perintah. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

Pemetaan (opsional)

Pemetaan dari kunci dan nilai-nilai terkait yang dapat Anda gunakan untuk menentukan nilai parameter bersyarat, mirip dengan tabel pencarian. Anda dapat mencocokkan kunci ke nilai yang sesuai dengan menggunakan fungsi intrinsik `Fn::FindInMap` dalam bagian `Resources` dan `Outputs`.

Bagian ini sesuai langsung dengan `Mappings` bagian AWS CloudFormation templat.

Kondisi (opsional)

Kondisi yang mengendalikan apakah sumber daya tertentu dibuat atau apakah properti sumber daya tertentu ditetapkan nilai selama pembuatan atau pembaruan tumpukan. Sebagai contoh, Anda dapat secara bersyarat membuat sumber daya yang bergantung pada apakah tumpukan untuk lingkungan produksi atau pengujian.

Bagian ini sesuai langsung dengan `Conditions` bagian AWS CloudFormation templat.

Sumber daya (wajib)

Sumber daya tumpukan dan properti mereka, seperti Instans Amazon Elastic Compute Cloud (Amazon EC2) atau bucket Amazon Simple Storage Service (Amazon S3). Anda dapat merujuk ke sumber daya dalam bagian `Resources` dan `Outputs` dari templat.

Bagian ini mirip dengan bagian `Resources` dari templat AWS CloudFormation . Dalam AWS SAM template, bagian ini dapat berisi AWS SAM sumber daya selain AWS CloudFormation sumber daya.

Output (opsional)

Nilai-nilai yang dikembalikan setiap kali Anda melihat properti tumpukan Anda. Misalnya, Anda dapat mendeklarasikan output untuk nama bucket S3, lalu memanggil perintah `aws cloudformation describe-stacks` AWS Command Line Interface (AWS CLI) untuk melihat nama.

Bagian ini berhubungan langsung dengan bagian `Outputs` dari templat AWS CloudFormation .

Langkah selanjutnya

Untuk mengunduh dan menyebarkan contoh aplikasi tanpa server yang berisi file AWS SAM templat, lihat [Memulai dengan AWS SAM](#) dan ikuti instruksi di [Tutorial: Men-deploy aplikasi Hello World](#)

Bagian global dari templat AWS SAM

Terkadang sumber daya yang Anda deklarasikan dalam templat AWS SAM memiliki konfigurasi umum. Misalnya, Anda mungkin memiliki aplikasi dengan beberapa sumber daya `AWS::Serverless::Function` yang memiliki konfigurasi `Runtime`, `Memory`, `VPCConfig`, `Environment`, dan `Cors` yang identik. Alih-alih menggandakan informasi ini di setiap sumber daya, Anda dapat mendeklarasikan mereka sekali pada bagian `Globals` dan biarkan sumber daya Anda mewarisi mereka.

`Globals` Bagian ini mendukung jenis AWS SAM sumber daya berikut:

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

Contoh:

```
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
```

Dalam contoh ini, HelloWorldFunction dan ThumbnailFunction menggunakan "nodejs12.x" untuk Runtime, "180" detik untuk Timeout, dan "index.handler" untuk Handler. HelloWorldFunction menambahkan variabel lingkungan MESSAGE, selain dari mewarisi TABLE_NAME. ThumbnailFunction mewarisi semua properti Globals dan menambahkan sumber peristiwa API.

Sumber daya dan properti yang didukung

AWS SAM mendukung sumber daya dan properti berikut.

```
Globals:
  Api:
    AccessLogSetting:
    Auth:
    BinaryMediaTypes:
    CacheClusterEnabled:
    CacheClusterSize:
    CanarySetting:
```


Cors:
DefinitionUri:
Domain:
EndpointConfiguration:
GatewayResponses:
MethodSettings:
MinimumCompressionSize:
Name:
OpenApiVersion:
PropagateTags:
TracingEnabled:
Variables:

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

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

```
SimpleTable:
  SSESpecification:

StateMachine:
  PropagateTags:
```

Note

Setiap sumber daya dan properti yang tidak termasuk dalam daftar sebelumnya tidak didukung. Beberapa alasan mereka tidak didukung dikarenakan: 1) Mereka membuka potensi masalah keamanan, atau 2) Mereka membuat templatnya sulit untuk dipahami.

API implisit

AWS SAM membuat API implisit ketika Anda mendeklarasikan API pada bagian `Events`. Anda dapat menggunakan `Globals` untuk menimpa semua properti dari API implisit.

Properti yang dapat ditimpa

Sumber daya dapat menimpa properti yang Anda deklarasikan pada bagian `Globals`. Misalnya, Anda dapat menambahkan variabel baru ke peta variabel lingkungan, atau Anda dapat mengganti variabel yang dideklarasikan secara global. Tetapi sumber daya tidak dapat menghapus properti yang ditentukan pada bagian `Globals`.

Secara umum, `Globals` mendeklarasikan properti yang dibagikan oleh semua sumber daya Anda. Beberapa sumber daya dapat memberikan nilai baru untuk properti yang dideklarasikan secara global, tetapi mereka tidak dapat menghapusnya. Jika beberapa sumber daya menggunakan properti namun yang lainnya tidak, maka Anda tidak harus mendeklarasikan mereka dalam bagian `Globals`.

Bagian berikut menjelaskan cara kerja utama untuk tipe data yang berbeda.

Tipe data primitif diganti

Tipe data primitif mencakup string, angka, Boolean, dan sebagainya.

Nilai yang ditentukan dalam bagian `Resources` menggantikan nilai dalam bagian `Globals`.

Contoh:

```
Globals:
```

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

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

Parameter Runtime untuk MyFunction diatur ke python3.9.

Peta digabung

Peta juga dikenal sebagai kamus atau koleksi pasangan nilai kunci.

Entri peta dalam bagian Resources digabung dengan entri peta global. Jika terdapat duplikat, bagian entri Resource menimpa bagian entri Globals.

Contoh:

```
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
```

Variabel lingkungan dari MyFunction ditetapkan sebagai berikut:

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

Daftar adalah aditif

Daftar juga dikenal sebagai array.

Daftar entri di bagian Globals ditambahkan ke daftar di bagian Resources.

Contoh:

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

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

Parameter SecurityGroupIds untuk MyFunction VpcConfig ditetapkan sebagai berikut:

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

Sumber daya dan referensi properti AWS SAM

Bagian ini berisi informasi referensi untuk sumber daya dan tipe properti AWS SAM.

Topik

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

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

AWS::Serverless::Api

Membuat koleksi sumber daya Amazon API Gateway dan metode yang dapat dipanggil melalui titik akhir HTTPS.

[AWS::Serverless::Api](#) Sumber daya tidak perlu ditambahkan secara eksplisit ke template Definisi Aplikasi AWS Tanpa Server. Sumber daya tipe ini secara implisit dibuat dari penyatuan peristiwa Api yang ditentukan pada sumber daya [AWS::Serverless::Function](#) yang ditentukan dalam templat yang tidak mengacu pada sumber daya [AWS::Serverless::Api](#).

[AWS::Serverless::Api](#) Resource harus digunakan untuk mendefinisikan dan mendokumentasikan penggunaan API OpenApi, yang memberikan lebih banyak kemampuan untuk mengonfigurasi sumber daya Amazon API Gateway yang mendasarinya.

Kami menyarankan Anda menggunakan AWS CloudFormation kait atau kebijakan IAM untuk memverifikasi bahwa sumber daya API Gateway memiliki otorisasi yang melekat padanya untuk mengontrol akses ke sumber daya tersebut.

Untuk informasi selengkapnya tentang penggunaan AWS CloudFormation kait, lihat [Mendaftarkan kait](#) di panduan pengguna AWS CloudFormation CLI dan repositori. [apigw-enforce-authorizer](#) GitHub

Untuk informasi selengkapnya tentang penggunaan kebijakan IAM, lihat [Mengharuskan rute API memiliki otorisasi dalam Panduan](#) Pengembang API Gateway.

Note

Ketika Anda menyebarkan ke AWS CloudFormation, AWS SAM mengubah AWS SAM sumber daya Anda menjadi AWS CloudFormation sumber daya. Untuk informasi selengkapnya, lihat [Sumber daya AWS CloudFormation yang dibuat](#).

Sintaks

Untuk mendeklarasikan entitas ini dalam template AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Type: AWS::Serverless::Api
Properties:
  AccessLogSetting: AccessLogSetting
  AlwaysDeploy: Boolean
  ApiKeySourceType: String
  Auth: ApiAuth
  BinaryMediaTypes: List
  CacheClusterEnabled: Boolean
  CacheClusterSize: String
  CanarySetting: CanarySetting
  Cors: String | CorsConfiguration
  DefinitionBody: JSON
  DefinitionUri: String | ApiDefinition
  Description: String
  DisableExecuteApiEndpoint: Boolean
  Domain: DomainConfiguration
  EndpointConfiguration: EndpointConfiguration
  FailOnWarnings: Boolean
  GatewayResponses: Map
  MergeDefinitions: Boolean
  MethodSettings: MethodSettings
  MinimumCompressionSize: Integer
  Mode: String
  Models: Map
  Name: String
  OpenApiVersion: String
  PropagateTags: Boolean
  StageName: String
  Tags: Map
  TracingEnabled: Boolean
  Variables: Map
```

Properti

AccessLogSetting

Mengonfigurasi Pengaturan Log Ac untuk tahap.

Jenis: [AccessLogSetting](#)

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [AccessLogSetting](#) properti `AWS::ApiGateway::Stage` sumber daya.

AlwaysDeploy

Selalu menerapkan API, bahkan ketika tidak ada perubahan pada API yang terdeteksi.

Tipe: Boolean

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

ApiKeySourceType

Sumber kunci API untuk permintaan pengukuran sesuai dengan rencana penggunaan. Nilai yang valid adalah HEADER dan AUTHORIZER.

Tipe: String

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [ApiKeySourceType](#) properti `AWS::ApiGateway::RestApi` sumber daya.

Auth

Konfigurasi otorisasi untuk mengendalikan akses ke API Gateway Anda.

Untuk informasi selengkapnya tentang mengonfigurasi akses menggunakan AWS SAM lihat [Pengendalian akses ke API dari API Gateway](#).

Jenis: [ApiAuth](#)

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

BinaryMediaTypes

Daftar tipe MIME yang dapat dikembalikan API Anda. Gunakan ini untuk mengaktifkan dukungan biner untuk API. Gunakan ~1 bukan / dalam tipe mime.

Tipe: Daftar

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini mirip dengan [BinaryMediaTypes](#) properti `AWS::ApiGateway::RestApi` sumber daya. Daftar `BinaryMediaTypes` ditambahkan ke AWS CloudFormation sumber daya dan dokumen OpenAPI.

CacheClusterEnabled

Menunjukkan apakah caching diaktifkan untuk panggung. Untuk cache respons, Anda juga harus mengatur `CachingEnabled` ke `true` bawah `MethodSettings`.

Tipe: Boolean

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [CacheClusterEnabled](#) properti `AWS::ApiGateway::Stage` sumber daya.

CacheClusterSize

Ukuran kluster cache dari tahap tersebut.

Tipe: String

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [CacheClusterSize](#) properti `AWS::ApiGateway::Stage` sumber daya.

CanarySetting

Konfigurasi pengaturan canary ke tahap deployment reguler.

Jenis: [CanarySetting](#)

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [CanarySetting](#) properti `AWS::ApiGateway::Stage` sumber daya.

Cors

Kelola Cross-origin resource sharing (CORS) untuk semua API Gateway Anda. Tentukan domain untuk diizinkan sebagai string atau tentukan kamus dengan konfigurasi `Cors` tambahan.

Note

CORS AWS SAM perlu memodifikasi definisi OpenAPI Anda. Buat definisi OpenAPI sebaris di untuk mengaktifkan `DefinitionBody` CORS.

Untuk informasi lebih lanjut tentang CORS, lihat [Aktifkan CORS untuk sumber daya API REST API Gateway](#) di Panduan Developer API Gateway.

Jenis: String | [CorsConfiguration](#)

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

DefinitionBody

Spesifikasi OpenAPI yang menggambarkan API Anda. Jika `DefinitionUri` dan `DefinitionBody` tidak ditentukan, SAM akan membuat `DefinitionBody` untuk Anda berdasarkan konfigurasi templat Anda.

Untuk mereferensikan OpenAPI file lokal yang mendefinisikan API Anda, gunakan `AWS::Include` transformasi. Untuk mempelajari selengkapnya, lihat [Unggah file lokal saat penerapan](#).

Tipe: JSON

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini mirip dengan [Body](#) properti `AWS::ApiGateway::RestApi` sumber daya. Jika properti tertentu disediakan, konten dapat dimasukkan atau dimodifikasi ke dalam `DefinitionBody` sebelum diteruskan ke CloudFormation. Properti termasuk `Auth`, `BinaryMediaTypesCors`, `GatewayResponses`, `Models`, dan `EventSource` tipe Api untuk yang sesuai `AWS::Serverless::Function`.

DefinitionUri

Amazon S3 Uri, jalur file lokal, atau objek lokasi dokumen OpenAPI menentukan API. Objek Amazon S3 yang dirujuk properti ini harus menjadi file OpenAPI yang valid. Jika `DefinitionUri` dan `DefinitionBody` tidak ditentukan, SAM akan membuat `DefinitionBody` untuk Anda berdasarkan konfigurasi templat Anda.

Jika lintasan file lokal disediakan, templat harus melalui alur kerja yang mencakup perintah `sam deploy` atau `sam package`, agar ketentuan diubah dengan benar.

Fungsi intrinsik tidak didukung dalam OpenApi file eksternal yang direferensikan oleh. `DefinitionUri` Gunakan sebagai gantinya `DefinitionBody` properti dengan [Include Transform](#) untuk mengimpor OpenApi definisi ke dalam template.

Jenis: String | [ApiDefinition](#)

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini mirip dengan [BodyS3Location](#) properti `AWS::ApiGateway::RestApi` sumber daya. Properti Amazon S3 nest diberi nama berbeda.

Description

Deskripsi sumber daya Api.

Tipe: String

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [Description](#) properti `AWS::ApiGateway::RestApi` sumber daya.

DisableExecuteApiEndpoint

Menentukan apakah klien dapat memanggil API dengan menggunakan titik akhir `execute-api` default. Secara default, klien dapat memanggil API Anda dengan default `https://
{api_id}.execute-api.{region}.amazonaws.com`. Untuk mengharuskan klien menggunakan nama domain khusus untuk menjalankan API Anda, tentukan `True`.

Tipe: Boolean

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini mirip dengan [DisableExecuteApiEndpoint](#) properti `AWS::ApiGateway::RestApi` sumber daya. Itu diteruskan langsung ke `disableExecuteApiEndpoint` properti [x-amazon-apigateway-endpoint-configuration](#) ekstensi, yang akan ditambahkan ke [Body](#) properti `AWS::ApiGateway::RestApi` sumber daya.

Domain

Mengonfigurasi domain khusus untuk API dari API Gateway ini.

Jenis: [DomainConfiguration](#)

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

EndpointConfiguration

Tipe titik akhir dari REST API.

Jenis: [EndpointConfiguration](#)

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini mirip dengan [EndpointConfiguration](#) properti AWS::ApiGateway::RestApi sumber daya. Properti konfigurasi nest diberi nama berbeda.

FailOnWarnings

Menentukan apakah akan memutar kembali penciptaan API (true) atau not (false) ketika peringatan ditemui. Nilai default-nya adalah false.

Tipe: Boolean

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [FailOnWarnings](#) properti AWS::ApiGateway::RestApi sumber daya.

GatewayResponses

Mengonfigurasi Respons Gateway untuk API. Respons Gateway adalah respons yang dikembalikan oleh API Gateway, secara langsung atau melalui penggunaan Lambda Authorizers. Untuk informasi selengkapnya, lihat dokumentasi untuk [OpenApi ekstensi Api Gateway untuk Tanggapan Gateway](#).

Tipe: Peta

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

MergeDefinitions

AWS SAM menghasilkan OpenAPI spesifikasi dari sumber peristiwa API Anda. Tentukan `true` untuk AWS SAM menggabungkan ini ke dalam OpenAPI spesifikasi sebaris yang ditentukan dalam sumber daya Anda `AWS::Serverless::Api`. Tentukan `false` untuk tidak bergabung.

`MergeDefinitions` membutuhkan `DefinitionBody` properti `AWS::Serverless::Api` untuk didefinisikan. `MergeDefinitions` tidak kompatibel dengan `DefinitionUri` properti untuk `AWS::Serverless::Api`.

Nilai default: `false`

Tipe: Boolean

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

MethodSettings

Mengonfigurasi semua pengaturan untuk tahap API termasuk Logging, Metrik, CacheTTL, Throttling.

Jenis: Daftar [MethodSetting](#)

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [MethodSettings](#) properti `AWS::ApiGateway::Stage` sumber daya.

MinimumCompressionSize

Izinkan kompresi isi respons berdasarkan header `Accept-Encoding` klien. Kompresi terpicu ketika ukuran tubuh respons lebih besar dari atau sama dengan ambang batas Anda yang telah dikonfigurasi. Ambang batas ukuran isi maksimum adalah 10 MB (10.485.760 Bit). - Tipe kompresi berikut didukung: `gzip`, `mengempis`, dan `identitas`.

Tipe: Integer

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [MinimumCompressionSize](#) properti `AWS::ApiGateway::RestApi` sumber daya.

Mode

Properti ini hanya berlaku bila Anda menggunakan OpenAPI untuk menentukan API REST Anda. Mode menentukan bagaimana API Gateway menangani pembaruan sumber daya. Untuk informasi selengkapnya, lihat Properti [mode](#) dari jenis [AWS::ApiGateway::RestApi](#) sumber daya.

Nilai yang valid: `overwrite` atau `merge`

Tipe: String

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [Mode](#) properti `AWS::ApiGateway::RestApi` sumber daya.

Models

Skema yang akan digunakan oleh metode API Anda. Skema ini dapat dijelaskan menggunakan JSON atau YAML. Lihat bagian Contoh di bagian bawah halaman ini misalnya model.

Tipe: Peta

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

Name

Nama untuk RestApi sumber daya API Gateway

Tipe: String

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [Name](#) properti `AWS::ApiGateway::RestApi` sumber daya.

OpenApiVersion

Versi OpenApi untuk digunakan. Ini bisa `2.0` untuk spesifikasi Swagger, atau salah satu versi OpenApi 3.0, seperti `3.0.1` Untuk informasi selengkapnya tentang OpenAPI, lihat [Spesifikasi OpenAPI](#).

Note

AWS SAM menciptakan tahap yang disebut secara Stage default. Menyetel properti ini ke nilai yang valid akan mencegah pembuatan panggungStage.

Tipe: String

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

PropagateTags

Tunjukkan apakah akan meneruskan tag dari Tags properti ke sumber daya yang Anda [AWS::Serverless::Api](#) hasilkan atau tidak. Tentukan `True` untuk menyebarkan tag di sumber daya yang Anda hasilkan.

Tipe: Boolean

Wajib: Tidak

Default: `False`

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

StageName

Nama tahap, yang digunakan API Gateway sebagai segmen jalur pertama dalam Uniform Resource Identifier (URI) panggil.

Untuk referensi sumber daya tahap, gunakan `<api-logical-id>.Stage`. Untuk informasi lebih lanjut tentang referensi sumber daya yang dibuat ketika sumber daya [AWS::Serverless::Api](#) ditentukan, lihat [AWS CloudFormation sumber daya yang dihasilkan saat AWS::Serverless::Api ditentukan](#). Untuk informasi umum tentang AWS CloudFormation sumber daya yang dihasilkan, lihat [Sumber daya AWS CloudFormation yang dibuat](#).

Tipe: String

Wajib: Ya

AWS CloudFormation kompatibilitas: Properti ini mirip dengan [StageName](#) properti `AWS::ApiGateway::Stage` sumber daya. Hal ini diperlukan dalam SAM, tetapi tidak diperlukan di API Gateway

Catatan tambahan: API implisit memiliki nama tahap “Prod”.

Tags

Sebuah peta (string ke string) yang menentukan tanda yang akan ditambahkan ke tahap API Gateway ini. Untuk detail tentang kunci dan nilai tag yang valid, lihat [Tag sumber daya](#) di Panduan AWS CloudFormation Pengguna.

Tipe: Peta

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini mirip dengan [Tags](#) properti `AWS::ApiGateway::Stage` sumber daya. Properti Tags di SAM terdiri dari pasangan Key:Value; di CloudFormation dalamnya terdiri dari daftar objek Tag.

TracingEnabled

Menunjukkan apakah penelusuran aktif dengan X-Ray diaktifkan untuk tahap tersebut. Untuk informasi selengkapnya tentang X-Ray, lihat [Melacak permintaan pengguna untuk REST API menggunakan X-Ray](#) di Panduan Developer API Gateway.

Tipe: Boolean

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [TracingEnabled](#) properti `AWS::ApiGateway::Stage` sumber daya.

Variables

Sebuah peta (string ke string) yang menentukan variabel tahap, dengan nama variabel adalah kunci dan nilai variabel adalah nilai. Nama variabel terbatas pada karakter alfanumerik. Nilai harus sesuai dengan ekspresi reguler berikut: `[A-Za-z0-9._~:/?#&=, -]+`.

Tipe: Peta

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [Variables](#) properti `AWS::ApiGateway::Stage` sumber daya.

Nilai Pengembalian

Ref

Ketika ID logis dari sumber daya ini disediakan untuk fungsi intrinsik Ref, ID logis mengembalikan ID dari API Gateway utama.

Untuk informasi lebih lanjut tentang menggunakan fungsi Ref, lihat [Ref](#) di Panduan Pengguna AWS CloudFormation .

Fn:: GetAtt

Fn :: GetAtt mengembalikan nilai untuk atribut yang ditentukan dari jenis ini. Berikut ini adalah atribut yang tersedia dan nilai-nilai kembali sampel.

Untuk informasi lebih lanjut tentang Fn :: GetAtt, lihat [Fn :: GetAtt](#) di Panduan Pengguna AWS CloudFormation

RootResourceId

ID sumber daya root untuk sumber daya RestApi, seperti a0bc123d4e.

Contoh-contoh

SimpleApiExample

File AWS SAM template Hello World yang berisi Fungsi Lambda dengan titik akhir API. Ini adalah file AWS SAM template lengkap untuk aplikasi tanpa server yang berfungsi.

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

Cuplikan AWS SAM template dengan API yang ditentukan dalam file Swagger eksternal bersama dengan integrasi Lambda dan konfigurasi CORS. Ini hanya sebagian dari file AWS SAM template yang menunjukkan [AWS::Serverless::Api](#) definisi.

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

Cuplikan AWS SAM template dengan API yang menggunakan Amazon Cognito untuk mengotorisasi permintaan terhadap API. Ini hanya sebagian dari file AWS SAM template yang menunjukkan [AWS::Serverless::Api](#) definisi.

YAML

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

ApiModelsExample

Cuplikan AWS SAM template dengan API yang menyertakan skema Model. Ini hanya sebagian dari file AWS SAM template, menunjukkan [AWS::Serverless::Api](#) definisi dengan dua skema model.

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
```

Contoh caching

File AWS SAM template Hello World yang berisi Fungsi Lambda dengan titik akhir API. API telah mengaktifkan caching untuk satu sumber daya dan metode. Untuk informasi selengkapnya tentang caching, lihat [Mengaktifkan caching API untuk meningkatkan daya tanggap](#) dalam Panduan Pengembang API Gateway.

YAML

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

  ApiFunction: # Adds a GET method at the root resource via an Api event
    Type: AWS::Serverless::Function
    Properties:
      Events:
        ApiEvent:
          Type: Api
          Properties:
            Path: /
            Method: get
```

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

ApiAuth

Konfigurasi otorisasi untuk mengendalikan akses ke API Gateway Anda.

Untuk informasi lebih lanjut dan contoh untuk mengonfigurasi akses menggunakan AWS SAM lihat [Pengendalian akses ke API dari API Gateway](#).

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

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

Properti

AddApiKeyRequiredToCorsPreflight

Jika Cors properti `ApiKeyRequired` dan disetel, maka pengaturan `AddApiKeyRequiredToCorsPreflight` akan menyebabkan kunci API ditambahkan ke `Options` properti.

Tipe: Boolean

Wajib: Tidak

Default: True

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

AddDefaultAuthorizerToCorsPreflight

Jika properti `DefaultAuthorizer` dan `Cors` ditetapkan, kemudian pengaturan `AddDefaultAuthorizerToCorsPreflight` akan menyebabkan otorisasi default yang akan ditambahkan ke properti `Options` di bagian `OpenAPI`.

Tipe: Boolean

Wajib: Tidak

Default: BETUL

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

ApiKeyRequired

Jika diatur ke BETUL, kunci API diperlukan untuk semua peristiwa API. Untuk informasi selengkapnya tentang kunci API lihat [Buat dan Gunakan Rencana Penggunaan dengan Kunci API](#) di Panduan Developer API Gateway.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Authorizers

Pengotorisasi yang digunakan untuk mengendalikan akses ke API dari API Gateway Anda.

Untuk informasi selengkapnya, lihat [Pengendalian akses ke API dari API Gateway](#).

Jenis: [CognitoAuthorizer](#) | [LambdaTokenAuthorizer](#) | [LambdaRequestAuthorizer](#)

Wajib: Tidak

Default: Tidak ada

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Catatan tambahan: SAM menambahkan Authorizers ke OpenApi definisi Api.

DefaultAuthorizer

Tentukan otorisasi default untuk API Gateway dari API, yang akan digunakan untuk otorisasi panggilan API secara default.

Note

Jika Api EventSource untuk fungsi yang terkait dengan API ini dikonfigurasi untuk menggunakan Izin IAM, maka properti ini harus disetel keAWS_IAM, jika tidak, kesalahan akan terjadi.

Tipe: String

Wajib: Tidak

Default: Tidak ada

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

InvokeRole

Mengatur kredensial integrasi untuk semua sumber daya dan metode untuk nilai ini.

CALLER_CREDENTIALS memetakan ke `arn:aws:iam::*:user/*`, yang menggunakan kredensial pemanggil untuk memanggil titik akhir.

Nilai yang valid:CALLER_CREDENTIALS,NONE, IAMRoleArn

Tipe: String

Wajib: Tidak

Default: CALLER_CREDENTIALS

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

ResourcePolicy

Konfigurasi Kebijakan Sumber Daya untuk semua metode dan jalur pada API.

Jenis: [ResourcePolicyStatement](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Catatan tambahan: Pengaturan ini juga dapat ditentukan pada `AWS::Serverless::Function` individu menggunakan [ApiFunctionAuth](#). Ini diperlukan untuk API dengan `EndpointConfiguration: PRIVATE`.

UsagePlan

Mengonfigurasi rencana penggunaan yang terkait dengan API ini. Untuk informasi selengkapnya tentang rencana penggunaan lihat [Buat dan Gunakan Rencana Penggunaan dengan Kunci API](#) di Panduan Developer API Gateway.

AWS SAM Properti ini menghasilkan tiga AWS CloudFormation sumber daya tambahan ketika properti ini disetel: an [AWS::ApiGateway::UsagePlan](#), an [AWS::ApiGateway::UsagePlanKey](#), dan an [AWS::ApiGateway::ApiKey](#). Untuk informasi selengkapnya tentang skenario ini, lihat [UsagePlanproperti ditentukan](#). Untuk informasi umum tentang sumber daya AWS CloudFormation yang dihasilkan, lihat [Sumber daya AWS CloudFormation yang dibuat](#).

Jenis: [ApiUsagePlan](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

CognitoAuth

Contoh Cognito Auth

YAML

```
Auth:
```

```

Authorizers:
  MyCognitoAuth:
    UserPoolArn:
      Fn::GetAtt:
        - MyUserPool
        - Arn
    AuthType: "COGNITO_USER_POOLS"
DefaultAuthorizer: MyCognitoAuth
InvokeRole: CALLER_CREDENTIALS
AddDefaultAuthorizerToCorsPreflight: false
ApiKeyRequired: false
ResourcePolicy:
  CustomStatements: [{
    "Effect": "Allow",
    "Principal": "*",
    "Action": "execute-api:Invoke",
    "Resource": "execute-api:/Prod/GET/pets",
    "Condition": {
      "IpAddress": {
        "aws:SourceIp": "1.2.3.4"
      }
    }
  }]
  IpRangeBlacklist:
    - "10.20.30.40"

```

ApiUsagePlan

Mengonfigurasi rencana penggunaan untuk API dari API Gateway. Untuk informasi selengkapnya tentang rencana penggunaan, lihat [Buat dan Gunakan Rencana Penggunaan dengan Kunci API](#) di Panduan Developer API Gateway.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```

CreateUsagePlan: String
Description: String
Quota: QuotaSettings
Tags: List

```


Throttle: [ThrottleSettings](#)

UsagePlanName: *String*

Properti

CreateUsagePlan

Menentukan cara rencana penggunaan ini dikonfigurasi. Nilai yang valid adalah PER_API, SHARED, dan NONE.

PER_API membuat [AWS::ApiGateway::UsagePlan](#), [AWS::ApiGateway::ApiKey](#), dan [AWS::ApiGateway::UsagePlanKey](#) sumber daya yang khusus untuk API ini. Masing-masing sumber daya ini memiliki ID logis *<api-logical-id>UsagePlan*, *<api-logical-id>ApiKey*, dan *<api-logical-id>UsagePlanKey*.

SHARED membuat [AWS::ApiGateway::UsagePlan](#), [AWS::ApiGateway::ApiKey](#), dan [AWS::ApiGateway::UsagePlanKey](#) sumber daya yang dibagikan di seluruh API apa pun yang juga ada `CreateUsagePlan: SHARED` di AWS SAM template yang sama. Masing-masing sumber daya ini memiliki ID logis `ServerlessUsagePlan`, `ServerlessApiKey`, dan `ServerlessUsagePlanKey`. Jika Anda menggunakan opsi ini, kami sarankan Anda menambahkan konfigurasi tambahan untuk rencana penggunaan ini hanya pada satu sumber daya API untuk menghindari ketentuan yang bertentangan dan keadaan tidak pasti.

NONE menonaktifkan pembuatan atau asosiasi rencana penggunaan dengan API ini. Ini hanya diperlukan jika SHARED atau PER_API ditentukan dalam [Bagian global dari templat AWS SAM](#).

Nilai yang valid adalah : PER_API, SHARED, and NONE

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Description

Deskripsi rencana penggunaan.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Description](#) dari sumber daya `AWS::ApiGateway::UsagePlan`.

Quota

Mengonfigurasi jumlah permintaan yang dapat dibuat pengguna dalam interval tertentu.

Jenis: [QuotaSettings](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Quota](#) dari sumber daya `AWS::ApiGateway::UsagePlan`.

Tags

Array tanda yang berubah-ubah (pasangan kunci-nilai) yang akan dikaitkan ke rencana penggunaan.

Properti ini menggunakan [Jenis CloudFormation Tag](#).

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Tags](#) dari sumber daya `AWS::ApiGateway::UsagePlan`.

Throttle

Mengonfigurasi tingkat permintaan keseluruhan (permintaan rata-rata per detik) dan kapasitas lonjakan.

Jenis: [ThrottleSettings](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Throttle](#) dari sumber daya `AWS::ApiGateway::UsagePlan`.

UsagePlanName

Sebuah nama untuk rencana penggunaan.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [UsagePlanName](#) dari sumber daya `AWS::ApiGateway::UsagePlan`.

Contoh-contoh

UsagePlan

Berikut ini adalah contoh rencana penggunaan.

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

Menentukan otorisasi Kolam Pengguna Amazon Cognito.

Untuk informasi selengkapnya dan contoh tambahan, lihat [Pengendalian akses ke API dari API Gateway](#).

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
AuthorizationScopes: List
Identity: CognitoAuthorizationIdentity
```

`UserPoolArn`: *String*

Properti

AuthorizationScopes

Daftar cakupan otorisasi untuk pemberi kuasa ini.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Identity

Properti ini dapat digunakan untuk menentukan IdentitySource dalam permintaan masuk untuk otorisasi.

Jenis: [CognitoAuthorizationIdentity](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

UserPoolArn

Dapat merujuk ke kolam pengguna/menentukan arn kolam pengguna tempat Anda ingin menambahkan pengotorisasi cognito ini

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

CognitoAuth

Contoh Cognito Auth

YAML

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

CognitoAuthorizationIdentity

Properti ini dapat digunakan untuk menentukan permintaan masuk untuk otorisasi. IdentitySource Untuk informasi selengkapnya, IdentitySource lihat [OpenApi ekstensi ApiGateway Authorizer](#).

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

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

Properti

Header

Tentukan nama header untuk Otorisasi dalam OpenApi definisi.

Tipe: String

Wajib: Tidak

Default: Otorisasi

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

ReauthorizeEvery

Periode time-to-live (TTL), dalam hitungan detik, yang menentukan berapa lama API Gateway menyimpan hasil otorisasi. Jika Anda menentukan nilai yang lebih besar dari 0, API Gateway akan menyimpan tanggapan pemberi kuasa. Secara default, API Gateway mengatur properti ini ke 300. Nilai maksimumnya adalah 3600, atau 1 jam.

Tipe: Integer

Wajib: Tidak

Default: 300

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

ValidationExpression

Tentukan ekspresi validasi untuk memvalidasi Identitas masuk

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

CognitoAuthIdentity

YAML

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

LambdaRequestAuthorizer

Mengonfigurasi Otorisasi Lambda untuk mengendalikan akses ke API Anda dengan fungsi Lambda.

Untuk informasi selengkapnya dan contoh tambahan, lihat [Pengendalian akses ke API dari API Gateway](#).

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

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

Properti

DisableFunctionDefaultPermissions

Tentukan `true` untuk AWS SAM mencegah membuat sumber daya secara otomatis untuk memberikan izin antara `AWS::Lambda::Permissions` sumber `AWS::Serverless::Api` daya Anda dan fungsi Lambda otorisasi.

Nilai default: `false`

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

FunctionArn

Tentukan fungsi ARN dari fungsi Lambda yang menyediakan otorisasi untuk API.

Note

AWS SAM akan secara otomatis membuat `AWS::Lambda::Permissions` sumber daya ketika `FunctionArn` ditentukan untuk `AWS::Serverless::Api`. Sumber

`AWS::Lambda::Permissions` daya menyediakan izin antara API Anda dan fungsi Lambda otorisasi.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

FunctionInvokeRole

Menambahkan kredensi otorisasi ke OpenApi definisi otorisasi Lambda.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

FunctionPayloadType

Properti ini dapat digunakan untuk menentukan tipe Otorisasi Lambda untuk API.

Nilai yang valid: TOKEN atau REQUEST

Tipe: String

Wajib: Tidak

Default: TOKEN

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Identity

Properti ini dapat digunakan untuk menentukan IdentitySource dalam permintaan masuk untuk otorisasi. Properti ini hanya diperlukan jika properti FunctionPayloadType diatur ke REQUEST.

Jenis: [LambdaRequestAuthorizationIdentity](#)

Wajib: Bersyarat

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

LambdaRequestAuth

YAML

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

LambdaRequestAuthorizationIdentity

Properti ini dapat digunakan untuk menentukan permintaan masuk untuk otorisasi. IdentitySource Untuk informasi selengkapnya, IdentitySource lihat [OpenApi ekstensi ApiGateway Authorizer](#).

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

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

Properti

Context

Mengonversi string konteks yang diberikan untuk ekspresi pemetaan `formatcontext.contextString`.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Headers

Mengonversi header untuk string yang dipisahkan koma dari ekspresi pemetaan format `method.request.header.name`.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

QueryString

Mengonversi string kueri yang diberikan menjadi string yang dipisahkan koma dari ekspresi pemetaan format `method.request.querystring.queryString`.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

ReauthorizeEvery

Periode time-to-live (TTL), dalam hitungan detik, yang menentukan berapa lama API Gateway menyimpan hasil otorisasi. Jika Anda menentukan nilai yang lebih besar dari 0, API Gateway akan menyimpan tanggapan pemberi kuasa. Secara default, API Gateway mengatur properti ini ke 300. Nilai maksimumnya adalah 3600, atau 1 jam.

Tipe: Integer

Wajib: Tidak

Default: 300

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

StageVariables

Mengonversi variabel tahap yang diberikan menjadi string yang dipisahkan koma dari ekspresi pemetaan format `stageVariables.stageVariable`.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

LambdaRequestIdentity

YAML

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

LambdaTokenAuthorizer

Mengonfigurasi Otorisasi Lambda untuk mengendalikan akses ke API Anda dengan fungsi Lambda.

Untuk informasi selengkapnya dan contoh tambahan, lihat [Pengendalian akses ke API dari API Gateway](#).

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

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

Properti

DisableFunctionDefaultPermissions

Tentukan `true` untuk AWS SAM mencegah membuat sumber daya secara otomatis untuk memberikan izin antara `AWS::Lambda::Permissions` sumber `AWS::Serverless::Api` daya Anda dan fungsi Lambda otorisasi.

Nilai default: `false`

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

FunctionArn

Tentukan fungsi ARN dari fungsi Lambda yang menyediakan otorisasi untuk API.

Note

AWS SAM akan secara otomatis membuat `AWS::Lambda::Permissions` sumber daya ketika `FunctionArn` ditentukan untuk `AWS::Serverless::Api`. Sumber `AWS::Lambda::Permissions` daya menyediakan izin antara API Anda dan fungsi Lambda otorisasi.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

FunctionInvokeRole

Menambahkan kredensi otorisasi ke OpenApi definisi otorisasi Lambda.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

FunctionPayloadType

Properti ini dapat digunakan untuk menentukan tipe Otorisasi Lambda untuk API.

Nilai yang valid: TOKEN atau REQUEST

Tipe: String

Wajib: Tidak

Default: TOKEN

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Identity

Properti ini dapat digunakan untuk menentukan IdentitySource dalam permintaan masuk untuk otorisasi. Properti ini hanya diperlukan jika properti FunctionPayloadType diatur ke REQUEST.

Jenis: [LambdaTokenAuthorizationIdentity](#)

Wajib: Bersyarat

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

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

Properti ini dapat digunakan untuk menentukan permintaan masuk untuk otorisasi. IdentitySource Untuk informasi selengkapnya, IdentitySource lihat [OpenApi ekstensi ApiGateway Authorizer](#).

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

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

ValidationExpression: *String*

Properti

Header

Tentukan nama header untuk Otorisasi dalam OpenApi definisi.

Tipe: String

Wajib: Tidak

Default: Otorisasi

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

ReauthorizeEvery

Periode time-to-live (TTL), dalam hitungan detik, yang menentukan berapa lama API Gateway menyimpan hasil otorisasi. Jika Anda menentukan nilai yang lebih besar dari 0, API Gateway akan menyimpan tanggapan pemberi kuasa. Secara default, API Gateway mengatur properti ini ke 300. Nilai maksimumnya adalah 3600, atau 1 jam.

Tipe: Integer

Wajib: Tidak

Default: 300

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

ValidationExpression

Tentukan ekspresi validasi untuk memvalidasi Identitas yang masuk.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

LambdaTokenIdentity

YAML

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

ResourcePolicyStatement

Mengonfigurasi kebijakan sumber daya untuk semua metode dan jalur API. Untuk informasi selengkapnya tentang kebijakan sumber daya, lihat [Mengendalikan akses ke API dengan kebijakan sumber daya API Gateway](#) di Panduan Developer API Gateway.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

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

Properti

AwsAccountBlacklist

Akun AWS yang akan diblokir.

Jenis: Daftar String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

AwsAccountWhitelist

Akun AWS yang akan diizinkan. Untuk contoh penggunaan properti ini, lihat bagian Contoh di bagian bawah halaman ini.

Jenis: Daftar String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

CustomStatements

Daftar pernyataan kebijakan sumber daya kustom untuk diterapkan ke API ini. Untuk contoh penggunaan properti ini, lihat bagian Contoh di bagian bawah halaman ini.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

IntrinsicVpcBlacklist

Daftar virtual private cloud (VPC) yang akan diblokir, dengan setiap VPC ditetapkan sebagai referensi seperti [referensi dinamis](#) atau [fungsi intrinsik](#) Ref. Untuk contoh penggunaan properti ini, lihat bagian Contoh di bagian bawah halaman ini.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

IntrinsicVpcWhitelist

Daftar VPC yang akan diizinkan, dengan setiap VPC ditetapkan sebagai referensi seperti [referensi dinamis](#) atau [fungsi intrinsik](#) Ref.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

IntrinsicVpceBlacklist

Daftar VPC endpoint yang akan diblokir, dengan setiap VPC endpoint ditetapkan sebagai referensi seperti [referensi dinamis](#) atau [fungsi intrinsik](#) Ref.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

IntrinsicVpceWhitelist

Daftar VPC endpoint yang akan diizinkan, dengan setiap VPC endpoint ditetapkan sebagai referensi seperti [referensi dinamis](#) atau [fungsi intrinsik](#) Ref. Untuk contoh penggunaan properti ini, lihat bagian Contoh di bagian bawah halaman ini.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

IpRangeBlacklist

Alamat IP atau jangkauan alamat yang akan diblokir. Untuk contoh penggunaan properti ini, lihat bagian Contoh di bagian bawah halaman ini.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

IpRangeWhitelist

Alamat IP atau jangkauan alamat yang akan diizinkan.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

SourceVpcBlacklist

Sumber VPC atau VPC endpoint yang akan diblokir. Nama VPC sumber harus dimulai dengan "vpc-" dan nama VPC endpoint sumber harus dimulai dengan "vpce-". Untuk contoh penggunaan properti ini, lihat bagian Contoh di bagian bawah halaman ini.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

SourceVpcWhitelist

VPC sumber atau VPC endpoint yang akan diizinkan. Nama VPC sumber harus dimulai dengan "vpc-" dan nama VPC endpoint sumber harus dimulai dengan "vpce-".

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

Contoh Kebijakan Sumber Daya

Contoh berikut blok dua alamat IP dan sumber VPC, dan mengizinkan akun AWS.

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

Dokumen OpenAPI yang menentukan API.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

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

Version: *String*

Properti

Bucket

Nama bucket Amazon S3 tempat file OpenAPI disimpan.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Bucket](#) dari tipe data `AWS::ApiGateway::RestApi S3Location`.

Key

Kunci Amazon S3 dari file OpenAPI.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Key](#) dari tipe data `AWS::ApiGateway::RestApi S3Location`.

Version

Untuk objek berversi, versi file OpenAPI.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Version](#) dari tipe data `AWS::ApiGateway::RestApi S3Location`.

Contoh-contoh

Contoh Uri Ketentuan

Contoh ketentuan API

YAML

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

CorsConfiguration

Kelola cross-origin resource sharing (CORS) untuk API Gateway Anda. Tentukan domain untuk diizinkan sebagai string atau tentukan kamus dengan konfigurasi Cors tambahan.

Note

CORS AWS SAM perlu memodifikasi definisi OpenAPI Anda. Buat definisi OpenAPI sebaris di untuk mengaktifkan `DefinitionBody` CORS. Jika `CorsConfiguration` disetel dalam definisi OpenAPI dan juga di tingkat properti, AWS SAM gabungkan mereka. Level properti lebih diutamakan daripada definisi OpenAPI.

Untuk informasi lebih lanjut tentang CORS, lihat [Aktifkan CORS untuk sumber daya API REST API Gateway](#) di Panduan Developer API Gateway.

Sintaks

Untuk mendeklarasikan entitas ini dalam template AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

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

Properti

AllowCredentials

Boolean yang menunjukkan apakah permintaan diizinkan untuk berisi kredensial.

Tipe: Boolean

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

AllowHeaders

String header yang akan diizinkan.

Tipe: String

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

AllowMethods

String yang berisi metode HTTP yang akan diizinkan.

Tipe: String

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

AllowOrigin

String asal yang akan diizinkan.

Tipe: String

Wajib: Ya

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

MaxAge

String yang berisi jumlah detik untuk men-cache permintaan CORS Preflight.

Tipe: String

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

Contoh-contoh

CorsConfiguration

Contoh Konfigurasi CORS. Ini hanyalah sebagian dari file AWS SAM template yang menunjukkan [AWS::Serverless::Api](#) definisi dengan CORS dikonfigurasi dan file. [AWS::Serverless::Function](#) Jika Anda menggunakan integrasi proxy Lambda atau integrasi proxy HTTP, backend Anda harus mengembalikan `Access-Control-Allow-Origin`, `Access-Control-Allow-Methods` dan header `Access-Control-Allow-Headers`

YAML

```
Resources:
  ApiGatewayApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      Cors:
        AllowMethods: "'POST, GET'"
        AllowHeaders: "'X-Forwarded-For'"
        AllowOrigin: "'www.example.com'"
        MaxAge: "'600'"
        AllowCredentials: true
  ApiFunction: # Adds a GET method at the root resource via an Api event
    Type: AWS::Serverless::Function
    Properties:
      Events:
        ApiEvent:
          Type: Api
          Properties:
            Path: /
            Method: get
            RestApiId:
              Ref: ApiGatewayApi
      Runtime: python3.10
```



```
Handler: index.handler
InlineCode: |
import json
def handler(event, context):
    return {
        'statusCode': 200,
        'headers': {
            'Access-Control-Allow-Headers': 'Content-Type',
            'Access-Control-Allow-Origin': 'www.example.com',
            'Access-Control-Allow-Methods': 'POST, GET'
        },
        'body': json.dumps('Hello from Lambda!')
    }
```

DomainConfiguration

Mengonfigurasi domain kustom untuk API.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
BasePath: List
NormalizeBasePath: Boolean
CertificateArn: String
DomainName: String
EndpointConfiguration: String
MutualTlsAuthentication: MutualTlsAuthentication
OwnershipVerificationCertificateArn: String
Route53: Route53Configuration
SecurityPolicy: String
```

Properti

BasePath

Daftar basepaths untuk mengonfigurasi dengan nama domain Amazon API Gateway.

Tipe: Daftar

Wajib: Tidak

Default: /

Kompatibilitas AWS CloudFormation: Properti ini mirip dengan properti [BasePath](#) dari sumber daya AWS::ApiGateway::BasePathMapping. AWS SAM membuat beberapa sumber daya AWS::ApiGateway::BasePathMapping, satu per BasePath ditentukan dalam properti ini.

NormalizeBasePath

Menunjukkan apakah karakter non-alfanumerik diizinkan dalam basepaths yang ditentukan oleh properti. BasePath Ketika diatur ke True, karakter non-alfanumerik dihapus dari basepaths.

Gunakan NormalizeBasePath dengan BasePath properti.

Tipe: Boolean

Wajib: Tidak

Default: BETUL

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

CertificateArn

Amazon Resource Name (ARN) dari sertifikat dikelola AWS dari titik akhir nama domain ini. AWS Certificate Manager adalah satu-satunya sumber yang didukung.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini mirip dengan properti [CertificateArn](#) dari sumber daya AWS::ApiGateway::DomainName. Jika [EndpointConfiguration](#) diatur ke REGIONAL (nilai default), CertificateArn peta ke [RegionalCertificateArn](#) dalam AWS::ApiGateway::DomainName. Jika [EndpointConfiguration](#) diatur ke EDGE, CertificateArn petakan ke [CertificateArn](#) dalam AWS::ApiGateway::DomainName.

Catatan tambahan: Untuk titik akhir EDGE, Anda harus membuat sertifikat di Wilayah us-east-1 AWS.

DomainName

Nama domain khusus untuk API dari API Gateway Anda. Huruf besar tidak didukung.

AWS SAM membuat sumber daya [AWS::ApiGateway::DomainName](#) ketika properti ini diatur. Untuk informasi selengkapnya tentang skenario ini, lihat [DomainNameproperti ditentukan](#). Untuk informasi tentang sumber daya AWS CloudFormation yang dibuat, lihat [Sumber daya AWS CloudFormation yang dibuat](#).

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [DomainName](#) dari sumber daya `AWS::ApiGateway::DomainName`.

EndpointConfiguration

Menentukan tipe titik akhir API Gateway untuk memetakan ke domain kustom. Nilai properti ini menentukan bagaimana properti `CertificateArn` dipetakan di AWS CloudFormation.

Nilai yang valid: REGIONAL atau EDGE

Tipe: String

Wajib: Tidak

Default: REGIONAL

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padananAWS CloudFormation.

MutualTlsAuthentication

Konfigurasi autentikasi Keamanan Lapisan Pengangkutan (TLS) yang saling terkait untuk nama domain kustom.

Jenis: [MutualTlsAuthentication](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [MutualTlsAuthentication](#) dari sumber daya `AWS::ApiGateway::DomainName`.

OwnershipVerificationCertificateArn

ARN sertifikat publik yang dikeluarkan oleh ACM untuk memvalidasi kepemilikan domain kustom Anda. Diperlukan hanya ketika Anda mengonfigurasi TLS timbal balik dan Anda menentukan ARN sertifikat CA yang diimpor atau pribadi ACM untuk ARN. `CertificateArn`

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [OwnershipVerificationCertificateArn](#) dari sumber daya `AWS::ApiGateway::DomainName`.

Route53

Menentukan konfigurasi Amazon Route 53.

Tipe: [Route53Configuration](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

SecurityPolicy

Versi TLS ditambah paket sandi untuk nama domain ini.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [SecurityPolicy](#) dari sumber daya `AWS::ApiGateway::DomainName`.

Contoh-contoh

DomainName

DomainName contoh

YAML

```
Domain:
```

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

Route53Configuration

Mengonfigurasi set catatan Route53 untuk API.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

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

Properti

DistributionDomainName

Mengonfigurasi distribusi kustom nama domain kustom API.

Tipe: String

Wajib: Tidak

Default: Gunakan distribusi API Gateway.

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [DNSName](#) dari sumber daya `AWS::Route53::RecordSetGroup` `AliasTarget`.

Catatan tambahan: Nama domain [CloudFrontdistribusi](#).

EvaluateTargetHealth

Kapan EvaluateTargetHealth benar, catatan alias mewarisi kesehatan AWS sumber daya yang direferensikan, seperti penyeimbang beban Elastic Load Balancing atau catatan lain di zona yang dihosting.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [EvaluateTargetHealth](#) dari sumber daya AWS::Route53::RecordSetGroup AliasTarget.

Catatan tambahan: Anda tidak dapat mengatur EvaluateTargetHealth ke true ketika target alias adalah CloudFront distribusi.

HostedZoneId

ID zona yang di-hosting tempat Anda ingin membuat catatan.

Tentukan HostedZoneName atau HostedZoneId, tapi tidak keduanya. Jika Anda memiliki beberapa zona yang di-hosting dengan nama domain yang sama, Anda harus menentukan zona yang di-hosting menggunakan HostedZoneId.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [HostedZoneId](#) dari sumber daya AWS::Route53::RecordSetGroup RecordSet.

HostedZoneName

Nama zona yang di-hosting tempat ingin Anda membuat catatan.

Tentukan HostedZoneName atau HostedZoneId, jangan keduanya. Jika Anda memiliki beberapa zona yang di-hosting dengan nama domain yang sama, Anda harus menentukan zona yang di-hosting menggunakan HostedZoneId.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [HostedZoneName](#) dari sumber daya AWS::Route53::RecordSetGroup RecordSet.

IPv6

Saat properti ini disetel, AWS SAM buat AWS::Route53::RecordSet sumber daya dan set [Type](#) AAAA untuk yang disediakan HostedZone.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Region

Hanya set catatan sumber daya berbasis latensi: Wilayah Amazon EC2 tempat Anda membuat sumber daya yang direferensikan set catatan sumber daya ini. Sumber daya biasanya adalah sumber daya AWS, seperti instans EC2 atau penyeimbang beban ELB, dan direferensikan oleh alamat IP atau nama domain DNS, bergantung pada jenis catatan.

Saat Amazon Route 53 menerima kueri DNS untuk nama domain dan jenis yang telah Anda buat set catatan sumber daya latensinya, Route 53 memilih set catatan sumber daya latensi yang memiliki latensi terendah antara pengguna akhir dan Wilayah Amazon EC2 terkait. Route 53 kemudian mengembalikan nilai yang terkait dengan set catatan sumber daya yang dipilih.

Perhatikan hal berikut:

- Anda hanya dapat menentukan satu ResourceRecord per set catatan sumber daya latensi.
- Anda hanya dapat membuat satu set catatan sumber daya latensi untuk setiap Wilayah Amazon EC2.
- Anda tidak diharuskan membuat set catatan sumber daya latensi untuk semua Wilayah Amazon EC2. Route 53 akan memilih wilayah dengan latensi terbaik dari antara wilayah yang Anda buat set catatan sumber daya latensi.
- Anda tidak dapat membuat set catatan sumber daya non-latensi yang memiliki nilai yang sama untuk elemen Name dan Type sebagai set catatan sumber daya latensi.

Tipe: String

Wajib: Tidak

AWS CloudFormationkompatibilitas: Properti ini diteruskan langsung ke [Region](#) properti tipe `AWS::Route53::RecordSetGroup RecordSet` data.

SetIdentifier

Set catatan sumber daya yang memiliki kebijakan perutean selain sederhana: Pengenal yang membedakan antara beberapa set catatan sumber daya yang memiliki kombinasi nama dan jenis yang sama, seperti beberapa set catatan sumber daya tertimbang bernama `acme.example.com` yang memiliki tipe A. Dalam grup set catatan sumber daya yang memiliki nama dan tipe yang sama, nilai `SetIdentifier` harus unik untuk setiap set catatan sumber daya.

Untuk informasi tentang kebijakan perutean, lihat [Memilih kebijakan perutean di Panduan Pengembang Amazon Route 53](#).

Tipe: String

Wajib: Tidak

AWS CloudFormationkompatibilitas: Properti ini diteruskan langsung ke [SetIdentifier](#) properti tipe `AWS::Route53::RecordSetGroup RecordSet` data.

Contoh-contoh

Contoh basic

Dalam contoh ini, kami mengonfigurasi domain kustom dan set rekaman Route 53 untuk API kami.

YAML

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



```
HostedZoneId: ABCDEFGHIJKLMNOP
```

EndpointConfiguration

Tipe titik akhir dari REST API.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Type: String  
VPCendpointIds: List
```

Properti

Type

Tipe titik akhir dari REST API.

Nilai yang valid: EDGE atau REGIONAL atau PRIVATE

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Types](#) dari tipe data `AWS::ApiGateway::RestApi EndpointConfiguration`.

VPCendpointIds

Daftar ID VPC endpoint dari API REST yang akan dibuatkan alias Route53.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [VpcEndpointIds](#) dari tipe data `AWS::ApiGateway::RestApi EndpointConfiguration`.

Contoh-contoh

EndpointConfiguration

Contoh Konfigurasi Titik Akhir

YAML

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

AWS::Serverless::Application

Menyematkan aplikasi nirserver dari [AWS Serverless Application Repository](#) atau dari bucket Amazon S3 sebagai aplikasi yang di-nest. Aplikasi bersarang digunakan sebagai [AWS::CloudFormation::Stack](#) sumber daya bersarang, yang dapat berisi beberapa sumber daya lain termasuk sumber daya lainnya. [AWS::Serverless::Application](#)

Note

Ketika Anda menyebarkan ke AWS CloudFormation, AWS SAM mengubah AWS SAM sumber daya Anda menjadi AWS CloudFormation sumber daya. Untuk informasi selengkapnya, lihat [Sumber daya AWS CloudFormation yang dibuat](#).

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Type: AWS::Serverless::Application
Properties:
  Location: String | ApplicationLocationObject
  NotificationARNs: List
```

```
Parameters: Map  
Tags: Map  
TimeoutInMinutes: Integer
```

Properti

Location

URL templat, jalur file, atau lokasi objek dari aplikasi yang di-nest.

Jika URL template disediakan, itu harus mengikuti format yang ditentukan dalam [CloudFormation TemplateUrl dokumentasi](#) dan berisi template yang valid CloudFormation atau SAM. [ApplicationLocationObject](#) dapat digunakan untuk menentukan aplikasi yang telah diterbitkan ke [AWS Serverless Application Repository](#).

Jika jalur file lokal tersedia, templat harus melalui alur kerja yang mencakup perintah `sam deploy` atau `sam package`, agar aplikasi dapat diubah dengan benar.

Jenis: String | [ApplicationLocationObject](#)

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini mirip dengan properti [TemplateURL](#) dari sumber daya `AWS::CloudFormation::Stack`. CloudFormation Versi tidak mengambil [ApplicationLocationObject](#) untuk mengambil aplikasi dari file. AWS Serverless Application Repository

NotificationARNs

Daftar topik Amazon SNS yang ada tempat notifikasi tentang peristiwa tumpukan dikirim.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [NotificationARNs](#) dari sumber daya `AWS::CloudFormation::Stack`.

Parameters

Nilai parameter aplikasi.

Tipe: Peta

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Parameters](#) dari sumber daya `AWS::CloudFormation::Stack`.

Tags

Sebuah peta (string ke string) yang menentukan tanda yang akan ditambahkan ke aplikasi ini. Kunci dan nilai terbatas pada karakter alfanumerik. Kunci dapat berisi 1 hingga 127 karakter Unicode dan tidak boleh diawali dengan `aws:`. Nilai dapat berisi 1 hingga 255 karakter Unicode.

Tipe: Peta

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini mirip dengan properti [Tags](#) dari sumber daya `AWS::CloudFormation::Stack`. Properti Tags di SAM terdiri dari pasangan Key:Value; di CloudFormation dalamnya terdiri dari daftar objek Tag. Ketika tumpukan dibuat, SAM secara otomatis akan menambahkan tanda `lambda:createdBy:SAM` untuk aplikasi ini. Selain itu, jika aplikasi ini berasal dari AWS Serverless Application Repository, SAM juga akan secara otomatis menambahkan dua tanda tambahan `serverlessrepo:applicationId:ApplicationId` dan `serverlessrepo:semanticVersion:SemanticVersion`.

TimeoutInMinutes

Lama waktu, dalam hitungan menit, AWS CloudFormation menunggu tumpukan yang di-nest untuk mencapai status `CREATE_COMPLETE`. Default tidak memiliki batas waktu. Saat AWS CloudFormation mendeteksi bahwa tumpukan yang di-nest telah mencapai status `CREATE_COMPLETE`, itu menandai sumber daya tumpukan yang di-nest sebagai `CREATE_COMPLETE` di tumpukan induk dan mengulang membuat tumpukan induk. Jika periode waktu habis berakhir sebelum tumpukan yang di-nest mencapai `CREATE_COMPLETE`, AWS CloudFormation menandai tumpukan yang di-nest sebagai gagal dan memutar kembali tumpukan yang di-nest dan tumpukan induk.

Tipe: Integer

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [TimeoutInMinutes](#) dari sumber daya `AWS::CloudFormation::Stack`.

Nilai Pengembalian

Ref

Ketika ID logis dari sumber daya ini disediakan untuk fungsi intrinsik Ref, ia mengembalikan nama sumber daya dari sumber daya `AWS::CloudFormation::Stack` utama.

Untuk informasi lebih lanjut tentang penggunaan fungsi Ref, lihat [Ref](#) di Panduan Pengguna AWS CloudFormation.

Fn::GetAtt

`Fn::GetAtt` mengembalikan nilai untuk atribut yang ditentukan dari jenis ini. Berikut ini adalah atribut yang tersedia dan nilai-nilai kembali sampel.

Untuk informasi lebih lanjut tentang menggunakan `Fn::GetAtt`, lihat [Fn::GetAtt](#) di Panduan Pengguna AWS CloudFormation.

`Outputs.ApplicationOutputName`

Nilai output tumpukan dengan nama *ApplicationOutputName*.

Contoh-contoh

Aplikasi SAR

Aplikasi yang menggunakan templat dari Serverless Application Repository

YAML

```
Type: AWS::Serverless::Application
Properties:
  Location:
    ApplicationId: 'arn:aws:serverlessrepo:us-east-1:012345678901:applications/my-
application'
    SemanticVersion: 1.0.0
  Parameters:
    StringParameter: parameter-value
    IntegerParameter: 2
```

Aplikasi-Normal

Aplikasi dari url S3

YAML

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

ApplicationLocationObject

Aplikasi yang telah diterbitkan ke [AWS Serverless Application Repository](#).

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

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

Properti

ApplicationId

Amazon Resource Name (ARN) dari aplikasi.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

SemanticVersion

Versi semantik aplikasi.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

my-application

Contoh objek lokasi aplikasi

YAML

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

AWS::Serverless::Connector

Mengkonfigurasi izin antara dua sumber daya. Untuk pengenalan konektor, lihat [Mengelola izin sumber daya dengan konektor AWS SAM](#).

Untuk informasi selengkapnya tentang AWS CloudFormation sumber daya yang dihasilkan, lihat [AWS CloudFormation sumber daya yang dihasilkan saat Anda menentukan AWS::Serverless::Connector](#).

Untuk memberikan umpan balik tentang konektor, [kirimkan masalah baru](#) di serverless-application-model AWS GitHub repositori.

Note

Ketika Anda menyebarkan ke AWS CloudFormation, AWS SAM mengubah AWS SAM sumber daya Anda menjadi AWS CloudFormation sumber daya. Untuk informasi selengkapnya, lihat [Sumber daya AWS CloudFormation yang dibuat](#).

Sintaks

Untuk mendeklarasikan entitas ini dalam template AWS Serverless Application Model (AWS SAM) Anda, gunakan salah satu sintaks berikut.

Note

Sebaiknya gunakan sintaks konektor tertanam untuk sebagian besar kasus penggunaan. Tertanam dalam sumber daya membuatnya lebih mudah untuk membaca dan memelihara dari waktu ke waktu. Bila Anda perlu mereferensikan sumber daya sumber yang tidak berada

dalam AWS SAM template yang sama, seperti sumber daya dalam tumpukan bersarang atau sumber daya bersama, gunakan `AWS::Serverless::Connector` sintaks.

Konektor tertanam

```
<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
```

Properti

Destination

Sumber daya tujuan.

Jenis: [ResourceReference](#) | Daftar [ResourceReference](#)

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Permissions

Jenis izin yang diizinkan untuk dilakukan sumber daya sumber daya pada sumber daya tujuan.

`Read` termasuk tindakan AWS Identity and Access Management (IAM) yang memungkinkan membaca data dari sumber daya.

`Write` termasuk tindakan IAM yang memungkinkan memulai dan menulis data ke sumber daya.

Nilai yang valid: Read atau Write

Tipe: Daftar

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Source

Sumber sumber daya. Diperlukan saat menggunakan `AWS::Serverless::Connector` sintaks.

Jenis: [ResourceReference](#)

Wajib: Bersyarat

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

SourceReference

Sumber sumber daya.

Note

Gunakan dengan sintaks konektor tertanam saat mendefinisikan properti tambahan untuk sumber daya sumber.

Jenis: [SourceReference](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

Konektor tertanam

Contoh berikut menggunakan konektor tertanam untuk menentukan koneksi Write data antara AWS Lambda fungsi dan tabel Amazon DynamoDB:

```

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

```

Contoh berikut menggunakan konektor tertanam untuk menentukan Read dan Write izin:

```

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
    ...

```

Contoh berikut menggunakan konektor tertanam untuk menentukan sumber daya sumber dengan properti selainId:

```

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

Contoh berikut menggunakan [AWS::Serverless::Connector](#) resource untuk memiliki AWS Lambda fungsi read from, dan write ke tabel Amazon DynamoDB:

```

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

```

Contoh berikut menggunakan [AWS::Serverless::Connector](#) sumber daya agar fungsi Lambda menulis ke topik Amazon SNS, dengan kedua sumber daya dalam templat yang sama:

```

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

```

- Write

Contoh berikut menggunakan [AWS::Serverless::Connector](#) sumber daya agar topik Amazon SNS menulis ke fungsi Lambda, yang kemudian menulis ke tabel Amazon DynamoDB, dengan semua sumber daya dalam templat yang sama:

```
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

```

Berikut ini adalah AWS CloudFormation template yang diubah dari contoh di atas:

```

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

```

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

ResourceReference

Referensi ke sumber daya yang digunakan jenis [AWS::Serverless::Connector](#) sumber daya.

Note

Untuk sumber daya dalam template yang sama, berikan `Id`. Untuk sumber daya yang tidak dalam template yang sama, gunakan kombinasi properti lainnya. Untuk informasi selengkapnya, lihat [AWS SAM referensi konektor](#).

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

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

Properti**Arn**

ARN sumber daya.

Jenis: Tali

Diperlukan: Bersyarat

AWS CloudFormation kompatibilitas: Properti ini unik bagi AWS SAM dan tidak memiliki AWS CloudFormation padanan.

Id

[ID logis](#) dari sumber daya dalam template yang sama.

Note

Ketika `Id` ditentukan, jika konektor menghasilkan AWS Identity and Access Management (IAM) kebijakan, peran IAM yang terkait dengan kebijakan tersebut akan disimpulkan dari sumber daya `Id`. Ketika `Id` tidak ditentukan, menyediakan `RoleName` sumber daya untuk konektor untuk melampirkan kebijakan IAM yang dihasilkan untuk peran IAM.

Jenis: Tali

Diperlukan: Bersyarat

AWS CloudFormation kompatibilitas: Properti ini unik bagi AWS SAM dan tidak memiliki AWS CloudFormation padanan.

Name

Nama sumber daya.

Jenis: Tali

Diperlukan: Bersyarat

AWS CloudFormation kompatibilitas: Properti ini unik bagi AWS SAM dan tidak memiliki AWS CloudFormation padanan.

Qualifier

Kualifikasi untuk sumber daya yang mempersempit ruang lingkungannya. `Qualifier` menggantikan* nilai pada akhir kendala sumber daya ARN. Sebagai contoh, lihat [API Gateway memanggil fungsi Lambda](#).

Note

Definisi kualifikasi bervariasi per jenis sumber daya. Untuk daftar jenis sumber daya dan tujuan yang didukung, lihat [AWS SAM referensi konektor](#).

Jenis: Tali

Diperlukan: Bersyarat

AWS CloudFormationkompatibilitas: Properti ini unik bagiAWS SAM dan tidak memilikiAWS CloudFormation padanan.

QueueUrl

URL Antrean Amazon SQS. Properti ini hanya berlaku untuk sumber daya Amazon SQS.

Jenis: Tali

Diperlukan: Bersyarat

AWS CloudFormationkompatibilitas: Properti ini unik bagiAWS SAM dan tidak memilikiAWS CloudFormation padanan.

ResourceId

ID sumber daya. Misalnya, API Gateway API ID.

Jenis: Tali

Diperlukan: Bersyarat

AWS CloudFormationkompatibilitas: Properti ini unik bagiAWS SAM dan tidak memilikiAWS CloudFormation padanan.

RoleName

Nama peran yang terkait dengan sumber daya.

Note

KetikaId ditentukan, jika konektor menghasilkan kebijakan IAM, peran IAM yang terkait dengan kebijakan tersebut akan disimpulkan dari sumber dayaId. Ketika tidakId ditentukan, menyediakanRoleName sumber daya untuk konektor untuk melampirkan kebijakan IAM yang dihasilkan untuk peran IAM.

Jenis: Tali

Diperlukan: Bersyarat

AWS CloudFormationkompatibilitas: Properti ini unik bagiAWS SAM dan tidak memilikiAWS CloudFormation padanan.

Type

AWS CloudFormation Jenis sumber daya. Untuk informasi selengkapnya, kunjungi [referensi jenis AWS sumber daya dan properti](#).

Jenis: Tali

Diperlukan: Bersyarat

AWS CloudFormation kompatibilitas: Properti ini unik bagi AWS SAM dan tidak memiliki AWS CloudFormation padanan.

Contoh

API Gateway memanggil fungsi Lambda

Contoh berikut menggunakan [AWS::Serverless::Connector](#) sumber daya untuk memungkinkan Amazon API Gateway untuk memanggil AWS Lambda fungsi.

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

Referensi ke sumber daya sumber daya yang menggunakan jenis [AWS::Serverless::Connector](#) sumber daya.

Sintaksis

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Qualifier: String
```

Properti

Qualifier

Kualifikasi untuk sumber daya yang mempersempit ruang lingkungannya. *Qualifier* menggantikan* nilai pada akhir kendala sumber daya ARN.

Note

Definisi kualifikasi bervariasi per jenis sumber daya. Untuk daftar jenis sumber Daya yang didukung, lihat [AWS SAM referensi konektor](#).

Tipe: String

Wajib: Bersyarat

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh

Contoh berikut menggunakan konektor tertanam untuk menentukan sumber daya sumber dengan properti selain **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

Membuat AWS Lambda fungsi, peran eksekusi AWS Identity and Access Management (IAM), dan pemetaan sumber peristiwa yang memicu fungsi.

[AWS::Serverless::Function](#) Resource juga mendukung atribut Metadata resource, sehingga Anda dapat menginstruksikan AWS SAM untuk membuat runtime kustom yang dibutuhkan aplikasi Anda. Untuk informasi selengkapnya tentang membangun waktu aktif kustom, lihat [Membangun waktu aktif kustom](#).

Note

Ketika Anda menyebarkan ke AWS CloudFormation, AWS SAM mengubah AWS SAM sumber daya Anda menjadi AWS CloudFormation sumber daya. Untuk informasi selengkapnya, lihat [Sumber daya AWS CloudFormation yang dibuat](#).

Sintaks

Untuk mendeklarasikan entitas ini dalam template AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Type: AWS::Serverless::Function
Properties:
  Architectures: List
  AssumeRolePolicyDocument: JSON
  AutoPublishAlias: String
  AutoPublishAliasAllProperties: Boolean
  AutoPublishCodeSha256: String
  CodeSigningConfigArn: String
  CodeUri: String | FunctionCode
  DeadLetterQueue: Map | DeadLetterQueue
  DeploymentPreference: DeploymentPreference
  Description: String
  Environment: Environment
  EphemeralStorage: EphemeralStorage
  EventInvokeConfig: EventInvokeConfiguration
  Events: EventSource
  FileSystemConfigs: List
  FunctionName: String
  FunctionUrlConfig: FunctionUrlConfig
  Handler: String
  ImageConfig: ImageConfig
  ImageUri: String
  InlineCode: String
  KmsKeyArn: String
  Layers: List
  LoggingConfig: LoggingConfig
  MemorySize: Integer
  PackageType: String
```

```
PermissionsBoundary: String  
Policies: String | List | Map  
PropagateTags: Boolean  
ProvisionedConcurrencyConfig: ProvisionedConcurrencyConfig  
ReservedConcurrentExecutions: Integer  
Role: String  
RolePath: String  
Runtime: String  
RuntimeManagementConfig: RuntimeManagementConfig  
SnapStart: SnapStart  
Tags: Map  
Timeout: Integer  
Tracing: String  
VersionDescription: String  
VpcConfig: VpcConfig
```

Properti

Architectures

Arsitektur set instruksi untuk fungsi tersebut.

Untuk informasi selengkapnya tentang properti ini, lihat [Arsitektur set instruksi Lambda di Panduan](#) Pengembang.AWS Lambda

Nilai yang valid: Salah satu x86_64 atau arm64

Tipe: Daftar

Wajib: Tidak

Default: x86_64

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [Architectures](#) properti AWS::Lambda::Function sumber daya.

AssumeRolePolicyDocument

Menambahkan AssumeRolePolicyDocument untuk default yang dibuat Role untuk fungsi ini. Jika properti ini tidak ditentukan, AWS SAM menambahkan peran asumsi default untuk fungsi ini.

Tipe: JSON

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini mirip dengan [AssumeRolePolicyDocument](#) properti `AWS::IAM::Role` sumber daya. AWS SAM menambahkan properti ini ke peran IAM yang dihasilkan untuk fungsi ini. Jika Amazon Resource Name (ARN) peran tersedia untuk fungsi ini, properti ini tidak melakukan apa-apa.

AutoPublishAlias

Nama alias Lambda. Untuk informasi selengkapnya tentang alias Lambda, lihat [Alias fungsi Lambda](#) di Panduan Developer AWS Lambda . Untuk contoh yang menggunakan properti ini, lihat [Men-deploy aplikasi nirserver secara bertahap](#).

AWS SAM menghasilkan [AWS::Lambda::Version](#) dan [AWS::Lambda::Alias](#) sumber daya saat properti ini disetel. Untuk informasi selengkapnya tentang skenario ini, lihat [AutoPublishAlias properti ditentukan](#). Untuk informasi umum tentang AWS CloudFormation sumber daya yang dihasilkan, lihat [Sumber daya AWS CloudFormation yang dibuat](#).

Tipe: String

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

AutoPublishAliasAllProperties

Menentukan ketika baru [AWS::Lambda::Version](#) dibuat. `Kapantrue`, versi Lambda baru dibuat ketika properti apa pun dalam fungsi Lambda diubah. `Kapanfalse`, versi Lambda baru dibuat hanya jika salah satu properti berikut dimodifikasi:

- `Environment`, `MemorySize`, atau `SnapStart`.
- Setiap perubahan yang menghasilkan pembaruan ke `Code` properti, seperti `CodeDict`, `ImageUri`, atau `InlineCode`.

Properti ini `AutoPublishAlias` harus didefinisikan.

Jika juga `AutoPublishSha256` ditentukan, perilakunya lebih diutamakan.

```
AutoPublishAliasAllProperties: true
```

Tipe: Boolean

Wajib: Tidak

Nilai default: `false`

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

AutoPublishCodeSha256

Saat digunakan, string ini bekerja dengan `CodeUri` nilai untuk menentukan apakah versi Lambda baru perlu dipublikasikan. Properti ini sering digunakan untuk mengatasi masalah penerapan berikut: Paket penerapan disimpan di lokasi Amazon S3 dan digantikan oleh paket penerapan baru dengan kode fungsi Lambda yang diperbarui tetapi properti tetap tidak berubah (sebagai lawan `CodeUri` dari paket penerapan baru yang diunggah ke lokasi Amazon S3 baru dan diubah ke lokasi baru). `CodeUri`

Masalah ini ditandai dengan AWS SAM template yang memiliki karakteristik sebagai berikut:

- `DeploymentPreferenceObjek` dikonfigurasi untuk penerapan bertahap (seperti yang dijelaskan dalam) [Men-deploy aplikasi nirserver secara bertahap](#)
- `AutoPublishAliasProperti` disetel dan tidak berubah di antara penerapan
- `CodeUriProperti` disetel dan tidak berubah di antara penerapan.

Dalam skenario ini, memperbarui `AutoPublishCodeSha256` menghasilkan versi Lambda baru yang berhasil dibuat. Namun, kode fungsi baru yang diterapkan ke Amazon S3 tidak akan dikenali. Untuk mengenali kode fungsi baru, pertimbangkan untuk menggunakan pembuatan versi di bucket Amazon S3 Anda. Tentukan `Version` properti untuk fungsi Lambda Anda dan konfigurasi bucket Anda agar selalu menggunakan paket penerapan terbaru.

Dalam skenario ini, agar berhasil memicu deployment bertahap, Anda harus memberikan nilai unik untuk `AutoPublishCodeSha256`.

Tipe: String

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

CodeSigningConfigArn

ARN [AWS::Lambda::CodeSigningConfig](#) sumber daya, digunakan untuk mengaktifkan penandatanganan kode untuk fungsi ini. Untuk informasi selengkapnya tentang penandatanganan kode, lihat [Mengonfigurasi penandatanganan kode untuk aplikasi AWS SAM](#).

Tipe: String


Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [CodeSigningConfigArn](#) properti `AWS::Lambda::Function` sumber daya.

CodeUri

Kode untuk fungsi. Nilai yang diterima meliputi:

- Fungsi Amazon S3 URI. Misalnya, `s3://bucket-123456789/sam-app/1234567890abcdefg`.
- Jalur lokal ke fungsi. Misalnya, `hello_world/`.
- Sebuah objek [FunctionCode](#).

 Note

Jika Anda menyediakan URI atau [FunctionCode](#) objek Amazon S3 fungsi, Anda harus mereferensikan paket penerapan [Lambda](#) yang valid.

Jika Anda menyediakan jalur file lokal, gunakan AWS SAMCLI untuk mengunggah file lokal saat penerapan. Untuk mempelajari selengkapnya, lihat [Menggunakan AWS SAMCLI untuk mengunggah file lokal saat penerapan](#).

Jika Anda menggunakan fungsi intrinsik di `CodeUri` properti, tidak AWS SAM akan dapat mengurai nilai dengan benar. Pertimbangkan untuk menggunakan [AWS::LanguageExtensions transform](#) sebagai gantinya.

Jenis: [String |[FunctionCode](#)]

Diperlukan: Bersyarat. Kapan `PackageType` diatur ke `Zip`, salah satu `CodeUri` atau `InlineCode` diperlukan.

AWS CloudFormation kompatibilitas: Properti ini mirip dengan [Code](#) properti `AWS::Lambda::Function` sumber daya. Properti Amazon S3 yang di-nest diberi nama berbeda.

DeadLetterQueue

Mengonfigurasi topik Amazon Simple Notification Service (Amazon SNS) atau antrean Amazon Simple Queue Service (Amazon SQS) tempat Lambda mengirimkan peristiwa yang tidak dapat diproses. Untuk informasi selengkapnya tentang fungsionalitas antrian huruf mati, lihat [Antrian huruf mati di Panduan Pengembang AWS Lambda](#)

Note

Jika sumber peristiwa fungsi Lambda Anda adalah antrean Amazon SQS, konfigurasi antrian huruf mati untuk antrean sumber, bukan untuk fungsi Lambda. Antrean surat mati yang dikonfigurasi untuk fungsi digunakan untuk antrean invokasi tidak sinkron fungsi, bukan untuk antrean sumber peristiwa.

Jenis: Peta | [DeadLetterQueue](#)

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini mirip dengan [DeadLetterConfig](#) properti `AWS::Lambda::Function` sumber daya. Dalam AWS CloudFormation jenis ini berasal dari `TargetArn`, sedangkan di AWS SAM Anda harus melewati tipe bersama dengan `TargetArn`.

DeploymentPreference

Pengaturan untuk mengaktifkan deployment Lambda bertahap.

Jika `DeploymentPreference` objek ditentukan, AWS SAM membuat [AWS::CodeDeploy::Application](#) dipanggil `ServerlessDeploymentApplication` (satu per tumpukan), [AWS::CodeDeploy::DeploymentGroup](#) dipanggil `<function-logical-id>DeploymentGroup`, dan [AWS::IAM::Role](#) dipanggil `CodeDeployServiceRole`.

Jenis: [DeploymentPreference](#)

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

Lihat juga: Untuk informasi selengkapnya tentang properti ini, lihat [Men-deploy aplikasi nirserver secara bertahap](#).

Description

Deskripsi fungsi.

Tipe: String

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [Description](#) properti `AWS::Lambda::Function` sumber daya.

Environment

Konfigurasi untuk lingkungan waktu aktif.

Tipe: [Lingkungan](#)

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [Environment](#) properti `AWS::Lambda::Function` sumber daya.

EphemeralStorage

Objek yang menentukan ruang disk, dalam MB, tersedia untuk fungsi Lambda Anda di `/tmp`

Untuk informasi selengkapnya tentang properti ini, lihat [lingkungan eksekusi Lambda](#) di Panduan AWS Lambda Pengembang.

Jenis: [EphemeralStorage](#)

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [EphemeralStorage](#) properti `AWS::Lambda::Function` sumber daya.

EventInvokeConfig

Objek yang menggambarkan peristiwa memanggil konfigurasi pada fungsi Lambda.

Jenis: [EventInvokeConfiguration](#)

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

Events

Menentukan peristiwa yang memicu fungsi ini. Peristiwa terdiri dari tipe dan satu set properti yang bergantung pada tipenya.

Jenis: [EventSource](#)

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

FileSystemConfigs

Daftar [FileSystemConfig](#) objek yang menentukan pengaturan koneksi untuk sistem file Amazon Elastic File System (Amazon EFS).

Jika template Anda berisi [AWS::EFS::MountTarget](#) sumber daya, Anda juga harus menentukan atribut DependsOn resource untuk memastikan bahwa target mount dibuat atau diperbarui sebelum fungsi.

Tipe: Daftar

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [FileSystemConfigs](#) properti `AWS::Lambda::Function` sumber daya.

FunctionName

Nama untuk fungsi. Jika Anda tidak menentukan nama, nama unik akan dibuat untuk Anda.

Tipe: String

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [FunctionName](#) properti `AWS::Lambda::Function` sumber daya.

FunctionUrlConfig

Objek yang menggambarkan URL fungsi. URL fungsi adalah titik akhir HTTPS yang dapat Anda gunakan untuk menjalankan fungsi Anda.

Untuk informasi selengkapnya, lihat [URL fungsi](#) di Panduan AWS Lambda Pengembang.

Jenis: [FunctionUrlConfig](#)

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

Handler

Fungsi dalam kode Anda yang dipanggil untuk memulai eksekusi. Properti ini hanya diperlukan jika properti `PackageType` diatur ke `Zip`.

Tipe: String

Wajib: Bersyarat

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [Handler](#) properti `AWS::Lambda::Function` sumber daya.

ImageConfig

Objek yang digunakan untuk mengonfigurasi pengaturan citra kontainer Lambda. Untuk informasi selengkapnya, lihat [Menggunakan citra kontainer dengan Lambda](#) di Panduan Developer AWS Lambda .

Jenis: [ImageConfig](#)

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [ImageConfig](#) properti `AWS::Lambda::Function` sumber daya.

ImageUri

URI repositori Amazon Elastic Container Registry (Amazon ECR) untuk citra kontainer fungsi Lambda ini. Properti ini hanya berlaku jika properti `PackageType` diatur ke `Image`, jika tidak akan diabaikan. Untuk informasi selengkapnya, lihat [Menggunakan citra kontainer dengan Lambda](#) di Panduan Developer AWS Lambda .

Note

Jika `PackageType` properti disetel ke `Image`, maka `ImageUri` diperlukan, atau Anda harus membangun aplikasi Anda dengan `Metadata` entri yang diperlukan dalam file AWS SAM template. Untuk informasi selengkapnya, lihat [Aplikasi Pembangunan](#).

Membangun aplikasi Anda dengan `Metadata` entri yang diperlukan lebih diutamakan `ImageUri`, jadi jika Anda menentukan keduanya maka `ImageUri` diabaikan.

Tipe: String

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [ImageUri](#) properti tipe `AWS::Lambda::Function` Code data.

InlineCode

Kode fungsi Lambda yang ditulis langsung dalam templat. Properti ini hanya berlaku jika properti `PackageType` diatur ke `Zip`, jika tidak akan diabaikan.

Note

Jika properti `PackageType` diatur ke `Zip` (default), salah satu `CodeUri` atau `InlineCode` harus ada.

Tipe: String

Wajib: Bersyarat

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [ZipFile](#) properti tipe `AWS::Lambda::Function` Code data.

KmsKeyArn

ARN dari kunci AWS Key Management Service (AWS KMS) yang digunakan Lambda untuk mengenkripsi dan mendekripsi variabel lingkungan fungsi Anda.

Tipe: String

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [KmsKeyArn](#) properti `AWS::Lambda::Function` sumber daya.

Layers

Daftar ARN `LayerVersion` yang harus digunakan fungsi ini. Urutan yang ditentukan di sini adalah urutan ketika urutan impor saat menjalankan fungsi Lambda.

Tipe: Daftar

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [Layers](#) properti `AWS::Lambda::Function` sumber daya.

LoggingConfig

Pengaturan konfigurasi Amazon CloudWatch Logs fungsi.

Jenis: [LoggingConfig](#)

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [LoggingConfig](#) properti `AWS::Lambda::Function` sumber daya.

MemorySize

Ukuran memori dalam MB yang dialokasikan per pemanggilan fungsi.

Tipe: Integer

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [MemorySize](#) properti `AWS::Lambda::Function` sumber daya.

PackageType

Tipe paket deployment fungsi Lambda. Untuk informasi lebih lanjut, lihat [Paket deployment Lambda](#) di Panduan Developer AWS Lambda .

Catatan:

1. Jika properti ini diatur ke Zip (default), `CodeUri` atau `InlineCode` berlaku, dan `ImageUri` diabaikan.
2. Jika properti ini diatur ke Image, hanya `ImageUri` yang berlaku, dan `CodeUri` dan `InlineCode` diabaikan. Repositori Amazon ECR yang diperlukan untuk menyimpan gambar kontainer fungsi dapat dibuat secara otomatis oleh file. AWS SAMCLI Untuk informasi selengkapnya, lihat [sam deploy](#).

Nilai yang valid: Zip atau Image

Tipe: String

Wajib: Tidak

Default: Zip

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [PackageType](#) properti `AWS::Lambda::Function` sumber daya.

PermissionsBoundary

ARN batas izin untuk digunakan untuk peran eksekusi fungsi ini. Properti ini bekerja hanya jika peran dibuat untuk Anda.

Tipe: String

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [PermissionsBoundary](#) properti `AWS::IAM::Role` sumber daya.

Policies

Kebijakan izin untuk fungsi ini. Kebijakan akan ditambahkan ke peran eksekusi default AWS Identity and Access Management (IAM) fungsi.

Properti ini menerima satu nilai atau daftar nilai. Nilai yang diizinkan meliputi:

- [AWS SAMtemplat kebijakan](#).
- Kebijakan ARN [AWS terkelola atau kebijakan](#) yang [dikelola pelanggan](#).
- Nama kebijakan AWS terkelola dari [daftar](#) berikut.
- [Kebijakan IAM sebaris](#) yang diformat YAML sebagai peta.

Note

Jika Anda menyetel `Role` properti, properti ini diabaikan.

Tipe: String | Daftar | Peta

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini mirip dengan [Policies](#) properti `AWS::IAM::Role` sumber daya.

PropagateTags

Tunjukkan apakah akan meneruskan tag dari Tags properti ke sumber daya yang Anda [AWS::Serverless::Function](#) hasilkan atau tidak. Tentukan `True` untuk menyebarkan tag di sumber daya yang Anda hasilkan.

Tipe: Boolean

Wajib: Tidak

Default: `False`

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

ProvisionedConcurrencyConfig

Konfigurasi konkurensi yang disediakan untuk alias fungsi.

Note

`ProvisionedConcurrencyConfig` dapat ditentukan hanya jika `AutoPublishAlias` diatur. Jika tidak, akan terjadi kesalahan.

Jenis: [ProvisionedConcurrencyConfig](#)

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [ProvisionedConcurrencyConfig](#) properti `AWS::Lambda::Alias` sumber daya.

ReservedConcurrentExecutions

Jumlah maksimum eksekusi bersamaan yang ingin Anda simpan untuk fungsi tersebut.

Untuk informasi lebih lanjut, lihat [Skala Fungsi Lambda](#) dalam Panduan Developer AWS Lambda .

Tipe: Integer

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [ReservedConcurrentExecutions](#) properti `AWS::Lambda::Function` sumber daya.

Role

ARN dari IAM role untuk digunakan sebagai peran eksekusi fungsi ini.

Tipe: String

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini mirip dengan [Role](#) properti `AWS::Lambda::Function` sumber daya. Ini diperlukan di AWS CloudFormation tetapi tidak di AWS SAM. Jika peran tidak ditentukan, satu dibuat untuk Anda dengan ID logis `<function-logical-id>Role`.

RolePath

Jalur ke peran eksekusi IAM fungsi.

Gunakan properti ini saat peran dibuat untuk Anda. Jangan gunakan saat peran ditentukan dengan `Role` properti.

Tipe: String

Wajib: Bersyarat

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [Path](#) properti `AWS::IAM::Role` sumber daya.

Runtime

Pengenal [waktu aktif](#) fungsi. Properti ini hanya diperlukan jika properti `PackageType` diatur ke `Zip`.

Note

Jika Anda menentukan `provided` identifier untuk properti ini, Anda dapat menggunakan atribut Metadata resource untuk menginstruksikan AWS SAM untuk membangun runtime kustom yang diperlukan fungsi ini. Untuk informasi selengkapnya tentang membangun waktu aktif kustom, lihat [Membangun waktu aktif kustom](#).

Tipe: String

Wajib: Bersyarat

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [Runtime](#) properti `AWS::Lambda::Function` sumber daya.

RuntimeManagementConfig

Konfigurasi opsi manajemen runtime untuk fungsi Lambda Anda seperti pembaruan lingkungan runtime, perilaku rollback, dan memilih versi runtime tertentu. Untuk mempelajari selengkapnya, lihat [Pembaruan runtime Lambda di Panduan Pengembang AWS Lambda](#)

Jenis: [RuntimeManagementConfig](#)

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [RuntimeManagementConfig](#) properti `AWS::Lambda::Function` sumber daya.

SnapStart

Buat snapshot dari versi fungsi Lambda baru. Snapshot adalah status cache dari fungsi inisialisasi Anda, termasuk semua dependensinya. Fungsi ini diinisialisasi hanya sekali dan status cache digunakan kembali untuk semua pemanggilan future, meningkatkan kinerja aplikasi dengan mengurangi berapa kali fungsi Anda harus diinisialisasi. Untuk mempelajari lebih lanjut, lihat [Meningkatkan kinerja startup dengan Lambda SnapStart di Panduan AWS Lambda Pengembang](#).

Jenis: [SnapStart](#)

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [SnapStart](#) properti `AWS::Lambda::Function` sumber daya.

Tags

Sebuah peta (string ke string) yang menentukan tanda ditambahkan ke fungsi ini. Untuk detail tentang kunci dan nilai yang valid untuk tag, lihat [Kunci Tag dan Persyaratan Nilai](#) di Panduan AWS Lambda Pengembang.

Saat tumpukan dibuat, AWS SAM secara otomatis menambahkan `lambda:createdBy: SAM` tag ke fungsi Lambda ini, dan ke peran default yang dihasilkan untuk fungsi ini.

Tipe: Peta

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini mirip dengan [Tags](#) properti `AWS::Lambda::Function` sumber daya. `Tags` Properti di AWS SAM terdiri dari pasangan kunci-nilai (sedangkan dalam properti AWS CloudFormation ini terdiri dari daftar Tag objek). Selain itu, AWS SAM secara otomatis menambahkan `lambda: createdBy: SAM` tag ke fungsi Lambda ini, dan ke peran default yang dihasilkan untuk fungsi ini.

Timeout

Waktu maksimum dalam detik bahwa fungsi dapat berjalan sebelum dihentikan.

Tipe: Integer

Wajib: Tidak

Default: 3

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [Timeout](#) properti `AWS::Lambda::Function` sumber daya.

Tracing

String yang menentukan mode pelacakan X-Ray fungsi.

- `Active`- Mengaktifkan penelusuran X-Ray untuk fungsi tersebut.
- `Disabled`- Menonaktifkan X-Ray untuk fungsi tersebut.
- `PassThrough`- Mengaktifkan penelusuran X-Ray untuk fungsi tersebut. Keputusan pengambilan sampel didelegasikan ke layanan hilir.

Jika ditentukan sebagai `Active` atau `PassThrough` dan `Role` properti tidak disetel, AWS SAM tambahkan `arn:aws:iam::aws:policy/AWSXrayWriteOnlyAccess` kebijakan ke peran eksekusi Lambda yang dibuatnya untuk Anda.

Untuk informasi selengkapnya tentang X-Ray, lihat [Menggunakan AWS Lambda dengan AWS X-Ray](#) di Panduan AWS Lambda Pengembang.

Nilai yang valid: `[Active|Disabled|PassThrough]`

Tipe: String

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini mirip dengan [TracingConfig](#) properti `AWS::Lambda::Function` sumber daya.

VersionDescription

Menentukan bidang `Description` yang ditambahkan pada sumber daya versi Lambda baru.

Tipe: String

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [Description](#) properti `AWS::Lambda::Version` sumber daya.

VpcConfig

Konfigurasi yang memungkinkan fungsi ini mengakses sumber daya privat dalam virtual private cloud (VPC) Anda.

Jenis: [VpcConfig](#)

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [VpcConfig](#) properti `AWS::Lambda::Function` sumber daya.

Nilai Pengembalian

Ref

Ketika ID logis dari sumber daya ini disediakan untuk fungsi intrinsik `Ref`, ID mengembalikan nama sumber daya dari fungsi Lambda utama.

Untuk informasi lebih lanjut tentang menggunakan fungsi `Ref`, lihat [Ref](#) diPanduan Pengguna AWS CloudFormation .

Fn::GetAtt

`Fn::GetAtt` mengembalikan nilai untuk atribut yang ditentukan dari jenis ini. Berikut ini adalah atribut yang tersedia dan nilai-nilai kembali sampel.

Untuk informasi lebih lanjut tentang menggunakan `Fn::GetAtt`, lihat [Fn::GetAtt](#) di Panduan Pengguna AWS CloudFormation .

Arn

ARN fungsi Lambda utama.

Contoh-contoh

Fungsi sederhana

Berikut ini adalah contoh dasar sumber daya [AWS::Serverless::Function](#) dari tipe paket Zip (default) dan kode fungsi dalam bucket Amazon S3.

YAML

```
Type: AWS::Serverless::Function
Properties:
  Handler: index.handler
  Runtime: python3.9
  CodeUri: s3://bucket-name/key-name
```

Contoh properti fungsi

Berikut ini adalah contoh-contoh [AWS::Serverless::Function](#) dari tipe paket Zip (default) yang menggunakan `InlineCode`, `Layers`, `Tracing`, `Policies`, Amazon EFS, dan sumber peristiwa `Api`.

YAML

```
Type: AWS::Serverless::Function
DependsOn: MyMountTarget      # This is needed if an AWS::EFS::MountTarget resource
                               is declared for EFS
Properties:
  Handler: index.handler
  Runtime: python3.9
  InlineCode: |
    def handler(event, context):
      print("Hello, world!")
  ReservedConcurrentExecutions: 30
  Layers:
    - Ref: MyLayer
```

```

Tracing: Active
Timeout: 120
FileSystemConfigs:
  - Arn: !Ref MyEfsFileSystem
    LocalMountPath: /mnt/EFS
Policies:
  - AWSLambdaExecute
  - Version: '2012-10-17'
    Statement:
      - Effect: Allow
        Action:
          - s3:GetObject
          - s3:GetObjectACL
        Resource: 'arn:aws:s3:::my-bucket/*'
Events:
  ApiEvent:
    Type: Api
    Properties:
      Path: /path
      Method: get

```

ImageConfigcontoh

Berikut ini adalah contoh-contoh ImageConfig untuk fungsi Lambda dari tipe paket 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 contoh

Fungsi Lambda yang dikonfigurasi untuk memperbarui lingkungan runtime-nya sesuai dengan perilaku saat ini:


```

TestFunction
  Type: AWS::Serverless::Function
  Properties:
    ...
    Runtime: python3.9
    RuntimeManagementConfig:
      UpdateRuntimeOn: Auto

```

Fungsi Lambda yang dikonfigurasi untuk memperbarui lingkungan runtime saat fungsi diperbarui:

```

TestFunction
  Type: AWS::Serverless::Function
  Properties:
    ...
    Runtime: python3.9
    RuntimeManagementConfig:
      UpdateRuntimeOn: FunctionUpdate

```

Fungsi Lambda yang dikonfigurasi untuk memperbarui lingkungan runtime-nya secara manual:

```

TestFunction
  Type: AWS::Serverless::Function
  Properties:
    ...
    Runtime: python3.9
    RuntimeManagementConfig:
      RuntimeVersionArn: arn:aws:lambda:us-
east-1::runtime:4c459dd0104ee29ec65dcad056c0b3ddb20d6db76b265ade7eda9a066859b1e
      UpdateRuntimeOn: Manual

```

SnapStartcontoh

Contoh fungsi Lambda dengan SnapStart dihidupkan untuk versi masa depan:

```

TestFunc
  Type: AWS::Serverless::Function
  Properties:
    ...
    SnapStart:
      ApplyOn: PublishedVersions

```

DeadLetterQueue

Menentukan antrian SQS atau SNS topik yang dikirim peristiwa oleh AWS Lambda (Lambda) ketika tidak dapat memprosesnya. Untuk informasi selengkapnya tentang fungsionalitas antrian surat mati, lihat [Antrian huruf mati di Panduan Pengembang](#). AWS Lambda

SAM akan secara otomatis menambahkan izin yang sesuai ke peran eksekusi fungsi Lambda Anda untuk memberikan layanan Lambda akses ke sumber daya. `sqs: SendMessage` akan ditambahkan untuk antrian SQS dan `SNS: Publish` untuk topik SNS.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
TargetArn: String  
Type: String
```

Properti

TargetArn

Amazon Resource Name (ARN) antrian Amazon SQS atau topik Amazon SNS.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [TargetArn](#) dari tipe data `AWS::Lambda::Function DeadLetterConfig`.

Type

Tipe antrian surat mati.

Nilai yang valid: SNS, SQS

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

DeadLetterQueue

Contoh Antrean Surat Mati untuk topik SNS.

YAML

```
DeadLetterQueue:  
  Type: SNS  
  TargetArn: arn:aws:sns:us-east-2:123456789012:my-topic
```

DeploymentPreference

Menentukan konfigurasi untuk mengaktifkan deployment Lambda bertahap. Untuk informasi selengkapnya tentang konfigurasi deployment Lambda secara bertahap, lihat [Men-deploy aplikasi nirserver secara bertahap](#).

Note

Anda harus menentukan `AutoPublishAlias` dalam Anda [AWS::Serverless::Function](#) untuk menggunakan `DeploymentPreference` objek, jika tidak kesalahan akan terjadi.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Alarms: List  
Enabled: Boolean  
Hooks: Hooks  
PassthroughCondition: Boolean  
Role: String
```

[TriggerConfigurations](#): *List*

[Type](#): *String*

Properti

Alarms

Daftar CloudWatch alarm yang ingin Anda picu oleh kesalahan apa pun yang muncul oleh penerapan.

Properti ini menerima fungsi intrinsik Fn : : If. Lihat bagian Contoh di bagian bawah topik ini untuk contoh templat yang menggunakan Fn : : If.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Enabled

Apakah preferensi deployment ini diaktifkan.

Tipe: Boolean

Wajib: Tidak

Default: BETUL

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Hooks

Validasi fungsi Lambda yang dijalankan sebelum dan sesudah pergeseran lalu lintas.

Tipe: [Hook](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

PassthroughCondition

Jika Benar, dan jika preferensi penerapan ini diaktifkan, Kondisi fungsi akan diteruskan ke CodeDeploy sumber daya yang dihasilkan. Umumnya, Anda harus mengatur ini ke True. Jika tidak, CodeDeploy sumber daya akan dibuat bahkan jika Kondisi fungsi diselesaikan menjadi False.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Role

Peran IAM ARN CodeDeploy yang akan digunakan untuk pergeseran lalu lintas. IAM role tidak akan dibuat jika ini disediakan.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

TriggerConfigurations

Daftar konfigurasi pemicu yang ingin Anda kaitkan dengan grup deployment. Digunakan untuk memberitahu topik SNS pada peristiwa siklus hidup.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [TriggerConfigurations](#) dari sumber daya AWS::CodeDeploy::DeploymentGroup.

Type

Ada dua kategori tipe deployment pada saat ini: Linear dan Canary. Untuk informasi selengkapnya tentang tipe deployment yang tersedia, lihat [Men-deploy aplikasi nirserver secara bertahap](#).

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

DeploymentPreference dengan kait sebelum dan sesudah lalu lintas.

Contoh preferensi deployment yang berisi kait pra dan pascalalulintas.

YAML

```

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

```

DeploymentPreference dengan Fn: :Jika fungsi intrinsik

Contoh preferensi deployment yang menggunakan Fn: :If untuk mengonfigurasi alarm. Dalam contoh ini, Alarm1 akan dikonfigurasi jika MyCondition adalah true, dan Alarm2 dan Alarm5 akan dikonfigurasi jika MyCondition adalah false.

YAML

```

DeploymentPreference:
  Enabled: true
  Type: Canary10Percent10Minutes
  Alarms:
    Fn::If:
      - MyCondition
      - - Alarm1
        - Alarm2

```

- Alarm5

Hooks

Validasi fungsi Lambda yang dijalankan sebelum dan sesudah pergeseran lalu lintas.

Note

Fungsi Lambda yang direferensikan dalam properti ini mengkonfigurasi `CodeDeployLambdaAliasUpdate` objek sumber daya yang dihasilkan.

[AWS::Lambda::Alias](#) Untuk informasi selengkapnya, lihat [CodeDeployLambdaAliasUpdate Kebijakan](#) di Panduan AWS CloudFormation Pengguna.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
PostTraffic: String  
PreTraffic: String
```

Properti

PostTraffic

Fungsi Lambda yang dijalankan setelah pergeseran lalu lintas.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

PreTraffic

Fungsi Lambda yang dijalankan sebelum pergeseran lalu lintas.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

Kait

Contoh fungsi kait

YAML

```
Hooks:
  PreTraffic:
    Ref: PreTrafficLambdaFunction
  PostTraffic:
    Ref: PostTrafficLambdaFunction
```

EventInvokeConfiguration

Opsi konfigurasi untuk pemanggilan alias atau versi Lambda [tidak sinkron](#).

Sintaksis

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
DestinationConfig: EventInvokeDestinationConfiguration
MaximumEventAgeInSeconds: Integer
MaximumRetryAttempts: Integer
```

Properti

`DestinationConfig`

Objek konfigurasi yang menentukan tujuan dari peristiwa setelah Lambda memprosesnya.

Jenis: [EventInvokeDestinationConfiguration](#)

Wajib: Tidak

AWS CloudFormation kesesuaian: Properti ini mirip dengan [DestinationConfig](#) properti dari `AWS::Lambda::EventInvokeConfig` sumber daya. SAM membutuhkan parameter tambahan, "Type", yang tidak ada di CloudFormation.

MaximumEventAgeInSeconds

Masa maksimum permintaan yang dikirimkan Lambda ke fungsi untuk diproses.

Jenis: Bulat

Wajib: Tidak

AWS CloudFormation kesesuaian: Properti ini diloloskan langsung ke [MaximumEventAgeInSeconds](#) properti dari `AWS::Lambda::EventInvokeConfig` sumber daya.

MaximumRetryAttempts

Jumlah waktu maksimum untuk mencoba kembali saat fungsi mengembalikan kesalahan.

Jenis: Bulat

Wajib: Tidak

AWS CloudFormation kesesuaian: Properti ini diloloskan langsung ke [MaximumRetryAttempts](#) properti dari `AWS::Lambda::EventInvokeConfig` sumber daya.

Contoh

MaximumEventAgeInSeconds

Contoh MaximumEventAgeInSeconds

YAML

```
EventInvokeConfig:
  MaximumEventAgeInSeconds: 60
  MaximumRetryAttempts: 2
  DestinationConfig:
    OnSuccess:
      Type: SQS
      Destination: arn:aws:sqs:us-west-2:012345678901:my-queue
```

```
OnFailure:  
  Type: Lambda  
  Destination: !GetAtt DestinationLambda.Arn
```

EventInvokeDestinationConfiguration

Objek konfigurasi yang menentukan tujuan dari peristiwa setelah Lambda memprosesnya.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
OnFailure: OnFailure  
OnSuccess: OnSuccess
```

Properti

OnFailure

Tujuan untuk peristiwa yang gagal diproses.

Jenis: [OnFailure](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini mirip dengan properti [OnFailure](#) dari sumber daya AWS::Lambda::EventInvokeConfig. Membutuhkan Type, properti hanya-SAM tambahan.

OnSuccess

Tujuan untuk peristiwa yang berhasil diproses.

Jenis: [OnSuccess](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini mirip dengan properti [OnSuccess](#) dari sumber daya AWS::Lambda::EventInvokeConfig. Membutuhkan Type, properti hanya-SAM tambahan.

Contoh-contoh

OnSuccess

OnSuccess contoh

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SQS
      Destination: arn:aws:sqs:us-west-2:012345678901:my-queue
    OnFailure:
      Type: Lambda
      Destination: !GetAtt DestinationLambda.Arn
```

OnFailure

Tujuan untuk peristiwa yang gagal diproses.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Destination: String
Type: String
```

Properti

Destination

Amazon Resource Name (ARN) dari sumber daya tujuan.

Tipe: String

Wajib: Bersyarat

Kompatibilitas AWS CloudFormation: Properti ini mirip dengan properti [OnFailure](#) dari sumber daya AWS::Lambda::EventInvokeConfig. SAM akan menambahkan izin yang diperlukan

untuk membuat IAM role yang dibuat secara otomatis terkait dengan fungsi ini untuk mengakses sumber daya yang dirujuk dalam properti ini.

Catatan tambahan: Jika jenisnya adalah Lambda/EventBridge, Tujuan diperlukan.

Type

Tipe sumber daya yang dirujuk di tujuan. Tipe yang didukung adalah SQS, SNS, Lambda, dan EventBridge.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Catatan tambahan: Jika tipe SQS/SNS dan properti `Destination` dibiarkan kosong, sumber daya SQS/SNS otomatis dibuat oleh SAM. Untuk referensi sumber daya, gunakan `<function-logical-id>.DestinationQueue` untuk SQS atau `<function-logical-id>.DestinationTopic` untuk SNS. Jika jenisnya adalah Lambda/EventBridge, `Destination` diperlukan.

Contoh-contoh

EventInvoke Contoh Konfigurasi dengan tujuan SQS dan Lambda

Dalam contoh ini tidak ada `Destination` yang diberikan untuk `OnSuccess` konfigurasi SQS, sehingga SAM secara implisit membuat antrian SQS dan menambahkan izin yang diperlukan. Juga untuk contoh ini, Tujuan untuk sumber daya Lambda yang dideklarasikan dalam file template ditentukan dalam `OnFailure` konfigurasi, jadi SAM menambahkan izin yang diperlukan ke fungsi Lambda ini untuk memanggil fungsi Lambda tujuan.

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SQS
    OnFailure:
      Type: Lambda
```

```
Destination: !GetAtt DestinationLambda.Arn # Arn of a Lambda function declared
in the template file.
```

EventInvoke Contoh Konfigurasi dengan tujuan SNS

Dalam contoh ini Tujuan diberikan untuk topik SNS yang dideklarasikan dalam file template untuk OnSuccess konfigurasi.

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SNS
      Destination:
        Ref: DestinationSNS # Arn of an SNS topic declared in the tempate file
```

OnSuccess

Tujuan untuk peristiwa yang berhasil diproses.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Destination: String
Type: String
```

Properti

Destination

Amazon Resource Name (ARN) dari sumber daya tujuan.

Tipe: String

Wajib: Bersyarat

Kompatibilitas AWS CloudFormation: Properti ini mirip dengan properti [OnSuccess](#) dari sumber daya AWS::Lambda::EventInvokeConfig. SAM akan menambahkan izin yang diperlukan

untuk membuat IAM role yang dibuat secara otomatis terkait dengan fungsi ini untuk mengakses sumber daya yang dirujuk dalam properti ini.

Catatan tambahan: Jika jenisnya adalah Lambda/EventBridge, Tujuan diperlukan.

Type

Tipe sumber daya yang dirujuk di tujuan. Tipe yang didukung adalah SQS, SNS, Lambda, dan EventBridge.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Catatan tambahan: Jika tipe SQS/SNS dan properti `Destination` dibiarkan kosong, sumber daya SQS/SNS otomatis dibuat oleh SAM. Untuk referensi sumber daya, gunakan `<function-logical-id>.DestinationQueue` untuk SQS atau `<function-logical-id>.DestinationTopic` untuk SNS. Jika jenisnya adalah Lambda/EventBridge, `Destination` diperlukan.

Contoh-contoh

EventInvoke Contoh Konfigurasi dengan tujuan SQS dan Lambda

Dalam contoh ini tidak ada `Destination` yang diberikan untuk `OnSuccess` konfigurasi SQS, sehingga SAM secara implisit membuat antrian SQS dan menambahkan izin yang diperlukan. Juga untuk contoh ini, Tujuan untuk sumber daya Lambda yang dideklarasikan dalam file template ditentukan dalam `OnFailure` konfigurasi, jadi SAM menambahkan izin yang diperlukan ke fungsi Lambda ini untuk memanggil fungsi Lambda tujuan.

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SQS
    OnFailure:
      Type: Lambda
```

```
Destination: !GetAtt DestinationLambda.Arn # Arn of a Lambda function declared
in the template file.
```

EventInvoke Contoh Konfigurasi dengan tujuan SNS

Dalam contoh ini Tujuan diberikan untuk topik SNS yang dideklarasikan dalam file template untuk OnSuccess konfigurasi.

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SNS
      Destination:
        Ref: DestinationSNS # Arn of an SNS topic declared in the tempate file
```

EventSource

Objek yang menggambarkan sumber peristiwa yang memicu fungsi. Setiap peristiwa terdiri dari tipe dan satu set properti yang bergantung pada tipe itu. Untuk informasi lebih lanjut tentang properti dari setiap sumber peristiwa, lihat topik yang sesuai dengan tipe tersebut.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Properties: AlexaSkill | Api | CloudWatchEvent | CloudWatchLogs | Cognito
| DocumentDB | DynamoDB | EventBridgeRule | HttpApi | IoTRule | Kinesis | MQ | MSK
| S3 | Schedule | ScheduleV2 | SelfManagedKafka | SNS | SQS
Type: String
```

Properti

Properties

Objek menggambarkan sifat pemetaan peristiwa ini. Set properti harus sesuai dengan tipe yang ditentukan.

Jenis : [AlexaSkill](#) | [Api](#) | [Cognito](#) | [CloudWatchEvent](#) | [CloudWatchLogs](#) | [DocumentDB](#) | [DynamoDB](#) | [EventBridge](#) | [Rule](#) | [Trule](#) | [Kinesis](#) | [MQ](#) | [MSK](#) | [S3](#) | [Jadwal HttpApi](#) | [Schedulev2](#) | [SNS](#) | [SQS](#) | [SelfManagedKafka](#)

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Type

Tipe peristiwa.

Nilai yang valid:

[AlexaSkill](#) | [Api](#) | [CloudWatchEvent](#) | [CloudWatchLogs](#) | [Cognito](#) | [DocumentDB](#) | [DynamoDB](#) | [EventBridge](#) | [SQS](#)

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

ApiEvent

Contoh menggunakan peristiwa API

YAML

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

AlexaSkill

Objek yang menggambarkan tipe sumber peristiwa AlexaSkill.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
SkillId: String
```

Properti

SkillId

ID Keterampilan Alexa untuk Keterampilan Alexa Anda. Untuk informasi selengkapnya tentang ID Keterampilan lihat [Konfigurasi pemicu untuk fungsi Lambda](#) dalam dokumentasi Kit Keterampilan Alexa.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

AlexaSkillTrigger

Contoh Peristiwa Keterampilan Alexa

YAML

```
AlexaSkillEvent:  
  Type: AlexaSkill
```

Api

Objek yang menggambarkan tipe sumber peristiwa Api. Jika sumber daya [AWS::Serverless::Api](#) didefinisikan, nilai jalur dan metode harus sesuai dengan operasi dalam ketentuan OpenAPI API.

Jika `AWS::Serverless::Api` tidak ditentukan, input dan output fungsi adalah representasi dari permintaan HTTP dan respons HTTP.

Misalnya, menggunakan JavaScript API, kode status dan isi respons dapat dikontrol dengan mengembalikan objek dengan kunci `StatusCode` dan `body`.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Auth: ApiFunctionAuth
Method: String
Path: String
RequestModel: RequestModel
RequestParameters: List of [ String | RequestParameter ]
RestApiId: String
TimeoutInMillis: Integer
```

Properti

Auth

Konfigurasi auth untuk Api+Path+Metode tertentu ini.

Berguna untuk membatalkan config auth pengaturan `DefaultAuthorizer` API pada jalur individu ketika `DefaultAuthorizer` tidak ditentukan atau membatalkan pengaturan `ApiKeyRequired` default.

Jenis: [ApiFunctionAuth](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Method

Metode HTTP yang membuat fungsi ini dipanggil.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Path

Jalur Uri yang membuat fungsi ini dipanggil. Harus dimulai dengan /.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

RequestModel

Permintaan model yang akan digunakan untuk Api+Path+Metode tertentu ini. Ini harus merujuk nama model yang ditentukan dalam bagian `Models` dari sumber daya [AWS::Serverless::Api](#).

Jenis: [RequestModel](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

RequestParameters

Konfigurasi parameter permintaan untuk Api+Path+Metode tertentu ini. Semua nama parameter harus dimulai dengan `method.request` dan harus terbatas pada `method.request.header`, `method.request.querystring`, atau `method.request.path`.

Daftar dapat berisi string nama parameter dan [RequestParameter](#) objek. Untuk string, Caching properti `Required` akan default ke `false`.

Jenis: Daftar [String | [RequestParameter](#)]

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

RestApiId

Pengidentifikasi RestApi sumber daya, yang harus berisi operasi dengan jalur dan metode yang diberikan. Biasanya, ini diatur untuk merujuk sumber daya [AWS::Serverless::Api](#) yang ditentukan dalam templat ini.

Jika Anda tidak menentukan properti ini, AWS SAM membuat sumber daya [AWS::Serverless::Api](#) default menggunakan sebuah dokumen OpenApi yang dihasilkan. Sumber daya tersebut berisi penyatuan dari semua jalur dan metode yang ditentukan oleh peristiwa Api dalam templat yang sama yang tidak menentukan RestApiId.

Ini tidak dapat merujuk sumber daya [AWS::Serverless::Api](#) yang ditentukan dalam templat lain.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

TimeoutInMillis

Waktu habis khusus antara 50 dan 29.000 milidetik.

Note

Saat Anda menentukan properti ini, AWS SAM memodifikasi definisi OpenAPI Anda. Definisi OpenAPI harus ditentukan sebaris menggunakan properti. `DefinitionBody`

Tipe: Integer

Wajib: Tidak

Default: 29.000 milidetik atau 29 detik

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

Contoh basic

YAML

```
Events:
  ApiEvent:
    Type: Api
    Properties:
      Path: /path
      Method: get
      RequestParameters:
        - method.request.header.Authorization
        - method.request.querystring.keyword:
            Required: true
            Caching: false
```

ApiFunctionAuth

Mengonfigurasi otorisasi di tingkat peristiwa, untuk API, jalur, dan metode tertentu.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
ApiKeyRequired: Boolean
AuthorizationScopes: List
Authorizer: String
InvokeRole: String
OverrideApiAuth: Boolean
ResourcePolicy: ResourcePolicyStatement
```

Properti

ApiKeyRequired

Memerlukan kunci API untuk API, jalur, dan metode ini.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

AuthorizationScopes

Cakupan otorisasi yang akan diterapkan ke API, path, dan metode ini.

Cakupan yang Anda tentukan akan membatalkan setiap cakupan yang diterapkan oleh properti `DefaultAuthorizer` jika Anda telah menentukannya.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Authorizer

`Authorizer` Untuk fungsi tertentu.

Jika Anda memiliki otorisasi global yang ditentukan untuk `AWS::Serverless::Api` sumber daya Anda, Anda dapat mengganti otorisasi dengan menyetelnya. `Authorizer NONE` Lihat contohnya di [Ganti otorisasi global untuk REST API Amazon API Gateway](#).

Note

Jika Anda menggunakan `DefinitionBody` properti `AWS::Serverless::Api` sumber daya untuk mendeskripsikan API Anda, Anda harus menggunakan `OverrideApiAuth` with `Authorizer` untuk mengganti otorisasi global Anda. Untuk informasi selengkapnya, lihat [OverrideApiAuth](#).

Nilai valid: `AWS_IAM,NONE`, atau ID logis untuk otorisasi apa pun yang ditentukan dalam AWS SAM template Anda.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

InvokeRole

Menentukan InvokeRole untuk digunakan untuk otorisasi AWS_IAM.

Tipe: String

Wajib: Tidak

Default: CALLER_CREDENTIALS

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Catatan tambahan: CALLER_CREDENTIALS memetakan ke `arn:aws:iam::*:user/*`, yang menggunakan kredensial pemanggil untuk memanggil titik akhir.

OverrideApiAuth

Tentukan `true` untuk mengganti konfigurasi otorisasi global sumber daya `AWS::Serverless::Api`. Properti ini hanya diperlukan jika Anda menentukan otorisasi global dan menggunakan `DefinitionBody` properti `AWS::Serverless::Api` sumber daya untuk mendeskripsikan API Anda.

Note

Saat Anda menentukan `OverrideApiAuth` sebagai `true`, AWS SAM akan mengganti otorisasi global Anda dengan nilai apa pun yang disediakan untuk `ApiKeyRequired`, `Authorizer`, atau `ResourcePolicy`. Oleh karena itu, setidaknya satu dari properti ini juga harus ditentukan saat menggunakan `OverrideApiAuth`. Lihat contohnya di [Ganti otorisasi global saat DefinitionBody for ditentukan AWS::Serverless::Api](#).

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

ResourcePolicy

Mengonfigurasi Kebijakan Sumber Daya untuk jalur ini pada API.

Jenis: [ResourcePolicyStatement](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

Function-Auth

Contoh berikut menentukan otorisasi pada tingkat fungsi.

YAML

```
Auth:
  ApiKeyRequired: true
  Authorizer: NONE
```

Ganti otorisasi global untuk REST API Amazon API Gateway

Anda dapat menentukan otorisasi global untuk `AWS::Serverless::Api` sumber daya Anda. Berikut ini adalah contoh yang mengonfigurasi otorisasi default global:

```
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
```


Untuk mengganti authorizer default untuk AWS Lambda fungsi Anda, Anda dapat menentukan Authorizer sebagai NONE Berikut ini adalah contohnya:

```

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
  
```

Ganti otorisasi global saat DefinitionBody for ditentukan AWS::Serverless::Api

Saat menggunakan DefinitionBody properti untuk mendeskripsikan AWS::Serverless::Api sumber daya Anda, metode penggantian sebelumnya tidak berfungsi. Berikut ini adalah contoh penggunaan DefinitionBody properti untuk sumber AWS::Serverless::Api daya:

```

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

```

Untuk mengganti otorisasi global, gunakan properti. `OverrideApiAuth` Berikut ini adalah contoh yang digunakan `OverrideApiAuth` untuk mengganti otorisasi global dengan nilai yang disediakan untuk: `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

Mengonfigurasi kebijakan sumber daya untuk semua metode dan jalur API. Untuk informasi selengkapnya tentang kebijakan sumber daya, lihat [Mengendalikan akses ke API dengan kebijakan sumber daya API Gateway](#) di Panduan Developer API Gateway.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

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

Properti

AwsAccountBlacklist

Akun AWS yang akan diblokir.

Jenis: Daftar String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

AwsAccountWhitelist

Akun AWS yang akan diizinkan. Untuk contoh penggunaan properti ini, lihat bagian Contoh di bagian bawah halaman ini.

Jenis: Daftar String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

CustomStatements

Daftar pernyataan kebijakan sumber daya kustom untuk diterapkan ke API ini. Untuk contoh penggunaan properti ini, lihat bagian Contoh di bagian bawah halaman ini.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

IntrinsicVpcBlacklist

Daftar virtual private cloud (VPC) yang akan diblokir, dengan setiap VPC ditetapkan sebagai referensi seperti [referensi dinamis](#) atau [fungsi intrinsik](#) Ref. Untuk contoh penggunaan properti ini, lihat bagian Contoh di bagian bawah halaman ini.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

IntrinsicVpcWhitelist

Daftar VPC yang akan diizinkan, dengan setiap VPC ditetapkan sebagai referensi seperti [referensi dinamis](#) atau [fungsi intrinsik](#) Ref.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

IntrinsicVpceBlacklist

Daftar VPC endpoint yang akan diblokir, dengan setiap VPC endpoint ditetapkan sebagai referensi seperti [referensi dinamis](#) atau [fungsi intrinsik](#) Ref.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

IntrinsicVpceWhitelist

Daftar VPC endpoint yang akan diizinkan, dengan setiap VPC endpoint ditetapkan sebagai referensi seperti [referensi dinamis](#) atau [fungsi intrinsik](#) Ref. Untuk contoh penggunaan properti ini, lihat bagian Contoh di bagian bawah halaman ini.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

IpRangeBlacklist

Alamat IP atau jangkauan alamat yang akan diblokir. Untuk contoh penggunaan properti ini, lihat bagian Contoh di bagian bawah halaman ini.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

IpRangeWhitelist

Alamat IP atau jangkauan alamat yang akan diizinkan.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

SourceVpcBlacklist

Sumber VPC atau VPC endpoint yang akan diblokir. Nama VPC sumber harus dimulai dengan "vpc-" dan nama VPC endpoint sumber harus dimulai dengan "vpce-". Untuk contoh penggunaan properti ini, lihat bagian Contoh di bagian bawah halaman ini.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

SourceVpcWhitelist

VPC sumber atau VPC endpoint yang akan diizinkan. Nama VPC sumber harus dimulai dengan "vpc-" dan nama VPC endpoint sumber harus dimulai dengan "vpce-".

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

Contoh Kebijakan Sumber Daya

Contoh berikut blok dua alamat IP dan sumber VPC, dan mengizinkan akun AWS.

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

Mengkonfigurasi Model Permintaan untuk Metode Api+Path+tertentu.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```

Model: String
Required: Boolean
ValidateBody: Boolean
ValidateParameters: Boolean

```

Properti

Model

Nama model yang ditentukan dalam properti Model dari [AWS::Serverless::Api](#).

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Required

Menambahkan `required` properti di bagian parameter OpenApi definisi untuk titik akhir API yang diberikan.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

ValidateBody

Menentukan apakah API Gateway menggunakan Model untuk memvalidasi badan permintaan. Untuk informasi selengkapnya, lihat [Mengaktifkan validasi permintaan di API Gateway](#) di Panduan Pengembang API Gateway.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

ValidateParameters

Menentukan apakah API Gateway menggunakan Model untuk memvalidasi parameter jalur permintaan, string kueri, dan header. Untuk informasi selengkapnya, lihat [Mengaktifkan validasi permintaan di API Gateway](#) di Panduan Pengembang API Gateway.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

Model Permintaan

Contoh Permintaan Model

YAML

```
RequestModel:  
  Model: User  
  Required: true  
  ValidateBody: true  
  ValidateParameters: true
```

RequestParameter

Mengonfigurasi Parameter Permintaan untuk Api+Path+Metode tertentu.

Properti `Required` atau `Caching` perlu ditentukan untuk parameter permintaan

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Caching: Boolean  
Required: Boolean
```

Properti

Caching

Menambahkan `cacheKeyParameters` bagian ke `OpenApi` definisi API Gateway

Tipe: Boolean

Wajib: Bersyarat

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Required

Bidang ini menentukan apakah parameter diperlukan

Tipe: Boolean

Wajib: Bersyarat

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

Parameter permintaan

Contoh Parameter Permintaan pengaturan

YAML

```
RequestParameters:
  - method.request.header.Authorization:
      Required: true
      Caching: true
```

CloudWatchEvent

Objek yang menggambarkan tipe sumber peristiwa CloudWatchEvent.

AWS Serverless Application Model(AWS SAM) menghasilkan [AWS::Events::Rules](#) sumber daya saat jenis acara ini disetel.

Catatan penting: [EventBridgeRule](#) adalah tipe sumber peristiwa yang diinginkan yang akan digunakan, bukan CloudWatchEvent. EventBridgeRule dan CloudWatchEvent menggunakan layanan, API, dan sumber daya AWS CloudFormation utama yang sama. Namun, AWS SAM akan menambahkan dukungan untuk fitur baru hanya untuk EventBridgeRule.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Enabled: Boolean  
EventBusName: String  
Input: String  
InputPath: String  
Pattern: EventPattern  
State: String
```

Properti

Enabled

Menunjukkan apakah aturan diaktifkan.

Untuk menonaktifkan aturan, tetapkan properti ini ke `false`.

Note

Tentukan salah satu `Enabled` atau `State` properti, tetapi tidak keduanya.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini mirip dengan properti [State](#) dari sumber daya `AWS::Events::Rule`. Jika properti ini ditetapkan ke `true` maka AWS SAM meneruskan `ENABLED`, jika tidak, properti ini meneruskan `DISABLED`.

EventBusName

Bus peristiwa yang akan dihubungkan dengan aturan ini. Jika Anda menghilangkan properti ini, AWS SAM menggunakan bus peristiwa default.

Tipe: String

Wajib: Tidak

Default: Bus peristiwa default

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [EventBusName](#) dari sumber daya `AWS::Events::Rule`.

Input

Teks JSON yang valid yang dilewatkan ke target. Jika Anda menggunakan properti ini, tidak ada dari teks peristiwa itu sendiri yang diteruskan ke target.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Input](#) dari sumber daya `AWS::Events::Rule` Target.

InputPath

Bila Anda tidak ingin meneruskan seluruh peristiwa yang sesuai ke target, gunakan properti `InputPath` untuk menggambarkan bagian mana dari peristiwa yang akan diteruskan.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [InputPath](#) dari sumber daya `AWS::Events::Rule` Target.

Pattern

Menjelaskan peristiwa yang dirutekan ke target yang ditentukan. Untuk informasi selengkapnya, lihat [Peristiwa dan Pola Peristiwa EventBridge di Panduan EventBridge Pengguna Amazon](#).

Jenis: [EventPattern](#)

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [EventPattern](#) dari sumber daya `AWS::Events::Rule`.

State

Keadaan aturan.

Nilai yang diterima: DISABLED | ENABLED

Note

Tentukan salah satu Enabled atau State properti, tetapi tidak keduanya.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [State](#) dari sumber daya `AWS::Events::Rule`.

Contoh-contoh

CloudWatchEvent

Berikut adalah contoh peristiwa dari tipe sumber peristiwa CloudWatchEvent.

YAML

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

CloudWatchLogs

Objek yang menggambarkan tipe sumber peristiwa CloudWatchLogs.

Acara ini menghasilkan [AWS::Logs::SubscriptionFilter](#) sumber daya dan menentukan filter langganan dan mengaitkannya dengan grup log yang ditentukan.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
FilterPattern: String
LogGroupName: String
```

Properti

FilterPattern

Ekspresi pemfilteran yang membatasi apa yang akan dikirim ke sumber daya AWS tujuan. Untuk informasi selengkapnya tentang sintaks pola filter, lihat [Sintaks Filter dan Pola..](#)

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [FilterPattern](#) dari sumber daya `AWS::Logs::SubscriptionFilter`.

LogGroupName

Grup log yang dikaitkan dengan filter langganan. Semua peristiwa log yang diunggah ke grup log ini difilter dan dikirim ke sumber daya AWS jika pola filter cocok dengan log acara tersebut.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [LogGroupName](#) dari sumber daya `AWS::Logs::SubscriptionFilter`.

Contoh-contoh

Filter Langganan Cloudwatchlogs

Contoh filter Langganan Cloudwatchlogs

YAML

```
CWLog:
  Type: CloudWatchLogs
  Properties:
    LogGroupName:
      Ref: CloudWatchLambdaLogsGroup
    FilterPattern: My pattern
```

Cognito

Objek yang menggambarkan tipe sumber peristiwa Cognito.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Trigger: List  
UserPool: String
```

Properti

Trigger

Lambda memicu informasi konfigurasi untuk kolam pengguna baru.

Tipe: Daftar

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [LambdaConfig](#) dari sumber daya `AWS::Cognito::UserPool`.

UserPool

Referensi untuk UserPool didefinisikan dalam template yang sama

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

Peristiwa Cognito

Contoh Cognito

YAML

```
CognitoUserPoolPreSignup:
```

```
Type: Cognito
Properties:
  UserPool:
    Ref: MyCognitoUserPool
  Trigger: PreSignUp
```

DocumentDB

Objek yang menggambarkan tipe sumber peristiwa DocumentDB. Untuk informasi selengkapnya, lihat [MenggunakanAWS Lambda dengan Amazon DocumentDB](#) di PanduanAWS Lambda Pengembang.

Sintaksis

Untuk mendeklarasikan entitas ini diAWS SAM templat Anda, gunakan sintaks berikut.

YAML

```
BatchSize: Integer
Cluster: String
CollectionName: String
DatabaseName: String
Enabled: Boolean
FilterCriteria: FilterCriteria
FullDocument: String
MaximumBatchingWindowInSeconds: Integer
SecretsManagerKmsKeyId: String
SourceAccessConfigurations: List
StartingPosition: String
StartingPositionTimestamp: Double
```

Properti

BatchSize

Jumlah maksimum item yang akan diambil dalam satu batch.

Tipe: Integer

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini dilewatkan langsung ke properti [BatchSize](#) dari sumber daya AWS::Lambda::EventSourceMapping.

Cluster

Amazon DocumentDB.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [EventSourceArn](#) dari sumber daya AWS::Lambda::EventSourceMapping.

CollectionName

Nama koleksi untuk mengkonsumsi dalam basis. Jika Anda tidak menentukan koleksi, Lambda mengkonsumsi semua koleksi.

Tipe: String

Wajib: Tidak

AWS CloudFormationkompatibilitas: Properti ini dilewatkan langsung ke [CollectionName](#) properti dari tipeAWS::Lambda::EventSourceMappingDocumentDBEventSourceConfig data.

DatabaseName

Nama Amazon DocumentDB.

Tipe: String

Wajib: Ya

AWS CloudFormationkompatibilitas: Properti ini dilewatkan langsung ke [DatabaseName](#) properti dari tipeAWS::Lambda::EventSourceMappingDocumentDBEventSourceConfig data.

Enabled

Jika `true`, pemetaan sumber peristiwa aktif. Untuk menjeda polling dan pemanggilan, atur ke `false`.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini dilewatkan langsung ke properti [Enabled](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

FilterCriteria

Objek yang mendefinisikan kriteria yang menentukan apakah Lambda harus memproses suatu peristiwa. Untuk informasi selengkapnya, lihat [Pemfilteran peristiwa Lambda](#) di Panduan AWS Lambda Pengembang.

Jenis: [FilterCriteria](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini dilewatkan langsung ke properti [FilterCriteria](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

FullDocument

Menentukan apa yang dikirim Amazon DocumentDB ke aliran peristiwa Anda selama operasi pembaruan dokumen. Jika diatur ke `UpdateLookup`, Amazon DocumentDB mengirimkan delta yang menjelaskan perubahan, bersama dengan salinan seluruh dokumen. Jika tidak, Amazon DocumentDB hanya mengirimkan sebagian dokumen yang berisi perubahan.

Tipe: String

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini dilewatkan langsung ke [FullDocument](#) properti dari tipe `AWS::Lambda::EventSourceMappingDocumentDBEventSourceConfig` data.

MaximumBatchingWindowInSeconds

Jumlah waktu maksimum untuk mengumpulkan rekaman sebelum memanggil fungsi, dalam hitungan detik.

Tipe: Integer

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini dilewatkan langsung ke properti [MaximumBatchingWindowInSeconds](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

SecretsManagerKmsKeyId

ID kunci AWS Key Management Service (AWS KMS) dari kunci terkelola pelanggan dari AWS Secrets Manager. Diperlukan ketika Anda menggunakan kunci terkelola pelanggan dari Secrets Manager dengan peran eksekusi Lambda yang tidak termasuk `kms:Decrypt` izin.

Nilai properti ini adalah UUID. Sebagai contoh: `1abc23d4-567f-8ab9-cde0-1fab234c5d67`.

Tipe: String

Wajib: Bersyarat

AWS CloudFormation Kompatibilitas: Properti ini unik bagi AWS SAM dan tidak memiliki AWS CloudFormation padanan.

SourceAccessConfigurations

Array dari protokol autentikasi atau host virtual. Tentukan ini menggunakan tipe [SourceAccessConfigurations](#) data.

Untuk jenis sumber `DocumentDB` peristiwa, satu-satunya jenis konfigurasi yang valid adalah `BASIC_AUTH`.

- `BASIC_AUTH`—Rahasia Secrets Manager yang menyimpan kredensial broker Anda. Untuk tipe ini, kredensialnya harus dalam format berikut: `{"username": "your-username", "password": "your-password"}`. Hanya satu objek dari tipe `BASIC_AUTH` diizinkan.

Tipe: Daftar

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [SourceAccessConfigurations](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

StartingPosition

Posisi dalam pengaliran tempat untuk mulai membaca.

- `AT_TIMESTAMP`—Tentukan waktu untuk mulai membaca catatan.
- `LATEST`—Baca hanya catatan baru.
- `TRIM_HORIZON`—Proses semua catatan yang tersedia.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [StartingPosition](#) dari sumber daya AWS::Lambda::EventSourceMapping.

StartingPositionTimestamp

Waktu untuk mulai membaca, dalam detik waktu Unix. Tentukan `StartingPositionTimestamp` kapan `StartingPosition` ditentukan sebagai `AT_TIMESTAMP`.

Tipe: Ganda

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini dilewatkan langsung ke properti [StartingPositionTimestamp](#) dari sumber daya AWS::Lambda::EventSourceMapping.

Contoh

Amazon DocumentDB

```
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

Objek yang menggambarkan tipe sumber peristiwa DynamoDB. Untuk informasi lebih lanjut, lihat [Menggunakan AWS Lambda dengan Amazon DynamoDB](#) di Panduan Developer AWS Lambda.

AWS SAM menghasilkan sumber [AWS::Lambda::EventSourceMapping](#) daya saat jenis acara ini disetel.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
BatchSize: Integer  
BisectBatchOnFunctionError: Boolean  
DestinationConfig: DestinationConfig  
Enabled: Boolean  
FilterCriteria: FilterCriteria  
FunctionResponseTypes: List  
MaximumBatchingWindowInSeconds: Integer  
MaximumRecordAgeInSeconds: Integer  
MaximumRetryAttempts: Integer  
ParallelizationFactor: Integer  
StartingPosition: String  
StartingPositionTimestamp: Double  
Stream: String  
TumblingWindowInSeconds: Integer
```

Properti

BatchSize

Jumlah maksimum item yang akan diambil dalam satu batch.

Tipe: Integer

Wajib: Tidak

Default: 100

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [BatchSize](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

Minimal: 1

Maksimum: 1000

BisectBatchOnFunctionError

Jika fungsi mengembalikan kesalahan, bagi batch menjadi dua dan coba lagi.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [BisectBatchOnFunctionError](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

DestinationConfig

Antrean Amazon Simple Queue Service (Amazon SQS) atau tujuan topik Amazon Simple Notification Service (Amazon SNS) untuk catatan yang dibuang.

Jenis: [DestinationConfig](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [DestinationConfig](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

Enabled

Menonaktifkan pemetaan sumber peristiwa untuk menunda polling dan pemanggilan.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Enabled](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

FilterCriteria

Objek yang mendefinisikan kriteria untuk menentukan apakah Lambda harus memproses suatu peristiwa. Untuk informasi selengkapnya, lihat [pemfilteran AWS Lambda acara](#) di Panduan AWS Lambda Pengembang.

Jenis: [FilterCriteria](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [FilterCriteria](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

FunctionResponseTypes

Daftar tipe jawaban saat ini yang diterapkan ke pemetaan sumber peristiwa. Untuk informasi selengkapnya, lihat [Melaporkan kegagalan item batch](#) di Panduan Developer AWS Lambda.

Nilai yang valid: `ReportBatchItemFailures`

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [FunctionResponseTypes](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

MaximumBatchingWindowInSeconds

Jumlah waktu maksimum untuk mengumpulkan rekaman sebelum memanggil fungsi, dalam hitungan detik.

Tipe: Integer

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [MaximumBatchingWindowInSeconds](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

MaximumRecordAgeInSeconds

Periode permintaan maksimum yang dikirimkan Lambda ke fungsi untuk diproses.

Tipe: Integer

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [MaximumRecordAgeInSeconds](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

MaximumRetryAttempts

Jumlah waktu maksimum untuk mencoba kembali saat fungsi mengembalikan kesalahan.

Tipe: Integer

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [MaximumRetryAttempts](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

ParallelizationFactor

Jumlah batch yang akan diproses dari tiap serpihan secara bersamaan.

Tipe: Integer

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [ParallelizationFactor](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

StartingPosition

Posisi dalam pengaliran tempat untuk mulai membaca.

- `AT_TIMESTAMP`— Tentukan waktu untuk mulai membaca catatan.
- `LATEST`— Baca hanya catatan baru.
- `TRIM_HORIZON`— Memproses semua catatan yang tersedia.

Nilai yang valid: `AT_TIMESTAMP` | `LATEST` | `TRIM_HORIZON`

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [StartingPosition](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

StartingPositionTimestamp

Waktu untuk mulai membaca, dalam detik waktu Unix. Tentukan `StartingPositionTimestamp` kapan `StartingPosition` ditentukan sebagai `AT_TIMESTAMP`.

Tipe: Ganda

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [StartingPositionTimestamp](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

Stream

Amazon Resource Name (ARN) dari pengaliran DynamoDB.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [EventSourceArn](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

TumblingWindowInSeconds

Durasi dalam detik dari periode pemrosesan. Kisaran yang valid adalah 1 sampai 900 (15 menit).

Untuk informasi selengkapnya, lihat [Periode Jatuh](#) di Panduan Developer AWS Lambda.

Tipe: Integer

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [TumblingWindowInSeconds](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

Contoh-contoh

Sumber peristiwa DynamoDB untuk tabel DynamoDB yang ada

Sumber peristiwa DynamoDB untuk tabel DynamoDB yang sudah ada di akun AWS.

YAML

```
Events:
  DDBEvent:
    Type: DynamoDB
    Properties:
```

```

Stream: arn:aws:dynamodb:us-east-1:123456789012:table/TestTable/
stream/2016-08-11T21:21:33.291
StartingPosition: TRIM_HORIZON
BatchSize: 10
Enabled: false

```

Peristiwa DynamoDB untuk tabel DynamoDB yang dinyatakan dalam Templat

Peristiwa DynamoDB untuk tabel DynamoDB yang dinyatakan dalam file templat yang sama.

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

Objek yang menjelaskan jenis sumber EventBridgeRule peristiwa, yang menetapkan fungsi tanpa server Anda sebagai target aturan Amazon. EventBridge Untuk informasi selengkapnya, lihat [Apa itu Amazon EventBridge?](#) di Panduan EventBridge Pengguna Amazon.

AWS SAM menghasilkan sumber [AWS::Events::Rule](#) daya saat jenis acara ini disetel.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```

DeadLetterConfig: DeadLetterConfig
EventBusName: String
Input: String

```

```
InputPath: String  
InputTransformer: InputTransformer  
Pattern: EventPattern  
RetryPolicy: RetryPolicy  
RuleName: String  
State: String  
Target: Target
```

Properti

DeadLetterConfig

Konfigurasi antrian Amazon Simple Queue Service (Amazon SQS) EventBridge tempat pengiriman peristiwa setelah pemanggilan target gagal. Pemanggilan dapat gagal, misalnya, saat mengirim acara ke fungsi Lambda yang tidak ada, atau ketika tidak EventBridge memiliki izin yang cukup untuk memanggil fungsi Lambda. Untuk informasi selengkapnya, lihat [Kebijakan percobaan ulang acara dan menggunakan antrian huruf mati di Panduan Pengguna](#) Amazon. EventBridge

Note

Jenis [AWS::Serverless::Function](#) sumber daya memiliki tipe data yang serupa `DeadLetterQueue`, yang menangani kegagalan yang terjadi setelah pemanggilan fungsi Lambda target berhasil. Contoh tipe kegagalan ini termasuk Lambda throttling, atau kesalahan yang dikembalikan oleh fungsi target Lambda. Untuk informasi selengkapnya tentang `DeadLetterQueue` properti fungsi, lihat [Antrian huruf mati di Panduan Pengembang](#). AWS Lambda

Jenis: [DeadLetterConfig](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [DeadLetterConfig](#) dari tipe data `AWS::Events::Rule` Target. Versi AWS SAM properti ini termasuk subproperti tambahan, jika Anda ingin AWS SAM membuat antrian surat mati untuk Anda.

EventBusName

Bus peristiwa yang akan dihubungkan dengan aturan ini. Jika Anda menghilangkan properti ini, AWS SAM menggunakan bus peristiwa default.

Tipe: String

Wajib: Tidak

Default: Bus peristiwa default

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [EventBusName](#) dari sumber daya `AWS::Events::Rule`.

Input

Teks JSON yang valid yang dilewatkan ke target. Jika Anda menggunakan properti ini, tidak ada dari teks peristiwa itu sendiri yang diteruskan ke target.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Input](#) dari sumber daya `AWS::Events::Rule Target`.

InputPath

Bila Anda tidak ingin meneruskan seluruh peristiwa yang sesuai ke target, gunakan properti `InputPath` untuk menggambarkan bagian mana dari peristiwa yang akan diteruskan.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [InputPath](#) dari sumber daya `AWS::Events::Rule Target`.

InputTransformer

Pengaturan untuk memungkinkan Anda memberikan input kustom ke target berdasarkan data peristiwa tertentu. Anda dapat mengekstrak satu atau beberapa pasangan nilai kunci dari peristiwa dan kemudian menggunakan data tersebut untuk mengirim input yang disesuaikan ke target. Untuk informasi selengkapnya, lihat [Transformasi EventBridge input Amazon](#) di Panduan EventBridge Pengguna Amazon.

Jenis: [InputTransformer](#)

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [InputTransformer](#) properti tipe `AWS::Events::Rule` Target data.

Pattern

Menjelaskan peristiwa yang dirutekan ke target yang ditentukan. Untuk informasi selengkapnya, lihat [EventBridge peristiwa Amazon](#) dan [pola EventBridge acara](#) di Panduan EventBridge Pengguna Amazon.

Jenis: [EventPattern](#)

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [EventPattern](#) dari sumber daya `AWS::Events::Rule`.

RetryPolicy

Objek `RetryPolicy` yang menyertakan informasi tentang pengaturan kebijakan coba lagi. Untuk informasi selengkapnya, lihat [Kebijakan percobaan ulang acara dan menggunakan antrian huruf mati di Panduan Pengguna](#) Amazon. EventBridge

Jenis: [RetryPolicy](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [RetryPolicy](#) dari tipe data `AWS::Events::Rule` Target.

RuleName

Nama aturan.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Name](#) dari sumber daya `AWS::Events::Rule`.

State

Keadaan aturan.

Nilai yang diterima: DISABLED | ENABLED

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [State](#) dari sumber daya `AWS::Events::Rule`.

Target

Sumber AWS daya yang EventBridge dipanggil ketika aturan dipicu. Anda dapat menggunakan properti ini untuk menentukan ID logis dari target. Jika properti ini tidak ditentukan, AWS SAM menghasilkan ID logis dari target.

Tipe: [Target](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini mirip dengan properti [Targets](#) dari sumber daya `AWS::Events::Rule`. Versi AWS SAM properti ini hanya mengizinkan Anda untuk menentukan ID logis dari satu target.

Contoh-contoh

EventBridgeRule

Berikut adalah contoh tipe sumber peristiwa EventBridgeRule.

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

Objek yang digunakan untuk menentukan antrean Amazon Simple Queue Service (Amazon SQS) EventBridge tempat mengirimkan peristiwa setelah pemanggilan target gagal. Permintaan dapat gagal, misalnya, ketika mengirim peristiwa ke fungsi Lambda yang tidak ada, atau izin tidak cukup untuk memanggil fungsi Lambda. Untuk informasi selengkapnya, lihat [Kebijakan percobaan ulang acara dan menggunakan antrian huruf mati di Panduan Pengguna](#) Amazon. EventBridge

Note

Jenis [AWS::Serverless::Function](#) sumber daya memiliki tipe data yang serupa, `DeadLetterQueue` yang menangani kegagalan yang terjadi setelah pemanggilan fungsi Lambda target berhasil. Contoh tipe kegagalan ini termasuk Lambda throttling, atau kesalahan dikembalikan oleh fungsi target Lambda. Untuk informasi selengkapnya tentang `DeadLetterQueue` properti fungsi, lihat [Antrian huruf mati di Panduan Pengembang](#). AWS Lambda

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Arn: String
QueueLogicalId: String
Type: String
```

Properti

Arn

Amazon Resource Name (ARN) dari antrean Amazon SQS yang ditetapkan sebagai target antrean surat mati.

 Note

Tentukan Type properti atau Arn properti, tetapi tidak keduanya.


Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Arn](#) dari tipe data `AWS::Events::Rule DeadLetterConfig`.

QueueLogicalId

Nama kustom dari antrean surat mati yang AWS SAM buat jika Type ditentukan.

 Note

Jika Type properti tidak disetel, properti ini diabaikan.


Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Type

Tipe antrean. Bila properti ini diatur, AWS SAM secara otomatis membuat antrean surat mati dan melampirkan [kebijakan berbasis sumber daya](#) yang diperlukan untuk memberikan izin untuk memerintahkan sumber daya untuk mengirim peristiwa ke antrean.

 Note

Tentukan Type properti atau Arn properti, tetapi tidak keduanya.

Nilai yang valid: SQS

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

DeadLetterConfig

DeadLetterConfig

YAML

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

Target

Mengkonfigurasi AWS sumber daya yang EventBridge dipanggil ketika aturan dipicu.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Id: String
```

Properti

Id

ID logis dari target.

Nilai dari Id dapat mencakup karakter alfanumerik, titik (.), tanda hubung (-), dan garis bawah (_).

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Id](#) dari tipe data `AWS::Events::Rule Target`.

Contoh-contoh

Target

YAML

```
EBRule:
  Type: EventBridgeRule
  Properties:
    Target:
      Id: MyTarget
```

HttpApi

Objek yang menggambarkan sumber peristiwa dengan tipe HttpApi.

Jika OpenApi definisi untuk jalur dan metode yang ditentukan ada di API, SAM akan menambahkan bagian integrasi dan keamanan Lambda (jika ada) untuk Anda.

Jika tidak ada OpenApi definisi untuk jalur dan metode yang ditentukan di API, SAM akan membuat definisi ini untuk Anda.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
ApiId: String
Auth: HttpApiFunctionAuth
Method: String
Path: String
PayloadFormatVersion: String
RouteSettings: RouteSettings
TimeoutInMillis: Integer
```

Properti

ApiId

Pengenal dari sumber daya [AWS::Serverless::HttpApi](#) yang ditentukan dalam templat ini.

Jika tidak ditentukan, [AWS::Serverless::HttpApi](#) sumber daya default dibuat dengan `ServerlessHttpApi` menggunakan OpenApi dokumen yang dihasilkan yang berisi gabungan semua jalur dan metode yang ditentukan oleh peristiwa Api yang ditentukan dalam templat ini yang tidak menentukan `ApiId`.

Ini tidak dapat merujuk sumber daya [AWS::Serverless::HttpApi](#) yang ditentukan dalam templat lain.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Auth

Konfigurasi auth untuk `Api+Path+Metode` tertentu ini.

Berguna untuk membatalkan `DefaultAuthorizer` API atau mengatur auth config pada jalur individu ketika `DefaultAuthorizer` tidak ditentukan.

Jenis: [HttpApiFunctionAuth](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Method

Metode HTTP yang membuat fungsi ini dipanggil.

Jika `Path` dan `Method` tidak ditentukan, SAM akan membuat jalur API default yang merutekan setiap permintaan yang tidak memetakan ke titik akhir yang berbeda untuk fungsi Lambda ini. Hanya satu dari jalur default ini yang boleh ada per API.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Path

Jalur Uri yang membuat fungsi ini dipanggil. Harus dimulai dengan /.

Jika Path dan Method tidak ditentukan, SAM akan membuat jalur API default yang merutekan setiap permintaan yang tidak memetakan ke titik akhir yang berbeda untuk fungsi Lambda ini. Hanya satu dari jalur default ini yang boleh ada per API.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

PayloadFormatVersion

Menentukan format muatan yang dikirim ke integrasi.

CATATAN: PayloadFormatVersion mengharuskan SAM untuk memodifikasi definisi OpenAPI Anda, sehingga hanya berfungsi dengan definisi sebaris di OpenApi properti. DefinitionBody

Tipe: String

Wajib: Tidak

Default: 2.0

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

RouteSettings

Pengaturan rute per rute untuk API HTTP ini. Untuk informasi selengkapnya tentang setelan rute, lihat [AWS::ApiGatewayV2::Stage RouteSettings](#) di Panduan Pengembang API Gateway.

Catatan: Jika `RouteSettings` ditentukan dalam `HttpApi` sumber daya dan sumber peristiwa, AWS SAM gabungkan mereka dengan properti sumber peristiwa yang diutamakan.

Jenis: [RouteSettings](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [RouteSettings](#) dari sumber daya `AWS::ApiGatewayV2::Stage`.

TimeoutInMillis

Waktu habis kustom antara 50 dan 29.000 milidetik.

CATATAN: `TimeoutInMillis` mengharuskan SAM untuk memodifikasi definisi OpenAPI Anda, sehingga hanya berfungsi dengan definisi sebaris di OpenAPI properti. `DefinitionBody`

Tipe: Integer

Wajib: Tidak

Default: 5000

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

HttpApi Acara Default

HttpApi Peristiwa yang menggunakan jalur default. Semua jalur dan metode yang belum dipetakan pada API ini akan merutekan ke titik akhir ini.

YAML

```
Events:
  HttpApiEvent:
    Type: HttpApi
```

HttpApi

HttpApi Peristiwa yang menggunakan jalur dan metode tertentu.

YAML

```
Events:
  HttpApiEvent:
    Type: HttpApi
    Properties:
      Path: /
      Method: GET
```

HttpApi Otorisasi

HttpApi Event yang menggunakan Authorizer.

YAML

```
Events:
  HttpApiEvent:
    Type: HttpApi
    Properties:
      Path: /authenticated
      Method: GET
    Auth:
      Authorizer: OpenIdAuth
      AuthorizationScopes:
        - scope1
        - scope2
```

HttpApiFunctionAuth

Mengonfigurasi otorisasi di tingkat peristiwa.

Mengonfigurasi Auth untuk API + Path + Metode tertentu

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
AuthorizationScopes: List
Authorizer: String
```

Properti

AuthorizationScopes

Cakupan otorisasi yang akan diterapkan ke API, path, dan metode ini.

Lingkup yang tercantum di sini akan membatalkan setiap cakupan yang diterapkan oleh `DefaultAuthorizer` jika ada.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Authorizer

`Authorizer` Untuk Fungsi tertentu. Untuk menggunakan otorisasi IAM, tentukan `AWS_IAM` dan tentukan `true` untuk `EnableIamAuthorizer` di `Globals` bagian template Anda.

Jika Anda telah menentukan Otorisasi Global pada API dan ingin membuat Fungsi tertentu terbuka untuk publik, batalkan dengan mengatur `Authorizer` menjadi `NONE`.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

Function-Auth

Menentukan Otorisasi di tingkat Fungsi

YAML

```
Auth:
  Authorizer: OpenIdAuth
  AuthorizationScopes:
    - scope1
    - scope2
```

Otorisasi IAM

Menentukan otorisasi IAM di tingkat acara. Untuk menggunakan `AWS_IAM` otorisasi di tingkat acara, Anda juga harus menentukan `true` untuk `EnableIamAuthorizer` di `Globals` bagian template Anda. Untuk informasi selengkapnya, lihat [Bagian global dari templat AWS SAM](#).

YAML

```
Globals:
  HttpApi:
    Auth:
      EnableIamAuthorizer: true

Resources:
  HttpApiFunctionWithIamAuth:
    Type: AWS::Serverless::Function
    Properties:
      Events:
        ApiEvent:
          Type: HttpApi
          Properties:
            Path: /iam-auth
            Method: GET
            Auth:
              Authorizer: AWS_IAM
      Handler: index.handler
      InlineCode: |
        def handler(event, context):
          return {'body': 'HttpApiFunctionWithIamAuth', 'statusCode': 200}
      Runtime: python3.9
```

IoTRule

Objek yang menggambarkan tipe sumber peristiwa `IoTRule`.

Menciptakan [AWS::IoT::TopicRule](#) sumber daya untuk mendeklarasikan aturan. AWS IoT Untuk informasi selengkapnya, lihat [dokumentasi AWS CloudFormation](#)

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
AwsIotSqlVersion: String  
Sql: String
```

Properti

AwsIotSqlVersion

Versi mesin aturan SQL yang harus digunakan saat mengevaluasi aturan.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [AwsIotSqlVersion](#) dari sumber daya `AWS::IoT::TopicRule TopicRulePayload`.

Sql

Pernyataan SQL yang digunakan untuk mengueri topik. Untuk informasi selengkapnya, lihat [Referensi SQL AWS IoT](#) di Panduan Developer AWS IoT.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Sql](#) dari sumber daya `AWS::IoT::TopicRule TopicRulePayload`.

Contoh-contoh

Aturan IOT

Contoh Aturan IOT

YAML

```
IoTRule:  
  Type: IoTRule  
  Properties:
```

```
Sql: SELECT * FROM 'topic/test'
```

Kinesis

Objek yang menggambarkan tipe sumber peristiwa Kinesis. Untuk informasi lebih lanjut, lihat [Menggunakan AWS Lambda dengan Amazon Kinesis](#) di Panduan Developer AWS Lambda.

AWS SAM menghasilkan sumber [AWS::Lambda::EventSourceMapping](#) daya saat jenis acara ini disetel.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
BatchSize: Integer  
BisectBatchOnFunctionError: Boolean  
DestinationConfig: DestinationConfig  
Enabled: Boolean  
FilterCriteria: FilterCriteria  
FunctionResponseTypes: List  
MaximumBatchingWindowInSeconds: Integer  
MaximumRecordAgeInSeconds: Integer  
MaximumRetryAttempts: Integer  
ParallelizationFactor: Integer  
StartingPosition: String  
StartingPositionTimestamp: Double  
Stream: String  
TumblingWindowInSeconds: Integer
```

Properti

BatchSize

Jumlah maksimum item yang akan diambil dalam satu batch.

Tipe: Integer

Wajib: Tidak

Default: 100

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [BatchSize](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

Minimal: 1

Maksimum: 10000

`BisectBatchOnFunctionError`

Jika fungsi mengembalikan kesalahan, bagi batch menjadi dua dan coba lagi.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [BisectBatchOnFunctionError](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

`DestinationConfig`

Antrean Amazon Simple Queue Service (Amazon SQS) atau tujuan topik Amazon Simple Notification Service (Amazon SNS) untuk catatan yang dibuang.

Jenis: [DestinationConfig](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [DestinationConfig](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

`Enabled`

Menonaktifkan pemetaan sumber peristiwa untuk menjeda polling dan pemanggilan.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Enabled](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

`FilterCriteria`

Objek yang mendefinisikan kriteria untuk menentukan apakah Lambda harus memproses suatu peristiwa. Untuk informasi selengkapnya, lihat [pemfilteran AWS Lambda acara](#) di Panduan AWS Lambda Pengembang.

Jenis: [FilterCriteria](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [FilterCriteria](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

FunctionResponseTypes

Daftar tipe jawaban saat ini yang diterapkan ke pemetaan sumber peristiwa. Untuk informasi selengkapnya, lihat [Melaporkan kegagalan item batch](#) di Panduan Developer AWS Lambda.

Nilai yang valid: `ReportBatchItemFailures`

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [FunctionResponseTypes](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

MaximumBatchingWindowInSeconds

Jumlah waktu maksimum untuk mengumpulkan rekaman sebelum memanggil fungsi, dalam hitungan detik.

Tipe: Integer

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [MaximumBatchingWindowInSeconds](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

MaximumRecordAgeInSeconds

Periode permintaan maksimum yang dikirimkan Lambda ke fungsi untuk diproses.

Tipe: Integer

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [MaximumRecordAgeInSeconds](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

MaximumRetryAttempts

Jumlah waktu maksimum untuk mencoba kembali saat fungsi mengembalikan kesalahan.

Tipe: Integer

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [MaximumRetryAttempts](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

ParallelizationFactor

Jumlah batch yang akan diproses dari tiap serpihan secara bersamaan.

Tipe: Integer

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [ParallelizationFactor](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

StartingPosition

Posisi dalam pengaliran tempat untuk mulai membaca.

- `AT_TIMESTAMP`— Tentukan waktu untuk mulai membaca catatan.
- `LATEST`— Baca hanya catatan baru.
- `TRIM_HORIZON`— Memproses semua catatan yang tersedia.

Nilai yang valid: `AT_TIMESTAMP` | `LATEST` | `TRIM_HORIZON`

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [StartingPosition](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

StartingPositionTimestamp

Waktu untuk mulai membaca, dalam detik waktu Unix. Tentukan `StartingPositionTimestamp` kapan `StartingPosition` ditentukan sebagai `AT_TIMESTAMP`.

Tipe: Ganda

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [StartingPositionTimestamp](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

Stream

Amazon Resource Name (ARN) dari aliran data atau konsumen pengaliran.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [EventSourceArn](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

TumblingWindowInSeconds

Durasi dalam detik dari periode pemrosesan. Kisaran yang valid adalah 1 sampai 900 (15 menit).

Untuk informasi selengkapnya, lihat [Periode Jatuh](#) di Panduan Developer AWS Lambda.

Tipe: Integer

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [TumblingWindowInSeconds](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

Contoh-contoh

Sumber peristiwa Kinesis

Berikut ini adalah contoh sumber peristiwa Kinesis.

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

Objek yang menggambarkan tipe sumber peristiwa MQ. Untuk informasi lebih lanjut, lihat [Menggunakan Lambda dengan Amazon MQ](#) dalam Panduan Developer AWS Lambda.

AWS Serverless Application Model(AWS SAM) menghasilkan [AWS::Lambda::EventSourceMapping](#) sumber daya saat jenis acara ini disetel.

Note

Untuk memiliki antrian Amazon MQ di cloud pribadi virtual (VPC) yang terhubung ke fungsi Lambda di jaringan publik, peran eksekusi fungsi Anda harus menyertakan izin berikut:

- `ec2:CreateNetworkInterface`
- `ec2>DeleteNetworkInterface`
- `ec2:DescribeNetworkInterfaces`
- `ec2:DescribeSecurityGroups`
- `ec2:DescribeSubnets`
- `ec2:DescribeVpcs`

Untuk informasi selengkapnya, lihat [Izin peran eksekusi](#) di Panduan Developer AWS Lambda.

Sintaks

Untuk mendeklarasikan entitas ini di AWS SAM template Anda, gunakan sintaks berikut.

YAML

```
BatchSize: Integer
```

Broker: *String*
DynamicPolicyName: *Boolean*
Enabled: *Boolean*
FilterCriteria: *FilterCriteria*
MaximumBatchingWindowInSeconds: *Integer*
Queues: *List*
SecretsManagerKmsKeyId: *String*
SourceAccessConfigurations: *List*

Properti

BatchSize

Jumlah maksimum item yang akan diambil dalam satu batch.

Tipe: Integer

Wajib: Tidak

Default: 100

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [BatchSize](#) dari sumber daya AWS::Lambda::EventSourceMapping.

Minimal: 1

Maksimum: 10000

Broker

Amazon Resource Name (ARN) broker Amazon MQ.

Tipe: String


Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [EventSourceArn](#) dari sumber daya AWS::Lambda::EventSourceMapping.

DynamicPolicyName

Secara default, nama kebijakan AWS Identity and Access Management (IAM) adalah `SamAutoGeneratedAMQPolicy` untuk kompatibilitas mundur. Tentukan `true` untuk

menggunakan nama yang dibuat secara otomatis untuk kebijakan IAM Anda. Nama ini akan menyertakan ID logis sumber peristiwa Amazon MQ.

 Note

Saat menggunakan lebih dari satu sumber peristiwa Amazon MQ, tentukan `true` untuk menghindari duplikat nama kebijakan IAM.

Tipe: Boolean

Wajib: Tidak

Default: `false`

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Enabled

Jika `true`, pemetaan sumber peristiwa aktif. Untuk menunda polling dan pemanggilan, atur ke `false`.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Enabled](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

FilterCriteria

Objek yang mendefinisikan kriteria yang menentukan apakah Lambda harus memproses suatu peristiwa. Untuk informasi selengkapnya, lihat [pemfilteran AWS Lambda acara](#) di Panduan AWS Lambda Pengembang.

Jenis: [FilterCriteria](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [FilterCriteria](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

MaximumBatchingWindowInSeconds

Jumlah waktu maksimum untuk mengumpulkan rekaman sebelum memanggil fungsi, dalam hitungan detik.

Tipe: Integer

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [MaximumBatchingWindowInSeconds](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

Queues

Nama antrean tujuan broker Amazon MQ yang akan digunakan.

Tipe: Daftar

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Queues](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

SecretsManagerKmsKeyId

ID kunci AWS Key Management Service (AWS KMS) dari kunci yang dikelola pelanggan dari AWS Secrets Manager. Diperlukan saat Anda menggunakan kunci yang dikelola pelanggan dari Secrets Manager dengan peran eksekusi Lambda yang tidak menyertakan izin `kms:Decrypt`

Nilai properti ini adalah UUID. Sebagai contoh: `1abc23d4-567f-8ab9-cde0-1fab234c5d67`.

Tipe: String

Wajib: Bersyarat

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

SourceAccessConfigurations

Array dari protokol autentikasi atau host virtual. Tentukan ini menggunakan tipe [SourceAccessConfigurations](#) data.

Untuk tipe sumber MQ acara, satu-satunya tipe konfigurasi yang valid adalah `BASIC_AUTH` dan `VIRTUAL_HOST`.

- **BASIC_AUTH**—Rahasia Secrets Manager yang menyimpan kredensi broker Anda. Untuk tipe ini, kredensialnya harus dalam format berikut: `{"username": "your-username", "password": "your-password"}`. Hanya satu objek dari tipe `BASIC_AUTH` diizinkan.
- **VIRTUAL_HOST**— Nama host virtual di broker RabbitMQ Anda. Lambda akan menggunakan host Rabbit MQ ini sebagai sumber peristiwa. Hanya satu objek dari tipe `VIRTUAL_HOST` diizinkan.

Tipe: Daftar

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [SourceAccessConfigurations](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

Contoh-contoh

Sumber peristiwa Amazon MQ

Berikut ini adalah contoh tipe sumber peristiwa MQ untuk broker Amazon MQ.

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

Objek yang menggambarkan tipe sumber peristiwa MSK. Untuk informasi lebih lanjut, lihat [Menggunakan AWS Lambda dengan Amazon MSK](#) di Panduan Developer AWS Lambda.

AWS Serverless Application Model(AWS SAM) menghasilkan [AWS::Lambda::EventSourceMapping](#) sumber daya saat jenis acara ini disetel.

Sintaks

Untuk mendeklarasikan entitas ini di AWS SAM template Anda, gunakan sintaks berikut.

YAML

```
ConsumerGroupId: String  
DestinationConfig: DestinationConfig  
FilterCriteria: FilterCriteria  
MaximumBatchingWindowInSeconds: Integer  
SourceAccessConfigurations: SourceAccessConfigurations  
StartingPosition: String  
StartingPositionTimestamp: Double  
Stream: String  
Topics: List
```

Properti

ConsumerGroupId

String yang mengonfigurasi bagaimana acara akan dibaca dari topik Kafka.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [AmazonManagedKafkaConfiguration](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

DestinationConfig

Objek konfigurasi yang menentukan tujuan dari peristiwa setelah Lambda memprosesnya.

Gunakan properti ini untuk menentukan tujuan pemanggilan gagal dari sumber peristiwa MSK Amazon.

Jenis: [DestinationConfig](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [DestinationConfig](#) dari sumber daya AWS::Lambda::EventSourceMapping.

FilterCriteria

Objek yang mendefinisikan kriteria yang menentukan apakah Lambda harus memproses suatu peristiwa. Untuk informasi selengkapnya, lihat [pemfilteran AWS Lambda acara](#) di Panduan AWS Lambda Pengembang.

Jenis: [FilterCriteria](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [FilterCriteria](#) dari sumber daya AWS::Lambda::EventSourceMapping.

MaximumBatchingWindowInSeconds

Jumlah waktu maksimum untuk mengumpulkan rekaman sebelum memanggil fungsi, dalam hitungan detik.

Tipe: Integer

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [MaximumBatchingWindowInSeconds](#) dari sumber daya AWS::Lambda::EventSourceMapping.

SourceAccessConfigurations

Susunan protokol autentikasi, komponen VPC, atau host virtual untuk mengamankan dan menentukan sumber peristiwa Anda.

Nilai yang valid: CLIENT_CERTIFICATE_TLS_AUTH

Jenis: Daftar [SourceAccessConfiguration](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [SourceAccessConfigurations](#) dari sumber daya AWS::Lambda::EventSourceMapping.

StartingPosition

Posisi dalam pengaliran tempat untuk mulai membaca.

- `AT_TIMESTAMP`— Tentukan waktu untuk mulai membaca catatan.
- `LATEST`— Baca hanya catatan baru.
- `TRIM_HORIZON`— Memproses semua catatan yang tersedia.

Nilai yang valid: `AT_TIMESTAMP` | `LATEST` | `TRIM_HORIZON`

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [StartingPosition](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

StartingPositionTimestamp

Waktu untuk mulai membaca, dalam detik waktu Unix. Tentukan `StartingPositionTimestamp` kapan `StartingPosition` ditentukan sebagai `AT_TIMESTAMP`.

Tipe: Ganda

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [StartingPositionTimestamp](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

Stream

Amazon Resource Name (ARN) dari aliran data atau konsumen pengaliran.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [EventSourceArn](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

Topics

Nama topik Kafka.

Tipe: Daftar

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Topics](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

Contoh-contoh

Contoh Amazon MSK untuk Klaster yang ada

Berikut ini adalah contoh jenis sumber MSK peristiwa untuk kluster MSK Amazon yang sudah ada di file Akun AWS.

YAML

```
Events:
  MSKEvent:
    Type: MSK
    Properties:
      StartingPosition: LATEST
      Stream: arn:aws:kafka:us-east-1:012345678012:cluster/exampleClusterName/
      abcdefab-1234-abcd-5678-cdef0123ab01-2
    Topics:
      - MyTopic
```

Amazon MSK contoh untuk Klaster yang dinyatakan dalam Templat yang sama

Berikut ini adalah contoh tipe sumber peristiwa MSK untuk kluster Amazon MSK yang dinyatakan dalam file templat yang sama.

YAML

```
Events:
  MSKEvent:
    Type: MSK
    Properties:
      StartingPosition: LATEST
      Stream:
        Ref: MyMskCluster # This must be the name of an MSK cluster declared in the
        same template file
    Topics:
```

- MyTopic

S3

Objek yang menggambarkan tipe sumber peristiwa S3.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Bucket: String  
Events: String | List  
Filter: NotificationFilter
```

Properti

Bucket

Nama bucket S3. Bucket ini harus ada dalam templat yang sama.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini mirip dengan properti [BucketName](#) dari sumber daya AWS::S3::Bucket. Ini adalah bidang yang diperlukan dalam SAM. Bidang ini hanya menerima referensi ke bucket S3 yang dibuat dalam templat ini

Events

Peristiwa bucket Amazon S3 yang akan dipanggilkan fungsi Lambda. Lihat [Tipe peristiwa yang didukung Amazon S3](#) untuk daftar nilai yang valid.

Tipe: String | Daftar

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Event](#) dari tipe data AWS::S3::Bucket LambdaConfiguration.

Filter

Aturan pemfilteran yang menentukan objek Amazon S3 yang memanggil fungsi Lambda. Untuk informasi tentang pemfilteran nama kunci Amazon S3, lihat Mengonfigurasi [Pemberitahuan Acara Amazon S3](#) di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

Jenis: [NotificationFilter](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Filter](#) dari tipe data `AWS::S3::Bucket LambdaConfiguration`.

Contoh-contoh

S3-Event

Contoh peristiwa S3.

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

Objek yang menjelaskan jenis sumber Schedule peristiwa, yang menetapkan fungsi tanpa server Anda sebagai target EventBridge aturan Amazon yang dipicu pada jadwal. Untuk informasi selengkapnya, lihat [Apa itu Amazon EventBridge?](#) di Panduan EventBridge Pengguna Amazon.

AWS Serverless Application Model(AWS SAM) menghasilkan [AWS::Events::Rules](#) sumber daya saat jenis acara ini disetel.

Note

EventBridge sekarang menawarkan kemampuan penjadwalan baru, [Amazon EventBridge Scheduler](#). Amazon EventBridge Scheduler adalah penjadwal tanpa server yang memungkinkan Anda membuat, menjalankan, dan mengelola tugas dari satu layanan terpusat dan terkelola. EventBridge Scheduler sangat dapat disesuaikan, dan menawarkan skalabilitas yang ditingkatkan dibandingkan aturan EventBridge terjadwal, dengan serangkaian operasi API target yang lebih luas dan. Layanan AWS Kami menyarankan Anda menggunakan EventBridge Scheduler untuk memanggil target pada jadwal. Untuk menentukan jenis sumber acara ini di AWS SAM templat Anda, lihat [ScheduleV2](#).

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
DeadLetterConfig: DeadLetterConfig  
Description: String  
Enabled: Boolean  
Input: String  
Name: String  
RetryPolicy: RetryPolicy  
Schedule: String  
State: String
```

Properti

DeadLetterConfig

Konfigurasi antrian Amazon Simple Queue Service (Amazon SQS) EventBridge tempat pengiriman peristiwa setelah pemanggilan target gagal. Pemanggilan dapat gagal, misalnya, saat mengirim acara ke fungsi Lambda yang tidak ada, atau ketika tidak EventBridge memiliki izin yang

cukup untuk memanggil fungsi Lambda. Untuk informasi selengkapnya, lihat [Kebijakan percobaan ulang acara dan menggunakan antrian huruf mati di Panduan Pengguna](#) Amazon. EventBridge

Note

Jenis [AWS::Serverless::Function](#) sumber daya memiliki tipe data yang serupa `DeadLetterQueue`, yang menangani kegagalan yang terjadi setelah pemanggilan fungsi Lambda target berhasil. Contoh tipe kegagalan ini termasuk Lambda throttling, atau kesalahan yang dikembalikan oleh fungsi target Lambda. Untuk informasi selengkapnya tentang `DeadLetterQueue` properti fungsi, lihat [Antrian huruf mati di Panduan Pengembang](#). AWS Lambda

Jenis: [DeadLetterConfig](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [DeadLetterConfig](#) dari tipe data `AWS::Events::RuleTarget`. Versi AWS SAM dari properti ini termasuk subproperti tambahan, jika Anda menginginkan agar AWS SAM membuat antrian surat mati untuk Anda.

Description

Deskripsi aturan.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Description](#) dari sumber daya `AWS::Events::Rule`.

Enabled

Menunjukkan apakah aturan diaktifkan.

Untuk menonaktifkan aturan, tetapkan properti ini ke `false`.

Note

Tentukan salah satu `Enabled` atau `State` properti, tetapi tidak keduanya.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini mirip dengan properti [State](#) dari sumber daya `AWS::Events::Rule`. Jika properti ini ditetapkan ke `true` maka AWS SAM meneruskan `ENABLED`, jika tidak, properti ini meneruskan `DISABLED`.

Input

Teks JSON yang valid diteruskan ke target. Jika Anda menggunakan properti ini, tidak ada dari teks peristiwa itu sendiri yang diteruskan ke target.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Input](#) dari sumber daya `AWS::Events::Rule Target`.

Name

Nama aturan. Jika Anda tidak menetapkan nama, AWS CloudFormation menghasilkan ID fisik unik dan menggunakan ID tersebut sebagai nama aturan.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Name](#) dari sumber daya `AWS::Events::Rule`.

RetryPolicy

Objek `RetryPolicy` yang menyertakan informasi tentang pengaturan kebijakan coba lagi. Untuk informasi selengkapnya, lihat [Kebijakan percobaan ulang acara dan menggunakan antrian huruf mati di Panduan Pengguna Amazon EventBridge](#)

Jenis: [RetryPolicy](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [RetryPolicy](#) dari tipe data `AWS::Events::Rule Target`.

Schedule

Ekspresi penjadwalan yang menentukan kapan dan seberapa sering aturan dijalankan. Untuk informasi lebih lanjut, lihat [Ekspresi Jadwal untuk Aturan](#).

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [ScheduleExpression](#) dari sumber daya `AWS::Events::Rule`.

State

Keadaan aturan.

Nilai yang diterima: DISABLED | ENABLED

Note

Tentukan salah satu Enabled atau State properti, tetapi tidak keduanya.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [State](#) dari sumber daya `AWS::Events::Rule`.

Contoh-contoh

CloudWatch Jadwal Acara

CloudWatch Jadwal Event Contoh

YAML

```
CWSchedule:
  Type: Schedule
  Properties:
```

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

DeadLetterConfig

Objek yang digunakan untuk menentukan antrean Amazon Simple Queue Service (Amazon SQS) EventBridge tempat mengirimkan peristiwa setelah pemanggilan target gagal. Permintaan dapat gagal, misalnya, ketika mengirim peristiwa ke fungsi Lambda yang tidak ada, atau izin tidak cukup untuk memanggil fungsi Lambda. Untuk informasi selengkapnya, lihat [Kebijakan percobaan ulang acara dan menggunakan antrian huruf mati di Panduan Pengguna](#) Amazon. EventBridge

Note

Jenis [AWS::Serverless::Function](#) sumber daya memiliki tipe data yang serupa, `DeadLetterQueue` yang menangani kegagalan yang terjadi setelah pemanggilan fungsi Lambda target berhasil. Contoh tipe kegagalan ini termasuk Lambda throttling, atau kesalahan dikembalikan oleh fungsi target Lambda. Untuk informasi selengkapnya tentang `DeadLetterQueue` properti fungsi, lihat [Antrian huruf mati di Panduan Pengembang](#). AWS Lambda

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Arn: String  
QueueLogicalId: String  
Type: String
```

Properti

Arn

Amazon Resource Name (ARN) dari antrean Amazon SQS yang ditetapkan sebagai target antrean surat mati.

 Note

Tentukan Type properti atau Arn properti, tetapi tidak keduanya.


Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Arn](#) dari tipe data `AWS::Events::Rule DeadLetterConfig`.

QueueLogicalId

Nama kustom dari antrean surat mati yang AWS SAM buat jika Type ditentukan.

 Note

Jika Type properti tidak disetel, properti ini diabaikan.


Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Type

Tipe antrean. Bila properti ini diatur, AWS SAM secara otomatis membuat antrean surat mati dan melampirkan [kebijakan berbasis sumber daya](#) yang diperlukan untuk memberikan izin untuk memerintahkan sumber daya untuk mengirim peristiwa ke antrean.

 Note

Tentukan Type properti atau Arn properti, tetapi tidak keduanya.

Nilai yang valid: SQS

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

DeadLetterConfig

DeadLetterConfig

YAML

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

ScheduleV2

Objek yang menjelaskan jenis sumber `ScheduleV2` peristiwa, yang menetapkan fungsi tanpa server Anda sebagai target peristiwa Amazon EventBridge Scheduler yang dipicu pada jadwal. Untuk informasi selengkapnya, lihat [Apa itu Amazon EventBridge Scheduler?](#) di Panduan Pengguna EventBridge Penjadwal.

AWS Serverless Application Model(AWS SAM) menghasilkan [AWS::Scheduler::Schedule](#) sumber daya saat jenis acara ini disetel.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

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
```

Properti

DeadLetterConfig

Konfigurasi antrian Amazon Simple Queue Service (Amazon SQS) EventBridge tempat pengiriman peristiwa setelah pemanggilan target gagal. Pemanggilan dapat gagal, misalnya, saat mengirim acara ke fungsi Lambda yang tidak ada, atau ketika tidak EventBridge memiliki izin yang cukup untuk memanggil fungsi Lambda. Untuk informasi selengkapnya, lihat [Mengonfigurasi antrian huruf mati untuk EventBridge Penjadwal di Panduan Pengguna Penjadwal](#). EventBridge

Note

Jenis [AWS::Serverless::Function](#) sumber daya memiliki tipe data yang serupa `DeadLetterQueue`, yang menangani kegagalan yang terjadi setelah pemanggilan fungsi Lambda target berhasil. Contoh tipe kegagalan ini termasuk Lambda throttling, atau kesalahan yang dikembalikan oleh fungsi target Lambda. Untuk informasi selengkapnya tentang `DeadLetterQueue` properti fungsi, lihat [Antrian huruf mati di Panduan Pengembang](#). AWS Lambda

Jenis: [DeadLetterConfig](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [DeadLetterConfig](#) dari tipe data `AWS::Scheduler::ScheduleTarget`. Versi AWS SAM properti ini termasuk subproperti tambahan, jika Anda ingin AWS SAM membuat antrian surat mati untuk Anda.

Description

Deskripsi jadwal.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Description](#) dari sumber daya `AWS::Scheduler::Schedule`.

EndDate

Tanggal, di UTC, sebelum jadwal dapat memanggil targetnya. Bergantung pada ekspresi pengulangan jadwal, pemanggilan mungkin berhenti, atau sebelum, yang Anda tentukan `EndDate`.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [EndDate](#) dari sumber daya `AWS::Scheduler::Schedule`.

FlexibleTimeWindow

Mengizinkan konfigurasi jendela di mana jadwal dapat dipanggil.

Jenis: [FlexibleTimeWindow](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [FlexibleTimeWindow](#) dari sumber daya `AWS::Scheduler::Schedule`.

GroupName

Nama grup jadwal untuk dikaitkan dengan jadwal ini. Jika tidak ditentukan, grup default digunakan.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [GroupName](#) dari sumber daya `AWS::Scheduler::Schedule`.

Input

Teks JSON yang valid yang dilewatkan ke target. Jika Anda menggunakan properti ini, tidak ada dari teks peristiwa itu sendiri yang diteruskan ke target.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Input](#) dari sumber daya `AWS::Scheduler::Schedule Target`.

KmsKeyArn

ARN untuk Kunci KMS yang akan digunakan untuk mengenkripsi data pelanggan.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [KmsKeyArn](#) dari sumber daya `AWS::Scheduler::Schedule`.

Name

Nama jadwal. Jika Anda tidak menentukan nama, AWS SAM buat nama dalam format *Function-Logical-IDEvent-Source-Name* dan gunakan ID tersebut untuk nama jadwal.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Name](#) dari sumber daya `AWS::Scheduler::Schedule`.

OmitName

Secara default, AWS SAM menghasilkan dan menggunakan nama jadwal dalam format *< event-source-name ><Function-logical-ID>*. Tetapkan properti ini `true` agar AWS CloudFormation menghasilkan ID fisik yang unik dan gunakan itu untuk nama jadwal sebagai gantinya.

Tipe: Boolean

Wajib: Tidak

Default: `false`

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

PermissionsBoundary

ARN kebijakan yang digunakan untuk mengatur batas izin untuk peran.

Note

Jika `PermissionsBoundary` didefinisikan, AWS SAM akan menerapkan batasan yang sama untuk peran IAM target jadwal penjadwal.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [PermissionsBoundary](#) dari sumber daya `AWS::IAM::Role`.

RetryPolicy

Objek `RetryPolicy` yang menyertakan informasi tentang pengaturan kebijakan coba lagi.

Jenis: [RetryPolicy](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [RetryPolicy](#) dari tipe data `AWS::Scheduler::Schedule Target`.

RoleArn

ARN dari peran IAM yang akan digunakan EventBridge Scheduler untuk target saat jadwal dipanggil.

Jenis: [RoleArn](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [RoleArn](#) dari tipe data `AWS::Scheduler::Schedule Target`.

ScheduleExpression

Ekspresi penjadwalan yang menentukan kapan dan seberapa sering acara jadwal penjadwal berjalan.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [ScheduleExpression](#) dari sumber daya `AWS::Scheduler::Schedule`.

ScheduleExpressionTimezone

Zona waktu di mana ekspresi penjadwalan dievaluasi.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [ScheduleExpressionTimezone](#) dari sumber daya `AWS::Scheduler::Schedule`.

StartDate

Tanggal, di UTC, setelah itu jadwal dapat mulai memanggil target. Bergantung pada ekspresi pengulangan jadwal, pemanggilan mungkin terjadi pada, atau setelah, yang Anda tentukan `StartDate`.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [StartDate](#) dari sumber daya `AWS::Scheduler::Schedule`.

State

Keadaan jadwal Scheduler.

Nilai yang diterima: DISABLED | ENABLED

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [State](#) dari sumber daya `AWS::Scheduler::Schedule`.

Contoh-contoh

Contoh dasar mendefinisikan sumber daya ScheduleV2

```
Resources:
```

```

Function:
  Properties:
    ...
  Events:
    ScheduleEvent:
      Type: ScheduleV2
      Properties:
        ScheduleExpression: "rate(1 minute)"
    ComplexScheduleEvent:
      Type: ScheduleV2
      Properties:
        ScheduleExpression: rate(1 minute)
        FlexibleTimeWindow:
          Mode: FLEXIBLE
          MaximumWindowInMinutes: 5
        StartDate: '2022-12-28T12:00:00.000Z'
        EndDate: '2023-01-28T12:00:00.000Z'
        ScheduleExpressionTimezone: UTC
        RetryPolicy:
          MaximumRetryAttempts: 5
          MaximumEventAgeInSeconds: 300
        DeadLetterConfig:
          Type: SQS

```

SelfManagedKafka

Objek yang menggambarkan tipe sumber peristiwa SelfManagedKafka. Untuk informasi selengkapnya, lihat [Menggunakan AWS Lambda dengan Apache Kafka yang dikelola sendiri di Panduan Pengembang](#). AWS Lambda

AWS Serverless Application Model(AWS SAM) menghasilkan [AWS::Lambda::EventSourceMapping](#) sumber daya saat jenis acara ini disetel.

Sintaks

Untuk mendeklarasikan entitas ini di AWS SAM template Anda, gunakan sintaks berikut.

YAML

```

BatchSize: Integer
ConsumerGroupId: String
DestinationConfig: DestinationConfig
Enabled: Boolean

```

FilterCriteria: [FilterCriteria](#)
KafkaBootstrapServers: *List*
SourceAccessConfigurations: [SourceAccessConfigurations](#)
StartingPosition: *String*
StartingPositionTimestamp: *Double*
Topics: *List*

Properti

BatchSize

Jumlah maksimum rekaman di setiap batch yang Lambda tarik dari aliran Anda dan dikirim ke fungsi Anda.

Tipe: Integer

Wajib: Tidak

Default: 100

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [BatchSize](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

Minimal: 1

Maksimum: 10000

ConsumerGroupId

String yang mengonfigurasi bagaimana acara akan dibaca dari topik Kafka.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [SelfManagedKafkaConfiguration](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

DestinationConfig

Objek konfigurasi yang menentukan tujuan dari peristiwa setelah Lambda memprosesnya.

Gunakan properti ini untuk menentukan tujuan pemanggilan gagal dari sumber acara Kafka yang dikelola sendiri.

Jenis: [DestinationConfig](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [DestinationConfig](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

Enabled

Menonaktifkan pemetaan sumber peristiwa untuk menjeda polling dan pemanggilan.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Enabled](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

FilterCriteria

Objek yang mendefinisikan kriteria untuk menentukan apakah Lambda harus memproses suatu peristiwa. Untuk informasi selengkapnya, lihat [pemfilteran AWS Lambda acara](#) di Panduan AWS Lambda Pengembang.

Jenis: [FilterCriteria](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [FilterCriteria](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

KafkaBootstrapServers

Daftar server bootstrap untuk broker Kafka Anda. Sertakan port, misalnya `broker.example.com:xxxx`

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

SourceAccessConfigurations

Susunan protokol autentikasi, komponen VPC, atau host virtual untuk mengamankan dan menentukan sumber peristiwa Anda.

Nilai yang valid: BASIC_AUTH | CLIENT_CERTIFICATE_TLS_AUTH | SASL_SCRAM_256_AUTH | SASL_SCRAM_512_AUTH | SERVER_ROOT_CA_CERTIFICATE

Jenis: Daftar [SourceAccessConfiguration](#)

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [SourceAccessConfigurations](#) dari sumber daya AWS::Lambda::EventSourceMapping.

StartingPosition

Posisi dalam pengaliran tempat untuk mulai membaca.

- AT_TIMESTAMP— Tentukan waktu untuk mulai membaca catatan.
- LATEST— Baca hanya catatan baru.
- TRIM_HORIZON— Memproses semua catatan yang tersedia.

Nilai yang valid: AT_TIMESTAMP | LATEST | TRIM_HORIZON

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [StartingPosition](#) dari sumber daya AWS::Lambda::EventSourceMapping.

StartingPositionTimestamp

Waktu untuk mulai membaca, dalam detik waktu Unix. Tentukan StartingPositionTimestamp kapan StartingPosition ditentukan sebagai AT_TIMESTAMP.

Tipe: Ganda

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [StartingPositionTimestamp](#) dari sumber daya AWS::Lambda::EventSourceMapping.

Topics

Nama topik Kafka.

Tipe: Daftar

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Topics](#) dari sumber daya AWS::Lambda::EventSourceMapping.

Contoh-contoh

Sumber acara Kafka yang dikelola sendiri

Berikut adalah contoh peristiwa dari tipe sumber peristiwa SelfManagedKafka.

YAML

```
Events:
  SelfManagedKafkaEvent:
    Type: SelfManagedKafka
    Properties:
      BatchSize: 1000
      Enabled: true
      KafkaBootstrapServers:
        - abc.xyz.com:xxxx
      SourceAccessConfigurations:
        - Type: BASIC_AUTH
          URI: arn:aws:secretsmanager:us-west-2:123456789012:secret:my-path/my-secret-name-1a2b3c
      Topics:
        - MyKafkaTopic
```

SNS

Objek yang menggambarkan tipe sumber peristiwa SNS.

SAM menghasilkan [AWS::SNS::Subscription](#) sumber daya saat jenis acara ini disetel

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
FilterPolicy: SnsFilterPolicy
```

```
FilterPolicyScope: String  
RedrivePolicy: Json  
Region: String  
SqsSubscription: Boolean | SqsSubscriptionObject  
Topic: String
```

Properti

FilterPolicy

Kebijakan filter JSON yang ditetapkan untuk langganan. Untuk informasi selengkapnya, lihat [GetSubscriptionAttributes](#) di Referensi API Layanan Pemberitahuan Sederhana Amazon.

Jenis: [SnsFilterPolicy](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [FilterPolicy](#) dari sumber daya `AWS::SNS::Subscription`.

FilterPolicyScope

Atribut ini memungkinkan Anda memilih lingkup penyaringan dengan menggunakan salah satu jenis nilai string berikut:

- `MessageAttributes`— Filter diterapkan pada atribut pesan.
- `MessageBody`— Filter diterapkan pada badan pesan.

Tipe: String

Wajib: Tidak

Default: `MessageAttributes`

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [FilterPolicyScope](#) dari sumber daya `AWS::SNS::Subscription`.

RedrivePolicy

Bila ditentukan, kirim pesan yang tidak terkirim ke antrean surat mati Amazon SQS yang ditentukan. Pesan yang tidak dapat dikirim karena kesalahan klien (misalnya, ketika titik akhir berlangganan tidak terjangkau) atau kesalahan server (misalnya, ketika layanan yang mendorong titik akhir langganan menjadi tidak tersedia) diadakan di antrean surat mati untuk analisis lebih lanjut atau pemrosesan ulang.

Untuk informasi selengkapnya tentang kebijakan redrive dan antrian surat mati, lihat [antrian surat mati Amazon SQS di Panduan Pengembang Layanan Antrian](#) Sederhana Amazon.

Jenis: Json

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [RedrivePolicy](#) dari sumber daya `AWS::SNS::Subscription`.

Region

Untuk langganan lintas wilayah, wilayah tempat topik berada.

Jika tidak ada wilayah yang ditentukan, CloudFormation gunakan wilayah pemanggil sebagai default.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Region](#) dari sumber daya `AWS::SNS::Subscription`.

SqsSubscription

Atur properti ini menjadi `BETUL`, atau tentukan `SqsSubscriptionObject` untuk mengaktifkan batching notifikasi topik SNS dalam antrian SQS. Mengatur properti ini menjadi `true` membuat antrian SQS baru, sedangkan menentukan `SqsSubscriptionObject` menggunakan antrian SQS yang ada.

Jenis: Boolean | [SqsSubscriptionObject](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Topic

ARN dari topik untuk dilanggan.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [TopicArn](#) dari sumber daya `AWS::SNS::Subscription`.

Contoh-contoh

Contoh Sumber Peristiwa SNS

Contoh Sumber Peristiwa SNS

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

Tentukan opsi antrean SQS yang ada untuk peristiwa SNS

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
BatchSize: String
Enabled: Boolean
QueueArn: String
QueuePolicyLogicalId: String
QueueUrl: String
```

Properti

BatchSize

Jumlah maksimum item yang akan diambil dalam satu batch untuk antrean SQS.

Tipe: String

Wajib: Tidak

Default: 10

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Enabled

Menonaktifkan pemetaan sumber peristiwa SQS untuk menjeda polling dan pemanggilan.

Tipe: Boolean

Wajib: Tidak

Default: BETUL

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

QueueArn

Tentukan arn antrean SQS yang ada.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

QueuePolicyLogicalId

Berikan nama LogicalId kustom untuk sumber daya. [AWS::SQS::QueuePolicy](#)

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

QueueUrl

Tentukan URL antrean yang terkait dengan properti QueueArn.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

SQS yang ada untuk peristiwa SNS

Contoh untuk menambahkan antrean SQS yang ada untuk pelanggan topik SNS.

YAML

```
QueuePolicyLogicalId: CustomQueuePolicyLogicalId
QueueArn:
  Fn::GetAtt: MyCustomQueue.Arn
QueueUrl:
  Ref: MyCustomQueue
BatchSize: 5
```

SQS

Objek yang menggambarkan tipe sumber peristiwa SQS. Untuk informasi lebih lanjut, lihat [Menggunakan AWS Lambda dengan Amazon SQS](#) di Panduan Developer AWS Lambda.

SAM menghasilkan [AWS::Lambda::EventSourceMapping](#) sumber daya saat jenis acara ini disetel

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
BatchSize: Integer
```

Enabled: *Boolean*
FilterCriteria: *FilterCriteria*
FunctionResponseTypes: *List*
MaximumBatchingWindowInSeconds: *Integer*
Queue: *String*
ScalingConfig: *ScalingConfig*

Properti

BatchSize

Jumlah maksimum item yang akan diambil dalam satu batch.

Tipe: Integer

Wajib: Tidak

Default: 10

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [BatchSize](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

Minimal: 1

Maksimum: 10000

Enabled

Menonaktifkan pemetaan sumber peristiwa untuk menjeda polling dan pemanggilan.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Enabled](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

FilterCriteria

Objek yang mendefinisikan kriteria untuk menentukan apakah Lambda harus memproses suatu peristiwa. Untuk informasi selengkapnya, lihat [pemfilteran AWS Lambda acara](#) di Panduan AWS Lambda Pengembang.

Jenis: [FilterCriteria](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [FilterCriteria](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

FunctionResponseTypes

Daftar tipe jawaban saat ini yang diterapkan ke pemetaan sumber peristiwa. Untuk informasi selengkapnya, lihat [Melaporkan kegagalan item batch](#) di Panduan AWS Lambda Pengembang.

Nilai yang valid: `ReportBatchItemFailures`

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [FunctionResponseTypes](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

MaximumBatchingWindowInSeconds

Jumlah waktu maksimum untuk mengumpulkan catatan sebelum memanggil fungsi, dalam hitungan detik.

Tipe: Integer

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [MaximumBatchingWindowInSeconds](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

Queue

ARN antrean.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [EventSourceArn](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

ScalingConfig

Konfigurasi penskalaan poller SQS untuk mengontrol laju pemanggilan dan menetapkan pemanggilan bersamaan maksimum.

Jenis: [ScalingConfig](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [ScalingConfig](#) dari sumber daya `AWS::Lambda::EventSourceMapping`.

Contoh-contoh

Acara SQS dasar

```
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"]}'
```

Konfigurasi pelaporan batch sebagian untuk antrian SQS Anda

```
Events:
  SQSEvent:
    Type: SQS
    Properties:
      Enabled: true
      FunctionResponseTypes:
        - ReportBatchItemFailures
      Queue: !GetAtt MySqsQueue.Arn
      BatchSize: 10
```

Fungsi Lambda dengan acara SQS yang telah dikonfigurasi penskalaan

```
MyFunction:
  Type: AWS::Serverless::Function
  Properties:
    ...
  Events:
    MySQSEvent:
```

```
Type: SQS
Properties:
  ...
ScalingConfig:
  MaximumConcurrency: 10
```

FunctionCode

[Paket deployment](#) untuk fungsi Lambda.

Sintaks

Untuk mendeklarasikan entitas ini dalam template AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

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

Properti

Bucket

Bucket Amazon S3 di AWS Wilayah yang sama dengan fungsi Anda.

Tipe: String

Wajib: Ya

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [S3Bucket](#) properti tipe `AWS::Lambda::Function Code` data.

Key

Kunci Amazon S3 dari paket deployment.

Tipe: String

Wajib: Ya

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [S3Key](#) properti tipe `AWS::Lambda::Function Code` data.

Version

Untuk objek berversi, versi objek paket deployment yang digunakan.

Tipe: String

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [S3ObjectVersion](#) properti tipe AWS::Lambda::Function Code data.

Contoh-contoh

FunctionCode

CodeUri: Contoh Kode Fungsi

YAML

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

FunctionUrlConfig

Membuat URL AWS Lambda fungsi dengan parameter konfigurasi yang ditentukan. URL fungsi Lambda adalah titik akhir HTTPS yang dapat Anda gunakan untuk menjalankan fungsi Anda.

Secara default, URL fungsi yang Anda buat menggunakan \$LATEST versi fungsi Lambda Anda. Jika Anda menentukan `AutoPublishAlias` untuk fungsi Lambda Anda, titik akhir terhubung ke alias fungsi yang ditentukan.

Untuk informasi selengkapnya, lihat [URL fungsi Lambda di Panduan Pengembang](#). AWS Lambda

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
AuthType: String
```

```
Cors: Cors  
InvokeMode: String
```

Properti

AuthType

Jenis otorisasi untuk URL fungsi Anda. Untuk menggunakan AWS Identity and Access Management (IAM) untuk mengotorisasi permintaan, setel ke. `AWS_IAM` Untuk akses terbuka, atur ke `NONE`.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [AuthType](#) dari sumber daya `AWS::Lambda::Url`.

Cors

Pengaturan berbagi sumber daya lintas asal (CORS) untuk URL fungsi Anda.

Jenis: [Cor](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Cors](#) dari sumber daya `AWS::Lambda::Url`.

InvokeMode

Mode URL fungsi Anda akan dipanggil. Agar fungsi Anda mengembalikan respons setelah pemanggilan selesai, setel ke. `BUFFERED` Agar fungsi Anda mengalirkan respons, setel ke `RESPONSE_STREAM`. Nilai default-nya adalah `BUFFERED`.

Nilai yang valid: `BUFFERED` atau `RESPONSE_STREAM`

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [InvokeMode](#) dari sumber daya `AWS::Lambda::Url`.

Contoh-contoh

URL fungsi

Contoh berikut membuat fungsi Lambda dengan URL fungsi. URL fungsi menggunakan otorisasi IAM.

YAML

```
HelloWorldFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: hello_world/
    Handler: index.handler
    Runtime: nodejs14.x
    FunctionUrlConfig:
      AuthType: AWS_IAM
      InvokeMode: RESPONSE_STREAM

Outputs:
  MyFunctionUrlEndpoint:
    Description: "My Lambda Function URL Endpoint"
    Value:
      Fn::GetAtt: HelloWorldFunctionUrl.FunctionUrl
```

AWS::Serverless::GraphQLApi

Gunakan tipe `AWS::Serverless::GraphQLApi` sumber daya AWS Serverless Application Model (AWS SAM) untuk membuat dan mengonfigurasi AWS AppSync GraphQL API untuk aplikasi tanpa server Anda.

Untuk mempelajari lebih lanjut tentang AWS AppSync, lihat [Apa itu AWS AppSync?](#) di Panduan AWS AppSync Pengembang.

Sintaks

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
```

Properti

ApiKeys

Buat kunci unik yang dapat digunakan untuk melakukan GraphQL operasi yang membutuhkan kunci API.

Jenis: [ApiKeys](#)

Wajib: Tidak

AWS CloudFormationkompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

Auth

Konfigurasi autentikasi untuk GraphQL API Anda.

Jenis: [Auth](#)

Wajib: Ya

AWS CloudFormationkompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

Cache

Input dari suatu CreateApiCache operasi.

Jenis: [AWS::AppSync::ApiCache](#)

Wajib: Tidak

AWS CloudFormationkompatibilitas: Properti ini diteruskan langsung ke [AWS::AppSync::ApiCache](#) sumber daya.

DataSources

Buat sumber data untuk fungsi AWS AppSync yang terhubung ke. AWS SAMmendukung Amazon DynamoDB AWS Lambda dan sumber data.

Jenis: [DataSource](#)

Wajib: Ya

AWS CloudFormationkompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

DomainName

Nama domain khusus untuk GraphQL API Anda.

Jenis: [AWS::AppSync::DomainName](#)

Wajib: Tidak

AWS CloudFormationkompatibilitas: Properti ini diteruskan langsung ke [AWS::AppSync::DomainName](#) sumber daya. AWS SAMsecara otomatis menghasilkan [AWS::AppSync::DomainNameApiAssociation](#) sumber daya.

Functions

Konfigurasi fungsi dalam GraphQL API untuk melakukan operasi tertentu.

Jenis: [Fungsi](#)

Wajib: Ya

AWS CloudFormationkompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

Logging

Mengonfigurasi CloudWatch pencatatan Amazon untuk GraphQL API Anda.

Jika Anda tidak menentukan properti ini, AWS SAM akan menghasilkan `CloudWatchLogsRoleArn` dan menetapkan nilai-nilai berikut:

- `ExcludeVerboseContent: true`
- `FieldLogLevel: ALL`

Untuk memilih keluar dari logging, tentukan yang berikut ini:

```
Logging: false
```

Jenis: [LogConfig](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [LogConfig](#) dari sumber daya `AWS::AppSync::GraphQLApi`.

LogicalId

Nama unik GraphQL API Anda.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Name](#) dari sumber daya `AWS::AppSync::GraphQLApi`.

Name

Nama GraphQL API Anda. Tentukan properti ini untuk mengganti `LogicalId` nilainya.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Name](#) dari sumber daya `AWS::AppSync::GraphQLApi`.

Resolvers

Konfigurasi resolver untuk bidang API Anda GraphQL. AWS SAM mendukung [resolver JavaScript pipa](#).

Jenis: [Resolver](#)

Wajib: Ya

AWS CloudFormationkompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

SchemaInline

Representasi teks dari GraphQL skema dalam SDL format.

Tipe: String

Diperlukan: Bersyarat. Anda harus menentukan SchemaInline atau SchemaUri.

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Definition](#) dari sumber daya AWS::AppSync::GraphQLSchema.

SchemaUri

URI bucket Amazon Simple Storage Service (Amazon S3) skema atau path ke folder lokal.

Jika Anda menentukan jalur ke folder lokal, AWS CloudFormation mengharuskan file tersebut diunggah terlebih dahulu ke Amazon S3 sebelum penerapan. Anda dapat menggunakan AWS SAM CLI untuk memfasilitasi proses ini. Untuk informasi selengkapnya, lihat [Menggunakan AWS SAMCLI untuk mengunggah file lokal saat penerapan](#).

Tipe: String

Diperlukan: Bersyarat. Anda harus menentukan SchemaInline atau SchemaUri.

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [DefinitionS3Location](#) dari sumber daya AWS::AppSync::GraphQLSchema.

Tags

Tag (pasangan nilai kunci) untuk API iniGraphQL. Gunakan tag untuk mengidentifikasi dan mengkategorikan sumber daya.

Jenis: Daftar [Tag](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Tag](#) dari sumber daya AWS::AppSync::GraphQLApi.

XrayEnabled

Tunjukkan apakah akan menggunakan [penelusuran AWS X-Ray](#) untuk sumber daya ini.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [XrayEnabled](#) dari sumber daya AWS::AppSync::GraphQLApi.

Contoh-contoh

GraphQL API dengan sumber data DynamoDB

Dalam contoh ini, kita membuat GraphQL API yang menggunakan tabel DynamoDB sebagai sumber data.

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

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  DynamoDBPostsTable:
    Type: AWS::Serverless::SimpleTable

  MyGraphQLAPI:
    Type: AWS::Serverless::GraphQLApi
    Properties:
      SchemaUri: ./sam_graphql_api/schema.graphql
      Auth:
        Type: AWS_IAM
      DataSources:
        DynamoDb:
          PostsDataSource:
            TableName: !Ref DynamoDBPostsTable
            TableArn: !GetAtt DynamoDBPostsTable.Arn
      Functions:
        preprocessPostItem:
          Runtime:
            Name: APPSYNC_JS
            Version: 1.0.0
          DataSource: NONE
          CodeUri: ./sam_graphql_api/preprocessPostItem.js
        createPostItem:
          Runtime:
            Name: APPSYNC_JS
            Version: "1.0.0"
          DataSource: PostsDataSource
          CodeUri: ./sam_graphql_api/createPostItem.js
        getPostFromTable:
          Runtime:
            Name: APPSYNC_JS
            Version: "1.0.0"
          DataSource: PostsDataSource
          CodeUri: ./sam_graphql_api/getPostFromTable.js
    Resolvers:
      Mutation:
        addPost:
          Runtime:
            Name: APPSYNC_JS
```

```

    Version: "1.0.0"
  Pipeline:
    - preprocessPostItem
    - createPostItem
  Query:
    getPost:
      CodeUri: ./sam_graphql_api/getPost.js
      Runtime:
        Name: APPSYNC_JS
        Version: "1.0.0"
      Pipeline:
        - getPostFromTable

```

createPostItem.js

```

import { util } from "@aws-appsync/utils";

export function request(ctx) {
  const { key, values } = ctx.prev.result;
  return {
    operation: "PutItem",
    key: util.dynamodb.toMapValues(key),
    attributeValues: util.dynamodb.toMapValues(values),
  };
}

export function response(ctx) {
  return ctx.result;
}

```

getPostFromTable.js

```

import { util } from "@aws-appsync/utils";

export function request(ctx) {
  return dynamoDBGetItemRequest({ id: ctx.args.id });
}

export function response(ctx) {
  return ctx.result;
}

/**

```

```
* A helper function to get a DynamoDB item
*/
function dynamoDBGetItemRequest(key) {
  return {
    operation: "GetItem",
    key: util.dynamodb.toMapValues(key),
  };
}
```

preprocessPostItem.js

```
import { util } from "@aws-appsync/utils";

export function request(ctx) {
  const id = util.autoId();
  const { ...values } = ctx.args;
  values.ups = 1;
  values.downs = 0;
  values.version = 1;
  return { payload: { key: { id }, values: values } };
}

export function response(ctx) {
  return ctx.result;
}
```

Berikut adalah kode resolver kami:

getPost.js

```
export function request(ctx) {
  return {};
}

export function response(ctx) {
  return ctx.prev.result;
}
```

GraphQLAPI dengan fungsi Lambda sebagai sumber data

Dalam contoh ini, kita membuat GraphQL API yang menggunakan fungsi Lambda sebagai sumber data.

template.yaml

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyLambdaFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: nodejs14.x
      CodeUri: ./lambda

  MyGraphQLAPI:
    Type: AWS::Serverless::GraphQLApi
    Properties:
      Name: MyApi
      SchemaUri: ./gql/schema.gql
      Auth:
        Type: API_KEY
      ApiKeys:
        MyApiKey:
          Description: my api key
      DataSources:
        Lambda:
          MyLambdaDataSource:
            FunctionArn: !GetAtt MyLambdaFunction.Arn
      Functions:
        lambdaInvoker:
          Runtime:
            Name: APPSYNC_JS
            Version: 1.0.0
          DataSource: MyLambdaDataSource
          CodeUri: ./gql/invoker.js
      Resolvers:
        Mutation:
          addPost:
            Runtime:
              Name: APPSYNC_JS
              Version: 1.0.0
            Pipeline:
              - lambdaInvoker
      Query:
        getPost:
```

```
Runtime:
  Name: APPSYNC_JS
  Version: 1.0.0
Pipeline:
  - lambdaInvoker
```

Outputs:

```
MyGraphQLAPI:
  Description: AppSync API
  Value: !GetAtt MyGraphQLAPI.GraphQLUrl
MyGraphQLAPIMyApiKey:
  Description: API Key for authentication
  Value: !GetAtt MyGraphQLAPIMyApiKey.ApiKey
```

schema.graphql

```
schema {
  query: Query
  mutation: Mutation
}
type Query {
  getPost(id: ID!): Post
}
type Mutation {
  addPost(id: ID!, author: String!, title: String, content: String): Post!
}
type Post {
  id: ID!
  author: String!
  title: String
  content: String
  ups: Int
  downs: Int
}
```

Berikut adalah fungsi kami:

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

Buat kunci unik yang dapat digunakan untuk melakukan GraphQL operasi yang membutuhkan kunci API.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
LogicalId:  
  ApiKeyId: String  
  Description: String  
  ExpiresOn: Double
```

Properti

ApiKeyId

Nama unik kunci API Anda. Tentukan untuk mengganti LogicalId nilai.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [ApiKeyId](#) dari sumber daya `AWS::AppSync::ApiKey`.

Description

Deskripsi kunci API Anda.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Description](#) dari sumber daya `AWS::AppSync::ApiKey`.

ExpiresOn

Waktu setelah kunci API berakhir. Tanggal direpresentasikan sebagai detik sejak jangka waktu, dibulatkan ke bawah ke waktu jam terdekat.

Tipe: Ganda

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Expires](#) dari sumber daya `AWS::AppSync::ApiKey`.

LogicalId

Nama unik kunci API Anda.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [ApiKeyId](#) dari sumber daya `AWS::AppSync::ApiKey`.

Autentikasi

Konfigurasi otorisasi untuk GraphQL API Anda.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Additional:
- AuthProvider
LambdaAuthorizer: LambdaAuthorizerConfig
OpenIDConnect: OpenIDConnectConfig
Type: String
UserPool: UserPoolConfig
```

Properti

Additional

Daftar jenis otorisasi tambahan untuk GraphQL API Anda.

Jenis: Daftar [AuthProvider](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

LambdaAuthorizer

Tentukan konfigurasi otorisasi opsional untuk otorisasi fungsi Lambda Anda. Anda dapat mengkonfigurasi properti opsional ini ketika Type ditentukan sebagai `AWS_LAMBDA`.

Jenis: [LambdaAuthorizerConfig](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [LambdaAuthorizerConfig](#) dari sumber daya `AWS::AppSync::GraphQLApi`.

OpenIDConnect

Tentukan konfigurasi otorisasi opsional untuk layanan yang OpenID Connect sesuai. Anda dapat mengkonfigurasi properti opsional ini ketika Type ditentukan sebagai `OPENID_CONNECT`.

Jenis: [OpenID ConnectConfig](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [OpenIDConnectConfig](#) dari sumber daya `AWS::AppSync::GraphQLApi`.

Type

Jenis otorisasi default antara aplikasi dan AWS AppSync GraphQL API Anda.

Untuk daftar dan deskripsi nilai yang diizinkan, lihat [Otorisasi dan otentikasi di Panduan AWS AppSync Pengembang](#).

Saat Anda menentukan Lambda authorizer (`AWS_LAMBDA`), AWS SAM buat kebijakan AWS Identity and Access Management (IAM) untuk menyediakan izin antara API GraphQL dan fungsi Lambda Anda.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [AuthenticationType](#) dari sumber daya `AWS::AppSync::GraphQLApi`.

UserPool

Tentukan konfigurasi otorisasi opsional untuk menggunakan kumpulan pengguna Amazon Cognito. Anda dapat mengkonfigurasi properti opsional ini ketika Type ditentukan sebagai `AMAZON_COGNITO_USER_POOLS`.

Jenis: [UserPoolConfig](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [UserPoolConfig](#) dari sumber daya `AWS::AppSync::GraphQLApi`.

Contoh-contoh

Konfigurasi jenis otorisasi default dan tambahan

Dalam contoh ini, kita mulai dengan mengonfigurasi otorisasi Lambda sebagai jenis otorisasi default untuk API kita. GraphQL

```
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
```

Selanjutnya, kami mengonfigurasi jenis otorisasi tambahan untuk GraphQL API kami dengan menambahkan yang berikut ke AWS SAM template kami:

```
Additional:
- Type: AWS_IAM
- Type: API_KEY
- Type: OPENID_CONNECT
```

```

OpenIDConnect:
  AuthTTL: 10
  ClientId: myId
  IatTTL: 10
  Issuer: prod

```

Ini menghasilkan AWS SAM template berikut:

```

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

Konfigurasi otorisasi opsional untuk jenis otorisasi GraphQL API tambahan Anda.

Sintaksis

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
LambdaAuthorizer: LambdaAuthorizerConfig
```

[OpenIDConnect](#): [OpenIDConnectConfig](#)

Type: *String*

[UserPool](#): [UserPoolConfig](#)

Properti

LambdaAuthorizer

Tentukan konfigurasi otorisasi opsional untuk otorisasi AWS Lambda fungsi Anda. Anda dapat mengkonfigurasi properti opsional ini ketika Type ditentukan sebagai `AWS_LAMBDA`.

Jenis: [LambdaAuthorizerConfig](#)

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini dilewatkan langsung ke [LambdaAuthorizerConfig](#) properti dari suatu `AWS::AppSync::GraphQLApiAdditionalAuthenticationProvider` objek.

OpenIDConnect

Tentukan konfigurasi otorisasi opsional untuk layanan yang OpenID Connect sesuai. Anda dapat mengkonfigurasi properti opsional ini ketika Type ditentukan sebagai `OPENID_CONNECT`.

Jenis: [OpenID ConnectConfig](#)

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini dilewatkan langsung ke [OpenIDConnectConfig](#) properti dari suatu `AWS::AppSync::GraphQLApiAdditionalAuthenticationProvider` objek.

Type

Jenis otorisasi default antara aplikasi dan AWS AppSync GraphQL API Anda.

Untuk daftar dan deskripsi nilai yang diizinkan, lihat [Otorisasi dan otentikasi](#) di Panduan AWS AppSync Pengembang.

Saat Anda menentukan otorisasi Lambda (`AWS_LAMBDA`), AWS SAM buat kebijakan AWS Identity and Access Management (IAM) untuk menyediakan izin antara API dan fungsi Lambda Anda. GraphQL

Tipe: String

Wajib: Ya

AWS CloudFormationkompatibilitas: Properti ini dilewatkan langsung ke [AuthenticationType](#) properti dari suatu `AWS::AppSync::GraphQLApi` [AdditionalAuthenticationProvider](#) objek.

UserPool

Tentukan konfigurasi otorisasi opsional untuk menggunakan kumpulan pengguna Amazon Cognito. Anda dapat mengkonfigurasi properti opsional ini ketika Type ditentukan sebagai `AMAZON_COGNITO_USER_POOLS`.

Jenis: [UserPoolConfig](#)

Wajib: Tidak

AWS CloudFormationkompatibilitas: Properti ini dilewatkan langsung ke [UserPoolConfig](#) properti dari suatu `AWS::AppSync::GraphQLApi` [AdditionalAuthenticationProvider](#) objek.

DataSource

Konfigurasi sumber data yang dapat dihubungkan dengan resolver GraphQL API Anda. Anda dapat menggunakan AWS Serverless Application Model (AWS SAM) template untuk mengkonfigurasi koneksi ke sumber data berikut:

- Amazon DynamoDB
- AWS Lambda

Untuk mempelajari lebih lanjut tentang sumber data, lihat [Melampirkan sumber data](#) di Panduan AWS AppSync Developer.

Sintaksis

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
DynamoDb: DynamoDb
```


[Lambda](#): [Lambda](#)

Properti

DynamoDb

Konfigurasi tabel DynamoDB sebagai sumber data untuk resolver GraphQL API Anda.

Jenis: [DynamoDb](#)

Wajib: Tidak

AWS CloudFormationkompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation setara.

Lambda

Konfigurasi fungsi Lambda sebagai sumber data untuk resolver GraphQL API Anda.

Tipe: [Lambda](#)

Wajib: Tidak

AWS CloudFormationkompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation setara.

DynamoDb

Konfigurasi tabel Amazon DynamoDB sebagai sumber data untuk GraphQL resolver API Anda.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
LogicalId:  
  DeltaSync: DeltaSyncConfig  
  Description: String  
  Name: String  
  Permissions: List  
  Region: String
```

```
ServiceRoleArn: String  
TableArn: String  
TableName: String  
UseCallerCredentials: Boolean  
Versioned: Boolean
```

Properti

DeltaSync

Menjelaskan konfigurasi Delta Sync.

Jenis: [DeltaSyncConfig](#)

Wajib: Tidak

AWS CloudFormationkompatibilitas: Properti ini diteruskan langsung ke [DeltaSyncConfig](#) properti `AWS::AppSync::DataSource DynamoDBConfig` objek.

Description

Deskripsi sumber data Anda.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Description](#) dari sumber daya `AWS::AppSync::DataSource`.

LogicalId

Nama unik sumber data Anda.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Name](#) dari sumber daya `AWS::AppSync::DataSource`.

Name

Nama sumber data Anda. Tentukan properti ini untuk mengganti LogicalId nilainya.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Name](#) dari sumber daya `AWS::AppSync::DataSource`.

Permissions

Menyediakan izin ke sumber data Anda menggunakan [AWS SAM konektor](#). Anda dapat memberikan salah satu nilai berikut dalam daftar:

- `Read`— Izinkan resolver Anda membaca sumber data Anda.
- `Write`— Izinkan resolver Anda untuk menulis ke sumber data Anda.

AWS SAM menggunakan `AWS::Serverless::Connector` sumber daya yang diubah saat penerapan untuk memberikan izin Anda. Untuk mempelajari tentang sumber daya yang dihasilkan, lihat [AWS CloudFormation sumber daya yang dihasilkan saat Anda menentukan `AWS::Serverless::Connector`](#).

Note

Anda dapat menentukan `Permissions` atau `ServiceRoleArn`, tapi tidak keduanya. Jika tidak ada yang ditentukan, AWS SAM akan menghasilkan nilai default `Read` dan `Write`. Untuk mencabut akses ke sumber data Anda, hapus objek DynamoDB dari template Anda. AWS SAM

Tipe: Daftar

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan. Ini mirip dengan [Permissions](#) properti sumber `AWS::Serverless::Connector` daya.

Region

Tabel Wilayah AWS DynamoDB Anda. Jika Anda tidak menentukannya, AWS SAM gunakan [AWS::Region](#).

Tipe: String

Wajib: Tidak

AWS CloudFormationkompatibilitas: Properti ini diteruskan langsung ke [AwsRegion](#) properti `AWS::AppSync::DataSource DynamoDBConfig` objek.

ServiceRoleArn

Peran layanan AWS Identity and Access Management (IAM) ARN untuk sumber data. Sistem mengasumsikan peran ini saat mengakses sumber data.

Anda dapat menentukan `Permissions` atau `ServiceRoleArn`, tapi tidak keduanya.

Tipe: String

Diperlukan: Tidak. Jika tidak ditentukan, AWS SAM menerapkan nilai default untuk `Permissions`.

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [ServiceRoleArn](#) dari sumber daya `AWS::AppSync::DataSource`.

TableArn

ARN untuk tabel DynamoDB.

Tipe: String

Diperlukan: Bersyarat. Jika Anda tidak menentukan `ServiceRoleArn`, `TableArn` diperlukan.

AWS CloudFormationkompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

TableName

Nama tabel.

Tipe: String

Wajib: Ya

AWS CloudFormationkompatibilitas: Properti ini diteruskan langsung ke [TableName](#) properti `AWS::AppSync::DataSource DynamoDBConfig` objek.

UseCallerCredentials

Setel `true` untuk menggunakan IAM dengan sumber data ini.

Tipe: Boolean

Wajib: Tidak

AWS CloudFormationkompatibilitas: Properti ini diteruskan langsung ke [UseCallerCredentials](#) properti `AWS::AppSync::DataSource DynamoDBConfig` objek.

Versioned

Setel `true` untuk menggunakan [Deteksi Konflik, Resolusi Konflik, dan Sinkronisasi](#) dengan sumber data ini.

Tipe: Boolean

Wajib: Tidak

AWS CloudFormationkompatibilitas: Properti ini diteruskan langsung ke [Versioned](#) properti `AWS::AppSync::DataSource DynamoDBConfig` objek.

Lambda

Konfigurasi AWS Lambda fungsi sebagai sumber data untuk resolver GraphQL API Anda.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
LogicalId:  
  Description: String  
  FunctionArn: String  
  Name: String  
  ServiceRoleArn: String
```

Properti

Description

Deskripsi sumber data Anda.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Description](#) dari sumber daya `AWS::AppSync::DataSource`.

FunctionArn

ARN untuk fungsi Lambda.

Tipe: String

Wajib: Tidak

AWS CloudFormationkompatibilitas: Properti ini diteruskan langsung ke [LambdaFunctionArn](#) properti `AWS::AppSync::DataSource` `LambdaConfig` objek.

LogicalId

Nama unik sumber data Anda.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Name](#) dari sumber daya `AWS::AppSync::DataSource`.

Name

Nama sumber data Anda. Tentukan properti ini untuk mengganti `LogicalId` nilainya.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Name](#) dari sumber daya `AWS::AppSync::DataSource`.

ServiceRoleArn

Peran layanan AWS Identity and Access Management (IAM) ARN untuk sumber data. Sistem mengasumsikan peran ini saat mengakses sumber data.

Note

Untuk mencabut akses ke sumber data Anda, hapus objek Lambda dari template Anda.
AWS SAM

Tipe: String

Diperlukan: Tidak. Jika tidak ditentukan, AWS SAM akan memberikan `Write` izin menggunakan [AWS SAM konektor](#).

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [ServiceRoleArn](#) dari sumber daya `AWS::AppSync::DataSource`.

Fungsi

Konfigurasi fungsi dalam GraphQL API untuk melakukan operasi tertentu.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
LogicalId:  
  CodeUri: String  
  DataSource: String  
  Description: String  
  Id: String  
  InlineCode: String  
  MaxBatchSize: Integer  
  Name: String  
  Runtime: Runtime  
  Sync: SyncConfig
```

Properti

CodeUri

Kode fungsi Amazon Simple Storage Service (Amazon S3) URI atau path ke folder lokal.

Jika Anda menentukan jalur ke folder lokal, AWS CloudFormation mengharuskan file tersebut diunggah terlebih dahulu ke Amazon S3 sebelum penerapan. Anda dapat menggunakan AWS SAM CLI untuk memfasilitasi proses ini. Untuk informasi selengkapnya, lihat [Menggunakan AWS SAM CLI untuk mengunggah file lokal saat penerapan](#).

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [CodeS3Location](#) dari sumber daya `AWS::AppSync::FunctionConfiguration`.

DataSource

Nama sumber data yang akan dilampirkan fungsi ini.

- Untuk mereferensikan sumber data dalam `AWS::Serverless::GraphQLApi` sumber daya, tentukan ID logisnya.
- Untuk mereferensikan sumber data di luar `AWS::Serverless::GraphQLApi` sumber daya, berikan Name atributnya menggunakan fungsi `Fn::GetAtt` intrinsik. Misalnya, `!GetAtt MyLambdaDataSource.Name`.
- Untuk mereferensikan sumber data dari tumpukan yang berbeda, gunakan [Fn::ImportValue](#).

Jika variasi [`NONE` | `None` | `none`] ditentukan, AWS SAM akan menghasilkan `None` nilai untuk `AWS::AppSync::DataSource` [Type](#) objek.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [DataSourceName](#) dari sumber daya `AWS::AppSync::FunctionConfiguration`.

Description

Deskripsi fungsi Anda.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Description](#) dari sumber daya `AWS::AppSync::FunctionConfiguration`.

Id

ID Fungsi untuk fungsi yang terletak di luar `AWS::Serverless::GraphQLApi` sumber daya.

- Untuk mereferensikan fungsi dalam AWS SAM template yang sama, gunakan fungsi `Fn::GetAtt` intrinsik. Sebagai contoh, `Id: !GetAtt createPostItemFunc.FunctionId`.
- Untuk mereferensikan fungsi dari tumpukan yang berbeda, gunakan [Fn::ImportValue](#).

Saat menggunakan `Id`, semua properti lainnya tidak diperbolehkan. AWS SAM akan secara otomatis melewati ID Fungsi dari fungsi referensi Anda.

Tipe: String

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

`InlineCode`

Kode fungsi yang berisi fungsi permintaan dan respons.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Code](#) dari sumber daya `AWS::AppSync::FunctionConfiguration`.

`LogicalId`

Nama unik dari fungsi Anda.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Name](#) dari sumber daya `AWS::AppSync::FunctionConfiguration`.

`MaxBatchSize`

Jumlah maksimum input permintaan resolver yang akan dikirim ke satu AWS Lambda fungsi dalam operasi. `BatchInvoke`

Tipe: Integer

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [MaxBatchSize](#) dari sumber daya `AWS::AppSync::FunctionConfiguration`.

`Name`

Nama fungsi. Tentukan untuk mengganti `LogicalId` nilai.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Name](#) dari sumber daya `AWS::AppSync::FunctionConfiguration`.

Runtime

Menjelaskan runtime yang digunakan oleh resolver atau fungsi AWS AppSync pipeline. AWS AppSync Menentukan nama dan versi runtime yang akan digunakan.

Jenis: [Runtime](#)

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation. Ini mirip dengan [Runtime](#) properti sumber daya `AWS::AppSync::FunctionConfiguration` daya.

Sync

Menjelaskan konfigurasi Sinkronisasi untuk suatu fungsi.

Menentukan strategi Deteksi Konflik dan strategi Resolusi mana yang akan digunakan saat fungsi dipanggil.

Jenis: [SyncConfig](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [SyncConfig](#) dari sumber daya `AWS::AppSync::FunctionConfiguration`.

Waktu Aktif

Runtime resolver atau fungsi pipeline Anda. Menentukan nama dan versi yang akan digunakan.

Sintaksis

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Name: String  
Version: String
```

Properti

Name

Nama runtime yang akan digunakan. Saat ini, satu-satunya nilai yang diizinkan adalah `APPSYNC_JS`.

Tipe: String

Wajib: Ya

AWS CloudFormation kompatibilitas: Properti ini dilewatkan langsung ke [Name](#) properti dari suatu `AWS::AppSync::FunctionConfiguration` `AppSyncRuntime` objek.

Version

Versi runtime yang akan digunakan. Saat ini, satu-satunya versi yang diizinkan adalah `1.0.0`.

Tipe: String

Wajib: Ya

AWS CloudFormation kompatibilitas: Properti ini dilewatkan langsung ke [RuntimeVersion](#) properti dari suatu `AWS::AppSync::FunctionConfiguration` `AppSyncRuntime` objek.

Penyelesai

Konfigurasi resolver untuk bidang API Anda GraphQL. AWS Serverless Application Model (AWS SAM) mendukung [JavaScript resolver pipa](#).

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
OperationType:
```

LogicalId:**Caching:** [CachingConfig](#)**CodeUri:** *String***FieldName:** *String***InlineCode:** *String***MaxBatchSize:** *Integer***Pipeline:** *List***Runtime:** [Runtime](#)**Sync:** [SyncConfig](#)

Properti

Caching

Konfigurasi caching untuk resolver yang telah caching diaktifkan.

Jenis: [CachingConfig](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [CachingConfig](#) dari sumber daya `AWS::AppSync::Resolver`.

CodeUri

Kode fungsi resolver adalah Amazon Simple Storage Service (Amazon S3) URI atau path ke folder lokal.

Jika Anda menentukan jalur ke folder lokal, AWS CloudFormation mengharuskan file tersebut diunggah terlebih dahulu ke Amazon S3 sebelum penerapan. Anda dapat menggunakan AWS SAM CLI untuk memfasilitasi proses ini. Untuk informasi selengkapnya, lihat [Menggunakan AWS SAM CLI untuk mengunggah file lokal saat penerapan](#).

Jika tidak `InlineCode` ada `CodeUri` atau disediakan, AWS SAM akan menghasilkan `InlineCode` yang mengalihkan permintaan ke fungsi pipeline pertama dan menerima respons dari fungsi pipeline terakhir.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [CodeS3Location](#) dari sumber daya `AWS::AppSync::Resolver`.

FieldName

Nama resolver Anda. Tentukan properti ini untuk mengganti LogicalId nilainya.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [FieldName](#) dari sumber daya `AWS::AppSync::Resolver`.

InlineCode

Kode resolver yang berisi fungsi permintaan dan respons.

Jika tidak InlineCode ada CodeUri atau disediakan, AWS SAM akan menghasilkan InlineCode yang mengalihkan permintaan ke fungsi pipeline pertama dan menerima respons dari fungsi pipeline terakhir.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Code](#) dari sumber daya `AWS::AppSync::Resolver`.

LogicalId

Nama unik untuk resolver Anda. Dalam GraphQL skema, nama resolver Anda harus cocok dengan nama bidang yang digunakan. Gunakan nama bidang yang sama untuk LogicalId.

Tipe: String

Wajib: Ya

AWS CloudFormationkompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

MaxBatchSize

Jumlah maksimum input permintaan resolver yang akan dikirim ke satu AWS Lambda fungsi dalam operasi. BatchInvoke

Tipe: Integer

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [MaxBatchSize](#) dari sumber daya `AWS::AppSync::Resolver`.

OperationType

Jenis GraphQL operasi yang terkait dengan resolver Anda. Sebagai contoh, Query, Mutation, atau Subscription. Anda dapat membuat sarang beberapa resolver LogicalId dalam satu.

OperationType

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [TypeName](#) dari sumber daya `AWS::AppSync::Resolver`.

Pipeline

Fungsi yang terkait dengan penyelesaian alur. Tentukan fungsi dengan ID logis dalam daftar.

Tipe: Daftar

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation. Ini mirip dengan [PipelineConfig](#) properti sumber `AWS::AppSync::Resolver` daya.

Runtime

Runtime resolver atau fungsi pipeline Anda. Menentukan nama dan versi yang akan digunakan.

Jenis: [Runtime](#)

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation. Ini mirip dengan [Runtime](#) properti sumber `AWS::AppSync::Resolver` daya.

Sync

Menjelaskan konfigurasi Sinkronisasi untuk penyelesaian.

Menentukan strategi Deteksi Konflik dan strategi Resolusi mana yang akan digunakan saat resolver dipanggil.

Jenis: [SyncConfig](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [SyncConfig](#) dari sumber daya `AWS::AppSync::Resolver`.

Contoh-contoh

Gunakan kode fungsi resolver yang AWS SAM dihasilkan dan simpan bidang sebagai variabel

Berikut adalah GraphQL skema untuk contoh kita:

```
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
}
```

Berikut adalah cuplikan template kami AWS SAM:

```
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

```

Dalam AWS SAM template kami, kami tidak menentukan `CodeUri` atau `InlineCode`. Saat penerapan, AWS SAM secara otomatis menghasilkan kode inline berikut untuk resolver kami:

```

export function request(ctx) {
  return {};
}

export function response(ctx) {
  return ctx.prev.result;
}

```

Kode resolver default ini mengalihkan permintaan ke fungsi pipeline pertama dan menerima respons dari fungsi pipeline terakhir.

Dalam fungsi pipeline pertama kita, kita dapat menggunakan `args` bidang yang disediakan untuk mengurai objek permintaan dan membuat variabel kita. Kita kemudian dapat menggunakan variabel-variabel ini dalam fungsi kita. Berikut adalah contoh `preprocessPostItem` fungsi kami:

```

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;  
}
```

Waktu Aktif

Runtime resolver atau fungsi pipeline Anda. Menentukan nama dan versi yang akan digunakan.

Sintaksis

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Name: String  
Version: String
```

Properti

Name

Nama runtime yang akan digunakan. Saat ini, satu-satunya nilai yang diizinkan adalah `APPSYNC_JS`.

Tipe: String

Wajib: Ya

AWS CloudFormation kompatibilitas: Properti ini dilewatkan langsung ke [Name](#) properti dari suatu `AWS::AppSync::Resolver` `AppSyncRuntime` objek.

Version

Versi runtime yang akan digunakan. Saat ini, satu-satunya versi yang diizinkan adalah `1.0.0`.

Tipe: String

Wajib: Ya

AWS CloudFormation kompatibilitas: Properti ini dilewatkan langsung ke [RuntimeVersion](#) properti dari suatu `AWS::AppSync::Resolver` `AppSyncRuntime` objek.

AWS::Serverless::HttpApi

Membuat API HTTP Amazon API Gateway, yang mengaktifkan Anda untuk membuat API RESTful dengan latensi yang lebih rendah dan biaya yang lebih rendah dari REST API. Untuk informasi selengkapnya, lihat [Bekerja dengan API HTTP](#) di Panduan Developer API Gateway.

Kami menyarankan Anda menggunakan AWS CloudFormation kait atau kebijakan IAM untuk memverifikasi bahwa sumber daya API Gateway memiliki otorisasi yang melekat padanya untuk mengontrol akses ke sumber daya tersebut.

Untuk informasi selengkapnya tentang penggunaan AWS CloudFormation kait, lihat [Mendaftarkan kait](#) di panduan pengguna AWS CloudFormation CLI dan repositori. [apigw-enforce-authorizer](#) GitHub

Untuk informasi selengkapnya tentang penggunaan kebijakan IAM, lihat [Mengharuskan rute API memiliki otorisasi dalam Panduan](#) Pengembang API Gateway.

Note

Ketika Anda menyebarkan ke AWS CloudFormation, AWS SAM mengubah AWS SAM sumber daya Anda menjadi AWS CloudFormation sumber daya. Untuk informasi selengkapnya, lihat [Sumber daya AWS CloudFormation yang dibuat](#).

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

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
```

Properti

AccessLogSettings

Pengaturan untuk log akses dalam tahap.

Jenis: [AccessLogSettings](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [AccessLogSettings](#) dari sumber daya `AWS::ApiGatewayV2::Stage`.

Auth

Mengonfigurasi otorisasi untuk mengendalikan akses ke API HTTP API Gateway Anda.

Untuk informasi selengkapnya, lihat [Mengendalikan akses ke API HTTP dengan otorisasi JWT](#) di Panduan Developer API Gateway.

Jenis: [HttpApiAuth](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

CorsConfiguration

Mengelola cross-origin resource sharing (CORS) untuk semua API HTTP API Gateway Anda. Tentukan domain untuk diizinkan sebagai string, atau menentukan objek `HttpApiCorsConfiguration`. Perhatikan bahwa CORS membutuhkan AWS SAM untuk memodifikasi ketentuan OpenAPI Anda, jadi CORS bekerja hanya jika properti `DefinitionBody` ditentukan.

Untuk informasi selengkapnya, lihat [Mengonfigurasi CORS untuk API HTTP](#) di Panduan Developer API Gateway.

 Note

Jika `CorsConfiguration` disetel baik dalam definisi OpenAPI maupun di tingkat properti, maka AWS SAM gabungkan kedua sumber konfigurasi dengan properti yang diutamakan. Jika properti ini disetel `true`, maka semua asal diizinkan.

Jenis: String | [HttpApiCorsConfiguration](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

DefaultRouteSettings

Pengaturan rute default untuk API HTTP ini. Pengaturan ini berlaku untuk semua rute kecuali dibatalkan oleh properti `RouteSettings` untuk rute tertentu.

Jenis: [RouteSettings](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [RouteSettings](#) dari sumber daya `AWS::ApiGatewayV2::Stage`.

DefinitionBody

Ketentuan OpenAPI yang menjelaskan API HTTP Anda. Jika Anda tidak menentukan `DefinitionUri` atau `DefinitionBody`, AWS SAM membuat `DefinitionBody` untuk Anda berdasarkan konfigurasi templat Anda.

Tipe: JSON

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini mirip dengan properti [Body](#) dari sumber daya `AWS::ApiGatewayV2::Api`. Jika properti tertentu tersedia, AWS SAM dapat memasukkan konten ke dalam atau memodifikasi `DefinitionBody` sebelum diteruskan

ke AWS CloudFormation. Properti termasuk Auth dan EventSource tipe HttpApi untuk `AWS::Serverless::Function` sumber daya yang sesuai.

DefinitionUri

URI Amazon Simple Storage Service (Amazon S3), jalur file lokal, atau objek lokasi ketentuan OpenAPI yang menentukan API HTTP. Objek Amazon S3 yang merujuk properti ini harus menjadi file ketentuan OpenAPI yang valid. Jika Anda tidak menentukan `DefinitionUri` atau `DefinitionBody` ditentukan, AWS SAM membuat `DefinitionBody` untuk Anda berdasarkan konfigurasi templat Anda.

Jika Anda menyediakan jalur file lokal, templat harus melalui alur kerja yang mencakup perintah `deploy` atau `sam package` untuk ketentuan agar berubah dengan benar.

Fungsi intrinsik tidak didukung dalam file OpenApi definisi eksternal yang Anda referensikan. `DefinitionUri` Untuk mengimpor OpenApi definisi ke dalam template, gunakan `DefinitionBody` properti dengan [transformasi Sertakan](#).

Jenis: String | [HttpApiDefinition](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini mirip dengan properti [BodyS3Location](#) sumber daya `AWS::ApiGatewayV2::Api`. Properti Amazon S3 nest diberi nama berbeda.

Description

Deskripsi sumber daya HTTP API.

Saat Anda menentukan `Description`, AWS SAM akan memodifikasi OpenApi definisi sumber daya HTTP API dengan menyetel `description` bidang. Skenario berikut akan menghasilkan kesalahan:

- `DefinitionBody`Properti ditentukan dengan `description` bidang yang disetel dalam definisi Open API — Ini menghasilkan konflik `description` bidang yang tidak AWS SAM akan diselesaikan.
- `DefinitionUri`Properti ditentukan — tidak AWS SAM akan mengubah definisi Open API yang diambil dari Amazon S3.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

DisableExecuteApiEndpoint

Menentukan apakah klien dapat memanggil API HTTP Anda dengan menggunakan `https://
{api_id}.execute-api.{region}.amazonaws.com` titik akhir `execute-api` default. Secara default, klien dapat memanggil API Anda dengan titik akhir default. Untuk meminta agar klien hanya menggunakan nama domain kustom untuk memanggil API Anda, nonaktifkan titik akhir default.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini mirip dengan properti [DisableExecuteApiEndpoint](#) dari sumber daya `AWS::ApiGatewayV2::Api`. Itu diteruskan langsung ke `disableExecuteApiEndpoint` properti [x-amazon-apigateway-endpoint-configuration](#) ekstensi, yang akan ditambahkan ke [Body](#) properti `AWS::ApiGatewayV2::Api` sumber daya.

Domain

Mengonfigurasi domain kustom untuk API HTTP API Gateway ini.

Jenis: [HttpApiDomainConfiguration](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

FailOnWarnings

Menentukan apakah akan memutar kembali pembuatan API HTTP (`true`) atau tidak (`false`) saat peringatan ditemui. Nilai default-nya adalah `false`.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [FailOnWarnings](#) dari sumber daya `AWS::ApiGatewayV2::Api`.

Name

Nama sumber daya HTTP API.

Saat Anda menentukan `Name`, AWS SAM akan memodifikasi definisi OpenAPI sumber daya HTTP API dengan menyetel bidang `title`. Skenario berikut akan menghasilkan kesalahan:

- `DefinitionBody` properti ditentukan dengan `title` bidang yang disetel dalam definisi Open API — Ini menghasilkan konflik `title` bidang yang tidak AWS SAM akan diselesaikan.
- `DefinitionUri` properti ditentukan — tidak AWS SAM akan mengubah definisi Open API yang diambil dari Amazon S3.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

PropagateTags

Tunjukkan apakah akan meneruskan tag dari `Tags` properti ke sumber daya yang Anda [AWS::Serverless::HttpApi](#) hasilkan atau tidak. Tentukan `True` untuk menyebarkan tag di sumber daya yang Anda hasilkan.

Tipe: Boolean

Wajib: Tidak

Default: `False`

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

RouteSettings

Pengaturan rute, per rute, untuk API HTTP ini. Untuk informasi selengkapnya, lihat [Bekerja dengan rute untuk API HTTP](#) di Panduan Developer API Gateway.

Jenis: [RouteSettings](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [RouteSettings](#) dari sumber daya `AWS::ApiGatewayV2::Stage`.

StageName

Nama tahap API. Jika tidak ada nama yang ditentukan, AWS SAM menggunakan tahap `$default` dari API Gateway.

Tipe: String

Wajib: Tidak

Default: `$default`

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [StageName](#) dari sumber daya `AWS::ApiGatewayV2::Stage`.

StageVariables

Sebuah peta yang menentukan variabel panggung. Nama variabel dapat memiliki karakter alfanumerik dan garis bawah. Nilai-nilai harus sesuai dengan `[A-Za-z0-9-._~:/? #&=,]+`.

Tipe: [Json](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [StageVariables](#) dari sumber daya `AWS::ApiGatewayV2::Stage`.

Tags

Sebuah peta (string ke string) yang menentukan tanda untuk ditambahkan ke tahap API Gateway ini. Kunci dapat berupa 1 hingga 128 karakter Unicode dan tidak dapat menyertakan awalan `aws:`. Anda dapat menggunakan salah satu karakter berikut: set huruf Unicode, angka, spasi putih, `_`, `.`, `/`, `=`, `+`, dan `-`. Nilai dapat berupa 1 hingga 256 karakter Unicode panjangnya.

Tipe: Peta

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Catatan tambahan: Properti `Tags` membutuhkan AWS SAM untuk mengubah ketentuan OpenAPI Anda, sehingga tanda ditambahkan hanya jika properti `DefinitionBody` ditentukan—tanda tidak ditambahkan jika properti `DefinitionUri` ditentukan. AWS SAM secara otomatis menambahkan tanda `httpapi:createdBy: SAM`. Tanda juga ditambahkan ke sumber daya

`AWS::ApiGatewayV2::Stage` dan sumber daya `AWS::ApiGatewayV2::DomainName` (jika `DomainName` ditentukan).

Nilai Pengembalian

Ref

Bila Anda melewati ID logis dari sumber daya ini ke fungsi Ref intrinsik, Ref mengembalikan ID API dari sumber daya `AWS::ApiGatewayV2::Api` utama, misalnya, `a1bcdef2gh`.

Untuk informasi lebih lanjut tentang penggunaan fungsi Ref, lihat [Ref](#) di Panduan Pengguna AWS CloudFormation.

Contoh-contoh

Sederhana HttpApi

Contoh berikut menunjukkan minimum yang diperlukan untuk mengatur titik akhir HTTP API yang didukung oleh fungsi Lambda. Contoh ini menggunakan API HTTP default yang AWS SAM buat.

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Description: AWS SAM template with a simple API definition
Resources:
  ApiFunction:
    Type: AWS::Serverless::Function
    Properties:
      Events:
        ApiEvent:
          Type: HttpApi
      Handler: index.handler
      InlineCode: |
        def handler(event, context):
          return {'body': 'Hello World!', 'statusCode': 200}
      Runtime: python3.7
    Transform: AWS::Serverless-2016-10-31
```

HttpApi dengan Auth

Contoh berikut menunjukkan cara mengatur otorisasi di titik akhir HTTP API.

YAML

```

Properties:
  FailOnWarnings: true
  Auth:
    DefaultAuthorizer: OAuth2
    Authorizers:
      OAuth2:
        AuthorizationScopes:
          - scope4
        JwtConfiguration:
          issuer: "https://www.example.com/v1/connect/oauth2"
          audience:
            - MyApi
        IdentitySource: "$request.querystring.param"

```

HttpApi dengan definisi OpenAPI

Contoh berikut menunjukkan cara menambahkan ketentuan OpenAPI ke templat.

Perhatikan bahwa AWS SAM mengisi integrasi Lambda yang hilang HttpApi untuk peristiwa yang mereferensikan API HTTP ini. AWS SAM juga menambahkan jalur yang hilang yang menjadi referensi HttpApi acara.

YAML

```

Properties:
  FailOnWarnings: true
  DefinitionBody:
    info:
      version: '1.0'
      title:
        Ref: AWS::StackName
    paths:
      "/":
        get:
          security:
            - OpenIdAuth:
                - scope1
                - scope2
          responses: {}
    openapi: 3.0.1

```

```

securitySchemes:
  OpenIdAuth:
    type: openIdConnect
    x-amazon-apigateway-authorizer:
      identitySource: "$request.querystring.param"
      type: jwt
      jwtConfiguration:
        audience:
          - MyApi
        issuer: https://www.example.com/v1/connect/oidc
        openIdConnectUrl: https://www.example.com/v1/connect/oidc/.well-known/openid-configuration

```

HttpApi dengan pengaturan konfigurasi

Contoh berikut menunjukkan cara menambahkan API HTTP dan konfigurasi persiapan ke templat.

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

Mengonfigurasi otorisasi untuk mengendalikan akses ke API HTTP dari Amazon API Gateway Anda.

Untuk informasi selengkapnya tentang mengonfigurasi akses ke API HTTP, lihat [Mengendalikan dan mengelola akses ke API HTTP di API Gateway](#) di Panduan Developer API Gateway.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Authorizers: OAuth2Authorizer | LambdaAuthorizer  
DefaultAuthorizer: String  
EnableIamAuthorizer: Boolean
```

Properti

Authorizers

Pengotorisasi yang digunakan untuk mengendalikan akses ke API dari API Gateway Anda.

Jenis: [OAuth2Auth2Authorizer](#) | [LambdaAuthorizer](#)

Wajib: Tidak

Default: Tidak ada

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Catatan tambahan: AWS SAM menambahkan otorisasi untuk ketentuan OpenAPI.

DefaultAuthorizer

Tentukan otorisasi default yang digunakan untuk mengotorisasi panggilan API ke API dari API Gateway Anda. Anda dapat menentukan `AWS_IAM` sebagai pengotorisasi default jika `EnableIamAuthorizer` disetel ke `true`. Jika tidak, tentukan otorisasi yang telah Anda tentukan.

Authorizers

Tipe: String

Wajib: Tidak

Default: Tidak ada

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

EnableIamAuthorizer

Tentukan apakah akan menggunakan otorisasi IAM untuk rute API.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

Otorisasi OAuth 2.0

Contoh otorisasi OAuth 2.0

YAML

```
Auth:
  Authorizers:
    OAuth2Authorizer:
      AuthorizationScopes:
        - scope1
        - scope2
      JwtConfiguration:
        issuer: "https://www.example.com/v1/connect/oauth2"
        audience:
          - MyApi
      IdentitySource: "$request.querystring.param"
  DefaultAuthorizer: OAuth2Authorizer
```

Pengotorisasi IAM

Contoh otorisasi IAM

YAML

```
Auth:
  EnableIamAuthorizer: true
```

```
DefaultAuthorizer: AWS_IAM
```

LambdaAuthorizer

Mengonfigurasi otorisasi Lambda untuk mengendalikan akses ke API HTTP dari Amazon API Gateway Anda dengan fungsi AWS Lambda.

Untuk informasi dan contoh selengkapnya, lihat [Bekerja dengan otorisasi AWS Lambda untuk API HTTP](#) di Panduan Developer API Gateway.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
AuthorizerPayloadFormatVersion: String  
EnableFunctionDefaultPermissions: Boolean  
EnableSimpleResponses: Boolean  
FunctionArn: String  
FunctionInvokeRole: String  
Identity: LambdaAuthorizationIdentity
```

Properti

AuthorizerPayloadFormatVersion

Menentukan format muatan yang dikirim ke pemberi otorisasi Lambda API HTTP. Diperlukan untuk Otorisasi Lambda API HTTP.

Ini dilewatkan ke bagian `authorizerPayloadFormatVersion` dari `x-amazon-apigateway-authorizer` di bagian `securitySchemes` dari ketentuan OpenAPI.

Nilai yang valid: `1.0` atau `2.0`

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

EnableFunctionDefaultPermissions

Secara default, sumber daya HTTP API tidak diberikan izin untuk memanggil otorisasi Lambda. Tentukan properti ini `true` untuk secara otomatis membuat izin antara sumber daya HTTP API dan otorisasi Lambda Anda.

Tipe: Boolean

Wajib: Tidak

Nilai default: `false`

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

EnableSimpleResponses

Menentukan apakah otorisasi Lambda mengembalikan respons dalam format sederhana. Secara default, Otorisasi Lambda harus mengembalikan kebijakan (IAM) AWS Identity and Access Management. Jika diaktifkan, Otorisasi Lambda dapat mengembalikan nilai boolean bukan kebijakan IAM.

Ini diteruskan ke bagian `enableSimpleResponses` dari `x-amazon-apigateway-authorizer` di bagian `securitySchemes` dari ketentuan OpenAPI.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

FunctionArn

Amazon Resource Name (ARN) dari fungsi Lambda yang menyediakan otorisasi untuk API.

Ini dilewatkan ke bagian `authorizerUri` dari `x-amazon-apigateway-authorizer` di bagian `securitySchemes` dari ketentuan OpenAPI.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

FunctionInvokeRole

ARN IAM role yang memiliki kredensial yang diperlukan untuk API Gateway untuk memanggil fungsi otorisasi. Tentukan parameter ini jika kebijakan berbasis sumber daya fungsi Anda tidak memberikan izin `lambda:InvokeFunction` API Gateway.

Ini diteruskan ke bagian `authorizerCredentials` dari `x-amazon-apigateway-authorizer` di bagian `securitySchemes` dari ketentuan OpenAPI.

Untuk informasi lebih lanjut, lihat [Buat otorisasi Lambda](#) di Panduan Developer API Gateway.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Identity

Menentukan `IdentitySource` dalam permintaan masuk untuk otorisasi.

Ini diteruskan ke bagian `identitySource` dari `x-amazon-apigateway-authorizer` di bagian `securitySchemes` dari ketentuan OpenAPI.

Jenis: [LambdaAuthorizationIdentity](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

LambdaAuthorizer

LambdaAuthorizer contoh

YAML

```
Auth:
```

```
Authorizers:
  MyLambdaAuthorizer:
    AuthorizerPayloadFormatVersion: 2.0
    FunctionArn:
      Fn::GetAtt:
        - MyAuthFunction
        - Arn
    FunctionInvokeRole:
      Fn::GetAtt:
        - LambdaAuthInvokeRole
        - Arn
    Identity:
      Headers:
        - Authorization
```

LambdaAuthorizationIdentity

Properti penggunaan dapat digunakan untuk menentukan permintaan masuk untuk otorisasi Lambda. IdentitySource Untuk informasi selengkapnya tentang sumber identitas, lihat [Sumber identitas](#) di Panduan Developer API Gateway.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

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

Properti

Context

Mengonversi string konteks yang diberikan ke daftar ekspresi pemetaan dalam format `$context.contextString`.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Headers

Mengonversi header ke daftar ekspresi pemetaan dalam format `$request.header.name`.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

QueryString

Mengonversi string kueri yang diberikan ke daftar ekspresi pemetaan dalam format `$request.querystring.queryString`.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

ReauthorizeEvery

Periode time-to-live (TTL), dalam hitungan detik, yang menentukan berapa lama API Gateway menyimpan hasil otorisasi. Jika Anda menentukan nilai lebih dari 0, API Gateway akan menyimpan respons pengirim. Nilai maksimumnya adalah 3600, atau 1 jam.

Tipe: Integer

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

StageVariables

Mengonversi variabel persiapan yang diberikan ke daftar ekspresi pemetaan dalam format `$stageVariables.stageVariable`.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

LambdaRequestIdentity

Contoh identitas permintaan Lambda

YAML

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

OAuth2Authorizer

Ketentuan untuk otorisasi OAuth 2.0, juga dikenal sebagai otorisasi JSON Web Token (JWT).

Untuk informasi selengkapnya, lihat [Mengendalikan akses ke API HTTP dengan otorisasi JWT](#) di Panduan Developer API Gateway.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
AuthorizationScopes: List
```

IdentitySource: *String*

JwtConfiguration: *Map*

Properti

AuthorizationScopes

Daftar cakupan otorisasi untuk pemberi kuasa ini.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

IdentitySource

Ekspresi sumber identitas untuk otorisasi ini.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

JwtConfiguration

Konfigurasi JWT untuk otorisasi ini.

Ini diteruskan ke bagian `jwtConfiguration` dari `x-amazon-apigateway-authorizer` di bagian `securitySchemes` dari ketentuan OpenAPI.

Note

Properti `issuer` dan `audience` tidak peka huruf besar/kecil dan dapat digunakan baik huruf kecil seperti di OpenAPI atau huruf besar dan seperti dalam. `Issuer Audience`

[AWS::ApiGatewayV2::Authorizer](#)

Tipe: Peta

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

Otorisasi OAuth 2.0

Contoh otorisasi OAuth 2.0

YAML

```
Auth:
  Authorizers:
    OAuth2Authorizer:
      AuthorizationScopes:
        - scope1
      JwtConfiguration:
        issuer: "https://www.example.com/v1/connect/oauth2"
        audience:
          - MyApi
      IdentitySource: "$request.querystring.param"
  DefaultAuthorizer: OAuth2Authorizer
```

HttpApiCorsConfiguration

Mengelola cross-origin resource sharing (CORS) untuk API HTTP Anda. Tentukan domain untuk diizinkan sebagai string atau tentukan kamus dengan konfigurasi Cors tambahan. CATATAN: Cor membutuhkan SAM untuk memodifikasi definisi OpenAPI Anda, sehingga hanya berfungsi dengan OpenApi sebaris yang ditentukan di properti. `DefinitionBody`

Untuk informasi lebih lanjut tentang CORS, lihat [Mengonfigurasi CORS untuk API HTTP](#) di Panduan Developer API Gateway.

Catatan: Jika `HttpApiCorsConfiguration` disetel baik di OpenAPI maupun di tingkat properti, AWS SAM gabungkan mereka dengan properti yang diutamakan.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
AllowCredentials: Boolean  
AllowHeaders: List  
AllowMethods: List  
AllowOrigins: List  
ExposeHeaders: List  
MaxAge: Integer
```

Properti

AllowCredentials

Menentukan apakah kredensial termasuk dalam permintaan CORS.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

AllowHeaders

Merupakan kumpulan header yang diizinkan.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

AllowMethods

Merupakan kumpulan metode HTTP yang diizinkan.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

AllowOrigins

Merupakan kumpulan asal yang diizinkan.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

ExposeHeaders

Merupakan kumpulan header yang terekspos.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

MaxAge

Jumlah detik ketika peramban harus men-cache hasil permintaan penerbangan awal.

Tipe: Integer

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

HttpApiCorsConfiguration

Contoh Konfigurasi Cors API HTTP.

YAML

```
CorsConfiguration:
  AllowOrigins:
    - "https://example.com"
  AllowHeaders:
```



```
- x-apigateway-header
AllowMethods:
  - GET
MaxAge: 600
AllowCredentials: true
```

HttpApiDefinition

Dokumen OpenAPI yang menentukan API.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

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

Properti

Bucket

Nama bucket Amazon S3 tempat file OpenAPI disimpan.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Bucket](#) dari tipe data `AWS::ApiGatewayV2::Api BodyS3Location`.

Key

Kunci Amazon S3 dari file OpenAPI.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Key](#) dari tipe data `AWS::ApiGatewayV2::Api BodyS3Location`.

Version

Untuk objek berversi, versi file OpenAPI.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Version](#) dari tipe data `AWS::ApiGatewayV2::Api BodyS3Location`.

Contoh-contoh

Contoh Uri Ketentuan

Contoh ketentuan API

YAML

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

HttpApiDomainConfiguration

Mengonfigurasi domain kustom untuk API.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
BasePath: List
CertificateArn: String
DomainName: String
EndpointConfiguration: String
MutualTlsAuthentication: MutualTlsAuthentication
OwnershipVerificationCertificateArn: String
Route53: Route53Configuration
```

`SecurityPolicy`: *String*

Properti

BasePath

Daftar basepaths untuk mengonfigurasi dengan nama domain Amazon API Gateway.

Tipe: Daftar

Wajib: Tidak

Default: /

Kompatibilitas AWS CloudFormation: Properti ini mirip dengan properti [ApiMappingKey](#) dari sumber daya `AWS::ApiGatewayV2::ApiMapping`. AWS SAM membuat beberapa sumber daya `AWS::ApiGatewayV2::ApiMapping`, satu per nilai yang ditentukan dalam properti ini.

CertificateArn

Amazon Resource Name (ARN) dari sertifikat terkelola AWS untuk titik akhir nama domain ini. AWS Certificate Manager adalah satu-satunya sumber yang didukung.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [CertificateArn](#) dari sumber daya `AWS::ApiGateway2::DomainNameDomainNameConfiguration`.

DomainName

Nama domain khusus untuk API dari API Gateway Anda. Huruf besar tidak didukung.

AWS SAM membuat sumber daya `AWS::ApiGatewayV2::DomainName` ketika properti ini diatur. Untuk informasi selengkapnya tentang skenario ini, lihat [Properti DomainName ditentukan](#). Untuk informasi tentang sumber daya AWS CloudFormation yang dibuat, lihat [Sumber daya AWS CloudFormation yang dibuat](#).

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [DomainName](#) dari sumber daya `AWS::ApiGateway2::DomainName`.

EndpointConfiguration

Menentukan tipe titik akhir API Gateway untuk memetakan ke domain kustom. Nilai properti ini menentukan bagaimana properti `CertificateArn` dipetakan di AWS CloudFormation.

Satu-satunya nilai valid untuk API HTTP adalah `REGIONAL`.

Tipe: String

Wajib: Tidak

Default: `REGIONAL`

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

MutualTlsAuthentication

Konfigurasi autentikasi Keamanan Lapisan Pengangkutan (TLS) bersama untuk nama domain kustom.

Jenis: [MutualTlsAuthentication](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [MutualTlsAuthentication](#) dari sumber daya `AWS::ApiGatewayV2::DomainName`.

OwnershipVerificationCertificateArn

ARN sertifikat publik yang dikeluarkan oleh ACM untuk memvalidasi kepemilikan domain kustom Anda. Diperlukan hanya ketika Anda mengonfigurasi TLS timbal balik dan Anda menentukan ARN sertifikat CA yang diimpor atau pribadi ACM untuk ARN `CertificateArn`.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [OwnershipVerificationCertificateArn](#) dari tipe data `AWS::ApiGatewayV2::DomainNameConfiguration`.

Route53

Menentukan konfigurasi Amazon Route 53.

Tipe: [Route53Configuration](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

SecurityPolicy

Versi TLS kebijakan keamanan untuk nama domain ini.

Satu-satunya nilai valid untuk API HTTP adalah TLS_1_2.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [SecurityPolicy](#) dari tipe data `AWS::ApiGatewayV2::DomainNameDomainNameConfiguration`.

Contoh-contoh

DomainName

DomainName contoh

YAML

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

Route53Configuration

Mengonfigurasi set catatan Route53 untuk API.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
DistributionDomainName: String  
EvaluateTargetHealth: Boolean  
HostedZoneId: String  
HostedZoneName: String  
IPv6: Boolean  
Region: String  
SetIdentifier: String
```

Properti

DistributionDomainName

Mengonfigurasi distribusi kustom nama domain kustom API.

Tipe: String

Wajib: Tidak

Default: Gunakan distribusi API Gateway.

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [DNSName](#) dari sumber daya `AWS::Route53::RecordSetGroup` `AliasTarget`.

Catatan tambahan: Nama domain [CloudFrontdistribusi](#).

EvaluateTargetHealth

Kapan `EvaluateTargetHealth` benar, catatan alias mewarisi kesehatan AWS sumber daya yang direferensikan, seperti penyeimbang beban Elastic Load Balancing atau catatan lain di zona yang dihosting.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [EvaluateTargetHealth](#) dari sumber daya AWS::Route53::RecordSetGroup `AliasTarget`.

Catatan tambahan: Anda tidak dapat mengatur EvaluateTargetHealth ke true ketika target alias adalah CloudFront distribusi.

HostedZoneId

ID zona yang di-hosting tempat Anda ingin membuat catatan.

Tentukan HostedZoneName atau HostedZoneId, tapi tidak keduanya. Jika Anda memiliki beberapa zona yang di-hosting dengan nama domain yang sama, Anda harus menentukan zona yang di-hosting menggunakan HostedZoneId.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [HostedZoneId](#) dari sumber daya AWS::Route53::RecordSetGroup `RecordSet`.

HostedZoneName

Nama zona yang di-hosting tempat Anda ingin membuat catatan. Anda harus menyertakan titik beruntun (misalnya, `www.example.com.`) sebagai bagian dari HostedZoneName.

Tentukan HostedZoneName atau HostedZoneId, tapi tidak keduanya. Jika Anda memiliki beberapa zona yang di-hosting dengan nama domain yang sama, Anda harus menentukan zona yang di-hosting menggunakan HostedZoneId.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [HostedZoneName](#) dari sumber daya AWS::Route53::RecordSetGroup `RecordSet`.

IpV6

Saat properti ini disetel, AWS SAM buat AWS::Route53::RecordSet sumber daya dan set [Type](#) AAAA untuk yang disediakan HostedZone.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Region

Hanya set catatan sumber daya berbasis latensi: Wilayah Amazon EC2 tempat Anda membuat sumber daya yang direferensikan set catatan sumber daya ini. Sumber daya biasanya adalah sumber daya AWS, seperti instans EC2 atau penyeimbang beban ELB, dan direferensikan oleh alamat IP atau nama domain DNS, bergantung pada jenis catatan.

Saat Amazon Route 53 menerima kueri DNS untuk nama domain dan jenis yang telah Anda buat set catatan sumber daya latensinya, Route 53 memilih set catatan sumber daya latensi yang memiliki latensi terendah antara pengguna akhir dan Wilayah Amazon EC2 terkait. Route 53 kemudian mengembalikan nilai yang terkait dengan set catatan sumber daya yang dipilih.

Perhatikan hal berikut:

- Anda hanya dapat menentukan satu `ResourceRecord` per set catatan sumber daya latensi.
- Anda hanya dapat membuat satu set catatan sumber daya latensi untuk setiap Wilayah Amazon EC2.
- Anda tidak diharuskan membuat set catatan sumber daya latensi untuk semua Wilayah Amazon EC2. Route 53 akan memilih wilayah dengan latensi terbaik dari antara wilayah yang Anda buat set catatan sumber daya latensi.
- Anda tidak dapat membuat set catatan sumber daya non-latensi yang memiliki nilai yang sama untuk elemen `Name` dan `Type` sebagai set catatan sumber daya latensi.

Tipe: String

Wajib: Tidak

AWS CloudFormationkompatibilitas: Properti ini diteruskan langsung ke [Region](#) properti tipe `AWS::Route53::RecordSetGroup RecordSet` data.

SetIdentifier

Set catatan sumber daya yang memiliki kebijakan perutean selain sederhana: Pengenal yang membedakan antara beberapa set catatan sumber daya yang memiliki kombinasi nama dan jenis yang sama, seperti beberapa set catatan sumber daya tertimbang bernama `acme.example.com`

yang memiliki tipe A. Dalam grup set catatan sumber daya yang memiliki nama dan tipe yang sama, nilai `SetIdentifier` harus unik untuk setiap set catatan sumber daya.

Untuk informasi tentang kebijakan perutean, lihat [Memilih kebijakan perutean di Panduan Pengembang Amazon Route 53](#).

Tipe: String

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [SetIdentifier](#) properti tipe `AWS::Route53::RecordSetGroup RecordSet` data.

Contoh-contoh

Contoh Konfigurasi Route 53

Contoh ini menunjukkan cara mengonfigurasi Route 53.

YAML

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

AWS::Serverless::LayerVersion

Membuat Lambda LayerVersion yang berisi pustaka atau kode runtime yang dibutuhkan oleh Fungsi Lambda.

[AWS::Serverless::LayerVersion](#) Resource juga mendukung atribut Metadata resource, sehingga Anda dapat menginstruksikan AWS SAM untuk membangun layer yang disertakan dalam aplikasi Anda. Untuk informasi selengkapnya tentang membangun lapisan, lihat [Membangun lapisan](#).

Catatan Penting: Karena rilis atribut [UpdateReplacePolicy](#) resource di AWS CloudFormation, [AWS::Lambda::LayerVersion](#) (disarankan) menawarkan manfaat yang sama seperti [AWS::Serverless::LayerVersion](#).

Ketika Tanpa Server LayerVersion diubah, SAM juga mengubah id logis sumber daya sehingga yang lama LayerVersions tidak dihapus secara otomatis CloudFormation ketika sumber daya diperbarui.

Note

Ketika Anda menyebarkan ke AWS CloudFormation, AWS SAM mengubah AWS SAM sumber daya Anda menjadi AWS CloudFormation sumber daya. Untuk informasi selengkapnya, lihat [Sumber daya AWS CloudFormation yang dibuat](#).

Sintaks

Untuk mendeklarasikan entitas ini dalam template AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Type: AWS::Serverless::LayerVersion
Properties:
  CompatibleArchitectures: List
  CompatibleRuntimes: List
  ContentUri: String | LayerContent
  Description: String
  LayerName: String
  LicenseInfo: String
  RetentionPolicy: String
```

Properti

CompatibleArchitectures

Menentukan arsitektur set instruksi yang didukung untuk versi lapisan.

Untuk informasi selengkapnya tentang properti ini, lihat [Arsitektur set instruksi Lambda di Panduan Pengembang.AWS Lambda](#)

Nilai yang valid: x86_64, arm64

Tipe: Daftar

Wajib: Tidak

Default: x86_64

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [CompatibleArchitectures](#) properti AWS::Lambda::LayerVersion sumber daya.

CompatibleRuntimes

Daftar runtime yang kompatibel dengan ini LayerVersion.

Tipe: Daftar

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [CompatibleRuntimes](#) properti AWS::Lambda::LayerVersion sumber daya.

ContentUri

Amazon S3 Uri, jalur ke folder lokal, atau LayerContent objek kode lapisan.

Jika Uri atau objek Amazon S3 disediakan, LayerContent Objek Amazon S3 yang direferensikan harus berupa arsip ZIP yang valid yang berisi konten lapisan Lambda.

Jika jalur ke folder lokal disediakan, untuk konten yang akan diubah dengan benar templat harus melalui alur kerja yang mencakup [sam build](#) diikuti oleh salah satu [sam deploy](#) atau [sam package](#). Secara default, jalur relatif diselesaikan sehubungan dengan lokasi AWS SAM template.

Jenis: String | [LayerContent](#)

Wajib: Ya

AWS CloudFormation kompatibilitas: Properti ini mirip dengan [Content](#) properti AWS::Lambda::LayerVersion sumber daya. Properti Amazon S3 nest diberi nama berbeda.

Description

Deskripsi dari lapisan ini.

Tipe: String

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [Description](#) properti AWS::Lambda::LayerVersion sumber daya.

LayerName

Amazon Resource Name (ARN) dari pengguna.

Tipe: String

Wajib: Tidak

Default: Sumber daya id logis

AWS CloudFormation kompatibilitas: Properti ini mirip dengan [LayerName](#) properti `AWS::Lambda::LayerVersion` sumber daya. Jika Anda tidak menentukan nama, id logis dari sumber daya akan digunakan sebagai nama.

LicenseInfo

Informasi tentang lisensi untuk ini `LayerVersion`.

Tipe: String

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [LicenseInfo](#) properti `AWS::Lambda::LayerVersion` sumber daya.

RetentionPolicy

Properti ini menentukan apakah versi lama Anda `LayerVersion` dipertahankan atau dihapus saat Anda menghapus sumber daya. Jika Anda perlu mempertahankan versi lama `LayerVersion` saat memperbarui atau mengganti sumber daya, Anda harus mengaktifkan `UpdateReplacePolicy` atribut tersebut. Untuk informasi tentang melakukan ini, lihat [UpdateReplacePolicyatribut](#) di Panduan AWS CloudFormation Pengguna.

Nilai yang valid: `Retain` atau `Delete`

Tipe: String

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

Catatan tambahan: Saat Anda menentukan `Retain`, AWS SAM tambahkan a [Atribut sumber daya](#) dari `DeletionPolicy`: `Retain` ke `AWS::Lambda::LayerVersion` sumber daya yang diubah.

Nilai Pengembalian

Ref

Ketika ID logis dari sumber daya ini disediakan untuk fungsi Ref intrinsik, ia mengembalikan ARN sumber daya dari Lambda yang mendasarinya. `LayerVersion`

Untuk informasi lebih lanjut tentang menggunakan fungsi Ref, lihat [Ref](#) di Panduan Pengguna AWS CloudFormation .

Contoh-contoh

LayerVersionExample

Contoh dari LayerVersion

YAML

```
Properties:
  LayerName: MyLayer
  Description: Layer description
  ContentUri: 's3://my-bucket/my-layer.zip'
  CompatibleRuntimes:
    - nodejs10.x
    - nodejs12.x
  LicenseInfo: 'Available under the MIT-0 license.'
  RetentionPolicy: Retain
```

LayerContent

Arsip ZIP yang berisi konten lapisan [lapisan Lambda](#).

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

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

Properti

Bucket

Bucket Amazon S3 dari arsip lapisan.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [S3Bucket](#) dari tipe data `AWS::Lambda::LayerVersion Content`.

Key

Kunci Amazon S3 dari arsip lapisan.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [S3Key](#) dari tipe data `AWS::Lambda::LayerVersion Content`.

Version

Untuk objek berversi, versi objek arsip lapisan yang digunakan.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [S3ObjectVersion](#) dari tipe data `AWS::Lambda::LayerVersion Content`.

Contoh-contoh

LayerContent

Contoh konten lapisan

YAML

```
LayerContent:
```

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

AWS::Serverless::SimpleTable

Membuat tabel DynamoDB dengan kunci primer atribut tunggal. Hal ini berguna ketika data hanya perlu diakses melalui kunci primer.

Untuk menggunakan fungsionalitas DynamoDB yang lebih canggih, gunakan [AWS::DynamoDB::Table](#) sumber daya sebagai gantinya.

Note

Ketika Anda menyebarkan ke AWS CloudFormation, AWS SAM mengubah AWS SAM sumber daya Anda menjadi AWS CloudFormation sumber daya. Untuk informasi selengkapnya, lihat [Sumber daya AWS CloudFormation yang dibuat](#).

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Type: AWS::Serverless::SimpleTable  
Properties:  
  PointInTimeRecoverySpecification: PointInTimeRecoverySpecification  
  PrimaryKey: PrimaryKeyObject  
  ProvisionedThroughput: ProvisionedThroughput  
  SSESpecification: SSESpecification  
  TableName: String  
  Tags: Map
```

Properti

PointInTimeRecoverySpecification

Pengaturan yang digunakan untuk mengaktifkan pemulihan titik waktu.

Jenis: [PointInTimeRecoverySpecification](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [PointInTimeRecoverySpecification](#) dari sumber daya AWS::DynamoDB::Table.

PrimaryKey

Nama dan tipe atribut yang akan digunakan sebagai kunci primer tabel. Jika tidak tersedia, kunci primer akan menjadi String dengan nilai id.

Note

Nilai properti ini tidak dapat diubah setelah sumber daya ini dibuat.

Jenis: [PrimaryKeyObject](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

ProvisionedThroughput

Baca dan tulis informasi penyediaan throughput.

Jika ProvisionedThroughput tidak ditentukan BillingMode akan ditentukan sebagai PAY_PER_REQUEST.

Jenis: [ProvisionedThroughput](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [ProvisionedThroughput](#) dari sumber daya AWS::DynamoDB::Table.

SSESpecification

Menentukan pengaturan untuk mengaktifkan enkripsi di sisi server.

Tipe: [SSESpecification](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [SSESpecification](#) dari sumber daya AWS::DynamoDB::Table.

TableName

Nama untuk tabel DynamoDB.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [TableName](#) dari sumber daya AWS::DynamoDB::Table.

Tags

Sebuah peta (string ke string) yang menentukan tag yang akan ditambahkan ke ini SimpleTable. Untuk detail tentang kunci dan nilai tag yang valid, lihat [Tag sumber daya](#) di Panduan AWS CloudFormation Pengguna.

Tipe: Peta

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini mirip dengan properti [Tags](#) dari sumber daya AWS::DynamoDB::Table. Properti Tags di SAM terdiri dari pasangan Key:Value; di CloudFormation dalamnya terdiri dari daftar objek Tag.

Nilai Pengembalian

Ref

Bila ID logis dari sumber daya ini disediakan untuk fungsi intrinsik Ref, ID mengembalikan nama sumber daya dari tabel DynamoDB utama.

Untuk informasi lebih lanjut tentang menggunakan fungsi Ref, lihat [Ref](#) di Panduan Pengguna AWS CloudFormation.

Contoh-contoh

SimpleTableExample

Contoh dari SimpleTable

YAML

```
Properties:
  TableName: my-table
  Tags:
    Department: Engineering
    AppType: Serverless
```

PrimaryKeyObject

Objek yang menggambarkan properti kunci primer.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Name: String
Type: String
```

Properti

Name

Nama atribut dari kunci primer.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [AttributeName](#) dari tipe data `AWS::DynamoDB::Table AttributeDefinition`.

Catatan tambahan: Properti ini juga diteruskan ke [AttributeName](#) properti tipe `AWS::DynamoDB::Table KeySchema` data.

Type

Tipe data untuk kunci primer.

Nilai yang valid:String,Number, Binary

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [AttributeType](#) dari tipe data AWS::DynamoDB::Table AttributeDefinition.

Contoh-contoh

PrimaryKey

Contoh kunci primer.

YAML

```
Properties:
  PrimaryKey:
    Name: MyPrimaryKey
    Type: String
```

AWS::Serverless::StateMachine

Membuat mesin status AWS Step Functions, yang dapat Anda gunakan untuk mengatur fungsi AWS Lambda dan sumber daya AWS lainnya untuk membentuk alur kerja yang kompleks dan tangguh.

Untuk informasi selengkapnya, lihat Step Functions dalam [Panduan Developer AWS Step Functions](#).

Note

Ketika Anda menyebarkan keAWS CloudFormation, AWS SAM mengubah AWS SAM sumber daya Anda menjadi AWS CloudFormation sumber daya. Untuk informasi selengkapnya, lihat [Sumber daya AWS CloudFormation yang dibuat](#).

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Type: AWS::Serverless::StateMachine
Properties:
  AutoPublishAlias: String
  Definition: Map
  DefinitionSubstitutions: Map
  DefinitionUri: String | S3Location
  DeploymentPreference: DeploymentPreference
  Events: EventSource
  Logging: LoggingConfiguration
  Name: String
  PermissionsBoundary: String
  Policies: String | List | Map
  PropagateTags: Boolean
  RolePath: String
  Role: String
  Tags: Map
  Tracing: TracingConfiguration
  Type: String
```

Properti

AutoPublishAlias

Nama alias mesin negara. Untuk mempelajari selengkapnya tentang penggunaan alias mesin status Step Functions, lihat [Mengelola penerapan berkelanjutan dengan versi dan alias](#) di Panduan Pengembang. AWS Step Functions

Gunakan `DeploymentPreference` untuk mengonfigurasi preferensi penerapan untuk alias Anda. Jika Anda tidak menentukan `DeploymentPreference`, AWS SAM akan mengkonfigurasi lalu lintas untuk beralih ke versi mesin status yang lebih baru sekaligus.

AWS SAM menetapkan versi `DeletionPolicy` dan `UpdateReplacePolicy` ke secara `Retain` default. Versi sebelumnya tidak akan dihapus secara otomatis.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Name](#) dari sumber daya `AWS::StepFunctions::StateMachineAlias`.

Definition

Definisi mesin status adalah sebuah objek, di mana format dari objek cocok dengan format file templat AWS SAM Anda, contohnya, JSON atau YAML. Ketentuan mesin status mematuhi [Bahasa Status Amazon](#).

Untuk contoh ketentuan mesin status inline, lihat [Contoh-contoh](#).

Anda harus menyediakan Definition atau DefinitionUri.

Tipe: Peta

Wajib: Bersyarat

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

DefinitionSubstitutions

string-to-string Peta yang menentukan pemetaan untuk variabel placeholder dalam definisi mesin status. Hal ini memungkinkan Anda untuk memasukkan nilai-nilai yang diperoleh pada saat waktu aktif (misalnya, dari fungsi intrinsik) ke dalam ketentuan mesin status.

Tipe: Peta

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini mirip dengan properti [DefinitionSubstitutions](#) dari sumber daya `AWS::StepFunctions::StateMachine`. Jika ada fungsi intrinsik yang ditentukan dalam ketentuan mesin status inline, AWS SAM menambahkan entri ke properti ini untuk memasukkannya ke ketentuan mesin status.

DefinitionUri

URI Amazon Simple Storage Service (Amazon S3) atau jalur file lokal dari ketentuan mesin status yang ditulis di [Bahasa Status Amazon](#).

Jika Anda memberikan jalur file lokal, templat harus melalui alur kerja yang mencakup perintah `sam deploy` atau `sam package` untuk mengubah ketentuan dengan benar. Untuk melakukannya, Anda harus menggunakan CLI AWS SAM versi 0.52.0 atau yang lebih baru.

Anda harus memberikan Definition atau DefinitionUri.

Tipe: String | [S3Location](#)

Wajib: Bersyarat

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [DefinitionS3Location](#) dari sumber daya `AWS::StepFunctions::StateMachine`.

DeploymentPreference

Pengaturan yang mengaktifkan dan mengonfigurasi penerapan mesin status bertahap. Untuk mempelajari selengkapnya tentang penerapan bertahap Step Functions, lihat [Mengelola penerapan berkelanjutan dengan versi dan alias](#) di Panduan Pengembang. AWS Step Functions

Tentukan `AutoPublishAlias` sebelum mengkonfigurasi properti ini.

`DeploymentPreference` Pengaturan Anda akan diterapkan ke alias yang ditentukan dengan `AutoPublishAlias`.

Saat Anda menentukan `DeploymentPreference`, AWS SAM menghasilkan nilai `StateMachineVersionArn` sub-properti secara otomatis.

Jenis: [DeploymentPreference](#)

Wajib: Tidak

AWS CloudFormation kompatibilitas: AWS SAM menghasilkan dan melampirkan nilai `StateMachineVersionArn` properti ke `DeploymentPreference` dan meneruskan `DeploymentPreference` ke [DeploymentPreference](#) properti `AWS::StepFunctions::StateMachineAlias` sumber daya.

Events

Menentukan peristiwa yang memicu mesin status ini. Peristiwa terdiri dari tipe dan satu set properti yang bergantung pada tipenya.

Jenis: [EventSource](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Logging

Menentukan peristiwa riwayat eksekusi mana yang dicatat dan tempatnya dicatat.

Jenis: [LoggingConfiguration](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [LoggingConfiguration](#) dari sumber daya `AWS::StepFunctions::StateMachine`.

Name

Nama mesin status.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [StateMachineName](#) dari sumber daya `AWS::StepFunctions::StateMachine`.

PermissionsBoundary

ARN batas izin untuk digunakan untuk peran eksekusi mesin status ini. Properti ini hanya bekerja jika peran tersebut dibuat untuk Anda.

Tipe: String

Wajib: Tidak


Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [PermissionsBoundary](#) dari sumber daya `AWS::IAM::Role`.

Policies

Kebijakan izin untuk mesin negara bagian ini. Kebijakan akan ditambahkan ke peran eksekusi default mesin negara AWS Identity and Access Management (IAM).

Properti ini menerima satu nilai atau daftar nilai. Nilai yang diizinkan meliputi:

- [AWS SAMtemplat kebijakan](#).
- Kebijakan ARN [AWSterkelola atau kebijakan](#) yang [dikelola pelanggan](#).
- Nama kebijakan AWS terkelola dari [daftar](#) berikut.
- [Kebijakan IAM sebaris](#) yang diformat YAML sebagai peta.

 Note

Jika Anda menyetel `Role` properti, properti ini diabaikan.

Tipe: String | Daftar | Peta

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

PropagateTags

Tunjukkan apakah akan meneruskan tag dari Tags properti ke sumber daya yang Anda [AWS::Serverless::StateMachine](#) hasilkan atau tidak. Tentukan True untuk menyebarkan tag di sumber daya yang Anda hasilkan.

Tipe: Boolean

Wajib: Tidak

Default: False

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Role

ARN IAM role untuk digunakan sebagai peran eksekusi mesin status ini.

Tipe: String

Wajib: Bersyarat

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [RoleArn](#) dari sumber daya `AWS::StepFunctions::StateMachine`.

RolePath

Jalur ke peran eksekusi IAM mesin negara.

Gunakan properti ini saat peran dibuat untuk Anda. Jangan gunakan saat peran ditentukan dengan Role properti.

Tipe: String

Wajib: Bersyarat

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Path](#) dari sumber daya `AWS::IAM::Role`.

Tags

string-to-string Peta yang menentukan tag yang ditambahkan ke mesin status dan peran eksekusi yang sesuai. Untuk informasi tentang kunci dan nilai tag yang valid, lihat properti [Tag](#) dari [AWS::StepFunctions::StateMachine](#) sumber daya.

Tipe: Peta

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini mirip dengan [Tags](#) properti `AWS::StepFunctions::StateMachine` sumber daya. AWS SAM secara otomatis menambahkan `stateMachine:createdBy:SAM` tag ke sumber daya ini, dan ke peran default yang dihasilkan untuknya.

Tracing

Memilih apakah AWS X-Ray diaktifkan untuk mesin status. Untuk informasi lebih lanjut tentang menggunakan X-Ray dengan Step Function, lihat [AWS X-Ray dan Step Functions](#) di Panduan Developer AWS Step Functions.

Jenis: [TracingConfiguration](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [TracingConfiguration](#) dari sumber daya `AWS::StepFunctions::StateMachine`.

Type

Tipe mesin status.

Nilai yang valid: STANDARD atau EXPRESS

Tipe: String

Wajib: Tidak

Default: STANDARD

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [StateMachineType](#) dari sumber daya `AWS::StepFunctions::StateMachine`.

Nilai Pengembalian

Ref

Bila Anda memberikan ID logis dari sumber daya ini ke fungsi intrinsik Ref, Ref mengembalikan Amazon Resource Name (ARN) dari sumber daya `AWS::StepFunctions::StateMachine` utama.

Untuk informasi lebih lanjut tentang penggunaan fungsi Ref, lihat [Ref](#) di Panduan Pengguna AWS CloudFormation.

Fn::GetAtt

`Fn::GetAtt` mengembalikan nilai untuk atribut yang ditentukan dari jenis ini. Berikut ini adalah atribut yang tersedia dan nilai-nilai kembali sampel.

Untuk informasi lebih lanjut cara menggunakan `Fn::GetAtt`, lihat [Fn::GetAtt](#) di Panduan Pengguna AWS CloudFormation

Name

Mengembalikan nama mesin status, seperti `HelloWorld-StateMachine`.

Contoh-contoh

File Ketentuan Mesin Status

Berikut ini adalah contoh mesin status yang ditentukan dengan file ketentuan. File `my_state_machine.asl.json` harus ditulis dalam [Bahasa Status Amazon](#).

Dalam contoh ini, entri `DefinitionSubstitution` mengizinkan mesin status untuk menyertakan sumber daya yang dinyatakan dalam file templat AWS SAM.

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
```

Ketentuan Mesin Status Inline

Berikut ini adalah contoh dari ketentuan mesin status inline.

Dalam contoh ini, file templat AWS SAM ditulis dalam YAML, sehingga ketentuan mesin status juga ada di YAML. Untuk mendeklarasikan ketentuan mesin status inline di JSON, tulis file templat AWS SAM Anda dalam JSON.

YAML

```
MySampleStateMachine:
  Type: AWS::Serverless::StateMachine
  Properties:
    Definition:
      StartAt: MyLambdaState
      States:
        MyLambdaState:
          Type: Task
          Resource: arn:aws:lambda:us-east-1:123456123456:function:my-sample-lambda-app
          End: true
    Role: arn:aws:iam::123456123456:role/service-role/my-sample-role
  Tracing:
    Enabled: true
```

EventSource

Objek yang menggambarkan sumber peristiwa yang memicu mesin status. Setiap peristiwa terdiri dari tipe dan satu set properti yang bergantung pada tipe itu. Untuk informasi lebih lanjut tentang properti dari setiap sumber peristiwa, lihat subtopik yang sesuai dengan tipe tersebut.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

[Properties](#): [Schedule](#) / [ScheduleV2](#) / [CloudWatchEvent](#) / [EventBridgeRule](#) / [Api](#)

Type: *String*

Properti

Properties

Sebuah objek yang menggambarkan properti pemetaan peristiwa ini. Set properti harus sesuai dengan Type yang ditentukan.

Jenis: [Jadwal | ScheduleV2 | | Api CloudWatchEventEventBridgeRule](#)

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Type

Tipe peristiwa.

Nilai yang valid: `Api`, `Schedule`, `ScheduleV2`, `CloudWatchEvent`, `EventBridgeRule`

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

API

Berikut adalah contoh peristiwa dari tipe API ini.

YAML

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

```
RestApiId:  
  Ref: MyApi
```

Api

Objek yang menggambarkan tipe sumber peristiwa Api. Jika sumber daya [AWS::Serverless::Api](#) didefinisikan, nilai jalur dan metode harus sesuai dengan operasi dalam ketentuan OpenAPI API.

Sintaks

Untuk mendeklarasikan entitas ini dalam template AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Auth: ApiStateMachineAuth  
Method: String  
Path: String  
RestApiId: String  
UnescapeMappingTemplate: Boolean
```

Properti

Auth

Konfigurasi otorisasi untuk API, jalur, dan metode ini.

Gunakan properti ini untuk membatalkan pengaturan `DefaultAuthorizer` API untuk jalur individu, ketika `DefaultAuthorizer` tidak ditentukan, atau untuk membatalkan pengaturan `ApiKeyRequired` default.

Jenis: [ApiStateMachineAuth](#)

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

Method

Metode HTTP yang membuat fungsi ini dipanggil.

Tipe: String

Wajib: Ya

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

Path

Jalur URI yang membuat fungsi ini dipanggil. Nilai harus dimulai dengan /.

Tipe: String

Wajib: Ya

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

RestApiId

Pengenal sumber daya RestApi, yang harus berisi operasi dengan jalur dan metode yang diberikan. Biasanya, ini diatur untuk merujuk sumber daya [AWS::Serverless::Api](#) yang ditentukan dalam templat ini.

Jika Anda tidak mendefinisikan properti ini, AWS SAM buat [AWS::Serverless::Api](#) sumber daya default menggunakan OpenApi dokumen yang dihasilkan. Sumber daya tersebut berisi penyatuan dari semua jalur dan metode yang ditentukan oleh peristiwa Api dalam templat yang sama yang tidak menentukan RestApiId.

Properti ini tidak dapat merujuk sumber daya [AWS::Serverless::Api](#) yang ditentukan dalam templat lain.

Tipe: String

Wajib: Tidak

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

UnescapeMappingTemplate

Menghapus tanda kutip tunggal, dengan mengganti \ ' dengan ' , pada input yang diteruskan ke mesin status. Gunakan saat masukan Anda berisi tanda kutip tunggal.

Note

Jika diatur ke `False` dan input Anda berisi tanda kutip tunggal, kesalahan akan terjadi.

Tipe: Boolean

Wajib: Tidak

Default: Salah

AWS CloudFormation kompatibilitas: Properti ini unik AWS SAM dan tidak memiliki AWS CloudFormation padanan.

Contoh-contoh

ApiEvent

Berikut adalah contoh peristiwa dari tipe `Api` ini.

YAML

```
Events:
  ApiEvent:
    Type: Api
    Properties:
      Path: /path
      Method: get
```

ApiStateMachineAuth

Mengonfigurasi otorisasi di tingkat peristiwa, untuk API, jalur, dan metode tertentu.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
ApiKeyRequired: Boolean
```

AuthorizationScopes: *List*
Authorizer: *String*
ResourcePolicy: *ResourcePolicyStatement*

Properti

ApiKeyRequired

Memerlukan kunci API untuk API, jalur, dan metode ini.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

AuthorizationScopes

Cakupan otorisasi yang akan diterapkan ke API, path, dan metode ini.

Cakupan yang Anda tentukan akan membatalkan setiap cakupan yang diterapkan oleh properti `DefaultAuthorizer` jika Anda telah menentukannya.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Authorizer

`Authorizer` untuk mesin status tertentu.

Jika Anda telah menentukan otorisasi global untuk API dan ingin membuat mesin status ini menjadi publik, batalkan otoritas global dengan menetapkan `Authorizer` ke `NONE`.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

ResourcePolicy

Konfigurasi kebijakan sumber daya untuk API dan jalur ini.

Jenis: [ResourcePolicyStatement](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

StateMachine-Auth

Contoh berikut menentukan otorisasi di tingkat mesin status.

YAML

```
Auth:
  ApiKeyRequired: true
  Authorizer: NONE
```

ResourcePolicyStatement

Mengonfigurasi kebijakan sumber daya untuk semua metode dan jalur API. Untuk informasi selengkapnya tentang kebijakan sumber daya, lihat [Mengendalikan akses ke API dengan kebijakan sumber daya API Gateway](#) di Panduan Developer API Gateway.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
AwsAccountBlacklist: List
AwsAccountWhitelist: List
CustomStatements: List
IntrinsicVpcBlacklist: List
IntrinsicVpcWhitelist: List
```

[IntrinsicVpceBlacklist](#): *List*
[IntrinsicVpceWhitelist](#): *List*
[IpRangeBlacklist](#): *List*
[IpRangeWhitelist](#): *List*
[SourceVpcBlacklist](#): *List*
[SourceVpcWhitelist](#): *List*

Properti

AwsAccountBlacklist

Akun AWS yang akan diblokir.

Jenis: Daftar String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

AwsAccountWhitelist

Akun AWS yang akan diizinkan. Untuk contoh penggunaan properti ini, lihat bagian Contoh di bagian bawah halaman ini.

Jenis: Daftar String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

CustomStatements

Daftar pernyataan kebijakan sumber daya kustom untuk diterapkan ke API ini. Untuk contoh penggunaan properti ini, lihat bagian Contoh di bagian bawah halaman ini.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

IntrinsicVpcBlacklist

Daftar virtual private cloud (VPC) yang akan diblokir, dengan setiap VPC ditetapkan sebagai referensi seperti [referensi dinamis](#) atau [fungsi intrinsik](#) Ref. Untuk contoh penggunaan properti ini, lihat bagian Contoh di bagian bawah halaman ini.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

IntrinsicVpcWhitelist

Daftar VPC yang akan diizinkan, dengan setiap VPC ditetapkan sebagai referensi seperti [referensi dinamis](#) atau [fungsi intrinsik](#) Ref.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

IntrinsicVpceBlacklist

Daftar VPC endpoint yang akan diblokir, dengan setiap VPC endpoint ditetapkan sebagai referensi seperti [referensi dinamis](#) atau [fungsi intrinsik](#) Ref.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

IntrinsicVpceWhitelist

Daftar VPC endpoint yang akan diizinkan, dengan setiap VPC endpoint ditetapkan sebagai referensi seperti [referensi dinamis](#) atau [fungsi intrinsik](#) Ref. Untuk contoh penggunaan properti ini, lihat bagian Contoh di bagian bawah halaman ini.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

IpRangeBlacklist

Alamat IP atau jangkauan alamat yang akan diblokir. Untuk contoh penggunaan properti ini, lihat bagian Contoh di bagian bawah halaman ini.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

IpRangeWhitelist

Alamat IP atau jangkauan alamat yang akan diizinkan.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

SourceVpcBlacklist

Sumber VPC atau VPC endpoint yang akan diblokir. Nama VPC sumber harus dimulai dengan "vpc-" dan nama VPC endpoint sumber harus dimulai dengan "vpce-". Untuk contoh penggunaan properti ini, lihat bagian Contoh di bagian bawah halaman ini.

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

SourceVpcWhitelist

VPC sumber atau VPC endpoint yang akan diizinkan. Nama VPC sumber harus dimulai dengan "vpc-" dan nama VPC endpoint sumber harus dimulai dengan "vpce-".

Tipe: Daftar

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

Contoh Kebijakan Sumber Daya

Contoh berikut blok dua alamat IP dan sumber VPC, dan mengizinkan akun AWS.

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

Objek yang menggambarkan tipe sumber peristiwa CloudWatchEvent.

AWS Serverless Application Model(AWS SAM) menghasilkan [AWS::Events::Rule](#) sumber daya saat jenis acara ini disetel.

Catatan penting: [EventBridgeRule](#) adalah tipe sumber peristiwa yang diinginkan yang akan digunakan, bukan `CloudWatchEvent`. `EventBridgeRule` dan `CloudWatchEvent` menggunakan layanan, API, dan sumber daya AWS CloudFormation utama yang sama. Namun, AWS SAM akan menambahkan dukungan untuk fitur baru hanya untuk `EventBridgeRule`.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
EventBusName: String  
Input: String  
InputPath: String  
Pattern: EventPattern
```

Properti

EventBusName

Bus peristiwa yang akan dihubungkan dengan aturan ini. Jika Anda menghilangkan properti ini, AWS SAM menggunakan bus peristiwa default.

Tipe: String

Wajib: Tidak

Default: Bus peristiwa default

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [EventBusName](#) dari sumber daya `AWS::Events::Rule`.

Input

Teks JSON yang valid yang dilewatkan ke target. Jika Anda menggunakan properti ini, tidak ada dari teks peristiwa itu sendiri yang diteruskan ke target.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Input](#) dari sumber daya `AWS::Events::Rule` Target.

InputPath

Bila Anda tidak ingin meneruskan seluruh peristiwa yang sesuai ke target, gunakan properti `InputPath` untuk menggambarkan bagian mana dari peristiwa yang akan diteruskan.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [InputPath](#) dari sumber daya `AWS::Events::Rule` Target.

Pattern

Menjelaskan peristiwa yang dirutekan ke target yang ditentukan. Untuk informasi selengkapnya, lihat [Peristiwa dan Pola Peristiwa EventBridge di Panduan EventBridge Pengguna Amazon](#).

Jenis: [EventPattern](#)

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [EventPattern](#) dari sumber daya `AWS::Events::Rule`.

Contoh-contoh

CloudWatchEvent

Berikut adalah contoh peristiwa dari tipe sumber peristiwa `CloudWatchEvent`.

YAML

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

EventBridgeRule

Objek yang menjelaskan jenis sumber EventBridgeRule peristiwa, yang menetapkan mesin status Anda sebagai target untuk EventBridge aturan Amazon. Untuk informasi selengkapnya, lihat [Apa itu Amazon EventBridge?](#) di Panduan EventBridge Pengguna Amazon.

AWS SAM menghasilkan sumber [AWS::Events::Rule](#) daya saat jenis acara ini disetel.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
DeadLetterConfig: DeadLetterConfig  
EventBusName: String  
Input: String  
InputPath: String  
InputTransformer: InputTransformer  
Pattern: EventPattern  
RetryPolicy: RetryPolicy  
RuleName: String  
State: String  
Target: Target
```

Properti

DeadLetterConfig

Konfigurasi antrian Amazon Simple Queue Service (Amazon SQS) EventBridge tempat pengiriman peristiwa setelah pemanggilan target gagal. Pemanggilan dapat gagal, misalnya, saat mengirim acara ke fungsi Lambda yang tidak ada, atau ketika tidak EventBridge memiliki izin yang cukup untuk memanggil fungsi Lambda. Untuk informasi selengkapnya, lihat [Kebijakan percobaan ulang acara dan menggunakan antrian huruf mati di Panduan Pengguna](#) Amazon. EventBridge

Jenis: [DeadLetterConfig](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [DeadLetterConfig](#) dari tipe data `AWS::Events::Rule` Target. Versi AWS SAM properti ini

termasuk subproperti tambahan, jika Anda ingin AWS SAM membuat antrean surat mati untuk Anda.

EventBusName

Bus peristiwa yang akan dihubungkan dengan aturan ini. Jika Anda menghilangkan properti ini, AWS SAM menggunakan bus peristiwa default.

Tipe: String

Wajib: Tidak

Default: Bus peristiwa default

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [EventBusName](#) dari sumber daya `AWS::Events::Rule`.

Input

Teks JSON yang valid yang dilewatkan ke target. Jika Anda menggunakan properti ini, tidak ada dari teks peristiwa itu sendiri yang diteruskan ke target.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Input](#) dari sumber daya `AWS::Events::Rule Target`.

InputPath

Bila Anda tidak ingin meneruskan seluruh peristiwa yang sesuai ke target, gunakan properti `InputPath` untuk menggambarkan bagian mana dari peristiwa yang akan diteruskan.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [InputPath](#) dari sumber daya `AWS::Events::Rule Target`.

InputTransformer

Pengaturan untuk memungkinkan Anda memberikan input kustom ke target berdasarkan data peristiwa tertentu. Anda dapat mengekstrak satu atau beberapa pasangan nilai kunci dari

peristiwa dan kemudian menggunakan data tersebut untuk mengirim input yang disesuaikan ke target. Untuk informasi selengkapnya, lihat [Transformasi EventBridge input Amazon](#) di Panduan EventBridge Pengguna Amazon.

Jenis: [InputTransformer](#)

Wajib: Tidak

AWS CloudFormationkompatibilitas: Properti ini diteruskan langsung ke [InputTransformer](#) properti tipe `AWS::Events::Rule` Target data.

Pattern

Menjelaskan peristiwa yang dirutekan ke target yang ditentukan. Untuk informasi selengkapnya, lihat [Peristiwa dan Pola Peristiwa EventBridge di](#) Panduan EventBridge Pengguna Amazon.

Jenis: [EventPattern](#)

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [EventPattern](#) dari sumber daya `AWS::Events::Rule`.

RetryPolicy

Objek `RetryPolicy` yang menyertakan informasi tentang pengaturan kebijakan coba lagi. Untuk informasi selengkapnya, lihat [Kebijakan percobaan ulang acara dan menggunakan antrian huruf mati di Panduan Pengguna](#) Amazon. EventBridge

Jenis: [RetryPolicy](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [RetryPolicy](#) dari tipe data `AWS::Events::Rule` Target.

RuleName

Nama aturan.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Name](#) dari sumber daya `AWS::Events::Rule`.

State

Keadaan aturan.

Nilai yang valid: [DISABLED | ENABLED]

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [State](#) dari sumber daya `AWS::Events::Rule`.

Target

Sumber AWS daya yang EventBridge dipanggil ketika aturan dipicu. Anda dapat menggunakan properti ini untuk menentukan ID logis dari target. Jika properti ini tidak ditentukan, AWS SAM menghasilkan ID logis dari target.

Tipe: [Target](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini mirip dengan properti [Targets](#) dari sumber daya `AWS::Events::Rule`. Versi AWS SAM properti ini hanya mengizinkan Anda untuk menentukan ID logis dari satu target.

Contoh-contoh

EventBridgeRule

Berikut adalah contoh tipe sumber peristiwa EventBridgeRule.

YAML

```
EBRule:
  Type: EventBridgeRule
  Properties:
    Input: '{"Key": "Value"}'
    Pattern:
```

```
detail:
  state:
    - terminated
```

DeadLetterConfig

Objek yang digunakan untuk menentukan antrean Amazon Simple Queue Service (Amazon SQS) tempat EventBridge mengirimkan peristiwa setelah pemanggilan target gagal. Ininvokasi dapat gagal, misalnya, ketika mengirim sebuah peristiwa ke status mesin yang tidak ada, izin yang tidak mencukupi untuk memanggil status mesin. Untuk informasi selengkapnya, lihat [Kebijakan coba lagi peristiwa dan menggunakan antrean surat mati](#) di Panduan EventBridge Pengguna Amazon.

Sintaksis

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Arn: String
QueueLogicalId: String
Type: String
```

Properti

Arn

Amazon Resource Name (ARN) dari antrean Amazon SQS yang ditetapkan sebagai target antrean surat mati.

Note

Tentukan `Type` properti atau `Arn` properti, tetapi tidak keduanya.

Jenis: Tali

Diperlukan: Tidak

AWS CloudFormation kompatibilitas: Properti ini diteruskan langsung ke [Arn](#) properti dari tipe `AWS::Events::RuleDeadLetterConfig` data.

QueueLogicalId

Nama kustom dari antrean surat mati yang AWS SAM buat jika Type ditentukan.

Note

Jika Type properti tidak diatur, properti ini diabaikan.

Jenis: Tali

Diperlukan: Tidak

AWS CloudFormation kompatibilitas: Properti ini unik bagi AWS SAM dan tidak memiliki AWS CloudFormation padanan.

Type

Tipe antrean. Bila properti ini diatur, AWS SAM secara otomatis membuat antrean surat mati dan melampirkan [kebijakan berbasis sumber daya](#) yang diperlukan untuk memberikan izin untuk memerintahkan sumber daya untuk mengirim peristiwa ke antrean.

Note

Tentukan Type properti atau Arn properti, tetapi tidak keduanya.

Nilai yang valid: SQS

Jenis: Tali

Diperlukan: Tidak

AWS CloudFormation kompatibilitas: Properti ini unik bagi AWS SAM dan tidak memiliki AWS CloudFormation padanan.

Contoh

DeadLetterConfig

DeadLetterConfig

YAML

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

Target

Mengonfigurasi AWS sumber daya yang EventBridge memanggil ketika suatu aturan dipicu.

Sintaksis

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Id: String
```

Properti

Id

ID logis dari target.

Nilai dari Id dapat mencakup karakter alfanumerik, titik (.), tanda hubung (-), dan garis bawah (_).

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Id](#) dari tipe data `AWS::Events::Rule Target`.

Contoh

Target

YAML

```
EBRule:
```

```
Type: EventBridgeRule
Properties:
  Target:
    Id: MyTarget
```

Schedule

Objek yang menjelaskan jenis sumber Schedule peristiwa, yang menetapkan mesin status Anda sebagai target EventBridge aturan yang memicu jadwal. Untuk informasi lebih lanjut, lihat [Apa itu Amazon EventBridge?](#) di Panduan EventBridge Pengguna Amazon.

AWS Serverless Application Model(AWS SAM) menghasilkan [AWS::Events::Rules](#) sumber daya saat jenis acara ini disetel.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
DeadLetterConfig: DeadLetterConfig
Description: String
Enabled: Boolean
Input: String
Name: String
RetryPolicy: RetryPolicy
RoleArn: String
Schedule: String
State: String
Target: Target
```

Properti

DeadLetterConfig

Konfigurasi antrian Amazon Simple Queue Service (Amazon SQS) EventBridge tempat pengiriman peristiwa setelah pemanggilan target gagal. Pemanggilan dapat gagal, misalnya, saat mengirim acara ke fungsi Lambda yang tidak ada, atau ketika tidak EventBridge memiliki izin yang cukup untuk memanggil fungsi Lambda. Untuk informasi selengkapnya, lihat [Kebijakan percobaan ulang acara dan menggunakan antrian huruf mati di Panduan Pengguna](#) Amazon. EventBridge

Jenis: [DeadLetterConfig](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [DeadLetterConfig](#) dari tipe data `AWS::Events::Rule` Target. Versi AWS SAM dari properti ini termasuk subproperti tambahan, jika Anda menginginkan agar AWS SAM membuat antrean surat mati untuk Anda.

Description

Deskripsi aturan.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Description](#) dari sumber daya `AWS::Events::Rule`.

Enabled

Menunjukkan apakah aturan diaktifkan.

Untuk menonaktifkan aturan, tetapkan properti ini ke `false`.

Note

Tentukan salah satu `Enabled` atau `State` properti, tetapi tidak keduanya.

Tipe: Boolean

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini mirip dengan properti [State](#) dari sumber daya `AWS::Events::Rule`. Jika properti ini ditetapkan ke `true` maka AWS SAM meneruskan `ENABLED`, jika tidak, properti ini meneruskan `DISABLED`.

Input

Teks JSON yang valid diteruskan ke target. Jika Anda menggunakan properti ini, tidak ada dari teks peristiwa itu sendiri yang diteruskan ke target.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Input](#) dari sumber daya `AWS::Events::Rule` Target.

Name

Nama aturan. Jika Anda tidak menetapkan nama, AWS CloudFormation menghasilkan ID fisik unik dan menggunakan ID tersebut sebagai nama aturan.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Name](#) dari sumber daya `AWS::Events::Rule`.

RetryPolicy

Objek `RetryPolicy` yang menyertakan informasi tentang pengaturan kebijakan coba lagi. Untuk informasi selengkapnya, lihat [Kebijakan percobaan ulang acara dan menggunakan antrian huruf mati di Panduan Pengguna](#) Amazon. EventBridge

Jenis: [RetryPolicy](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [RetryPolicy](#) dari tipe data `AWS::Events::Rule` Target.

RoleArn

ARN dari peran IAM yang akan digunakan EventBridge Scheduler untuk target saat jadwal dipanggil.

Jenis: [RoleArn](#)

Diperlukan: Tidak. Jika tidak disediakan, peran baru akan dibuat dan digunakan.

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [RoleArn](#) dari tipe data `AWS::Scheduler::Schedule` Target.

Schedule

Ekspresi penjadwalan yang menentukan kapan dan seberapa sering aturan dijalankan. Untuk informasi lebih lanjut, lihat [Ekspresi Jadwal untuk Aturan](#).

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [ScheduleExpression](#) dari sumber daya `AWS::Events::Rule`.

State

Keadaan aturan.

Nilai yang diterima: DISABLED | ENABLED

Note

Tentukan salah satu Enabled atau State properti, tetapi tidak keduanya.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [State](#) dari sumber daya `AWS::Events::Rule`.

Target

Sumber AWS daya yang EventBridge dipanggil ketika aturan dipicu. Anda dapat menggunakan properti ini untuk menentukan ID logis dari target. Jika properti ini tidak ditentukan, AWS SAM menghasilkan ID logis dari target.

Tipe: [Target](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini mirip dengan properti [Targets](#) dari sumber daya `AWS::Events::Rule`. Versi AWS SAM properti ini hanya mengizinkan Anda untuk menentukan ID logis dari satu target.

Contoh-contoh

CloudWatch Jadwal Acara

CloudWatch Jadwal Contoh Acara

YAML

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

DeadLetterConfig

Objek yang digunakan untuk menentukan antrean Amazon Simple Queue Service (Amazon SQS) EventBridge tempat mengirimkan peristiwa setelah pemanggilan target gagal. Ininvokasi dapat gagal, misalnya, ketika mengirim sebuah peristiwa ke status mesin yang tidak ada, izin yang tidak mencukupi untuk memanggil status mesin. Untuk informasi selengkapnya, lihat [Kebijakan percobaan ulang acara dan menggunakan antrian huruf mati di Panduan Pengguna](#) Amazon. EventBridge

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Arn: String
QueueLogicalId: String
Type: String
```

Properti

Arn

Amazon Resource Name (ARN) dari antrean Amazon SQS yang ditetapkan sebagai target antrean surat mati.

 Note

Tentukan Type properti atau Arn properti, tetapi tidak keduanya.


Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Arn](#) dari tipe data `AWS::Events::Rule DeadLetterConfig`.

QueueLogicalId

Nama kustom dari antrean surat mati yang AWS SAM buat jika Type ditentukan.

 Note

Jika Type properti tidak disetel, properti ini diabaikan.


Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Type

Tipe antrean. Bila properti ini diatur, AWS SAM secara otomatis membuat antrean surat mati dan melampirkan [kebijakan berbasis sumber daya](#) yang diperlukan untuk memberikan izin untuk memerintahkan sumber daya untuk mengirim peristiwa ke antrean.

 Note

Tentukan Type properti atau Arn properti, tetapi tidak keduanya.

Nilai yang valid: SQS

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

Contoh-contoh

DeadLetterConfig

DeadLetterConfig

YAML

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

Target

Mengonfigurasi AWS sumber daya yang EventBridge memanggil ketika aturan dipicu.

Sintaksis

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

YAML

```
Id: String
```

Properti

Id

ID logis dari target.

Nilai dari Id dapat mencakup karakter alfanumerik, titik (.), tanda hubung (-), dan garis bawah (_).

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Id](#) dari tipe data `AWS::Events::Rule Target`.

Contoh

Target

YAML

```
EBRule:
  Type: Schedule
  Properties:
    Target:
      Id: MyTarget
```

ScheduleV2

Objek yang menjelaskan jenis sumber `ScheduleV2` peristiwa, yang menetapkan mesin status Anda sebagai target peristiwa Amazon EventBridge Scheduler yang dipicu pada jadwal. Untuk informasi selengkapnya, lihat [Apa itu Amazon EventBridge Scheduler?](#) di Panduan Pengguna EventBridge Penjadwal.

AWS Serverless Application Model(AWS SAM) menghasilkan [AWS::Scheduler::Schedules](#) sumber daya saat jenis acara ini disetel.

Sintaks

Untuk mendeklarasikan entitas ini di templat AWS Serverless Application Model (AWS SAM) Anda, gunakan sintaks berikut.

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
```

Properti

DeadLetterConfig

Konfigurasi antrian Amazon Simple Queue Service (Amazon SQS) EventBridge tempat pengiriman peristiwa setelah pemanggilan target gagal. Pemanggilan dapat gagal, misalnya, saat mengirim acara ke fungsi Lambda yang tidak ada, atau ketika tidak EventBridge memiliki izin yang cukup untuk memanggil fungsi Lambda. Untuk informasi selengkapnya, lihat [Mengonfigurasi antrian huruf mati untuk EventBridge Penjadwal di Panduan Pengguna Penjadwal](#). EventBridge

Jenis: [DeadLetterConfig](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [DeadLetterConfig](#) dari tipe data `AWS::Scheduler::ScheduleTarget`. Versi AWS SAM properti ini termasuk subproperti tambahan, jika Anda ingin AWS SAM membuat antrian surat mati untuk Anda.

Description

Deskripsi jadwal.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Description](#) dari sumber daya `AWS::Scheduler::Schedule`.

EndDate

Tanggal, di UTC, sebelum jadwal dapat memanggil targetnya. Bergantung pada ekspresi pengulangan jadwal, pemanggilan mungkin berhenti, atau sebelum, yang Anda tentukan `EndDate`.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [EndDate](#) dari sumber daya `AWS::Scheduler::Schedule`.

FlexibleTimeWindow

Mengizinkan konfigurasi jendela di mana jadwal dapat dipanggil.

Jenis: [FlexibleTimeWindow](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [FlexibleTimeWindow](#) dari sumber daya `AWS::Scheduler::Schedule`.

GroupName

Nama grup jadwal untuk dikaitkan dengan jadwal ini. Jika tidak ditentukan, grup default digunakan.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [GroupName](#) dari sumber daya `AWS::Scheduler::Schedule`.

Input

Teks JSON yang valid yang dilewatkan ke target. Jika Anda menggunakan properti ini, tidak ada dari teks peristiwa itu sendiri yang diteruskan ke target.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Input](#) dari sumber daya `AWS::Scheduler::Schedule Target`.

KmsKeyArn

ARN untuk Kunci KMS yang akan digunakan untuk mengenkripsi data pelanggan.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [KmsKeyArn](#) dari sumber daya `AWS::Scheduler::Schedule`.

Name

Nama jadwal. Jika Anda tidak menentukan nama, AWS SAM buat nama dalam format *StateMachine-Logical-IDEvent-Source-Name* dan gunakan ID tersebut untuk nama jadwal.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [Name](#) dari sumber daya `AWS::Scheduler::Schedule`.

OmitName

Secara default, AWS SAM menghasilkan dan menggunakan nama jadwal dalam format *<State-machine-logical event-source-name -ID>< >*. Tetapkan properti ini `true` agar AWS CloudFormation menghasilkan ID fisik yang unik dan gunakan itu untuk nama jadwal sebagai gantinya.

Tipe: Boolean


Wajib: Tidak

Default: `false`

Kompatibilitas AWS CloudFormation: Properti ini unik bagi AWS SAM dan tidak memiliki padanan AWS CloudFormation.

PermissionsBoundary

ARN kebijakan yang digunakan untuk mengatur batas izin untuk peran.

 Note

Jika `PermissionsBoundary` didefinisikan, AWS SAM akan menerapkan batasan yang sama untuk peran IAM target jadwal penjadwal.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [PermissionsBoundary](#) dari sumber daya `AWS::IAM::Role`.

RetryPolicy

Objek `RetryPolicy` yang menyertakan informasi tentang pengaturan kebijakan coba lagi.

Jenis: [RetryPolicy](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [RetryPolicy](#) dari tipe data `AWS::Scheduler::Schedule Target`.

RoleArn

ARN dari peran IAM yang akan digunakan EventBridge Scheduler untuk target saat jadwal dipanggil.

Jenis: [RoleArn](#)

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [RoleArn](#) dari tipe data `AWS::Scheduler::Schedule Target`.

ScheduleExpression

Ekspresi penjadwalan yang menentukan kapan dan seberapa sering jadwal berjalan.

Tipe: String

Wajib: Ya

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [ScheduleExpression](#) dari sumber daya `AWS::Scheduler::Schedule`.

ScheduleExpressionTimezone

Zona waktu di mana ekspresi penjadwalan dievaluasi.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [ScheduleExpressionTimezone](#) dari sumber daya `AWS::Scheduler::Schedule`.

StartDate

Tanggal, di UTC, setelah itu jadwal dapat mulai memanggil target. Bergantung pada ekspresi pengulangan jadwal, pemanggilan mungkin terjadi pada, atau setelah, yang Anda tentukan `StartDate`.

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [StartDate](#) dari sumber daya `AWS::Scheduler::Schedule`.

State

Status jadwal.

Nilai yang diterima: DISABLED | ENABLED

Tipe: String

Wajib: Tidak

Kompatibilitas AWS CloudFormation: Properti ini diteruskan langsung ke properti [State](#) dari sumber daya `AWS::Scheduler::Schedule`.

Contoh-contoh

Contoh dasar mendefinisikan sumber daya `ScheduleV2`

```
StateMachine:
  Type: AWS::Serverless::StateMachine
  Properties:
    Name: MyStateMachine
    Events:
      ScheduleEvent:
        Type: ScheduleV2
        Properties:
```

```
    ScheduleExpression: "rate(1 minute)"
  ComplexScheduleEvent:
    Type: ScheduleV2
    Properties:
      ScheduleExpression: rate(1 minute)
      FlexibleTimeWindow:
        Mode: FLEXIBLE
        MaximumWindowInMinutes: 5
      StartDate: '2022-12-28T12:00:00.000Z'
      EndDate: '2023-01-28T12:00:00.000Z'
      ScheduleExpressionTimezone: UTC
      RetryPolicy:
        MaximumRetryAttempts: 5
        MaximumEventAgeInSeconds: 300
      DeadLetterConfig:
        Type: SQS
  DefinitionUri:
    Bucket: sam-demo-bucket
    Key: my-state-machine.asl.json
    Version: 3
  Policies:
    - LambdaInvokePolicy:
      FunctionName: !Ref MyFunction
```

Untuk informasi referensi untuk semua jenis dan AWS SAM dukungan AWS sumber daya AWS CloudFormation dan properti, lihat [referensi jenis AWS sumber daya dan properti](#) di Panduan AWS CloudFormation Pengguna.

Atribut sumber daya

Atribut sumber daya adalah atribut yang dapat Anda tambahkan ke AWS SAM dan sumber daya AWS CloudFormation untuk mengontrol perilaku dan hubungan tambahan. Untuk informasi selengkapnya tentang atribut sumber daya, lihat [Referensi Sumber Daya](#) di Panduan Pengguna AWS CloudFormation.

AWS SAM mendukung subset atribut sumber daya yang didefinisikan oleh AWS CloudFormation. Dari atribut sumber daya yang didukung, beberapa dari itu hanya disalin ke basis sumber daya AWS CloudFormation yang dibuat sesuai dengan sumber daya AWS SAM, dan beberapa disalin ke semua sumber daya AWS CloudFormation yang dibuat dihasilkan dari sumber daya AWS SAM. Untuk informasi selengkapnya tentang sumber daya AWS CloudFormation yang dibuat dari sumber daya AWS SAM yang sesuai, lihat [Sumber daya AWS CloudFormation yang dibuat](#).

Tabel berikut merangkum atribut sumber daya yang didukung oleh AWS SAM, patuh pada [Pengecualian](#) yang tercantum di bawah ini.

Atribut sumber daya	Beberapa sumber daya yang dibuat pada tujuan
DependsOn Metadata ^{1, 2}	Hanya sumber daya AWS CloudFormation dasar yang dibuat saja. Untuk informasi tentang pemetaan antara sumber daya AWS SAM dan sumber daya AWS CloudFormation basis, lihat Skenario sumber daya AWS CloudFormation yang dibuat .
Kondisi DeletionPolicy UpdateReplacePolicy	Semua sumber daya AWS CloudFormation dibuat dari sumber daya AWS SAM. Untuk informasi tentang skenario untuk sumber daya AWS CloudFormation dibuat, lihat Skenario sumber daya AWS CloudFormation yang dibuat .

Catatan:

1. Untuk informasi selengkapnya tentang penggunaan atribut sumber daya Metadata dengan tipe sumber daya `AWS::Serverless::Function`, lihat [Membangun waktu aktif kustom](#).
2. Untuk informasi selengkapnya tentang penggunaan atribut sumber daya Metadata dengan tipe sumber daya `AWS::Serverless::LayerVersion`, lihat [Membangun lapisan](#).

Pengecualian

Terdapat sejumlah pengecualian untuk aturan atribut sumber daya yang dijelaskan sebelumnya:

- Pada `AWS::Lambda::LayerVersion`, AWS SAM-hanya bidang kustom `RetentionPolicy` yang mengatur `DeletionPolicy` untuk yang sumber daya AWS CloudFormation yang dibuat. Hal ini memiliki prioritas yang lebih tinggi dari `DeletionPolicy` kode itu sendiri. Jika tidak diatur, maka secara default `DeletionPolicy` diatur ke `Retain`.
- Pada `AWS::Lambda::Version`, jika `DeletionPolicy` tidak ditentukan, default-nya adalah `Retain`.
- Untuk skenario ketika `DeploymentPreferences` ditentukan untuk fungsi nirserver, atribut sumber daya tidak disalin ke sumber daya AWS CloudFormation yang dibuat berikut:

- `AWS::CodeDeploy::Application`
- `AWS::CodeDeploy::DeploymentGroup`
- Parameter `AWS::IAM::Role` bernama `CodeDeployServiceRole` yang dibuat untuk skenario ini
- Jika AWS SAM template Anda berisi beberapa fungsi dengan sumber peristiwa API yang dibuat secara implisit, maka fungsi tersebut akan membagikan sumber daya yang dihasilkan `AWS::ApiGateway::RestApi`. Pada skenario ini, jika fungsi memiliki atribut sumber daya yang berbeda, maka untuk sumber daya `AWS::ApiGateway::RestApi` yang dibuat, AWS SAM menyalin atribut sumber daya sesuai dengan daftar prioritas berikut:
 - `UpdateReplacePolicy`:
 1. Retain
 2. Snapshot
 3. Delete
 - `DeletionPolicy`:
 1. Retain
 2. Delete

Fungsi intrinsik

Fungsi intrinsik adalah fungsi bawaan yang memungkinkan Anda menetapkan nilai ke properti yang hanya tersedia saat runtime. AWS SAM memiliki dukungan terbatas untuk properti fungsi intrinsik tertentu, sehingga tidak dapat menyelesaikan beberapa fungsi intrinsik. Oleh karena itu, kami sarankan untuk menambahkan `AWS::LanguageExtensions` transformasi untuk menyelesaikan ini:

Transform:

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

Note

Catatan: Jika Anda menggunakan fungsi intrinsik di `CodeUri` properti, tidak AWS SAM akan dapat mengurai nilai dengan benar. Pertimbangkan untuk menggunakan `AWS::LanguageExtensions` transformasi sebagai gantinya.

Untuk informasi lebih lanjut, lihat [bagian Properti dari `AWS::Serverless::Function`](#).

Untuk informasi selengkapnya tentang fungsi intrinsik, lihat [Referensi Fungsi Intrinsik](#) di Panduan Pengguna AWS CloudFormation .

Sumber daya AWS CloudFormation yang dibuat

Saat AWS Serverless Application Model (AWS SAM) memproses file templat AWS SAM, itu membuat satu atau lebih sumber daya AWS CloudFormation. Kumpulan sumber daya AWS CloudFormation yang dibuat AWS SAM tidak sama tergantung pada skenario yang Anda tentukan. Skenario adalah kombinasi dari sumber daya AWS SAM dan properti yang ditentukan dalam file templat Anda. Anda dapat mereferensikan sumber daya AWS CloudFormation yang dibuat di tempat lain dalam file templat Anda, mirip dengan cara Anda mereferensikan sumber daya yang Anda deklarasikan secara eksplisit dalam file templat Anda.

Misalnya, jika Anda menentukan sumber daya `AWS::Serverless::Function` di file templat AWS SAM Anda, AWS SAM selalu membuat sebuah sumber daya dasar `AWS::Lambda::Function`. Jika Anda juga menentukan properti `AutoPublishAlias` opsional, AWS SAM juga membuat sumber daya `AWS::Lambda::Alias` dan `AWS::Lambda::Version`.

Bagian ini berisi daftar skenario dan sumber daya AWS CloudFormation yang mereka buat, dan menunjukkan cara mereferensikan sumber daya AWS CloudFormation yang dibuat di file templat AWS SAM Anda.

Mereferensikan sumber daya AWS CloudFormation yang dibuat

Anda mempunyai dua pilihan untuk mereferensikan sumber daya AWS CloudFormation yang dibuat di dalam file templat AWS SAM Anda, dengan `LogicalId` atau dengan properti yang direferensikan.

Referensi AWS CloudFormation sumber daya yang dihasilkan oleh `LogicalId`

Sumber daya AWS CloudFormation yang dibuat AWS SAM masing-masing mempunyai [LogicalId](#), yang merupakan pengidentifikasi alfanumerik (A-Z, a-z, 0-9) yang unik dalam file templat. AWS SAM menggunakan `LogicalIds` dari sumber daya AWS SAM dalam file templat Anda untuk membangun `LogicalIds` dari sumber daya AWS CloudFormation yang dibuat. Anda dapat menggunakan `LogicalId` dari sumber daya AWS CloudFormation yang dibuat untuk mengakses properti dari sumber daya tersebut dalam file templat Anda, seperti yang Anda lakukan untuk sumber daya AWS CloudFormation yang telah Anda nyatakan secara eksplisit. Untuk informasi selengkapnya tentang `LogicalIds` dalam AWS CloudFormation serta templat AWS SAM, lihat [Sumber Daya](#) di Panduan Pengguna AWS CloudFormation.

Note

Parameter `LogicalIds` dari beberapa sumber daya yang dibuat mencakup nilai hash unik untuk menghindari konflik namespace. Parameter `LogicalIds` dari sumber daya ini berasal ketika tumpukan dibuat. Anda dapat mengambil mereka hanya setelah tumpukan telah dibuat menggunakan AWS Management Console, AWS CLI, atau salah satu dari AWS SDK. Kami tidak menyarankan mereferensikan sumber daya ini dengan `LogicalId` karena nilai hash mungkin berubah.

Mereferensikan sumber daya AWS CloudFormation yang dibuat dengan properti yang dapat direferensikan

Untuk beberapa sumber daya yang dibuat, AWS SAM menyediakan properti yang dapat direferensikan dari sumber daya AWS SAM. Anda dapat menggunakan properti ini untuk referensi sebuah sumber daya AWS CloudFormation yang dibuat dan propertinya dalam file templat AWS SAM.

Note

Tidak semua sumber daya AWS CloudFormation yang dibuat memiliki properti yang dapat direferensikan. Untuk sumber daya tersebut, Anda harus menggunakan `LogicalId`.

Skenario sumber daya AWS CloudFormation yang dibuat

Tabel berikut merangkum sumber daya dan properti AWS SAM yang membentuk skenario yang membuat sumber daya AWS CloudFormation. Topik dalam kolom Skenario memberikan detail tentang sumber daya AWS CloudFormation tambahan yang dibuat AWS SAM untuk skenario tersebut.

Sumber daya AWS SAM	Sumber daya AWS CloudFormation dasar	Skenario
<u>AWS::Serverless::Api</u>	<u>AWS::ApiGateway::RestApi</u>	<ul style="list-style-type: none"> • <u>DomainName</u> properti ditentukan • <u>UsagePlan</u> properti ditentukan
<u>AWS::Serverless::Application</u>	<u>AWS::CloudFormation::Stack</u>	<ul style="list-style-type: none"> • Selain membuat sumber daya AWS CloudFormation dasar, tidak ada skenario tambahan untuk sumber daya nirserver ini.
		<ul style="list-style-type: none"> • <u>AutoPublishAlias</u> properti ditentukan • <u>Properti peran</u> tidak ditentukan • <u>DeploymentPreference</u> properti ditentukan • <u>Sumber peristiwa Api</u> ditentukan • <u>Sumber HttpApi acara</u> ditentukan
<u>AWS::Serverless::Function</u>	<u>AWS::Lambda::Function</u>	<ul style="list-style-type: none"> • <u>Sumber peristiwa streaming</u> ditentukan • <u>Sumber event bridge (atau bus peristiwa)</u> ditentukan • <u>Sumber IoTRule acara</u> ditentukan • <u>OnSuccess(atau OnFailure) properti</u> ditentukan untuk acara Amazon SNS • <u>OnSuccess(atau OnFailure) properti</u> ditentukan untuk peristiwa Amazon SQS

Sumber daya AWS SAM	Sumber daya AWS CloudFormation dasar	Skenario
<u>AWS::Serverless::HttpApi</u>	<u>AWS::ApiGatewayV2::Api</u>	<ul style="list-style-type: none"> • <u>Properti StageName ditentukan</u> • <u>Properti StageName tidak ditentukan</u> • <u>Properti DomainName ditentukan</u>
<u>AWS::Serverless::LayerVersion</u>	<u>AWS::Lambda::LayerVersion</u>	<ul style="list-style-type: none"> • Selain membuat sumber daya AWS CloudFormation dasar, tidak ada skenario tambahan untuk sumber daya nirserver ini.
<u>AWS::Serverless::SimpleTable</u>	<u>AWS::DynamoDB::Table</u>	<ul style="list-style-type: none"> • Selain membuat sumber daya AWS CloudFormation dasar, tidak ada skenario tambahan untuk sumber daya nirserver ini.
<u>AWS::Serverless::StateMachine</u>	<u>AWS::StepFunctions::StateMachine</u>	<ul style="list-style-type: none"> • <u>Properti peran tidak ditentukan</u> • <u>Sumber peristiwa Api ditentukan</u> • <u>Sumber event bridge (atau bus peristiwa) ditentukan</u>

Topik

- [AWS CloudFormation sumber daya yang dihasilkan saat AWS::Serverless::Api ditentukan](#)
- [Sumber daya AWS CloudFormation dihasilkan ketika AWS::Serverless::Application ditentukan](#)
- [AWS CloudFormation sumber daya yang dihasilkan saat Anda menentukan AWS::Serverless::Connector](#)
- [AWS CloudFormation sumber daya yang dihasilkan saat AWS::Serverless::Function ditentukan](#)
- [AWS CloudFormation sumber daya yang dihasilkan saat AWS::Serverless::GraphQLApi ditentukan](#)
- [Sumber daya AWS CloudFormation dibuat ketika AWS::Serverless::HttpApi ditentukan](#)

- [AWS CloudFormation sumber daya yang dihasilkan saat `AWS::Serverless::LayerVersion` ditentukan](#)
- [AWS CloudFormation sumber daya yang dihasilkan saat `AWS::Serverless::SimpleTable` ditentukan](#)
- [AWS CloudFormation sumber daya yang dibuat saat `AWS::Serverless::StateMachine` ditentukan](#)

AWS CloudFormation sumber daya yang dihasilkan saat `AWS::Serverless::Api` ditentukan

Saat `AWS::Serverless::Api` ditentukan, AWS Serverless Application Model (AWS SAM) selalu membuat sebuah `AWS::ApiGateway::RestApi` dari sumber daya AWS CloudFormation dasar. Selain itu, ini juga selalu membuat `AWS::ApiGateway::Stage` dan sebuah sumber daya `AWS::ApiGateway::Deployment`.

`AWS::ApiGateway::RestApi`

LogicalId: `<api-LogicalId>`

Properti yang dapat direferensikan: N/A (Anda harus menggunakan `LogicalId` untuk mereferensikan sumber daya AWS CloudFormation ini)

`AWS::ApiGateway::Stage`

LogicalId: `<api-LogicalId><stage-name>Stage`

`<stage-name>` adalah string dari properti `StageName` yang diatur kepadanya. Misalnya, jika Anda mengatur `StageName` ke `Gamma`, sehingga `LogicalId` adalah `MyRestApiGammaStage`.

Properti yang dapat direferensikan: `<api-LogicalId>.Stage`

`AWS::ApiGateway::Deployment`

LogicalId: `<api-LogicalId>Deployment<sha>`

`<sha>` adalah nilai hash unik yang dibuat ketika tumpukan dibuat. Misalnya, `MyRestApiDeployment926eeb5ff1`.

Properti yang dapat direferensikan: `<api-LogicalId>.Deployment`

Selain dari sumber daya AWS CloudFormation, ketika `AWS::Serverless::Api` ditentukan, AWS SAM membuat sumber daya AWS CloudFormation tambahan untuk skenario berikut.

Skenario

- [DomainNameproperty ditentukan](#)
- [UsagePlanproperty ditentukan](#)

DomainNameproperty ditentukan

Saat property DomainName dari property Domain yang berasal dari `AWS::Serverless::Api` telah ditentukan, AWS SAM membuat sumber daya AWS CloudFormation `AWS::ApiGateway::DomainName` dari sumber daya AWS CloudFormation.

AWS::ApiGateway::DomainName

LogicalId: `ApiGatewayDomainName<sha>`

`<sha>` adalah nilai hash unik yang dibuat ketika tumpukan dibuat. Misalnya: `ApiGatewayDomainName926eeb5ff1`.

Property yang dapat direferensikan: `<api-LogicalId>.DomainName`

UsagePlanproperty ditentukan

Saat property UsagePlan dari property Auth yang berasal dari `AWS::Serverless::Api` telah ditentukan, AWS SAM membuat sumber daya AWS CloudFormation berikut:

`AWS::ApiGateway::UsagePlan`, `AWS::ApiGateway::UsagePlanKey`, dan `AWS::ApiGateway::ApiKey`.

AWS::ApiGateway::UsagePlan

LogicalId: `<api-LogicalId>UsagePlan`

Property yang dapat direferensikan: `<api-LogicalId>.UsagePlan`

AWS::ApiGateway::UsagePlanKey

LogicalId: `<api-LogicalId>UsagePlanKey`

Property yang dapat direferensikan: `<api-LogicalId>.UsagePlanKey`

AWS::ApiGateway::ApiKey

LogicalId: `<api-LogicalId>ApiKey`

Property yang dapat direferensikan: `<api-LogicalId>.ApiKey`

Sumber daya AWS CloudFormation dihasilkan ketika `AWS::Serverless::Application` ditentukan

Saat `AWS::Serverless::Application` ditentukan, AWS Serverless Application Model (AWS SAM) membuat sebuah `AWS::CloudFormation::Stack` dari sumber daya AWS CloudFormation dasar.

`AWS::CloudFormation::Stack`

LogicalId: `<application-LogicalId>`

Properti yang dapat direferensikan: N/A (Anda harus menggunakan `LogicalId` referensi ini AWS CloudFormation sumber daya)

AWS CloudFormation sumber daya yang dihasilkan saat Anda menentukan `AWS::Serverless::Connector`

Note

Ketika Anda mendefinisikan konektor melalui `Connectors` properti tertanam, pertama-tama diubah menjadi `AWS::Serverless::Connector` sumber daya sebelum menghasilkan sumber daya ini.

Saat Anda menentukan `AWS::Serverless::Connector` sumber daya dalam AWS SAM templat, AWS SAM hasilkan AWS CloudFormation sumber daya berikut sesuai kebutuhan.

`AWS::IAM::ManagedPolicy`

LogicalId: `<connector-LogicalId>Policy`

Properti yang dapat direferensikan: N/A (Untuk mereferensikan AWS CloudFormation sumber daya ini, Anda harus menggunakan.) `LogicalId`

`AWS::SNS::TopicPolicy`

LogicalId: `<connector-LogicalId>TopicPolicy`

Properti yang dapat direferensikan: N/A (Untuk mereferensikan AWS CloudFormation sumber daya ini, Anda harus menggunakan.) `LogicalId`

AWS::SQS::QueuePolicy

LogicalId: *<connector-LogicalId>*QueuePolicy

Properti yang dapat direferensikan: N/A (Untuk mereferensikan AWS CloudFormation sumber daya ini, Anda harus menggunakan.) LogicalId

AWS::Lambda::Permission

LogicalId: *<connector-LogicalId>**<permission>*LambdaPermission

<permission> adalah izin yang ditentukan oleh Permissions properti. Misalnya, Write.

Properti yang dapat direferensikan: N/A (Untuk mereferensikan AWS CloudFormation sumber daya ini, Anda harus menggunakan.) LogicalId

AWS CloudFormation sumber daya yang dihasilkan saat AWS::Serverless::Function ditentukan

Saat `AWS::Serverless::Function` ditentukan, AWS Serverless Application Model (AWS SAM) selalu membuat sebuah `AWS::Lambda::Function` dari sumber daya AWS CloudFormation dasar.

AWS::Lambda::Function

LogicalId: *<function-LogicalId>*

Properti yang dapat direferensikan: N/A (Anda harus menggunakan LogicalId untuk mereferensikan sumber daya AWS CloudFormation ini)

Selain dari sumber daya AWS CloudFormation ini, ketika `AWS::Serverless::Function` telah ditentukan, AWS SAM juga membuat sumber daya AWS CloudFormation untuk skenario berikut.

Skenario

- [AutoPublishAlias properti ditentukan](#)
- [Properti peran tidak ditentukan](#)
- [DeploymentPreference properti ditentukan](#)
- [Sumber peristiwa Api ditentukan](#)
- [Sumber HttpApi acara ditentukan](#)
- [Sumber peristiwa streaming ditentukan](#)

- [Sumber event bridge \(atau bus peristiwa\) ditentukan](#)
- [Sumber IoTRule acara ditentukan](#)
- [OnSuccess\(atau OnFailure\) properti ditentukan untuk acara Amazon SNS](#)
- [OnSuccess\(atau OnFailure\) properti ditentukan untuk peristiwa Amazon SQS](#)

AutoPublishAlias properti ditentukan

Saat properti `AutoPublishAlias` dari `AWS::Serverless::Function` telah ditentukan, AWS SAM membuat sumber daya AWS CloudFormation berikut: `AWS::Lambda::Alias` dan `AWS::Lambda::Version`.

AWS::Lambda::Alias

LogicalId: <function-LogicalId>Alias<alias-name>

<alias-name> adalah string dari `AutoPublishAlias` yang diatur kepadanya. Misalnya, jika Anda mengatur `AutoPublishAlias` kelive, `LogicalId` adalah: *MyFunctionAlias hidup*.

Properti yang dapat direferensikan: *<function-LogicalId>.Alias*

AWS::Lambda::Version

LogicalId: <function-LogicalId>Version<sha>

<sha> adalah nilai hash unik yang dibuat ketika tumpukan dibuat. Misalnya, *MyFunctionVersi 926eeb5ff1*.

Properti yang dapat direferensikan: *<function-LogicalId>.Version*

Properti peran tidak ditentukan

Saat properti `Role` dari `AWS::Serverless::Function` tidak ditentukan, AWS SAM membuat sebuah `AWS::IAM::Role` dari sumber daya AWS CloudFormation.

AWS::IAM::Role

LogicalId: <function-LogicalId>Role

Properti yang dapat direferensikan: N/A (Anda harus menggunakan `LogicalId` untuk mereferensikan sumber daya AWS CloudFormation ini)

DeploymentPreference properti ditentukan

Saat properti DeploymentPreference dari `AWS::Serverless::Function` telah ditentukan, AWS SAM membuat sumber daya AWS CloudFormation sumber daya berikut: `AWS::CodeDeploy::Application` dan `AWS::CodeDeploy::DeploymentGroup`. Selain itu, jika properti Role dari objek DeploymentPreference tidak ditentukan, AWS SAM juga membuat sebuah `AWS::IAM::Role` dari sumber daya AWS CloudFormation.

AWS::CodeDeploy::Application

LogicalId: ServerlessDeploymentApplication

Properti yang dapat direferensikan: N/A (Anda harus menggunakan LogicalId untuk mereferensikan sumber daya AWS CloudFormation ini)

AWS::CodeDeploy::DeploymentGroup

LogicalId: `<function-LogicalId>DeploymentGroup`

Properti yang dapat direferensikan: N/A (Anda harus menggunakan LogicalId untuk mereferensikan sumber daya AWS CloudFormation ini)

AWS::IAM::Role

LogicalId: CodeDeployServiceRole

Properti yang dapat direferensikan: N/A (Anda harus menggunakan LogicalId untuk mereferensikan sumber daya AWS CloudFormation ini)

Sumber peristiwa Api ditentukan

Saat properti Event dari `AWS::Serverless::Function` diatur ke Api, tetapi properti RestApiId tidak ditentukan, AWS SAM membuat `AWS::ApiGateway::RestApi` dari sumber daya AWS CloudFormation.

AWS::ApiGateway::RestApi

LogicalId: ServerlessRestApi

Properti yang dapat direferensikan: N/A (Anda harus menggunakan LogicalId untuk mereferensikan sumber daya AWS CloudFormation ini)

Sumber HttpApi acara ditentukan

Saat properti Event dari `AWS::Serverless::Function` diatur ke `HttpApi`, tetapi properti `ApiId` tidak ditentukan, AWS SAM membuat `AWS::ApiGatewayV2::Api` dari sumber daya AWS CloudFormation.

AWS::ApiGatewayV2::Api

LogicalId: `ServerlessHttpApi`

Properti yang dapat direferensikan: N/A (Anda harus menggunakan `LogicalId` untuk mereferensikan sumber daya AWS CloudFormation ini)

Sumber peristiwa streaming ditentukan

Saat properti Event dari `AWS::Serverless::Function` diatur ke salah satu tipe streaming, AWS SAM membuat `AWS::Lambda::EventSourceMapping` dari sumber daya AWS CloudFormation. Ini berlaku untuk jenis berikut: `DynamoDB`, `Kinesis`, `MQ`, `MSK`, dan `SQS`.

AWS::Lambda::EventSourceMapping

LogicalId: `<function-LogicalId><event-LogicalId>`

Properti yang dapat direferensikan: N/A (Anda harus menggunakan `LogicalId` untuk mereferensikan sumber daya AWS CloudFormation ini)

Sumber event bridge (atau bus peristiwa) ditentukan

Saat properti Event dari `AWS::Serverless::Function` diatur ke salah satu jenis event bridge (atau bus peristiwa), AWS SAM membuat `AWS::Events::Rule` dari sumber daya AWS CloudFormation. Ini berlaku untuk tipe berikut: `EventBridgeRule`, `Schedule`, dan `CloudWatchEvents`.

AWS::Events::Rule

LogicalId: `<function-LogicalId><event-LogicalId>`

Properti yang dapat direferensikan: N/A (Anda harus menggunakan `LogicalId` untuk mereferensikan sumber daya AWS CloudFormation ini)

Sumber `IoT::TopicRule` acara ditentukan

Saat properti `Event` dari `AWS::Serverless::Function` diatur ke `IoT::TopicRule`, AWS SAM membuat `AWS::IoT::TopicRule` dari sumber daya AWS CloudFormation.

AWS::IoT::TopicRule

LogicalId: `<function-LogicalId><event-LogicalId>`

Properti yang dapat direferensikan: N/A (Anda harus menggunakan `LogicalId` untuk mereferensikan sumber daya AWS CloudFormation ini)

`OnSuccess`(atau `OnFailure`) properti ditentukan untuk acara Amazon SNS

Saat properti `OnSuccess` (atau `OnFailure`) dari properti `DestinationConfig` yang berasal dari properti `EventInvokeConfig` dari `AWS::Serverless::Function` telah ditentukan, dan tipe tujuannya adalah SNS namun tujuan ARN tidak ditentukan, AWS SAM membuat sumber daya AWS CloudFormation berikut: `AWS::Lambda::EventInvokeConfig` dan `AWS::SNS::Topic`.

AWS::Lambda::EventInvokeConfig

LogicalId: `<function-LogicalId>EventInvokeConfig`

Properti yang dapat direferensikan: N/A (Anda harus menggunakan `LogicalId` untuk mereferensikan sumber daya AWS CloudFormation ini)

AWS::SNS::Topic

LogicalId: `<function-LogicalId>OnSuccessTopic`
(atau `<function-LogicalId>OnFailureTopic`)

Properti yang dapat direferensikan: `<function-LogicalId>.DestinationTopic`

Jika `OnSuccess` dan `OnFailure` telah ditentukan untuk peristiwa Amazon SNS, agar membedakan antara sumber daya yang dibuat, Anda harus menggunakan `LogicalId`.

`OnSuccess`(atau `OnFailure`) properti ditentukan untuk peristiwa Amazon SQS

Saat properti `OnSuccess` (atau `OnFailure`) dari properti `DestinationConfig` yang berasal dari properti `EventInvokeConfig` dari `AWS::Serverless::Function` telah ditentukan, dan tipe

tujuannya adalah SQS namun tujuan ARN tidak ditentukan, AWS SAM membuat sumber daya AWS CloudFormation berikut: `AWS::Lambda::EventInvokeConfig` dan `AWS::SQS::Queue`.

AWS::Lambda::EventInvokeConfig

LogicalId: `<function-LogicalId>EventInvokeConfig`

Properti yang dapat direferensikan: N/A (Anda harus menggunakan `LogicalId` untuk mereferensikan sumber daya AWS CloudFormation ini)

AWS::SQS::Queue

LogicalId: `<function-LogicalId>OnSuccessQueue`
(atau `<function-LogicalId>OnFailureQueue`)

Properti yang dapat direferensikan: `<function-LogicalId>.DestinationQueue`

Jika `OnSuccess` dan `OnFailure` telah ditentukan untuk peristiwa Amazon SQS, agar membedakan antara sumber daya yang dibuat, Anda harus menggunakan `LogicalId`.

AWS CloudFormation sumber daya yang dihasilkan saat `AWS::Serverless::GraphQLApi` ditentukan

Saat Anda menentukan `AWS::Serverless::GraphQLApi` sumber daya dalam templat AWS Serverless Application Model (AWS SAM), AWS SAM selalu buat AWS CloudFormation sumber daya dasar berikut.

AWS::AppSync::DataSource

LogicalId: `<graphqlapi-LogicalId><datasource-RelativeId><datasource-Type>DataSource`

Properti yang dapat direferensikan: N/A (Anda harus menggunakan `LogicalId` untuk mereferensikan sumber daya AWS CloudFormation ini)

AWS::AppSync::FunctionConfiguration

LogicalId: `<graphqlapi-LogicalId><function-RelativeId>`

Properti yang dapat direferensikan: N/A (Anda harus menggunakan `LogicalId` untuk mereferensikan sumber daya AWS CloudFormation ini)

AWS::AppSync::GraphQLApi

LogicalId: <graphqlapi-LogicalId>

Properti yang dapat direferensikan: N/A (Anda harus menggunakan LogicalId untuk mereferensikan sumber daya AWS CloudFormation ini)

AWS::AppSync::GraphQLSchema

LogicalId: <graphqlapi-LogicalId>Schema

Properti yang dapat direferensikan: N/A (Anda harus menggunakan LogicalId untuk mereferensikan sumber daya AWS CloudFormation ini)

AWS::AppSync::Resolver

LogicalId: <graphqlapi-LogicalId><OperationType><resolver-RelativeId>

Properti yang dapat direferensikan: N/A (Anda harus menggunakan LogicalId untuk mereferensikan sumber daya AWS CloudFormation ini)

Selain AWS CloudFormation sumber daya ini, ketika AWS::Serverless::GraphQLApi ditentukan, juga AWS SAM dapat menghasilkan AWS CloudFormation sumber daya berikut.

AWS::AppSync::ApiCache

LogicalId: <graphqlapi-LogicalId>ApiCache

Properti yang dapat direferensikan: N/A (Anda harus menggunakan LogicalId untuk mereferensikan sumber daya AWS CloudFormation ini)

AWS::AppSync::ApiKey

LogicalId: <graphqlapi-LogicalId><apikey-RelativeId>

Properti yang dapat direferensikan: N/A (Anda harus menggunakan LogicalId untuk mereferensikan sumber daya AWS CloudFormation ini)

AWS::AppSync::DomainName

LogicalId: <graphqlapi-LogicalId>DomainName

Properti yang dapat direferensikan: N/A (Anda harus menggunakan LogicalId untuk mereferensikan sumber daya AWS CloudFormation ini)

AWS::AppSync::DomainNameApiAssociation

LogicalId: <graphqlapi-LogicalId>DomainNameApiAssociation

Properti yang dapat direferensikan: N/A (Anda harus menggunakan LogicalId untuk mereferensikan sumber daya AWS CloudFormation ini)

AWS SAM juga dapat menggunakan `AWS::Serverless::Connector` sumber daya untuk memberikan izin. Lihat informasi yang lebih lengkap di [AWS CloudFormation sumber daya yang dihasilkan saat Anda menentukan AWS::Serverless::Connector](#).

Sumber daya AWS CloudFormation dibuat ketika AWS::Serverless::HttpApi ditentukan

Saat `AWS::Serverless::HttpApi` ditentukan, AWS Serverless Application Model (AWS SAM) membuat sebuah `AWS::ApiGatewayV2::Api` dari sumber daya AWS CloudFormation dasar.

AWS::ApiGatewayV2::Api

LogicalId: <httpapi-LogicalId>

Properti yang dapat direferensikan: N/A (Anda harus menggunakan LogicalId referensi ini AWS CloudFormation sumber daya)

Selain dari sumber daya AWS CloudFormation ini, ketika `AWS::Serverless::HttpApi` telah ditentukan, AWS SAM juga membuat sumber daya AWS CloudFormation untuk skenario berikut:

Skenario

- [Properti StageName ditentukan](#)
- [Properti StageName tidak ditentukan](#)
- [Properti DomainName ditentukan](#)

Properti StageName ditentukan

Saat properti `StageName` dari `AWS::Serverless::HttpApi` telah ditentukan, AWS SAM membuat sebuah `AWS::ApiGatewayV2::Stage` dari sumber daya AWS CloudFormation.

AWS::ApiGatewayV2::Stage

LogicalId: `<httpapi-LogicalId><stage-name>Stage`

`<stage-name>` adalah string dari properti `StageName` yang diatur kepadanya. Misalnya, jika Anda mengatur `StageName` kepada `Gamma`, yang `LogicalId` adalah: `MyHttpAPIGammaPanggung`.

Properti yang dapat direferensikan: `<httpapi-LogicalId>.Stage`

Properti `StageName` tidak ditentukan

Saat properti `StageName` dari `AWS::Serverless::HttpApi` tidak ditentukan, AWS SAM membuat `AWS::ApiGatewayV2::Stage` dari sumber daya AWS CloudFormation.

AWS::ApiGatewayV2::Stage

LogicalId: `<httpapi-LogicalId>ApiGatewayDefaultStage`

Properti yang dapat direferensikan: `<httpapi-LogicalId>.Stage`

Properti `DomainName` ditentukan

Saat properti `DomainName` dari properti `Domain` yang berasal dari `AWS::Serverless::HttpApi` telah ditentukan, AWS SAM membuat `AWS::ApiGatewayV2::DomainName` dari sumber daya AWS CloudFormation.

AWS::ApiGatewayV2::DomainName

LogicalId: `ApiGatewayDomainNameV2<sha>`

`<sha>` adalah nilai hash unik yang dibuat ketika tumpukan dibuat. Misalnya, `ApiGatewayDomainNameV2926eeb5ff1`.

Properti yang dapat direferensikan: `<httpapi-LogicalId>.DomainName`

AWS CloudFormation sumber daya yang dihasilkan saat `AWS::Serverless::LayerVersion` ditentukan

Saat `AWS::Serverless::LayerVersion` ditentukan, AWS Serverless Application Model (AWS SAM) membuat sebuah `AWS::Lambda::LayerVersion` dari sumber daya AWS CloudFormation dasar.

AWS::Lambda::LayerVersion

LogicalId: <layerversion-LogicalId>

Properti yang dapat direferensikan: N/A (Anda harus menggunakan `LogicalId` referensi ini AWS CloudFormation sumber daya)

AWS CloudFormation sumber daya yang dihasilkan saat `AWS::Serverless::SimpleTable` ditentukan

Saat `AWS::Serverless::SimpleTable` ditentukan, AWS Serverless Application Model (AWS SAM) membuat sebuah `AWS::DynamoDB::Table` dari sumber daya AWS CloudFormation dasar.

AWS::DynamoDB::Table

LogicalId: <simpletable-LogicalId>

Properti yang dapat direferensikan: N/A (Anda harus menggunakan `LogicalId` referensi ini AWS CloudFormation sumber daya)

AWS CloudFormation sumber daya yang dibuat saat `AWS::Serverless::StateMachine` ditentukan

Saat `AWS::Serverless::StateMachine` ditentukan, AWS Serverless Application Model (AWS SAM) membuat sebuah `AWS::StepFunctions::StateMachine` dari sumber daya AWS CloudFormation dasar.

AWS::StepFunctions::StateMachine

LogicalId: <statemachine-LogicalId>

Properti yang dapat direferensikan: N/A (Anda harus menggunakan `LogicalId` referensi ini AWS CloudFormation sumber daya)

Selain dari sumber daya AWS CloudFormation ini, ketika `AWS::Serverless::StateMachine` telah ditentukan, AWS SAM juga membuat sumber daya AWS CloudFormation untuk skenario berikut:

Skenario

- [Properti peran tidak ditentukan](#)
- [Sumber peristiwa Api ditentukan](#)
- [Sumber event bridge \(atau bus peristiwa\) ditentukan](#)

Properti peran tidak ditentukan

Saat properti `Role` dari `AWS::Serverless::StateMachine` tidak ditentukan, AWS SAM membuat sebuah `AWS::IAM::Role` dari sumber daya AWS CloudFormation.

AWS::IAM::Role

LogicalId: `<statemachine-LogicalId>Role`

Properti yang dapat direferensikan: N/A (Anda harus menggunakan `LogicalId` referensi ini AWS CloudFormation sumber daya)

Sumber peristiwa Api ditentukan

Saat properti `Event` dari `AWS::Serverless::StateMachine` diatur ke `Api`, tetapi properti `RestApiId` tidak ditentukan, AWS SAM membuat `AWS::ApiGateway::RestApi` dari sumber daya AWS CloudFormation.

AWS::ApiGateway::RestApi

LogicalId: `ServerlessRestApi`

Properti yang dapat direferensikan: N/A (Anda harus menggunakan `LogicalId` referensi ini AWS CloudFormation sumber daya)

Sumber event bridge (atau bus peristiwa) ditentukan

Saat properti Event dari `AWS::Serverless::StateMachine` diatur ke salah satu jenis event bridge (atau bus peristiwa), AWS SAM membuat `AWS::Events::Rule` dari sumber daya AWS CloudFormation. Ini berlaku untuk tipe berikut: `EventBridgeRule`, `Schedule`, dan `CloudWatchEvents`.

`AWS::Events::Rule`

LogicalId: `<statemachine-LogicalId><event-LogicalId>`

Properti yang dapat direferensikan: N/A (Anda harus menggunakan `LogicalId` referensi ini AWS CloudFormation sumber daya)

Ekstensi API Gateway

Ekstensi API Gateway adalah ekstensi untuk spesifikasi OpenAPI yang mendukung otorisasi khusus AWS dan integrasi API khusus API Gateway. Untuk informasi selengkapnya tentang ekstensi API Gateway, lihat [Ekstensi API Gateway ke OpenAPI..](#)

AWS SAM mendukung subset ekstensi API Gateway. Untuk melihat ekstensi API Gateway yang didukung oleh AWS SAM, lihat tabel berikut.

Ekstensi API Gateway	Didukung oleh AWS SAM
x-amazon-apigateway-any-metode Objek	Ya
x-amazon-apigateway-apiProperti -key-source	Tidak
x-amazon-apigateway-auth Objek	Ya
x-amazon-apigateway-authorizer Objek	Ya
x-amazon-apigateway-authtype Properti	Ya
x-amazon-apigateway-binary-media-types Properti	Ya
x-amazon-apigateway-documentation Objek	Tidak

x-amazon-apigateway-endpoint-konfigurasi Objek	Tidak
x-amazon-apigateway-gateway-Responses Obyek	Ya
x-amazon-apigateway-gateway-Responses.GatewayResponse Objek	Ya
x-amazon-apigateway-gateway-Responses.ResponseParameters Objek	Ya
x-amazon-apigateway-gateway-Responses.ResponseTemplates Objek	Ya
x-amazon-apigateway-integration Objek	Ya
x-amazon-apigateway-integration.RequestTemplates Obyek	Ya
x-amazon-apigateway-integration.RequestParameters Obyek	Tidak
x-amazon-apigateway-integration.response Obyek	Ya
x-amazon-apigateway-integration.response Obyek	Ya
x-amazon-apigateway-integration.ResponseTemplates Obyek	Ya
x-amazon-apigateway-integration.ResponseParameters Objek	Ya
x-amazon-apigateway-requestProperti -validator	Tidak
x-amazon-apigateway-request-validator Obyek	Tidak
x-amazon-apigateway-request-Validators.RequestValidator Objek	Tidak

Mengotorisasi aplikasi nirserver

Ketika Anda mengotorisasi aplikasi nirserver menggunakan AWS SAM, Anda membuat templat AWS SAM untuk menyatakan dan mengonfigurasi komponen aplikasi Anda.

Bagian ini berisi topik tentang memvalidasi templat AWS SAM dan membangun aplikasi Anda dengan dependensi. Bagian tersebut juga mencakup topik tentang penggunaan AWS SAM untuk kasus penggunaan tertentu seperti bekerja dengan lapisan Lambda, menggunakan aplikasi yang di-nest, mengendalikan akses ke API Gateway, mengatur sumber daya AWS dengan Step Functions, dan penandatanganan kode aplikasi Anda.

Topik

- [Mengelola akses dan izin sumber daya](#)
- [Menjadwalkan acara dengan EventBridge Scheduler](#)
- [Memvalidasi file templat AWS SAM](#)
- [Cara menggunakan lapisan](#)
- [Menggunakan aplikasi nest](#)
- [Pengendalian akses ke API dari API Gateway](#)
- [Mengorkestrasi sumber daya AWS dengan AWS Step Functions](#)
- [Mengonfigurasi penandatanganan kode untuk aplikasi AWS SAM](#)

Mengelola akses dan izin sumber daya

Agar AWS sumber daya Anda berinteraksi satu sama lain, akses dan izin yang tepat harus dikonfigurasi di antara sumber daya Anda, yang memerlukan konfigurasi pengguna, peran, dan kebijakan AWS Identity and Access Management (IAM) untuk menyelesaikan interaksi Anda dengan cara yang aman. Untuk mempelajari lebih lanjut, lihat [Mengontrol akses dengan AWS Identity and Access Management](#) di Panduan AWS CloudFormation Pengguna.

The AWS Serverless Application Model (AWS SAM) menyediakan dua opsi yang menyederhanakan pengelolaan akses dan izin untuk aplikasi tanpa server Anda.

1. AWS SAMkonektor
2. Templat kebijakan AWS SAM

AWS SAMkonektor

Konektor adalah cara penyediaan izin antara dua sumber daya. Anda melakukan ini dengan menjelaskan bagaimana mereka harus berinteraksi satu sama lain dalam AWS SAM template Anda. Mereka dapat didefinisikan menggunakan atribut `Connectors` sumber daya atau tipe `AWS::Serverless::Connector` sumber daya. Konektor mendukung penyediaan `Read` dan `Write` akses data dan peristiwa antara kombinasi sumber daya. AWS Untuk mempelajari lebih lanjut tentang AWS SAM konektor, lihat [Mengelola izin sumber daya dengan konektor AWS SAM](#).

Templat kebijakan AWS SAM

AWS SAMtemplat kebijakan adalah kumpulan izin yang telah ditentukan sebelumnya yang dapat Anda tambahkan ke AWS SAM templat untuk mengelola akses dan izin antara AWS Lambda fungsi, mesin AWS Step Functions status, dan sumber daya yang berinteraksi dengannya. Untuk mempelajari lebih lanjut tentang templat AWS SAM kebijakan, lihat [AWS SAMtemplat kebijakan](#).

AWS CloudFormationmekanisme

AWS CloudFormationMekanisme mencakup konfigurasi pengguna IAM, peran, dan kebijakan untuk mengelola izin antar sumber daya Anda. AWS Untuk mempelajari selengkapnya, lihat [Mengelola izin denganAWS CloudFormation mekanisme](#).

Praktik terbaik

Di seluruh aplikasi tanpa server, Anda dapat menggunakan beberapa metode untuk mengonfigurasi izin antar sumber daya Anda. Oleh karena itu, Anda dapat memilih opsi terbaik untuk setiap skenario dan menggunakan beberapa opsi bersama di seluruh aplikasi Anda. Berikut adalah beberapa hal yang perlu dipertimbangkan ketika memilih opsi terbaik untuk Anda:

- AWS SAMkonektor dan templat kebijakan mengurangi keahlian IAM yang diperlukan untuk memfasilitasi interaksi aman antara AWS sumber daya Anda. Gunakan konektor dan templat kebijakan saat didukung.
- AWS SAMkonektor menyediakan sintaks singkat yang sederhana dan intuitif untuk menentukan izin di AWS SAM template Anda dan membutuhkan keahlian IAM paling sedikit. Saat AWS SAM konektor dan templat kebijakan didukung, gunakan AWS SAM konektor.
- AWS SAMkonektor dapat menyediakan `Read` dan `Write` mengakses data dan peristiwa antara sumber yang didukung dan AWS SAM sumber daya tujuan. Untuk daftar sumber daya yang didukung, lihat [AWS SAMreferensi konektor](#). Saat didukung, gunakan AWS SAM konektor.

- Meskipun templat AWS SAM kebijakan terbatas pada izin antara fungsi Lambda, mesin status Step Functions, dan sumber daya AWS yang berinteraksi dengannya, templat kebijakan mendukung semua operasi CRUD. Saat didukung, dan bila templat AWS SAM kebijakan untuk skenario Anda tersedia, gunakan templat AWS SAM kebijakan. Untuk daftar templat kebijakan yang tersedia, lihat [AWS SAMtemplat kebijakan](#).
- Untuk semua skenario lain, atau ketika granularitas diperlukan, gunakan AWS CloudFormation mekanisme.

Mengelola izin sumber daya dengan konektor AWS SAM

Topik

- [Apa itu AWS SAM konektor?](#)
- [Contoh konektor](#)
- [Koneksi yang didukung antara sumber dan sumber daya tujuan](#)
- [Menggunakan konektor](#)
- [Cara kerja konektor](#)
- [Manfaat AWS SAM konektor](#)
- [Pelajari selengkapnya](#)
- [Berikan umpan balik](#)

Apa itu AWS SAM konektor?

Konektor adalah tipe sumber daya abstrak AWS Serverless Application Model (AWS SAM), diidentifikasi sebagai `AWS::Serverless::Connector`, yang menyediakan izin sederhana dan tercakup dengan baik antara sumber daya aplikasi tanpa server Anda. Gunakan atribut `Connectors` sumber daya dengan menyematkannya dalam sumber daya sumber. Kemudian, tentukan sumber daya tujuan Anda dan jelaskan bagaimana data atau peristiwa harus mengalir di antara sumber daya tersebut. AWS SAMkemudian menyusun kebijakan akses yang diperlukan untuk memfasilitasi interaksi yang diperlukan.

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

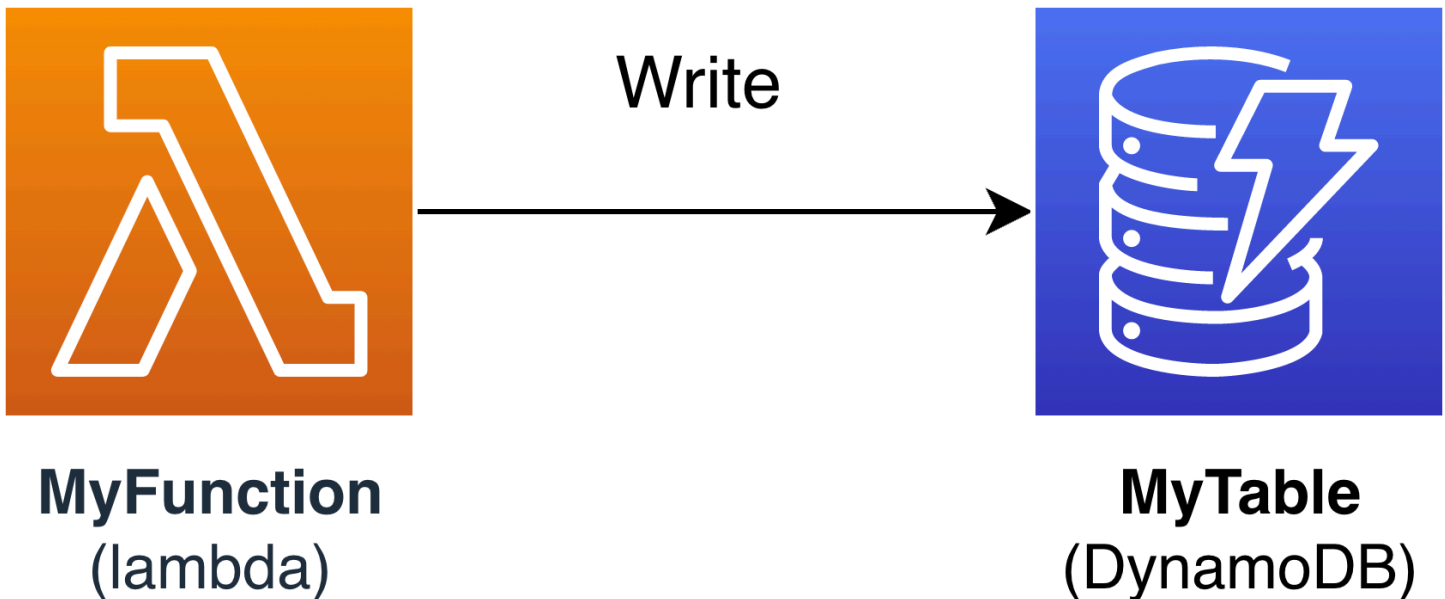
```

<source-resource-logical-id>:
  Type: <resource-type>
  ...
  Connectors:
    <connector-name>:
      Properties:
        Destination:
          <properties-that-identify-destination-resource>
        Permissions:
          <permission-types-to-provision>
  ...

```

Contoh konektor

Dalam contoh ini, kita menggunakan konektor untuk menulis data dari AWS Lambda fungsi ke tabel Amazon DynamoDB.



```

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

```

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

Atribut `Connectors` resource disematkan dalam sumber daya fungsi Lambda. Tabel DynamoDB didefinisikan sebagai sumber daya tujuan menggunakan properti. Id Konektor akan memberikan `Write` izin antara dua sumber daya ini.

Saat Anda menerapkan AWS SAM template Anda AWS CloudFormation, secara otomatis AWS SAM akan menyusun kebijakan akses yang diperlukan yang diperlukan agar koneksi ini berfungsi.

Koneksi yang didukung antara sumber dan sumber daya tujuan

Dukungan konektor `Read` dan jenis izin `Write` data dan acara antara kombinasi pilihan koneksi sumber dan sumber daya tujuan. Misalnya, konektor mendukung `Write` koneksi antara `AWS::ApiGateway::RestApi` sumber daya sumber dan sumber daya `AWS::Lambda::Function` tujuan.

Sumber dan sumber daya tujuan dapat didefinisikan dengan menggunakan kombinasi properti yang didukung. Persyaratan properti akan tergantung pada koneksi yang Anda buat dan di mana sumber daya ditentukan.

Note

Konektor dapat menyediakan izin antara jenis sumber daya tanpa server dan non-server yang didukung.

Untuk daftar koneksi sumber daya yang didukung dan persyaratan propertinya, lihat [Jenis sumber daya dan tujuan yang didukung untuk konektor](#).

Menggunakan konektor

Tentukan izin Baca dan Tulis

Readdan Write izin dapat disediakan dalam satu konektor:

```
AWS::Serverless::Function
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
```

Tentukan sumber daya dengan menggunakan properti lain yang didukung

Untuk sumber daya sumber dan tujuan, ketika didefinisikan dalam template yang sama, gunakan Id properti. Secara opsional, a Qualifier dapat ditambahkan untuk mempersempit ruang lingkup sumber daya yang Anda tentukan. Ketika sumber daya tidak berada dalam template yang sama, gunakan kombinasi properti yang didukung.

- Untuk daftar kombinasi properti yang didukung untuk sumber daya sumber dan tujuan, lihat [Jenis sumber daya dan tujuan yang didukung untuk konektor](#).

- Untuk deskripsi properti yang dapat Anda gunakan dengan konektor, lihat [AWS::Serverless::Connector](#).

Saat Anda menentukan sumber daya sumber dengan properti selain `Id`, gunakan `SourceReference` properti tersebut.

```

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>

```

Berikut adalah contoh, menggunakan a `Qualifier` untuk mempersempit cakupan sumber daya Amazon API Gateway:

```

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

```

```
...
```

Berikut adalah contoh, menggunakan kombinasi yang didukung dari Arn dan Type untuk menentukan sumber daya tujuan dari template lain:

```
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
    ...
```

Buat beberapa konektor dari satu sumber

Dalam sumber daya sumber, Anda dapat menentukan beberapa konektor, masing-masing dengan sumber daya tujuan yang berbeda.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      BucketConn:
        Properties:
          Destination:
            Id: MyBucket
          Permissions:
            - Read
            - Write
      SQSConn:
        Properties:
          Destination:
            Id: MyQueue
          Permissions:
```

```

    - Read
    - Write
  TableConn:
    Properties:
      Destination:
        Id: MyTable
      Permissions:
        - Read
        - Write
  TableConnWithTableArn:
    Properties:
      Destination:
        Type: AWS::DynamoDB::Table
        Arn: !GetAtt MyTable.Arn
      Permissions:
        - Read
        - Write
  ...

```

Buat konektor multi-tujuan

Dalam sumber daya sumber, Anda dapat menentukan konektor tunggal dengan beberapa sumber daya tujuan. Berikut adalah contoh sumber daya fungsi Lambda yang terhubung ke bucket Amazon Simple Storage Service (Amazon S3) dan tabel DynamoDB:

```

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
```

Tentukan atribut sumber daya dengan konektor

Atribut sumber daya dapat didefinisikan untuk sumber daya untuk menentukan perilaku dan hubungan tambahan. Untuk mempelajari lebih lanjut tentang atribut sumber daya, lihat [Referensi atribut sumber daya](#) di Panduan AWS CloudFormation Pengguna.

Anda dapat menambahkan atribut sumber daya ke konektor tertanam dengan mendefinisikannya pada level yang sama dengan properti konektor Anda. Saat AWS SAM template Anda diubah saat penerapan, atribut akan diteruskan ke sumber daya yang dihasilkan.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      MyConn:
        DeletionPolicy: Retain
        DependsOn: AnotherFunction
        Properties:
          ...
```

Cara kerja konektor

Note

Bagian ini menjelaskan bagaimana konektor menyediakan sumber daya yang diperlukan di balik layar. Ini terjadi untuk Anda secara otomatis saat menggunakan konektor.

Pertama, atribut `Connectors` sumber daya tertanam diubah menjadi tipe `AWS::Serverless::Connector` sumber daya. ID logisnya secara otomatis dibuat sebagai `<source-resource-logical-id><embedded-connector-logical-id>`.

Misalnya, berikut adalah konektor tertanam:

```
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

```

Ini akan menghasilkan `AWS::Serverless::Connector` sumber daya berikut:

```

Transform: AWS::Serverless-2016-10-31
Resources:
  ...
  MyFunctionMyConn:
    Type: AWS::Serverless::Connector
    Properties:
      Source:
        Id: MyFunction
      Destination:
        Id: MyTable
      Permissions:
        - Read
        - Write

```

Note

Anda juga dapat menentukan konektor dalam AWS SAM template Anda dengan menggunakan sintaks ini. Ini disarankan ketika sumber daya Anda ditentukan pada templat terpisah dari konektor Anda.

Selanjutnya, kebijakan akses yang diperlukan untuk koneksi ini disusun secara otomatis. Untuk informasi selengkapnya tentang sumber daya yang dihasilkan oleh konektor, lihat [AWS CloudFormation sumber daya yang dihasilkan saat Anda menentukan `AWS::Serverless::Connector`](#).

Manfaat AWS SAM konektor

Dengan secara otomatis menyusun kebijakan akses yang sesuai antar sumber daya, konektor memberi Anda kemampuan untuk membuat aplikasi tanpa server Anda dan fokus pada arsitektur aplikasi Anda tanpa memerlukan keahlian dalam kemampuan AWS otorisasi, bahasa kebijakan, dan pengaturan keamanan khusus layanan. Oleh karena itu, konektor adalah manfaat besar bagi pengembang yang mungkin baru dalam pengembangan tanpa server, atau pengembang berpengalaman yang ingin meningkatkan kecepatan pengembangan mereka.

Pelajari selengkapnya

Untuk informasi selengkapnya tentang penggunaan AWS SAM konektor, lihat [AWS::Serverless::Connector](#).

Berikan umpan balik

Untuk memberikan umpan balik tentang konektor, [kirimkan masalah baru](#) di serverless-application-model AWS GitHub repositori.

AWS SAMtemplat kebijakan

AWS Serverless Application Model (AWS SAM) memungkinkan Anda memilih dari daftar templat kebijakan untuk mencakup izin fungsi Lambda AWS Step Functions dan mesin status ke sumber daya yang digunakan oleh aplikasi Anda.

AWS SAM aplikasi dalam AWS Serverless Application Repository yang menggunakan templat kebijakan tidak memerlukan pengakuan pelanggan khusus untuk menyebarkan aplikasi dari AWS Serverless Application Repository

Jika Anda ingin meminta agar templat kebijakan baru ditambahkan, lakukan hal berikut:

1. Kirim permintaan tarik terhadap file sumber `policy_templates.json` di cabang proyek. `develop` AWS SAM GitHub Anda dapat menemukan file sumber di [policy_templates.json](#) di situs web. GitHub
2. Kirimkan masalah dalam AWS SAM GitHub proyek yang mencakup alasan permintaan tarik Anda dan tautan ke permintaan tersebut. Gunakan tautan ini untuk mengirimkan masalah baru: [AWS Serverless Application Model: Permasalahan](#).

Sintaks

Untuk setiap templat kebijakan yang Anda tentukan dalam file AWS SAM templat, Anda harus selalu menentukan objek yang berisi nilai placeholder templat kebijakan. Jika templat kebijakan tidak memerlukan nilai pengganti, Anda harus menentukan objek yang kosong.

YAML

```
MyFunction:
  Type: AWS::Serverless::Function
  Properties:
    Policies:
      - PolicyTemplateName1:      # Policy template with placeholder value
        Key1: Value1
      - PolicyTemplateName2: {}   # Policy template with no placeholder value
```

Contoh-contoh

Contoh 1: Templat kebijakan dengan nilai pengganti

Contoh berikut menunjukkan bahwa templat kebijakan [SQSPollerPolicy](#) mengasumsikan QueueName sebagai sumber daya. AWS SAM Template mengambil nama antrian Amazon SQS MyQueue "", yang dapat Anda buat dalam aplikasi yang sama atau diminta sebagai parameter untuk aplikasi.

```
MyFunction:
  Type: 'AWS::Serverless::Function'
  Properties:
    CodeUri: ${codeuri}
    Handler: hello.handler
    Runtime: python2.7
    Policies:
      - SQSPollerPolicy:
        QueueName:
          !GetAtt MyQueue.QueueName
```

Contoh 2: Templat kebijakan tanpa nilai pengganti

Contoh berikut berisi templat kebijakan [CloudWatchPutMetricPolicy](#), yang tidak memiliki nilai pengganti.

Note

Meskipun tidak ada nilai pengganti, Anda harus menentukan objek yang kosong, jika tidak ditentukan akan muncul sebuah kesalahan.

```
MyFunction:
  Type: 'AWS::Serverless::Function'
  Properties:
    CodeUri: ${codeuri}
    Handler: hello.handler
    Runtime: python2.7
    Policies:
      - CloudWatchPutMetricPolicy: {}
```

Tabel templat kebijakan

Berikut ini adalah tabel templat kebijakan yang tersedia.

Templat kebijakan	Deskripsi		
AcmGetCertificatePolicy	Memberikan izin untuk membaca sertifikat dari AWS Certificate Manager.		
AMIDescriptionPolicy	Memberikan izin untuk menjelaskan Amazon Machine Images (AMI).		
AthenaQueryPolicy	Memberikan izin untuk mengeksekusi kueri Athena.		
AWSSecretsManagerGetSecretValuePolicy	Memberikan izin agar mendapatkan nilai rahasia untuk rahasia AWS Secrets Manager yang ditentukan.		

Templat kebijakan	Deskripsi		
AWSSecretsManagerRotationPolicy	Memberikan izin untuk merotasi rahasia di AWS Secrets Manager.		
CloudFormationDescribeStacksPolicy	Memberikan izin untuk menggambarkan AWS CloudFormation tumpukan.		
CloudWatchDashboardPolicy	Memberikan izin untuk menempatkan metrik untuk beroperasi di CloudWatch dasbor.		
CloudWatchDescribeAlarmHistoryPolicy	Memberikan izin untuk menggambarkan riwayat CloudWatch alarm.		
CloudWatchPutMetricPolicy	Memberikan izin untuk mengirim metrik ke CloudWatch.		
CodeCommitCrudPolicy	Memberikan izin untuk membuat/membaca/memperbarui/menghapus objek dalam repositori tertentu. CodeCommit		
CodeCommitReadPolicy	Memberikan izin untuk membaca objek dalam CodeCommit repositori tertentu.		
CodePipelineLambdaExecutionPolicy	Memberikan izin untuk fungsi Lambda yang dipanggil oleh CodePipeline untuk melaporkan status pekerjaan.		

Templat kebijakan	Deskripsi		
CodePipelineReadOnlyPolicy	Memberikan izin baca untuk mendapatkan detail tentang CodePipeline pipa.		
ComprehendBasicAccessPolicy	Memberikan izin untuk mendeteksi entitas, frasa kunci, bahasa, dan sentimen.		
CostExplorerReadOnlyPolicy	Memberikan izin baca-saja pada baca-saja Cost Explorer API untuk riwayat penagihan .		
DynamoDBBackupFullAccessPolicy	Memberikan izin membaca dan menulis pada pencadangan sesuai permintaan DynamoDB untuk tabel.		
DynamoDBCrudPolicy	Memberikan izin untuk membuat, membaca, memperbarui, dan menghapus pada tabel Amazon DynamoDB.		
DynamoDBReadOnlyPolicy	Memberikan izin baca-saja pada tabel DynamoDB.		
DynamoDBReconfigurePolicy	Memberikan izin untuk mengonfigurasi ulang tabel DynamoDB.		
DynamoDBRestoreFromBackupPolicy	Memberikan izin untuk memulihkan tabel DynamoDB dari cadangan.		
DynamoDBStreamReadPolicy	Memberikan izin untuk menjelaskan dan membaca aliran dan catatan DynamoDB.		

Templat kebijakan	Deskripsi		
DynamoDBWritePolicy	Memberikan izin tulis-saja pada tabel DynamoDB.		
EC2CopyImagePolicy	Memberikan izin untuk menyalin citra Amazon EC2.		
EC2DescribePolicy	Memberikan izin untuk menjelaskan instans Amazon Elastic Compute Cloud (Amazon EC2).		
EcsRunTaskPolicy	Memberikan izin agar dapat memulai tugas baru untuk definisi tugas.		
EFSWriteAccessPolicy	Memberikan izin untuk memasang sistem file Amazon EFS dengan akses tulis.		
EKSDescribePolicy	Memberikan izin untuk menjelaskan atau membuat daftar klaster Amazon EKS.		
ElasticMapReduceAddJobFlowStepsPolicy	Memberikan izin untuk menambahkan langkah-langkah baru agar dapat menjalankan klaster.		
ElasticMapReduceCancelStepsPolicy	Memberikan izin untuk membatalkan langkah yang tertunda atau langkah-langkah pada klaster yang sedang berjalan.		
ElasticMapReduceModifyInstancesFleetPolicy	Memberikan izin untuk membuat daftar detail dan mengubah kapasitas untuk armada instans dalam sebuah klaster.		

Templat kebijakan	Deskripsi		
ElasticMapReduceInstanceGroupsPolicy	Memberikan izin untuk membuat daftar detail dan mengubah pengaturan pada grup instans dalam klaster.		
ElasticMapReduceSecurityTerminationProtectionPolicy	Memberikan izin untuk mengatur perlindungan terminasi pada klaster.		
ElasticMapReduceTerminateJobsFlowsPolicy	Memberikan izin untuk menutup klaster.		
ElasticsearchHttpPostPolicy	Memberikan izin POST ke Amazon OpenSearch Service.		
EventBridgePutEventsPolicy	Memberikan izin untuk mengirim acara ke EventBridge.		
FilterLogEventsPolicy	Memberikan izin untuk memfilter peristiwa CloudWatch Log dari grup log tertentu.		
FirehoseCreatePolicy	Memberikan izin untuk membuat, menulis, memperbarui, dan menghapus aliran pengiriman Firehose.		
FirehoseWritePolicy	Memberikan izin untuk menulis ke aliran pengiriman Firehose.		

Templat kebijakan	Deskripsi		
KinesisCreatePolicy	Memberikan izin untuk membuat, memublikasikan, dan menghapus aliran Amazon Kinesis.		
KinesisStreamReadPolicy	Memberikan izin untuk membuat daftar dan membaca aliran Amazon Kinesis.		
KMSTDecryptPolicy	Memberikan izin untuk mendekripsi dengan kunci AWS Key Management Service (AWS KMS).		
KMSEncryptPolicy	Memberikan izin untuk mengenkripsi dengan kunci AWS Key Management Service (AWS KMS).		
LambdaInvokePolicy	Memberikan izin untuk memanggil AWS Lambda fungsi, alias, atau versi.		
MobileAnalyticsWriteOnlyAccessPolicy	Memberikan izin tulis-saja agar dapat menempatkan data kejadian untuk semua sumber daya aplikasi.		
OrganizationsListAccountsPolicy	Memberikan izin baca-saja pada daftar nama akun anak dan ID.		
PinpointEndpointAccessPolicy	Memberikan izin agar mendapatkan serta memperbarui titik akhir untuk aplikasi Amazon Pinpoint.		
PollyFullAccessPolicy	Memberikan izin akses penuh ke sumber leksikon Amazon Polly.		

Templat kebijakan	Deskripsi		
Rekogniti onDetectOnlyPolicy	Memberikan izin untuk mendeteksi wajah, label, dan teks.		
Rekogniti onFacesManagementPolicy	Memberikan izin untuk menambah, menghapus, dan mencari wajah di koleksi Amazon Rekognition.		
Rekogniti onFacesPolicy	Memberikan izin untuk membandingkan dan mendeteksi wajah serta label.		
Rekogniti onLabelsPolicy	Memberikan izin untuk mendeteksi objek dan moderasi pada label.		
Rekogniti onNoDataAccessPolicy	Memberikan izin untuk membandingkan dan mendeteksi wajah serta label.		
Rekogniti onReadPolicy	Memberikan izin untuk membuat daftar dan mencari wajah.		
Rekogniti onWriteOnIyAccessPolicy	Memberikan izin untuk membuat koleksi dan indeks wajah.		
Route53ChangeResourceRecordSetsPolicy	Memberikan izin untuk mengubah set catatan sumber daya di Route 53.		
S3CrudPolicy	Memberikan izin untuk membuat, membaca, memperbarui, dan menghapus untuk bertindak di objek dalam bucket Amazon S3.		

Templat kebijakan	Deskripsi		
S3FullAccessPolicy	Memberikan izin akses penuh agar dapat bertindak pada objek dalam bucket Amazon S3.		
S3ReadPolicy	Memberikan izin baca-saja untuk membaca objek di bucket Amazon Simple Storage Service (Amazon S3).		
S3WritePolicy	Memberikan izin menulis untuk menulis objek ke dalam bucket Amazon S3.		
SageMakerCreateEndpointConfigurationPolicy	Memberikan izin untuk membuat konfigurasi titik akhir di SageMaker.		
SageMakerCreateEndpointPolicy	Memberikan izin untuk membuat titik akhir di SageMaker.		
ServerlessRepoReadWriteAccessPolicy	Memberikan izin untuk membuat dan mendaftarkan aplikasi dalam AWS Serverless Application Repository layanan.		
SESBulkTemplatedCreatePolicy	Memberikan izin untuk mengirimkan email, templat email, templat email massal, dan melakukan verifikasi identitas.		
SESBulkTemplatedCreatePolicy_v2	Memberikan izin untuk mengirim email Amazon SES, email yang ditemplat, dan templat email massal serta agar menjalankan verifikasi identitas.		
SESCrudPolicy	Memberikan izin untuk mengirim email dan memverifikasi identitas.		

Templat kebijakan	Deskripsi		
SESEmailTemplateCrudPolicy	Memberikan izin untuk membuat, mendapatkan, membuat daftar, memperbarui dan menghapus templat email Amazon SES.		
SESSendBouncePolicy	Memberikan SendBounce izin ke identitas Amazon Simple Email Service (Amazon SES).		
SNSCrudPolicy	Memberikan izin untuk membuat, memublikasikan, dan berlangganan pada topik Amazon SNS.		
SNSPublishMessagePolicy	Memberikan izin untuk memublikasikan pesan ke topik Amazon Simple Notification Service (Amazon SNS).		
SQSPollerPolicy	Memberikan izin untuk polling antrean Amazon Simple Queue Service (Amazon SQS).		
SQSSendMessagePolicy	Memberikan izin untuk mengirim pesan ke antrean Amazon SQS.		
SSMParameterReadPolicy	Memberikan izin untuk mengakses parameter dari penyimpanan parameter Amazon EC2 Systems Manager (SSM) untuk memuat rahasia di akun ini. Gunakan ketika nama parameter tidak memiliki awalan garis miring.		

Templat kebijakan	Deskripsi		
SSMParameterWithSlashPrefixReadPolicy	Memberikan izin untuk mengakses parameter dari penyimpanan parameter Amazon EC2 Systems Manager (SSM) untuk memuat rahasia di akun ini. Gunakan ketika nama parameter memiliki awalan garis miring.		
StepFunctionsExecutionPolicy	Memberikan izin untuk memulai eksekusi pada mesin keadaan Step Functions.		
TextractDetectAndAnalyzePolicy	Memberikan akses untuk mendeteksi dan menganalisa dokumen menggunakan Amazon Textract.		
TextractGetResultPolicy	Memberikan akses untuk mendapatkan dokumen yang terdeteksi dan teranalisis dari Amazon Textract.		
TextractPolicy	Memberikan akses penuh menuju Amazon Textract.		
VPCAccessPolicy	Memberikan akses untuk membuat, menghapus, menjelaskan, dan melepaskan antarmuka jaringan elastis.		

Memecahkan masalah

Kesalahan SAM CLI: "Harus menentukan nilai parameter yang valid untuk templat kebijakan policy-template-name '< >'"

Saat mengeksekusi `sam build`, Anda akan melihat kesalahan berikut:

```
"Must specify valid parameter values for policy template '<policy-template-name>'"
```

Kesalahan tersebut berarti bahwa Anda tidak dapat melewati objek kosong ketika menyatakan templat kebijakan yang tidak memiliki nilai pengganti.

Untuk memperbaikinya, nyatakan kebijakan seperti contoh berikut untuk [CloudWatchPutMetricPolicy](#).

```
MyFunction:
  Policies:
    - CloudWatchPutMetricPolicy: {}
```

Daftar templat kebijakan

Berikut ini adalah templat kebijakan yang tersedia, bersama dengan izin yang diterapkan untuk masing-masing templat kebijakan. AWS Serverless Application Model (AWS SAM) secara otomatis mengisi item placeholder (seperti AWS Wilayah dan ID akun) dengan informasi yang sesuai.

Topik

- [AcmGetCertificatePolicy](#)
- [AMIDescribePolicy](#)
- [AthenaQueryPolicy](#)
- [AWSecretsManagerGetSecretValuePolicy](#)
- [AWSecretsManagerRotationPolicy](#)
- [CloudFormationDescribeStacksPolicy](#)
- [CloudWatchDashboardPolicy](#)
- [CloudWatchDescribeAlarmHistoryPolicy](#)
- [CloudWatchPutMetricPolicy](#)
- [CodePipelineLambdaExecutionPolicy](#)
- [CodePipelineReadOnlyPolicy](#)
- [CodeCommitCrudPolicy](#)
- [CodeCommitReadPolicy](#)
- [ComprehendBasicAccessPolicy](#)
- [CostExplorerReadOnlyPolicy](#)
- [DynamoDBBackupFullAccessPolicy](#)

- [DynamoDBCrudPolicy](#)
- [DynamoDBReadPolicy](#)
- [DynamoDBReconfigurePolicy](#)
- [DynamoDBRestoreFromBackupPolicy](#)
- [DynamoDBStreamReadPolicy](#)
- [DynamoDBWritePolicy](#)
- [EC2CopyImagePolicy](#)
- [EC2DescribePolicy](#)
- [EcsRunTaskPolicy](#)
- [EFSWriteAccessPolicy](#)
- [EKSDescribePolicy](#)
- [ElasticMapReduceAddJobFlowStepsPolicy](#)
- [ElasticMapReduceCancelStepsPolicy](#)
- [ElasticMapReduceModifyInstanceFleetPolicy](#)
- [ElasticMapReduceModifyInstanceGroupsPolicy](#)
- [ElasticMapReduceSetTerminationProtectionPolicy](#)
- [ElasticMapReduceTerminateJobFlowsPolicy](#)
- [ElasticsearchHttpPostPolicy](#)
- [EventBridgePutEventsPolicy](#)
- [FilterLogEventsPolicy](#)
- [FirehoseCrudPolicy](#)
- [FirehoseWritePolicy](#)
- [KinesisCrudPolicy](#)
- [KinesisStreamReadPolicy](#)
- [KMSTDecryptPolicy](#)
- [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

Memberikan izin untuk membaca sertifikat dari AWS Certificate Manager.

```
"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": [  
      "acm:GetCertificate"  
    ],  
    "Resource": {  
      "Fn::Sub": [  
        "${certificateArn}",  
        {  
          "certificateArn": {  
            "Ref": "CertificateArn"  
          }  
        }  
      ]  
    }  
  }  
]
```

AMIDescribePolicy

Memberikan izin untuk menjelaskan Amazon Machine Images (AMI).

```
"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": [  
      "ec2:DescribeImages"  
    ],  
    "Resource": "*"  
  }  
]
```

AthenaQueryPolicy

Memberikan izin untuk mengeksekusi kueri Athena.

```

"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"
          }
        }
      ]
    }
  }
]

```

```

    }
  ]
}
]

```

AWSecretsManagerGetSecretValuePolicy

Memberikan izin untuk mendapatkan nilai rahasia untuk AWS Secrets Manager rahasia yang ditentukan.

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "secretsmanager:GetSecretValue"
    ],
    "Resource": {
      "Fn::Sub": [
        "${secretArn}",
        {
          "secretArn": {
            "Ref": "SecretArn"
          }
        }
      ]
    }
  }
]

```

AWSecretsManagerRotationPolicy

Memberikan izin untuk merotasi rahasia di 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

Memberikan izin untuk menggambarkan AWS CloudFormation tumpukan.

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "cloudformation:DescribeStacks"
    ],
    "Resource": {
      "Fn::Sub": "arn:${AWS::Partition}:cloudformation:${AWS::Region}:
${AWS::AccountId}:stack/*"
    }
  }
]

```



```
]
```

CloudWatchDashboardPolicy

Memberikan izin untuk menempatkan metrik untuk beroperasi di CloudWatch dasbor.

```
"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": [  
      "cloudwatch:GetDashboard",  
      "cloudwatch:ListDashboards",  
      "cloudwatch:PutDashboard",  
      "cloudwatch:ListMetrics"  
    ],  
    "Resource": "*"  
  }  
]
```

CloudWatchDescribeAlarmHistoryPolicy

Memberikan izin untuk menggambarkan riwayat CloudWatch alarm Amazon.

```
"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": [  
      "cloudwatch:DescribeAlarmHistory"  
    ],  
    "Resource": "*"  
  }  
]
```

CloudWatchPutMetricPolicy

Memberikan izin untuk mengirim metrik ke CloudWatch.

```
"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": [  
      "cloudwatch:PutMetricData"  
    ]  
  }  
]
```

```

    ],
    "Resource": "*"
  }
]

```

CodePipelineLambdaExecutionPolicy

Memberikan izin untuk fungsi Lambda yang dipanggil oleh AWS CodePipeline untuk melaporkan status pekerjaan.

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "codepipeline:PutJobSuccessResult",
      "codepipeline:PutJobFailureResult"
    ],
    "Resource": "*"
  }
]

```

CodePipelineReadOnlyPolicy

Memberikan izin baca untuk mendapatkan detail tentang CodePipeline pipa.

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "codepipeline:ListPipelineExecutions"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:codepipeline:${AWS::Region}:${AWS::AccountId}:
${pipelinename}",
        {
          "pipelinename": {
            "Ref": "PipelineName"
          }
        }
      ]
    }
  }
]

```

]

CodeCommitCrudPolicy

Memberikan izin untuk membuat, membaca, memperbarui, dan menghapus objek dalam CodeCommit repositori tertentu.

```
"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

Memberikan izin untuk membaca objek dalam CodeCommit repositori tertentu.

```
"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

Memberikan izin untuk mendeteksi entitas, frasa kunci, bahasa, dan sentimen.

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "comprehend:BatchDetectKeyPhrases",
      "comprehend:DetectDominantLanguage",
      "comprehend:DetectEntities",
      "comprehend:BatchDetectEntities",
      "comprehend:DetectKeyPhrases",
      "comprehend:DetectSentiment",
      "comprehend:BatchDetectDominantLanguage",
      "comprehend:BatchDetectSentiment"
    ],
    "Resource": "*"
  }
]

```

CostExplorerReadOnlyPolicy

Memberikan izin hanya-baca ke API hanya-baca (Cost AWS Cost Explorer Explorer) untuk riwayat penagihan.

```

"Statement": [
  {

```

```

    "Effect": "Allow",
    "Action": [
      "ce:GetCostAndUsage",
      "ce:GetDimensionValues",
      "ce:GetReservationCoverage",
      "ce:GetReservationPurchaseRecommendation",
      "ce:GetReservationUtilization",
      "ce:GetTags"
    ],
    "Resource": "*"
  }
]

```

DynamoDBBackupFullAccessPolicy

Memberikan izin membaca dan menulis pada pencadangan sesuai permintaan DynamoDB untuk tabel.

```

"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

Memberikan izin untuk membuat, membaca, memperbarui, dan menghapus pada tabel Amazon DynamoDB.

```

"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

Memberikan izin baca-saja pada tabel DynamoDB.

```

"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

Memberikan izin untuk mengonfigurasi ulang tabel DynamoDB.

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:UpdateTable"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}",
        {
          "tableName": {
            "Ref": "TableName"
          }
        }
      ]
    }
  }
]

```

DynamoDBRestoreFromBackupPolicy

Memberikan izin untuk memulihkan tabel DynamoDB dari cadangan.

```
"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

Memberikan izin untuk menjelaskan dan membaca aliran dan catatan DynamoDB.

```
"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

Memberikan izin tulis-saja pada tabel DynamoDB.

```
"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

Memberikan izin untuk menyalin citra Amazon EC2.

```
"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": [  
      "ec2:CopyImage"  
    ],  
    "Resource": {  
      "Fn::Sub": [  
        "arn:${AWS::Partition}:ec2:${AWS::Region}:${AWS::AccountId}:image/${imageId}",  
        {  
          "imageId": {  
            "Ref": "ImageId"  
          }  
        }  
      ]  
    }  
  }  
]
```

EC2DescribePolicy

Memberikan izin untuk menjelaskan instans Amazon Elastic Compute Cloud (Amazon EC2).

```
"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": [  
      "ec2:DescribeRegions",  
      "ec2:DescribeInstances"  
    ],  
    "Resource": "*"  
  }  
]
```

EcsRunTaskPolicy

Memberikan izin agar dapat memulai tugas baru untuk definisi tugas.

```
"Statement": [  
  {
```

```

{
  "Action": [
    "ecs:RunTask"
  ],
  "Resource": {
    "Fn::Sub": [
      "arn:${AWS::Partition}:ecs:${AWS::Region}:${AWS::AccountId}:task-definition/
${taskDefinition}",
      {
        "taskDefinition": {
          "Ref": "TaskDefinition"
        }
      }
    ]
  },
  "Effect": "Allow"
}
]

```

EFSWriteAccessPolicy

Memberikan izin untuk memasang sistem file Amazon EFS dengan akses tulis.

```

"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

Memberikan izin untuk menjelaskan atau membuat daftar kluster Amazon Elastic Kubernetes Service (Amazon EKS).

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "eks:DescribeCluster",
      "eks:ListClusters"
    ],
    "Resource": "*"
  }
]

```

ElasticMapReduceAddJobFlowStepsPolicy

Memberikan izin untuk menambahkan langkah-langkah baru agar dapat menjalankan kluster.

```

"Statement": [
  {
    "Action": "elasticmapreduce:AddJobFlowSteps",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",

```



```

    {
      "clusterId": {
        "Ref": "ClusterId"
      }
    }
  ],
  "Effect": "Allow"
}
]

```

ElasticMapReduceCancelStepsPolicy

Memberikan izin untuk membatalkan langkah yang tertunda atau langkah-langkah pada klaster yang sedang berjalan.

```

"Statement": [
  {
    "Action": "elasticmapreduce:CancelSteps",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]

```

ElasticMapReduceModifyInstanceFleetPolicy

Memberikan izin untuk membuat daftar detail dan mengubah kapasitas untuk armada instans dalam sebuah klaster.

```

"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

Memberikan izin untuk membuat daftar detail dan mengubah pengaturan pada grup instans dalam klaster.

```

"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

Memberikan izin untuk mengatur perlindungan terminasi pada klaster.

```
"Statement": [  
  {  
    "Action": "elasticmapreduce:SetTerminationProtection",  
    "Resource": {  
      "Fn::Sub": [  
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:  
${AWS::AccountId}:cluster/${clusterId}",  
        {  
          "clusterId": {  
            "Ref": "ClusterId"  
          }  
        }  
      ]  
    },  
    "Effect": "Allow"  
  }  
]
```

ElasticMapReduceTerminateJobFlowsPolicy

Memberikan izin untuk menutup klaster.

```
"Statement": [  
  {  
    "Action": "elasticmapreduce:TerminateJobFlows",  
    "Resource": {  
      "Fn::Sub": [  
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:  
${AWS::AccountId}:cluster/${clusterId}",  
        {  
          "clusterId": {  
            "Ref": "ClusterId"  
          }  
        }  
      ]  
    },  
    "Effect": "Allow"  
  }  
]
```

ElasticsearchHttpPostPolicy

Memberikan izin POST dan PUT ke OpenSearch Layanan Amazon.

```
"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

Memberikan izin untuk mengirim acara ke Amazon EventBridge.

```
"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": "events:PutEvents",  
    "Resource": {  
      "Fn::Sub": [  
        "arn:${AWS::Partition}:events:${AWS::Region}:${AWS::AccountId}:event-bus/  
${eventBusName}",  
        {  
          "eventBusName": {  
            "Ref": "EventBusName"  
          }  
        }  
      ]  
    }  
  }  
]
```

```

    }
  }
]

```

FilterLogEventsPolicy

Memberikan izin untuk memfilter peristiwa CloudWatch Log dari grup log tertentu.

```

"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

Memberikan izin untuk membuat, menulis, memperbarui, dan menghapus aliran pengiriman Firehose.

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "firehose:CreateDeliveryStream",
      "firehose>DeleteDeliveryStream",
      "firehose:DescribeDeliveryStream",
      "firehose:PutRecord",
      "firehose:PutRecordBatch",
      "firehose:UpdateDestination"
    ]
  }
]

```

```

    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:firehose:${AWS::Region}:
        ${AWS::AccountId}:deliverystream/${deliveryStreamName}",
        {
          "deliveryStreamName": {
            "Ref": "DeliveryStreamName"
          }
        }
      ]
    }
  }
}
]

```

FirehoseWritePolicy

Memberikan izin untuk menulis ke aliran pengiriman Firehose.

```

"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

Memberikan izin untuk membuat, memublikasikan, dan menghapus aliran Amazon Kinesis.

```

"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

Memberikan izin untuk membuat daftar dan membaca aliran Amazon Kinesis.

```

"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

Memberikan izin untuk mendekripsi dengan kunci AWS Key Management Service (AWS KMS). Perhatikan bahwa keyId harus berupa ID AWS KMS kunci, dan bukan alias kunci.

```

"Statement": [
  {
    "Action": "kms:Decrypt",
    "Effect": "Allow",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:kms:${AWS::Region}:${AWS::AccountId}:key/${keyId}",
        {
          "keyId": {
            "Ref": "KeyId"
          }
        }
      ]
    }
  }
]

```



```

    }
  }
]
}
]

```

KMSEncryptPolicy

Memberikan izin untuk mengenkripsi dengan AWS KMS kunci. Perhatikan bahwa KeyID harus berupa ID AWS KMS kunci, dan bukan alias kunci.

```

"Statement": [
  {
    "Action": "kms:Encrypt",
    "Effect": "Allow",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:kms:${AWS::Region}:${AWS::AccountId}:key/${keyId}",
        {
          "keyId": {
            "Ref": "KeyId"
          }
        }
      ]
    }
  }
]

```

LambdaInvokePolicy

Memberikan izin untuk memanggil AWS Lambda fungsi, alias, atau versi.

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "lambda:InvokeFunction"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:lambda:${AWS::Region}:${AWS::AccountId}:function:
        ${functionName}*",

```

```

    {
      "functionName": {
        "Ref": "FunctionName"
      }
    }
  ]
}
]

```

MobileAnalyticsWriteOnlyAccessPolicy

Memberikan izin tulis-saja agar dapat menempatkan data kejadian untuk semua sumber daya aplikasi.

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "mobileanalytics:PutEvents"
    ],
    "Resource": "*"
  }
]

```

OrganizationsListAccountsPolicy

Memberikan izin baca-saja pada daftar nama akun anak dan ID.

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "organizations:ListAccounts"
    ],
    "Resource": "*"
  }
]

```

PinpointEndpointAccessPolicy

Memberikan izin agar mendapatkan serta memperbarui titik akhir untuk aplikasi Amazon Pinpoint.

```

"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

Memberikan izin akses penuh ke sumber leksikon Amazon Polly.

```

"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

Memberikan izin untuk mendeteksi wajah, label, dan teks.

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:DetectFaces",
      "rekognition:DetectLabels",
      "rekognition:DetectModerationLabels",
      "rekognition:DetectText"
    ],
    "Resource": "*"
  }
]

```

RekognitionFacesManagementPolicy

Memberikan izin untuk menambah, menghapus, dan mencari wajah di koleksi Amazon Rekognition.

```

"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

Memberikan izin untuk membandingkan dan mendeteksi wajah serta label.

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:CompareFaces",
      "rekognition:DetectFaces"
    ],
    "Resource": "*"
  }
]

```

RekognitionLabelsPolicy

Memberikan izin untuk mendeteksi objek dan moderasi pada label.

```

"Statement": [

```

```

{
  "Effect": "Allow",
  "Action": [
    "rekognition:DetectLabels",
    "rekognition:DetectModerationLabels"
  ],
  "Resource": "*"
}
]

```

RekognitionNoDataAccessPolicy

Memberikan izin untuk membandingkan dan mendeteksi wajah serta label.

```

"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

Memberikan izin untuk membuat daftar dan mencari wajah.

```

"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

Memberikan izin untuk membuat koleksi dan indeks wajah.

```

"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

Memberikan izin untuk mengubah set catatan sumber daya di Route 53.

```
"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": [  
      "route53:ChangeResourceRecordSets"  
    ],  
    "Resource": {  
      "Fn::Sub": [  
        "arn:${AWS::Partition}:route53::hostedzone/${HostedZoneId}",  
        {  
          "HostedZoneId": {  
            "Ref": "HostedZoneId"  
          }  
        }  
      ]  
    }  
  }  
]
```

S3CrudPolicy

Memberikan izin untuk membuat, membaca, memperbarui, dan menghapus untuk bertindak di objek dalam bucket Amazon S3.

```
"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

Memberikan izin akses penuh agar dapat bertindak pada objek dalam bucket Amazon S3.

```

"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

Memberikan izin baca-saja untuk membaca objek di bucket Amazon Simple Storage Service (Amazon S3).

```

"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

Memberikan izin menulis untuk menulis objek ke dalam bucket Amazon S3.

```

"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

Memberikan izin untuk membuat konfigurasi titik akhir di SageMaker.

```

"Statement": [
  {
    "Action": [
      "sagemaker:CreateEndpointConfig"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sagemaker:${AWS::Region}:${AWS::AccountId}:endpoint-
        config/${endpointConfigName}",
        {
          "endpointConfigName": {

```

```

        "Ref": "EndpointConfigName"
      }
    }
  ],
  "Effect": "Allow"
}
]

```

SageMakerCreateEndpointPolicy

Memberikan izin untuk membuat titik akhir di SageMaker.

```

"Statement": [
  {
    "Action": [
      "sagemaker:CreateEndpoint"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sagemaker:${AWS::Region}:${AWS::AccountId}:endpoint/
${endpointName}",
        {
          "endpointName": {
            "Ref": "EndpointName"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]

```

ServerlessRepoReadWriteAccessPolicy

Memberikan izin untuk membuat dan mendaftar aplikasi di layanan AWS Serverless Application Repository (AWS SAM).

```

"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

Memberikan izin untuk mengirim email Amazon SES, email yang ditemplat, dan templat email massal serta agar menjalankan verifikasi identitas.

Note

ses:SendTemplatedEmailTindakan ini membutuhkan template ARN. Gunakan SESBulkTemplatedCrudPolicy_v2 sebagai gantinya.

```

"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

Memberikan izin untuk mengirim email Amazon SES, email yang ditemplat, dan templat email massal serta agar menjalankan verifikasi identitas.

```

"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

Memberikan izin untuk mengirim email dan memverifikasi identitas.

```

"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

Memberikan izin untuk membuat, mendapatkan, mendaftar, memperbarui, dan menghapus templat email Amazon SES.

```
"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": [  
      "ses:CreateTemplate",  
      "ses:GetTemplate",  
      "ses:ListTemplates",  
      "ses:UpdateTemplate",  
      "ses>DeleteTemplate",  
      "ses:TestRenderTemplate"  
    ],  
    "Resource": "*"   
  }  
]
```

SESSendBouncePolicy

Memberikan SendBounce izin ke identitas Amazon Simple Email Service (Amazon SES).

```
"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": [  
      "ses:SendBounce"  
    ],  
    "Resource": {  
      "Fn::Sub": [  
        "arn:${AWS::Partition}:ses:${AWS::Region}:${AWS::AccountId}:identity/  
${identityName}",  
        {  
          "identityName": {  
            "Ref": "IdentityName"  
          }  
        }  
      ]  
    }  
  }  
]
```

SNSCrudPolicy

Memberikan izin untuk membuat, memublikasikan, dan berlangganan pada topik Amazon SNS.

```
"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

Memberikan izin untuk memublikasikan pesan ke topik Amazon Simple Notification Service (Amazon SNS).

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "sns:Publish"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sns:${AWS::Region}:${AWS::AccountId}:${topicName}",
        {
          "topicName": {
```

```

        "Ref": "TopicName"
      }
    }
  ]
}
]

```

SQSPollerPolicy

Memberikan izin untuk polling antrean Amazon Simple Queue Service (Amazon SQS).

```

"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

Memberikan izin untuk mengirim pesan ke antrean Amazon SQS.

```

"Statement": [
  {
    "Effect": "Allow",

```

```

    "Action": [
      "sqs:SendMessage*"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sqs:${AWS::Region}:${AWS::AccountId}:${queueName}",
        {
          "queueName": {
            "Ref": "QueueName"
          }
        }
      ]
    }
  ]
}
]

```

SSMParameterReadPolicy

Memberikan izin untuk mengakses parameter dari penyimpanan parameter Amazon EC2 Systems Manager (SSM) untuk memuat rahasia di akun ini. Gunakan ketika nama parameter tidak memiliki awalan garis miring.

Note

Jika Anda tidak menggunakan kunci default, Anda juga akan membutuhkan kebijakan `KMSDecryptPolicy`.

```

"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

Memberikan izin untuk mengakses parameter dari penyimpanan parameter Amazon EC2 Systems Manager (SSM) untuk memuat rahasia di akun ini. Gunakan ketika nama parameter memiliki awalan garis miring.

Note

Jika Anda tidak menggunakan kunci default, Anda juga akan membutuhkan kebijakan `KMSDecryptPolicy`.

```

"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

Memberikan izin untuk memulai eksekusi pada mesin keadaan Step Functions.

```

"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "states:StartExecution"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:states:${AWS::Region}:${AWS::AccountId}:stateMachine:
        ${stateMachineName}",
        {
          "stateMachineName": {
            "Ref": "StateMachineName"
          }
        }
      ]
    }
  }
]

```

TextractDetectAnalyzePolicy

Memberikan akses untuk mendeteksi dan menganalisa dokumen menggunakan Amazon Textract.

```

"Statement": [

```

```
{
  "Effect": "Allow",
  "Action": [
    "textract:DetectDocumentText",
    "textract:StartDocumentTextDetection",
    "textract:StartDocumentAnalysis",
    "textract:AnalyzeDocument"
  ],
  "Resource": "*"
}
```

TextractGetResultPolicy

Memberikan akses untuk mendapatkan dokumen yang terdeteksi dan teranalisis dari Amazon Textract.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "textract:GetDocumentTextDetection",
      "textract:GetDocumentAnalysis"
    ],
    "Resource": "*"
  }
]
```

TextractPolicy

Memberikan akses penuh menuju Amazon Textract.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "textract:*"
    ],
    "Resource": "*"
  }
]
```

VPCAccessPolicy

Memberikan akses untuk membuat, menghapus, menjelaskan, dan melepaskan antarmuka jaringan elastis.

```
"Statement": [  
  {  
    "Effect": "Allow",  
    "Action": [  
      "ec2:CreateNetworkInterface",  
      "ec2>DeleteNetworkInterface",  
      "ec2:DescribeNetworkInterfaces",  
      "ec2:DetachNetworkInterface"  
    ],  
    "Resource": "*"    
  }  
]
```

Mengelola izin dengan AWS CloudFormation mekanisme

Untuk mengontrol akses ke AWS sumber daya, AWS Serverless Application Model (AWS SAM) dapat menggunakan mekanisme yang sama seperti AWS CloudFormation. Untuk informasi selengkapnya, lihat [Mengontrol akses dengan AWS Identity and Access Management](#) dalam Panduan Pengguna AWS CloudFormation.

Ada tiga opsi utama untuk memberikan izin pengguna untuk mengelola aplikasi nirserver. Setiap opsi menyediakan pengguna dengan tingkat yang berbeda dari kontrol akses.

- Berikan izin administrator.
- Lampirkan kebijakan terkelola AWS yang diperlukan.
- Berikan izin AWS Identity and Access Management (IAM) tertentu.

Tergantung pada opsi yang Anda pilih, pengguna hanya dapat mengelola aplikasi nirserver yang berisi sumber daya AWS yang dimilikinya untuk izin akses.

Bagian berikut menjelaskan setiap opsi secara lebih mendetail.

Berikan izin administrator

Jika Anda memberikan izin administrator untuk pengguna, mereka dapat mengelola aplikasi nirserver yang berisi kombinasi sumber daya AWS. Ini adalah opsi yang paling sederhana, namun juga memberi pengguna set izin terluas, yang karenanya memungkinkan administrator melakukan tindakan dengan dampak tertinggi.

Untuk informasi selengkapnya tentang memberikan izin administrator ke pengguna, lihat [Membuat pengguna dan grup admin IAM pertama Anda](#) dalam Panduan Pengguna IAM.

Lampirkan Kebijakan terkelola AWS yang diperlukan

Anda dapat memberikan pengguna subset izin menggunakan [Kebijakan terkelola AWS](#), daripada memberikan izin administrator penuh. Jika Anda menggunakan opsi ini, pastikan bahwa set kebijakan terkelola AWS mencakup semua tindakan dan sumber daya yang diperlukan untuk aplikasi nirserver yang dikelola pengguna.

Misalnya, kebijakan terkelola AWS berikut cukup untuk [men-deploy sampel aplikasi Hello World](#):

- `AWSCloudFormationFullAccess`
- `IAMFullAccess`
- `AWSLambda_FullAccess`
- `AmazonAPIGatewayAdministrator`
- `AmazonS3FullAccess`
- `AmazonEC2ContainerRegistryFullAccess`

Untuk informasi tentang melampirkan kebijakan ke pengguna IAM, lihat [Mengubah izin untuk pengguna IAM](#) dalam Panduan Pengguna IAM.

Berikan izin IAM tertentu

Untuk tingkat kontrol akses terperinci, Anda dapat memberikan izin IAM tertentu untuk pengguna yang menggunakan [pernyataan kebijakan](#). Jika Anda menggunakan opsi ini, pastikan bahwa pernyataan kebijakan mencakup semua tindakan dan sumber daya yang diperlukan untuk aplikasi nirserver yang dikelola pengguna.

Praktik terbaik dengan opsi ini adalah menolak izin pengguna untuk membuat peran, termasuk peran eksekusi Lambda, sehingga peran tersebut tidak dapat memberikan izin yang meningkat. Jadi, Anda sebagai administrator harus membuat terlebih dahulu [Peran eksekusi Lambda](#) yang

akan ditentukan dalam aplikasi nirserver yang akan pengguna kelola. Untuk informasi selengkapnya tentang membuat peran eksekusi Lambda, lihat [Membuat peran eksekusi di konsol IAM](#).

Untuk [sampel aplikasi Hello World AWSLambdaBasicExecutionRole](#) cukup untuk menjalankan aplikasi. Setelah Anda membuat peran eksekusi Lambda, ubah file templat AWS SAM aplikasi Hello World sampel untuk menambahkan properti berikut ke sumber daya `AWS::Serverless::Function`:

```
Role: lambda-execution-role-arn
```

Pernyataan kebijakan berikut memberikan izin yang cukup bagi pengguna untuk men-deploy, memperbarui, dan menghapus aplikasi dengan diubah aplikasi Hello World di tempat:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CloudFormationTemplate",
      "Effect": "Allow",
      "Action": [
        "cloudformation:CreateChangeSet"
      ],
      "Resource": [
        "arn:aws:cloudformation:*:aws:transform/Serverless-2016-10-31"
      ]
    },
    {
      "Sid": "CloudFormationStack",
      "Effect": "Allow",
      "Action": [
        "cloudformation:CreateChangeSet",
        "cloudformation:CreateStack",
        "cloudformation>DeleteStack",
        "cloudformation:DescribeChangeSet",
        "cloudformation:DescribeStackEvents",
        "cloudformation:DescribeStacks",
        "cloudformation:ExecuteChangeSet",
        "cloudformation:GetTemplateSummary",
        "cloudformation:ListStackResources",
        "cloudformation:UpdateStack"
      ],
      "Resource": [
```

```
        "arn:aws:cloudformation:*:111122223333:stack/*"
    ]
},
{
    "Sid": "S3",
    "Effect": "Allow",
    "Action": [
        "s3:CreateBucket",
        "s3:GetObject",
        "s3:PutObject"
    ],
    "Resource": [
        "arn:aws:s3:::*/*"
    ]
},
{
    "Sid": "ECRRepository",
    "Effect": "Allow",
    "Action": [
        "ecr:BatchCheckLayerAvailability",
        "ecr:BatchGetImage",
        "ecr:CompleteLayerUpload",
        "ecr:CreateRepository",
        "ecr>DeleteRepository",
        "ecr:DescribeImages",
        "ecr:DescribeRepositories",
        "ecr:GetDownloadUrlForLayer",
        "ecr:GetRepositoryPolicy",
        "ecr:InitiateLayerUpload",
        "ecr:ListImages",
        "ecr:PutImage",
        "ecr:SetRepositoryPolicy",
        "ecr:UploadLayerPart"
    ],
    "Resource": [
        "arn:aws:ecr:*:111122223333:repository/*"
    ]
},
{
    "Sid": "ECRAuthToken",
    "Effect": "Allow",
    "Action": [
        "ecr:GetAuthorizationToken"
    ],
}
```

```

    "Resource": [
      "*"
    ]
  },
  {
    "Sid": "Lambda",
    "Effect": "Allow",
    "Action": [
      "lambda:AddPermission",
      "lambda:CreateFunction",
      "lambda>DeleteFunction",
      "lambda:GetFunction",
      "lambda:GetFunctionConfiguration",
      "lambda:ListTags",
      "lambda:RemovePermission",
      "lambda:TagResource",
      "lambda:UntagResource",
      "lambda:UpdateFunctionCode",
      "lambda:UpdateFunctionConfiguration"
    ],
    "Resource": [
      "arn:aws:lambda:*:111122223333:function:*"
    ]
  },
  {
    "Sid": "IAM",
    "Effect": "Allow",
    "Action": [
      "iam:CreateRole",
      "iam:AttachRolePolicy",
      "iam>DeleteRole",
      "iam:DetachRolePolicy",
      "iam:GetRole",
      "iam:TagRole"
    ],
    "Resource": [
      "arn:aws:iam:*:111122223333:role/*"
    ]
  },
  {
    "Sid": "IAMPassRole",
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": "*",

```

```
    "Condition": {
      "StringEquals": {
        "iam:PassedToService": "lambda.amazonaws.com"
      }
    },
    {
      "Sid": "APIGateway",
      "Effect": "Allow",
      "Action": [
        "apigateway:DELETE",
        "apigateway:GET",
        "apigateway:PATCH",
        "apigateway:POST",
        "apigateway:PUT"
      ],
      "Resource": [
        "arn:aws:apigateway:*:*:*"
      ]
    }
  ]
}
```

Note

Pernyataan kebijakan contoh di bagian ini memberikan izin yang cukup bagi Anda untuk men-deploy, memperbarui, dan menghapus [sampel aplikasi Hello World](#). Jika menambahkan jenis sumber daya tambahan ke aplikasi, Anda perlu memperbarui pernyataan kebijakan untuk menyertakan yang berikut ini:

1. Izin untuk aplikasi Anda untuk memanggil tindakan layanan.
2. Kepala layanan, jika diperlukan untuk tindakan layanan.

Misalnya, jika Anda menambahkan alur kerja Step Functions, Anda mungkin perlu menambahkan izin untuk tindakan yang tercantum [di sini](#), dan `principalstates.amazonaws.com` layanan.

Untuk informasi selengkapnya tentang kebijakan IAM terkelola, lihat [Mengelola kebijakan IAM](#) dalam Panduan Pengguna IAM.

Menjadwalkan acara dengan EventBridge Scheduler

Apa itu Amazon EventBridge Scheduler?

Amazon EventBridge Scheduler adalah layanan penjadwalan yang memungkinkan Anda membuat, memulai, dan mengelola puluhan juta acara dan tugas di semua layanan. AWS Untuk mempelajari selengkapnya tentang Amazon EventBridge Scheduler, lihat [Apa itu Amazon EventBridge Scheduler?](#) di Panduan Pengguna EventBridge Penjadwal.

Topik

- [EventBridge Dukungan penjadwal di AWS SAM](#)
- [Membuat acara EventBridge Scheduler di AWS SAM](#)
- [Contoh-contoh](#)
- [Pelajari selengkapnya](#)

EventBridge Dukungan penjadwal di AWS SAM

Spesifikasi template AWS Serverless Application Model (AWS SAM) menyediakan sintaks sederhana dan singkat yang dapat Anda gunakan untuk menjadwalkan acara dengan EventBridge Scheduler untuk dan. AWS Lambda AWS Step Functions

Membuat acara EventBridge Scheduler di AWS SAM

Tetapkan `ScheduleV2` properti sebagai jenis acara di AWS SAM template Anda untuk menentukan acara EventBridge Scheduler Anda. Properti ini mendukung `AWS::Serverless::Function` dan jenis `AWS::Serverless::StateMachine` sumber daya.

```
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 Penjadwalan acara Scheduler juga mendukung antrian surat mati (DLQ) untuk acara yang belum diproses. Untuk informasi selengkapnya tentang antrian huruf mati, lihat [Mengonfigurasi antrian huruf mati untuk Penjadwal di Panduan](#) Pengguna Penjadwal. EventBridge EventBridge

Ketika ARN DLQ ditentukan, AWS SAM mengonfigurasi izin untuk jadwal Scheduler untuk mengirim pesan ke DLQ. Ketika ARN DLQ tidak ditentukan, AWS SAM akan membuat sumber daya DLQ.

Contoh-contoh

Contoh dasar mendefinisikan acara EventBridge Scheduler dengan AWS SAM

```

Transform: AWS::Serverless-2016-10-31
Resources:
  MyLambdaFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: python3.8
      InlineCode: |
        def handler(event, context):
            print(event)
            return {'body': 'Hello World!', 'statusCode': 200}
    MemorySize: 128
    Events:
      Schedule:
        Type: ScheduleV2
        Properties:
          ScheduleExpression: rate(1 minute)
          Input: '{"hello": "simple"}'

```

```
MySFNFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: index.handler
    Runtime: python3.8
    InlineCode: |
      def handler(event, context):
        print(event)
        return {'body': 'Hello World!', 'statusCode': 200}
    MemorySize: 128

StateMachine:
  Type: AWS::Serverless::StateMachine
  Properties:
    Type: STANDARD
  Definition:
    StartAt: MyLambdaState
    States:
      MyLambdaState:
        Type: Task
        Resource: !GetAtt MySFNFunction.Arn
        End: true
  Policies:
    - LambdaInvokePolicy:
        FunctionName: !Ref MySFNFunction
  Events:
    Events:
      Schedule:
        Type: ScheduleV2
        Properties:
          ScheduleExpression: rate(1 minute)
          Input: '{"hello": "simple"}
```

Pelajari selengkapnya

Untuk mempelajari lebih lanjut tentang mendefinisikan properti ScheduleV2 EventBridge Scheduler, lihat:

- [ScheduleV2](#) untuk `AWS::Serverless::Function`.
- [ScheduleV2](#) untuk `AWS::Serverless::StateMachine`.

Memvalidasi file templat AWS SAM

Validasi templat Anda dengan [sam validate](#). Saat ini, perintah ini memvalidasi bahwa templat yang disediakan adalah JSON / YAML yang valid. Seperti kebanyakan AWS SAM CLI perintah, ia mencari `template.[yaml|yml]` file di direktori kerja Anda saat ini secara default. Anda dapat menentukan file/lokasi templat yang berbeda dengan opsi `-t` atau `--template`.

Contoh:

```
$ sam validate
<path-to-template>/template.yaml is a valid SAM Template
```

Note

Perintah `sam validate` memerlukan kredensial AWS untuk dikonfigurasi. Lihat informasi yang lebih lengkap di [Mengkonfigurasi AWS SAMCLI](#).

Cara menggunakan lapisan

Menggunakan AWS SAM, Anda dapat menyertakan lapisan dalam aplikasi nirserver Anda. Untuk informasi lebih lanjut mengenai lapisan, lihat [lapisan AWS Lambda](#) di Panduan Developer AWS Lambda.

Bagian ini menyediakan informasi tentang hal berikut:

- Termasuk lapisan dalam aplikasi Anda
- Bagaimana lapisan di-cache secara lokal

Untuk informasi lebih lanjut tentang membuat lapisan kustom, lihat [Membangun lapisan](#).

Termasuk lapisan dalam aplikasi Anda

Untuk menyertakan lapisan dalam aplikasi Anda, gunakan `Layers` Properti dari [AWS::Serverless::Function](#) jenis sumber daya.

Berikut adalah contoh AWS SAM templat dengan fungsi Lambda yang mencakup sebuah lapisan:

```

ServerlessFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: .
    Handler: my_handler
    Runtime: Python3.7
    Layers:
      - <LayerVersion ARN>

```

Bagaimana lapisan di-cache secara lokal

Ketika Anda meminta fungsi Anda menggunakan salah satu `local`, paket lapisan fungsi Anda diunduh dan di-cache pada host lokal Anda.

Tabel berikut menunjukkan lokasi direktori cache default untuk sistem operasi yang berbeda.

OS	Lokasi
Windows 7	C:\Users\ <user>\AppData\Roaming\AWS SAM</user>
Windows 8	C:\Users\ <user>\AppData\Roaming\AWS SAM</user>
Windows 10	C:\Users\ <user>\AppData\Roaming\AWS SAM</user>
macOS	~/.aws-sam/layers-pkg
Unix	~/.aws-sam/layers-pkg

Setelah paket di-cache, lapisan AWS SAM CLI melapisi layer ke image Docker yang digunakan untuk memanggil fungsi Anda. Itu AWS SAM CLI menghasilkan nama-nama gambar yang dibangunnya, serta LayerVersions yang disimpan di cache. Anda dapat menemukan detail lebih lanjut tentang skema di bagian berikut.

Untuk memeriksa lapisan overlay, jalankan perintah berikut untuk memulai sesi bash pada gambar yang ingin Anda periksa:

```

docker run -it --entrypoint=/bin/bash samcli/lambda:<Tag following the schema outlined
in Docker Image Tag Schema> -i

```

Skema nama Direktori Caching Layer

Mengingat LayerVersionArn yang didefinisikan dalam template Anda, AWS SAM CLI ekstrak LayerName dan Versi dari ARN. Ini menciptakan sebuah direktori untuk menempatkan isi lapisan di dalam nama LayerName-Version-`<first 10 characters of sha256 of ARN>`.

Contoh:

```
ARN = arn:aws:lambda:us-west-2:111111111111:layer:myLayer:1
Directory name = myLayer-1-926eeb5ff1
```

Skema tag Gambar Docker

Untuk komputasi lapisan unik hash, gabungkan semua nama lapisan unik dengan tanda pembatas '-', ambil hash SHA256, kemudian ambil 10 karakter pertama.

Contoh:

```
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
```

Nama-nama unik dikomputasi sama dengan skema nama pada Lapisan Direktori Caching:

```
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
```

Untuk komputasi lapisan unik hash, gabungkan semua nama lapisan unik dengan tanda pembatas '-', ambil hash sha256, kemudian ambil 25 karakter pertama:

```
myLayer-1-926eeb5ff1-mySecondLayer-1-6bc1022bdf = 2dd7ac5ffb30d515926aef
```

Kemudian gabungkan nilai ini dengan runtime dan arsitektur fungsi, dengan pembatas '-':

```
python3.7-x86_64-2dd7ac5ffb30d515926aefffd
```

Menggunakan aplikasi nest

Sebuah aplikasi nirserver dapat mencakup satu aplikasi nest atau lebih. Anda dapat men-deploy aplikasi nest sebagai artefak yang berdiri sendiri atau sebagai komponen dari aplikasi yang lebih besar.

Seiring pertumbuhan arsitektur nirserver, pola umum muncul pada komponen yang sama didefinisikan dalam beberapa templat aplikasi. Anda sekarang dapat memisahkan pola umum sebagai aplikasi khusus, lalu menyusunnya sebagai bagian dari templat aplikasi baru atau yang sudah ada. Dengan aplikasi nest, Anda tetap dapat lebih fokus pada logika bisnis yang unik untuk aplikasi Anda.

Untuk menentukan aplikasi nest di aplikasi nirserver Anda, gunakan tipe sumber daya [AWS::Serverless::Application](#).

Anda dapat menentukan aplikasi nest dari dua sumber berikut:

- Aplikasi AWS Serverless Application Repository – Anda dapat menentukan aplikasi nest dengan menggunakan aplikasi yang tersedia untuk akun Anda di AWS Serverless Application Repository. Ini dapat berupa aplikasi privat di akun Anda, aplikasi yang dibagikan secara privat dengan akun Anda, atau aplikasi yang dibagikan secara publik di AWS Serverless Application Repository. Untuk informasi selengkapnya tentang tingkat izin deployment yang berbeda, lihat [Izin Deployment Aplikasi](#) dan [Memublikasikan Aplikasi](#) di Panduan Developer AWS Serverless Application Repository.
- Aplikasi lokal – Anda dapat menentukan aplikasi nest dengan menggunakan aplikasi yang disimpan di sistem file lokal Anda.

Lihat bagian berikut untuk detail tentang cara menggunakan AWS SAM untuk menentukan kedua tipe aplikasi nest ini di aplikasi nirserver Anda.

Note

Jumlah maksimum aplikasi yang dapat di-nest dalam aplikasi nirserver adalah 200.
Jumlah maksimum parameter yang dapat dimiliki aplikasi nest adalah 60.

Mendefinisikan aplikasi nest dari AWS Serverless Application Repository

Anda dapat menentukan aplikasi nest dengan menggunakan aplikasi yang tersedia di AWS Serverless Application Repository. Anda juga dapat menyimpan dan mendistribusikan aplikasi yang berisi aplikasi nest menggunakan AWS Serverless Application Repository. Untuk meninjau detail aplikasi nest di AWS Serverless Application Repository, Anda dapat menggunakan SDK AWS, AWS CLI, atau konsol Lambda.

Untuk menentukan aplikasi yang di-hosting di AWS Serverless Application Repository dalam templat AWS SAM aplikasi nirserver Anda, gunakan tombol Salin sebagai Sumber Daya SAM di halaman detail setiap aplikasi AWS Serverless Application Repository. Untuk melakukannya, ikuti langkah-langkah berikut:

1. Pastikan Anda masuk ke AWS Management Console.
2. Temukan aplikasi yang ingin Anda nest di AWS Serverless Application Repository dengan menggunakan langkah-langkah di bagian [Menjelajah, Mencari, dan Men-deploy Aplikasi](#) dari Panduan Developer AWS Serverless Application Repository.
3. Pilih tombol Salin sebagai Sumber Daya SAM. Bagian templat SAM untuk aplikasi yang Anda lihat sekarang ada di clipboard Anda.
4. Tempelkan bagian templat SAM ke bagian Resources: dari file templat SAM untuk aplikasi yang ingin Anda nest di aplikasi ini.

Berikut ini adalah contoh bagian templat SAM untuk aplikasi nest yang di-hosting di 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
```

Jika tidak ada pengaturan parameter yang diperlukan, Anda dapat menghilangkan bagian `Parameters:` pada templat.

Important

Aplikasi yang berisi aplikasi nest yang di-hosting di AWS Serverless Application Repository mewarisi pembatasan berbagi aplikasi nest.

Misalnya, anggaplah aplikasi dibagikan secara publik, tetapi berisi aplikasi nest yang hanya dibagikan secara privat dengan akun AWS yang membuat aplikasi induk. Dalam hal ini, jika akun AWS Anda tidak memiliki izin untuk men-deploy aplikasi bersarang, Anda tidak dapat men-deploy aplikasi induk. Untuk informasi lebih lanjut tentang izin untuk men-deploy aplikasi, lihat [Izin Deployment Aplikasi](#) dan [Memublikasikan Aplikasi](#) di Panduan Developer AWS Serverless Application Repository.

Mendefinisikan aplikasi nest dari sistem file lokal

Anda dapat menentukan aplikasi nest dengan menggunakan aplikasi yang disimpan di sistem file lokal Anda. Anda melakukan ini dengan menentukan jalur ke file templat AWS SAM yang disimpan di sistem file lokal Anda.

Berikut ini adalah contoh bagian templat SAM untuk aplikasi lokal nest:

```
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
```

Jika tidak ada pengaturan parameter, Anda dapat menghilangkan bagian `Parameters:` pada templat.

Men-deploy aplikasi nest

Anda dapat menerapkan aplikasi bersarang Anda dengan menggunakan perintah. AWS SAM CLI `aws sam deploy` Untuk detail selengkapnya, lihat [Men-deploy aplikasi nirserver](#).

Note

Saat Anda men-deploy aplikasi yang berisi aplikasi nest, Anda harus mengakuinya. Anda melakukan ini dengan meneruskan `CAPABILITY_AUTO_EXPAND` ke [CreateCloudFormationChangeSet API](#), atau menggunakan perintah. `aws serverlessrepo create-cloud-formation-change-set` AWS CLI Untuk informasi selengkapnya tentang mengakui aplikasi nest, lihat [Mengakui IAM Role, Kebijakan Sumber Daya, dan Aplikasi Bersarang saat Men-deploy Aplikasi](#) di Panduan Developer AWS Serverless Application Repository.

Pengendalian akses ke API dari API Gateway

Untuk mengendalikan siapa saja yang dapat mengakses API dari Amazon API Gateway, Anda dapat mengaktifkan otorisasi dalam templat AWS SAM Anda.

AWS SAM mendukung beberapa mekanisme untuk mengontrol akses ke API Gateway API Anda. Kumpulan mekanisme yang didukung berbeda antara tipe sumber daya `AWS::Serverless::HttpApi` dan `AWS::Serverless::Api`.

Tabel berikut merangkum mekanisme yang didukung oleh setiap tipe sumber daya.

Mekanisme untuk mengendalikan akses	<code>AWS::Serverless::HttpApi</code>	<code>AWS::Serverless::Api</code>
Otorisator Lambda	✓	✓
Izin IAM		✓
Kumpulan pengguna Amazon Cognito	✓ *	✓
Kunci API		✓

Mekanisme untuk mengendalikan akses	AWS::Serverless::HttpApi	AWS::Serverless::Api
Kebijakan sumber daya		✓
OAuth 2.0/otorisasi JWT	✓	

* Anda dapat menggunakan Amazon Cognito sebagai penerbit JSON Web Token (JWT) dengan tipe sumber daya `AWS::Serverless::HttpApi`.

- Otorisasi Lambda – Otorisasi Lambda (sebelumnya dikenal sebagai custom authorizer) adalah fungsi Lambda yang Anda berikan untuk mengendalikan akses ke API Anda. Saat API Anda dipanggil, fungsi Lambda ini dipanggil dengan konteks permintaan atau token otorisasi yang disediakan aplikasi klien. Fungsi Lambda merespons apakah pemanggil diotorisasi untuk melakukan operasi yang diminta.

Tipe sumber daya `AWS::Serverless::HttpApi` dan `AWS::Serverless::Api` mendukung otorisasi Lambda.

Untuk informasi selengkapnya tentang otorisasi Lambda dengan `AWS::Serverless::HttpApi`, lihat [Bekerja dengan otorisasi AWS Lambda untuk API HTTP](#) di Panduan Developer API Gateway. Untuk informasi selengkapnya tentang otorisasi Lambda dengan `AWS::Serverless::Api`, lihat [Gunakan otorisasi Lambda API Gateway](#) di Panduan Developer API Gateway.

Untuk contoh otorisasi Lambda untuk kedua tipe sumber daya, lihat [Contoh otorisasi Lambda](#).

- Izin IAM – Anda dapat mengendalikan siapa saja yang dapat menggunakan API Anda menggunakan [Izin \(IAM\)AWS Identity and Access Management](#). Pengguna yang memanggil API Anda harus diautentikasi dengan kredensial IAM. Panggilan ke API Anda berhasil hanya jika ada kebijakan IAM yang dilampirkan ke pengguna IAM yang mewakili pemanggil API, grup IAM yang berisi pengguna, atau IAM role yang diasumsikan pengguna.

Hanya tipe sumber daya `AWS::Serverless::Api` yang mendukung izin IAM.

Untuk informasi selengkapnya, lihat [Mengendalikan akses ke API dengan izin IAM](#) di Panduan Developer API Gateway. Sebagai contoh, lihat [Contoh izin IAM](#).

- Kolam pengguna Amazon Cognito – Kolam pengguna Amazon Cognito adalah direktori pengguna di Amazon Cognito. Klien API Anda harus terlebih dahulu membuat pengguna masuk ke kolam

pengguna dan mendapatkan identitas atau token akses untuk pengguna tersebut. Kemudian klien memanggil API Anda dengan salah satu token yang dikembalikan. Panggilan API hanya berhasil jika token yang diperlukan valid.

Tipe sumber daya `AWS::Serverless::Api` mendukung kolam pengguna Amazon Cognito. Tipe sumber daya `AWS::Serverless::HttpApi` mendukung penggunaan Amazon Cognito sebagai penerbit JWT.

Untuk informasi selengkapnya, lihat [Mengendalikan akses ke REST API menggunakan kolam pengguna Amazon Cognito sebagai pemberi otorisasi](#) di Panduan Developer API Gateway. Sebagai contoh, lihat [Contoh kolam pengguna Amazon Cognito](#).

- Kunci API – Kunci API adalah nilai string alfanumerik yang Anda distribusi ke pelanggan developer aplikasi untuk memberikan akses ke API Anda.

Hanya tipe sumber daya `AWS::Serverless::Api` yang mendukung kunci API.

Untuk informasi selengkapnya tentang kunci API, lihat [Membuat dan menggunakan rencana penggunaan dengan kunci API](#) di Panduan Developer API Gateway. Untuk contoh kunci API, lihat [Contoh kunci API](#).

- Kebijakan sumber daya – Kebijakan sumber daya adalah dokumen kebijakan JSON yang dapat Anda lampirkan ke API dari API Gateway. Gunakan kebijakan sumber daya untuk mengendalikan apakah principal tertentu (biasanya pengguna atau peran IAM) dapat memanggil API.

Hanya tipe sumber daya `AWS::Serverless::Api` yang mendukung kebijakan sumber daya sebagai mekanisme untuk mengendalikan akses ke API Gateway.

Untuk informasi selengkapnya tentang kebijakan sumber daya, lihat [Mengendalikan akses ke API dengan kebijakan sumber daya API Gateway](#) di Panduan Developer API Gateway. Untuk contoh kebijakan sumber daya, lihat [Contoh kebijakan sumber daya](#).

- Otorisasi OAuth 2.0/JWT – Anda dapat menggunakan JWTs sebagai bagian dari kerangka kerja [OpenID Connect \(OIDC\)](#) dan [OAuth 2.0](#) untuk mengendalikan akses ke API Anda. API Gateway memvalidasi JWT yang dikirimkan klien dengan permintaan API, dan mengizinkan atau menolak permintaan berdasarkan validasi token dan, secara opsional, cakupan dalam token.

Hanya tipe sumber daya `AWS::Serverless::HttpApi` yang mendukung otorisasi OAuth 2.0/JWT.

Untuk informasi selengkapnya, lihat [Mengendalikan akses ke API HTTP dengan otorisasi JWT](#) di Panduan Developer API Gateway. Sebagai contoh, lihat [Contoh otorisasi OAuth 2.0/JWT](#).

Memilih mekanisme untuk mengendalikan akses

Mekanisme yang Anda pilih yang digunakan untuk mengendalikan akses ke API dari API Gateway bergantung pada beberapa faktor. Misalnya, jika Anda memiliki proyek greenfield tanpa menyiapkan otorisasi atau kendali akses, maka kolam pengguna Amazon Cognito mungkin menjadi pilihan terbaik Anda. Ini karena saat Anda menyiapkan kolam pengguna, Anda juga secara otomatis menyiapkan autentikasi dan kendali akses.

Namun, jika aplikasi Anda sudah menyiapkan autentikasi, maka menggunakan otorisasi Lambda mungkin merupakan pilihan terbaik Anda. Ini karena Anda dapat memanggil layanan autentikasi yang ada dan mengembalikan dokumen kebijakan berdasarkan respons. Selain itu, jika aplikasi Anda memerlukan autentikasi khusus atau logika kendali akses yang tidak didukung oleh kolam pengguna, maka otorisasi Lambda mungkin merupakan pilihan terbaik Anda.

Ketika Anda telah memilih mekanisme mana yang akan digunakan, lihat bagian yang sesuai [Contoh-contoh](#) untuk cara menggunakan AWS SAM untuk mengonfigurasi aplikasi Anda untuk menggunakan mekanisme itu.

Menyesuaikan respons kesalahan

Anda dapat menggunakan AWS SAM untuk menyesuaikan konten dari beberapa respons kesalahan API Gateway. Hanya tipe sumber daya `AWS::Serverless::Api` yang mendukung respons API Gateway yang disesuaikan.

Untuk informasi selengkapnya tentang respons API Gateway, lihat [Respon gateway di API Gateway](#) di Panduan Developer API Gateway. Untuk contoh respons yang disesuaikan, lihat [Contoh respons yang disesuaikan](#).

Contoh-contoh

- [Contoh otorisasi Lambda](#)
- [Contoh izin IAM](#)
- [Contoh kolam pengguna Amazon Cognito](#)
- [Contoh kunci API](#)

- [Contoh kebijakan sumber daya](#)
- [Contoh otorisasi OAuth 2.0/JWT](#)
- [Contoh respons yang disesuaikan](#)

Contoh otorisasi Lambda

Tipe sumber daya `AWS::Serverless::Api` yang mendukung dua tipe otorisasi Lambda: otorisasi `TOKEN` dan otorisasi `REQUEST`. Tipe sumber daya `AWS::Serverless::HttpApi` hanya mendukung otorisasi `REQUEST`. Berikut ini adalah contoh dari setiap tipe.

Contoh **TOKEN** otorisasi Lambda () `AWS::Serverless::Api`

Anda dapat mengontrol akses ke API Anda dengan menentukan otorisasi `TOKEN` Lambda dalam template Anda. AWS SAM Untuk melakukannya, Anda menggunakan tipe data [ApiAuth](#).

Berikut ini adalah contoh bagian AWS SAM template untuk Authorizer Lambda `TOKEN`:

Note

Dalam contoh berikut, SAM dihasilkan `FunctionRole` secara implisit.

```
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

```

Untuk informasi selengkapnya tentang otorisasi Lambda, lihat [Gunakan otorisasi Lambda API Gateway](#) di Panduan Developer API Gateway.

Contoh **REQUEST** otorisasi Lambda () AWS::Serverless::Api

Anda dapat mengontrol akses ke API Anda dengan menentukan otorisasi REQUEST Lambda dalam template Anda. AWS SAM Untuk melakukannya, Anda menggunakan tipe data [ApiAuth](#).

Berikut ini adalah contoh bagian AWS SAM template untuk Authorizer LambdaREQUEST:

```

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
```

Untuk informasi selengkapnya tentang otorisasi Lambda, lihat [Gunakan otorisasi Lambda API Gateway](#) di Panduan Developer API Gateway.

Contoh otorisasi Lambda () AWS::Serverless::HttpApi

Anda dapat mengontrol akses ke API HTTP Anda dengan mendefinisikan otorisasi Lambda dalam template Anda. AWS SAM Untuk melakukannya, Anda menggunakan tipe data [HttpApiAuth](#).

Berikut ini adalah contoh bagian AWS SAM template untuk Authorizer Lambda:

```
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
```

Contoh izin IAM

Anda dapat mengendalikan akses ke API dengan menentukan izin IAM di dalam templat AWS SAM . Untuk melakukannya, Anda menggunakan tipe data [ApiAuth](#).

Berikut ini adalah contoh AWS SAM template yang digunakan untuk izin IAM:

```
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}
```

Untuk informasi selengkapnya tentang izin IAM, lihat [Mengendalikan akses untuk menerapkan API](#) di Panduan Developer API Gateway.

Contoh kolam pengguna Amazon Cognito

Anda dapat mengendalikan akses ke API dengan menentukan kolam pengguna Amazon Cognito di dalam templat AWS SAM . Untuk melakukannya, Anda menggunakan tipe data [ApiAuth](#).

Berikut ini adalah contoh bagian AWS SAM template untuk kumpulan pengguna:

```
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
```

Untuk informasi selengkapnya tentang kolam pengguna Amazon Cognito, lihat [Mengendalikan akses ke REST API menggunakan kolam pengguna Amazon Cognito sebagai pemberi otorisasi](#) di Panduan Developer API Gateway.

Contoh kunci API

Anda dapat mengontrol akses ke API Anda dengan mewajibkan kunci API dalam AWS SAM template Anda. Untuk melakukannya, Anda menggunakan tipe data [ApiAuth](#).

Berikut ini adalah contoh bagian AWS SAM template untuk kunci API:

```
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
```

Untuk informasi selengkapnya tentang kunci API, lihat [Membuat dan menggunakan rencana penggunaan dengan kunci API](#) di Panduan Developer API Gateway.

Contoh kebijakan sumber daya

Anda dapat mengendalikan akses ke API dengan melampirkan kebijakan sumber daya di dalam templat AWS SAM . Untuk melakukannya, Anda menggunakan tipe data [ApiAuth](#).

Berikut ini adalah contoh AWS SAM template untuk API pribadi. API pribadi harus memiliki kebijakan sumber daya untuk diterapkan.

```
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: '*'
    }
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

```

Untuk informasi selengkapnya tentang kebijakan sumber daya, lihat [Mengendalikan akses ke API dengan kebijakan sumber daya API Gateway](#) di Panduan Developer API Gateway. Untuk informasi selengkapnya tentang API pribadi, lihat [Membuat API pribadi di Amazon API Gateway](#) di Panduan Pengembang API Gateway.

Contoh otorisasi OAuth 2.0/JWT

Anda dapat mengendalikan akses ke API menggunakan JWTs sebagai bagian dari kerangka kerja [Connect OpenID \(OIDC\)](#) dan [OAuth 2.0](#). Untuk melakukannya, Anda menggunakan tipe data [HttpApiAuth](#).

Berikut ini adalah bagian contoh AWS SAM template untuk otorisasi OAuth 2.0/JWT:

```

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

```

Untuk informasi selengkapnya tentang otorisasi OAuth 2.0/JWT, lihat [Mengendalikan akses ke API HTTP dengan otorisasi JWT](#) di Panduan Developer API Gateway.

Contoh respons yang disesuaikan

Anda dapat menyesuaikan beberapa respons kesalahan API Gateway dengan menentukan header respons dalam templat AWS SAM Anda. Untuk melakukannya, Anda menggunakan tipe data [Gateway Response Object](#).

Berikut ini adalah contoh AWS SAM template yang membuat respons khusus untuk DEFAULT_5XX kesalahan tersebut.

```

AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  MyApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod

```

```
GatewayResponses:
  DEFAULT_5XX:
    ResponseParameters:
      Headers:
        Access-Control-Expose-Headers: "'WWW-Authenticate'"
        Access-Control-Allow-Origin: "'*'"
        ErrorHeader: "'MyCustomErrorHeader'"
      ResponseTemplates:
        application/json: "{\"message\": \"Error on the $context.resourcePath
resource\" }"

GetFunction:
  Type: AWS::Serverless::Function
  Properties:
    Runtime: python3.10
    Handler: index.handler
    InlineCode: |
      def handler(event, context):
        raise Exception('Check out the new response!')
  Events:
    GetResource:
      Type: Api
      Properties:
        Path: /error
        Method: get
        RestApiId: !Ref MyApi
```

Untuk informasi selengkapnya tentang respons API Gateway, lihat [Respon gateway di API Gateway](#) di Panduan Developer API Gateway.

Mengorkestrasi sumber daya AWS dengan AWS Step Functions

Anda dapat menggunakan [AWS Step Functions](#) untuk mengatur fungsi AWS Lambda dan sumber daya AWS lainnya untuk membentuk alur kerja yang kompleks dan tangguh.

Note

Untuk mengelola AWS SAM template yang berisi mesin status Step Functions, Anda harus menggunakan versi 0.52.0 atau yang lebih baru. AWS SAM CLI Untuk memeriksa versi yang Anda miliki, jalankan perintah `sam --version`.

Step Functions didasarkan pada konsep [tugas](#) dan [mesin status](#). Anda menentukan mesin status yang menggunakan [Amazon States Language](#) berbasis JSON. [Konsol Step Functions](#) menampilkan tampilan grafis dari struktur mesin status sehingga Anda dapat secara visual memeriksa logika mesin status Anda dan memantau eksekusi.

Dengan dukungan Step Functions di AWS Serverless Application Model (AWS SAM), Anda dapat melakukan hal-hal berikut:

- Tentukan mesin status, baik secara langsung dalam templat AWS SAM atau dalam file terpisah
- Buat peran eksekusi mesin status melalui templat kebijakan AWS SAM, kebijakan sebaris, atau kebijakan terkelola
- Memicu eksekusi mesin status dengan API Gateway atau EventBridge peristiwa Amazon, sesuai jadwal dalam AWS SAM templat, atau dengan memanggil API secara langsung
- Gunakan [Templat Kebijakan AWS SAM](#) yang tersedia untuk pola pengembangan Step Functions yang umum.

Contoh

Cuplikan contoh berikut dari file templat AWS SAM menentukan mesin status Step Functions dalam file penentuan. Perhatikan bahwa file `my_state_machine.asl.json` harus ditulis dalam [Amazon States Language](#).

```
AWSTemplateFormatVersion: "2010-09-09"
Transform: AWS::Serverless-2016-10-31
Description: Sample SAM template with Step Functions State Machine

Resources:
  MyStateMachine:
    Type: AWS::Serverless::StateMachine
    Properties:
      DefinitionUri: statemachine/my_state_machine.asl.json
      ...
```

Untuk mengunduh aplikasi sampel AWS SAM yang mencakup mesin status Step Functions, lihat [Buat Mesin Status Step Functions Menggunakan AWS SAM](#) di Panduan Developer AWS Step Functions.

Informasi lain

Untuk mempelajari selengkapnya tentang Step Functions dan menggunakannya dengan AWS SAM, lihat berikut ini:

- [Bagaimana cara AWS Step Functions kerja](#)
- [AWS Step Functions dan AWS Serverless Application Model](#)
- [Tutorial: Membuat Mesin Status Step Functions Menggunakan AWS SAM](#)
- [AWS SAMSpesifikasi: AWS::Serverless::StateMachine](#)

Mengonfigurasi penandatanganan kode untuk aplikasi AWS SAM

Anda dapat menggunakan AWS SAM untuk mengaktifkan penandatanganan kode dengan aplikasi nirserver untuk membantu memastikan bahwa hanya kode tepercaya yang di-deploy. Untuk informasi selengkapnya tentang fitur penandatanganan kode, lihat [Mengonfigurasi penandatanganan kode untuk fungsi Lambda](#) di Panduan Developer AWS Lambda.

Sebelum Anda dapat mengonfigurasi penandatanganan kode untuk aplikasi nirserver, Anda harus membuat profil penandatanganan menggunakan AWS Signer. Anda menggunakan profil penandatanganan ini untuk tugas-tugas berikut:

1. Membuat konfigurasi penandatanganan kode – Menyatakan sumber daya [AWS::Lambda::CodeSigningConfig](#) untuk menentukan profil penandatanganan penerbit tepercaya dan untuk mengatur tindakan kebijakan untuk pemeriksaan validasi. Anda dapat menyatakan objek ini di templat AWS SAM yang sama seperti fungsi nirserver Anda, dalam templat AWS SAM yang berbeda, atau dalam templat AWS CloudFormation. Anda kemudian mengaktifkan penandatanganan kode untuk fungsi nirserver dengan menentukan properti [CodeSigningConfigArn](#) fungsi tersebut dengan Amazon Resource Name (ARN) dari sumber daya [AWS::Lambda::CodeSigningConfig](#).
2. Menandatangani kode Anda – Gunakan perintah [sam package](#) atau [sam deploy](#) dengan opsi `--signing-profiles`.

Note

Agar kode Anda berhasil ditandatangani dengan perintah `sam package` atau `sam deploy`, versioning untuk bucket Amazon S3 yang Anda gunakan harus diaktifkan dengan perintah ini.

Jika Anda menggunakan Bucket Amazon S3 yang dibuat AWS SAM untuk Anda, versioning diaktifkan secara otomatis. Untuk informasi selengkapnya tentang pembuatan versi bucket Amazon S3 dan petunjuk untuk mengaktifkan pembuatan versi di bucket Amazon S3 yang Anda berikan, lihat [Menggunakan pembuatan versi di bucket Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon](#).

Saat Anda men-deploy aplikasi nirserver, Lambda melakukan pemeriksaan validasi pada semua fungsi yang penandatanganan kodenya telah Anda aktifkan. Lambda juga melakukan pemeriksaan validasi pada setiap lapisan yang fungsi lambda-nya bergantung pada lapisan tersebut. Untuk informasi selengkapnya tentang pemeriksaan validasi Lambda, lihat [Validasi tanda tangan](#) di Panduan Developer AWS Lambda.

Contoh

Membuat profil penandatanganan

Untuk membuat profil penandatanganan, jalankan perintah berikut:

```
aws signer put-signing-profile --platform-id "AWSLambda-SHA384-ECDSA" --profile-name MySigningProfile
```

Jika perintah sebelumnya berhasil, Anda akan melihat ARN profil penandatanganan dikembalikan. Sebagai contoh:

```
{
  "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"
}
```

Bidang `profileVersionArn` berisi ARN untuk digunakan saat Anda membuat konfigurasi penandatanganan kode.

Membuat konfigurasi penandatanganan kode dan mengaktifkan penandatanganan kode untuk suatu fungsi

Contoh templat AWS SAM berikut menyatakan sumber daya

[AWS::Lambda::CodeSigningConfig](#) dan mengaktifkan penandatanganan kode untuk fungsi

Lambda. Dalam contoh ini, jika pemeriksaan tanda tangan gagal, akan ditemukan satu profil tepercaya, dan penolakan deployment.

```
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"
```

Menandatangani kode Anda

Anda dapat menandatangani kode saat mengemas atau men-deploy aplikasi Anda. Tentukan opsi `--signing-profiles` dengan perintah `sam package` atau `sam deploy`, seperti yang ditunjukkan pada contoh perintah berikut.

Menandatangani kode fungsi saat mengemas aplikasi Anda:

```
sam package --signing-profiles HelloWorld=MySigningProfile --s3-bucket test-bucket --
output-template-file packaged.yaml
```

Menandatangani kode fungsi dan lapisan tempat fungsi Anda bergantung, saat mengemas aplikasi Anda:

```
sam package --signing-profiles HelloWorld=MySigningProfile MyLayer=MySigningProfile --
s3-bucket test-bucket --output-template-file packaged.yaml
```

Menandatangani kode fungsi dan lapisan Anda, lalu melakukan deployment:


```
sam deploy --signing-profiles HelloWorld=MySigningProfile MyLayer=MySigningProfile --  
s3-bucket test-bucket --template-file packaged.yaml --stack-name --region us-east-1 --  
capabilities CAPABILITY_IAM
```

Note

Agar berhasil menandatangani kode Anda dengan perintah `sam package` atau `sam deploy`, versioning untuk bucket Amazon S3 yang Anda gunakan harus diaktifkan dengan perintah ini. Jika Anda menggunakan Bucket Amazon S3 yang dibuat AWS SAM untuk Anda, versioning diaktifkan secara otomatis. Untuk informasi selengkapnya tentang pembuatan versi bucket Amazon S3 dan petunjuk untuk mengaktifkan pembuatan versi di bucket Amazon S3 yang Anda berikan, lihat [Menggunakan pembuatan versi di bucket Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon](#).

Menyediakan profil penandatanganan dengan `sam deploy --guided`

Saat Anda menjalankan perintah `sam deploy --guided` dengan aplikasi nirserver yang dikonfigurasi dengan penandatanganan kode, AWS SAM meminta Anda untuk memberikan profil penandatanganan yang akan digunakan untuk penandatanganan kode. Untuk informasi selengkapnya tentang `sam deploy --guided` prompt, lihat [sam deploy](#) di referensi AWS SAM CLI perintah.

Membangun aplikasi nirserver

Membangun aplikasi nirserver Anda melibatkan pengambilan file templat AWS SAM, kode aplikasi, dan file khusus bahasa yang berlaku serta dependensi, dan penempatan semua artefak bangunan dalam format dan lokasi yang tepat untuk langkah-langkah berikutnya dalam alur kerja Anda.

Misalnya, Anda mungkin ingin menguji aplikasi secara lokal, atau Anda mungkin ingin menerapkan aplikasi Anda menggunakan aplikasi. AWS SAM CLI Kedua aktivitas ini menggunakan artefak bangunan aplikasi Anda sebagai input.

Bagian ini menunjukkan cara menggunakan perintah [sam build](#) untuk membangun aplikasi nirserver menggunakan AWS SAM. Anda memiliki opsi untuk membangun semua fungsi dan lapisan aplikasi Anda, atau komponen individual dari aplikasi Anda, seperti fungsi atau lapisan tertentu.

Untuk pengantar menggunakan `sam build` perintah, lihat [Menggunakan sam build](#).

Topik

- [Aplikasi Pembangunan](#)
- [Membangun lapisan](#)
- [Membangun waktu aktif kustom](#)

Aplikasi Pembangunan

Untuk membangun aplikasi nirserver Anda, gunakan perintah [sam build](#). Perintah ini juga mengumpulkan artefak bangunan dependensi aplikasi Anda dan menempatkannya dalam format dan lokasi yang tepat untuk langkah berikutnya, seperti pengujian lokal, pengemasan, dan deployment.

Anda menentukan dependensi aplikasi dalam file manifes, seperti `requirements.txt` (Python) atau `package.json` (Node.js), atau dengan menggunakan properti `Layers` dari sumber daya fungsi. Properti `Layers` berisi daftar sumber daya [lapisan AWS Lambda](#) yang fungsi Lambda-nya bergantung pada lapisan tersebut.

Format artefak bangunan aplikasi Anda bergantung pada setiap fungsi properti `PackageType`. Opsi untuk properti ini adalah:

- **Zip** - Arsip file `.zip` yang berisi kode aplikasi Anda dan dependensinya. Jika Anda mengemas kode Anda sebagai arsip file `.zip`, Anda harus menentukan waktu aktif Lambda untuk fungsi Anda.

- **Image** – Citra kontainer termasuk sistem operasi dasar, waktu aktif, dan ekstensi, selain kode aplikasi Anda dan dependensinya.

Untuk informasi selengkapnya tentang tipe paket Lambda, lihat [Paket deployment Lambda](#) di Panduan Developer AWS Lambda.

Topik

- [Membangun arsip file .zip](#)
- [Membangun citra kontainer](#)
- [File variabel lingkungan kontainer](#)
- [Mempercepat waktu pembuatan dengan membangun proyek Anda di folder sumber](#)
- [Contoh-contoh](#)
- [Membangun fungsi di luar AWS SAM](#)
- [Membangun fungsi Node.js Lambda dengan esbuild](#)
- [Membangun fungsi.NET 7 Lambda dengan kompilasi AOT Asli](#)
- [Membangun fungsi Lambda Karat dengan Cargo Lambda](#)

Membangun arsip file .zip

Untuk membangun aplikasi nirserver Anda sebagai arsip file .zip, nyatakan `PackageType: Zip` untuk fungsi nirserver Anda.

AWS SAM membangun aplikasi Anda untuk [arsitektur](#) yang Anda tentukan. Jika Anda tidak menentukan arsitektur, AWS SAM gunakan secara `x86_64` default.

Jika fungsi Lambda Anda bergantung pada paket yang telah dikompilasi secara native, gunakan bendera `--use-container`. Bendera ini secara lokal mengompilasi fungsi Anda dalam kontainer Docker yang berperilaku seperti lingkungan Lambda, sehingga kontainer Docker berada dalam format yang tepat ketika Anda men-deploy kontainer Docker ke AWS Cloud.

Ketika Anda menggunakan opsi `--use-container`, secara default AWS SAM menarik citra kontainer dari [Publik Amazon ECR](#). Jika Anda ingin menarik gambar kontainer dari repositori lain, misalnya DockerHub, Anda dapat menggunakan `--build-image` opsi dan memberikan URI gambar kontainer alternatif. Berikut ini adalah dua contoh perintah untuk membangun aplikasi menggunakan gambar kontainer dari DockerHub repositori:

```
# Build a Node.js 12 application using a container image pulled from DockerHub
sam build --use-container --build-image amazon/aws-sam-cli-build-image-nodejs12.x

# Build a function resource using the Python 3.8 container image pulled from DockerHub
sam build --use-container --build-image Function1=amazon/aws-sam-cli-build-image-python3.8
```

Untuk daftar URI yang dapat Anda gunakan `--build-image`, lihat URI [Repositori citra](#) mana yang berisi DockerHub URI untuk sejumlah runtime yang didukung.

Untuk contoh tambahan membangun aplikasi arsip file .zip, nanti Anda dapat melihatnya di bagian Contoh dalam topik ini.

Membangun citra kontainer

Untuk membangun aplikasi nirserver Anda sebagai citra kontainer, nyatakan `PackageType: Image` untuk fungsi nirserver Anda. Anda juga harus menyatakan atribut sumber daya Metadata dengan entri berikut:

Dockerfile

Nama Dockerfile yang berkaitan dengan fungsi Lambda.

DockerContext

Lokasi Dockerfile.

DockerTag

(Opsional) tanda untuk diterapkan pada citra yang dibangun.

DockerBuildArgs

Bangun argumen untuk bangunan tersebut.

Berikut ini adalah contoh bagian atribut sumber daya Metadata:

```
Metadata:
  Dockerfile: Dockerfile
  DockerContext: ./hello_world
  DockerTag: v1
```

Untuk mengunduh aplikasi sampel yang dikonfigurasi dengan tipe paket Image, lihat [Tutorial: Men-deploy aplikasi Hello World](#) di Tutorial: Men-deploy aplikasi Hello World. Saat prompt mempertanyakan tipe paket yang ingin Anda instal, pilih Image.

Note

Jika Anda menentukan image dasar multi-arsitektur di Dockerfile Anda, AWS SAM buat image container Anda untuk arsitektur mesin host Anda. Untuk membangun arsitektur yang berbeda, tentukan gambar dasar yang menggunakan arsitektur target tertentu.

File variabel lingkungan kontainer

Untuk menyediakan file JSON yang berisi variabel lingkungan untuk kontainer bangunan, gunakan argumen `--container-env-var-file` dengan perintah `sam build`. Anda dapat memberikan variabel lingkungan tunggal yang berlaku untuk semua sumber daya nirserver, atau variabel lingkungan yang berbeda untuk setiap sumber daya.

format

Format untuk meneruskan variabel lingkungan ke kontainer bangunan tergantung pada berapa banyak variabel lingkungan yang Anda berikan untuk sumber daya Anda.

Untuk menyediakan variabel lingkungan tunggal pada semua sumber daya, tentukan objek `Parameters` seperti berikut:

```
{
  "Parameters": {
    "GITHUB_TOKEN": "TOKEN_GLOBAL"
  }
}
```

Untuk menyediakan variabel lingkungan yang berbeda pada setiap sumber daya, tentukan objek untuk setiap sumber daya seperti berikut:

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

```
"MyFunction2": {  
  "GIT_HUB_TOKEN": "TOKEN2"  
}
```

Simpan variabel lingkungan Anda sebagai file, misalnya, dengan nama `env.json`. Perintah berikut ini menggunakan file ini untuk meneruskan variabel lingkungan Anda ke kontainer bangunan:

```
sam build --use-container --container-env-var-file env.json
```

Precedence

- Variabel lingkungan yang Anda berikan untuk sumber daya tertentu lebih diutamakan daripada variabel lingkungan tunggal untuk semua sumber daya.
- Variabel lingkungan yang Anda berikan pada baris perintah lebih diutamakan daripada variabel lingkungan dalam sebuah file.

Mempercepat waktu pembuatan dengan membangun proyek Anda di folder sumber

Untuk runtime dan metode build yang didukung, Anda dapat menggunakan `--build-in-source` opsi untuk membangun proyek secara langsung di folder sumber. Secara default, AWS SAM CLI build dalam direktori sementara, yang melibatkan penyalinan kode sumber dan file proyek. Dengan `--build-in-source`, AWS SAM CLI build langsung di folder sumber Anda, yang mempercepat proses pembuatan dengan menghapus kebutuhan untuk menyalin file ke direktori sementara.

Untuk daftar runtime dan metode build yang didukung, lihat [--build-in-source](#).

Contoh-contoh

Contoh 1: Arsip file .zip

Perintah `sam build` berikut membangun arsip file .zip:

```
# Build all functions and layers, and their dependencies  
sam build
```

```
# Run the build process inside a Docker container that functions like a Lambda
environment
sam build --use-container

# Build a Node.js 12 application using a container image pulled from DockerHub
sam build --use-container --build-image amazon/aws-sam-cli-build-image-nodejs12.x

# Build a function resource using the Python 3.8 container image pulled from DockerHub
sam build --use-container --build-image Function1=amazon/aws-sam-cli-build-image-
python3.8

# Build and run your functions locally
sam build && sam local invoke

# For more options
sam build --help
```

Contoh 2: Citra kontainer

Templat AWS SAM berikut membangun sebagai citra kontainer:

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      PackageType: Image
      ImageConfig:
        Command: ["app.lambda_handler"]
    Metadata:
      Dockerfile: Dockerfile
      DockerContext: ./hello_world
      DockerTag: v1
```

Berikut ini adalah contoh Dockerfile:

```
FROM public.ecr.aws/lambda/python:3.8

COPY app.py requirements.txt ./

RUN python3.8 -m pip install -r requirements.txt

# Overwrite the command by providing a different command directly in the template.
```

```
CMD ["app.lambda_handler"]
```

Contoh 3: npm ci

Untuk aplikasi Node.js, Anda dapat menggunakan `npm ci` alih-alih `npm install` menginstal dependensi. Untuk menggunakan `npm ci`, tentukan `UseNpmCi: True` `BuildProperties` di bawah atribut `Metadata` sumber daya fungsi Lambda Anda. Untuk menggunakannya `npm ci`, aplikasi Anda harus memiliki `npm-shrinkwrap.json` file `package-lock.json` atau yang ada di fungsi `CodeUri` untuk Lambda Anda.

Contoh berikut digunakan `npm ci` untuk menginstal dependensi saat Anda menjalankan: `sam build`

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello-world/
      Handler: app.handler
      Runtime: nodejs14.x
      Architectures:
        - x86_64
      Events:
        HelloWorld:
          Type: Api
          Properties:
            Path: /hello
            Method: get
    Metadata:
      BuildProperties:
        UseNpmCi: True
```

Membangun fungsi di luar AWS SAM

Secara default, saat Anda menjalankan `sam build`, AWS SAM membangun semua sumber daya fungsi Anda. Pilihan lain termasuk:

- Membangun semua sumber daya fungsi di luar AWS SAM — Jika Anda membangun semua sumber daya fungsi Anda secara manual atau melalui alat lain, tidak `sam build` diperlukan. Anda dapat melewati `sam build` dan melanjutkan ke langkah berikutnya dalam proses Anda, seperti melakukan pengujian lokal atau menerapkan aplikasi Anda.

- Bangun beberapa sumber daya fungsi di luar AWS SAM — Jika Anda AWS SAM ingin membangun beberapa sumber daya fungsi Anda sambil memiliki sumber daya fungsi lain yang dibangun di luar AWS SAM, Anda dapat menentukan ini di AWS SAM template Anda.

Membangun beberapa sumber daya fungsi di luar AWS SAM

Untuk AWS SAM melewati fungsi saat menggunakan `sam build`, konfigurasi yang berikut ini di AWS SAM template Anda:

1. Tambahkan properti `SkipBuild: True` metadata ke fungsi Anda.
2. Tentukan jalur ke sumber daya fungsi bawaan Anda.

Berikut adalah contoh, dengan `TestFunction` dikonfigurasi untuk dilewati. Sumber daya yang dibangun terletak di `built-resources/TestFunction.zip`.

```
TestFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: built-resources/TestFunction.zip
    Handler: TimeHandler::handleRequest
    Runtime: java11
  Metadata:
    SkipBuild: True
```

Sekarang, ketika Anda menjalankan `sam build`, AWS SAM akan melakukan hal berikut:

1. AWS SAM akan melewati fungsi yang dikonfigurasi dengan `SkipBuild: True`.
2. AWS SAM akan membangun semua sumber daya fungsi lainnya dan menyimpannya di direktori `.aws-sam build`.
3. Untuk fungsi yang dilewati, templatnya di direktori `.aws-sam build` akan diperbarui secara otomatis untuk mereferensikan jalur yang ditentukan ke sumber daya fungsi bawaan Anda.

Berikut adalah contoh template cache untuk `TestFunction` di direktori `.aws-sam build`:

```
TestFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: ../../built-resources/TestFunction.zip
    Handler: TimeHandler::handleRequest
```

```
Runtime: java11
Metadata:
  SkipBuild: True
```

Membangun fungsi Node.js Lambda dengan esbuild

Untuk membangun dan mengemas AWS Lambda fungsi Node.js, Anda dapat menggunakan AWS SAM CLI JavaScript bundler dengan esbuild. Bundler esbuild mendukung fungsi Lambda yang Anda tulis. TypeScript

Untuk membangun fungsi Lambda Node.js dengan esbuild, tambahkan Metadata objek ke `AWS::Serverless::Function` sumber daya Anda dan tentukan esbuild untuk `BuildMethod`. Saat Anda menjalankan `sam build` perintah, AWS SAM gunakan esbuild untuk menggabungkan kode fungsi Lambda Anda.

Properti metadata

MetadataObjek mendukung properti berikut untuk esbuild.

BuildMethod

Menentukan bundler untuk aplikasi Anda. Satu-satunya nilai yang di-support adalah `esbuild`.

BuildProperties

Menentukan properti build untuk kode fungsi Lambda Anda.

`BuildProperties`Objek mendukung properti berikut untuk esbuild. Semua properti adalah opsional. Secara default, AWS SAM gunakan penanganan fungsi Lambda Anda untuk titik masuk.

EntryPoints

Menentukan titik masuk untuk aplikasi Anda.

Eksternal

Menentukan daftar paket untuk menghilangkan dari build. Untuk informasi lebih lanjut, lihat [Eksternal](#) di esbuildsitus web.

format

Menentukan format output dari JavaScript file yang dihasilkan dalam aplikasi Anda. Untuk informasi selengkapnya, lihat [Format](#) di situs web esbuild.

Loader

Menentukan daftar konfigurasi untuk memuat data untuk jenis file tertentu.

MainFields

Menentukan `package.json` bidang untuk mencoba mengimpor ketika menyelesaikan paket. Nilai default-nya adalah `main,module`.

Mengecilkan

Menentukan apakah untuk mengecilkan kode output dibundel. Nilai default-nya adalah `true`.

OutExtension

Kustomisasi ekstensi file dari file yang dihasilkan `esBUILD`. Untuk informasi lebih lanjut, lihat [Ekstensi keluar](#) di situs web `esbuild`.

Peta sumber

Menentukan apakah bundler menghasilkan file peta sumber. Nilai default-nya adalah `false`.

Ketika diatur ke `true`, `NODE_OPTIONS: --enable-source-maps` ditambahkan ke variabel lingkungan fungsi Lambda, dan peta sumber dihasilkan dan disertakan dalam fungsi.

Atau, ketika `NODE_OPTIONS: --enable-source-maps` disertakan dalam variabel lingkungan fungsi, secara otomatis `Sourcemap` diatur ke `true`.

Saat berkonflik, lebih `Sourcemap: false` diutamakan. `NODE_OPTIONS: --enable-source-maps`

Note

Secara default, Lambda mengenkripsi semua variabel lingkungan saat istirahat dengan `(.)`. AWS Key Management Service (AWS KMS) Saat menggunakan peta sumber, agar penerapan berhasil, peran eksekusi fungsi Anda harus memiliki izin untuk melakukan `kms:Encrypt` tindakan.

SourcesContent

Menentukan apakah akan menyertakan kode sumber Anda dalam file peta sumber Anda. Konfigurasi properti ini saat `Sourcemap` disetel ke `'true'`.

- Tentukan `SourcesContent: 'true'` untuk menyertakan semua kode sumber.
- Tentukan `SourcesContent: 'false'` untuk mengecualikan semua kode sumber. Ini menghasilkan ukuran file peta sumber yang lebih kecil, yang berguna dalam produksi dengan mengurangi waktu start-up. Namun, kode sumber tidak akan tersedia di debugger.

Nilai default-nya adalah `SourcesContent: true`.

Untuk informasi selengkapnya, lihat [Konten sumber](#) di situs web esbuild.

Target

Menentukan versi ECMAScript target. Nilai default-nya adalah `es2020`.

TypeScript Contoh fungsi Lambda

Contoh cuplikan AWS SAM template berikut menggunakan esbuild untuk membuat fungsi Lambda Node.js dari kode di TypeScript `hello-world/app.ts`

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello-world/
      Handler: app.handler
      Runtime: nodejs14.x
      Architectures:
        - x86_64
      Events:
        HelloWorld:
          Type: Api
          Properties:
            Path: /hello
            Method: get
      Environment:
        Variables:
          NODE_OPTIONS: --enable-source-maps
    Metadata:
      BuildMethod: esbuild
      BuildProperties:
        Format: esm
        Minify: false
        OutExtension:
```

```
- .js=.mjs
Target: "es2020"
SourceMap: true
EntryPoints:
- app.ts
External:
- "<package-to-exclude>"
```

Membangun fungsi.NET 7 Lambda dengan kompilasi AOT Asli

Buat dan kemas AWS Lambda fungsi.NET 7 Anda dengan AWS Serverless Application Model (AWS SAM), menggunakan kompilasi Native Ahead-of-Time (AOT) untuk meningkatkan waktu mulai dingin. AWS Lambda

Topik

- [Ikhtisar AOT asli .NET 7](#)
- [Menggunakan AWS SAM dengan fungsi.NET 7 Lambda](#)
- [Instal prasyarat](#)
- [Tentukan fungsi.NET 7 Lambda di template Anda AWS SAM](#)
- [Membangun aplikasi Anda dengan AWS SAMCLI](#)
- [Pelajari selengkapnya](#)

Ikhtisar AOT asli .NET 7

Secara historis, fungsi.NET Lambda memiliki waktu mulai dingin yang memengaruhi pengalaman pengguna, latensi sistem, dan biaya penggunaan aplikasi tanpa server Anda. Dengan .NET 7, Microsoft menambahkan dukungan untuk kompilasi Native AOT, yang mempercepat kinerja dengan mengkompilasi kode.NET langsung ke mesin dan kode asli sistem operasi, menghilangkan kompilasi Just-in-Time (JIT) saat runtime. Dengan kompilasi.NET 7 Native AOT, Anda dapat meningkatkan waktu mulai dingin fungsi Lambda Anda. Untuk mempelajari lebih lanjut tentang Native AOT untuk.NET 7, lihat [Menggunakan AOT Asli](#) di repositori GitHub Dotnet.

Menggunakan AWS SAM dengan fungsi.NET 7 Lambda

Lakukan hal berikut untuk mengonfigurasi fungsi.NET 7 Lambda Anda dengan AWS Serverless Application Model (AWS SAM):

- Instal prasyarat pada mesin pengembangan Anda.

- Tentukan fungsi.NET 7 Lambda di template AndaAWS SAM.
- Bangun aplikasi Anda dengan AWS SAMCLI.

Instal prasyarat

Berikut ini adalah prasyarat yang diperlukan:

- The AWS SAM CLI
- CLI INTI .NET
- Alat Global Inti Amazon.Lambda.Tools .NET
- Docker

Instal AWS SAMCLI

1. Untuk memeriksa apakah Anda sudah AWS SAM CLI menginstal, jalankan yang berikut ini:

```
sam --version
```

2. Untuk menginstal AWS SAMCLI, lihat[Instalasi AWS SAMCLI](#).
3. Untuk memutakhirkan versi terinstal AWS SAMCLI, lihat[Memutakhirkan AWS SAMCLI](#).

Instal CLI CLI.NET

1. Untuk mengunduh dan menginstal .NET Core CLI, lihat [Mengunduh.NET](#) dari situs web Microsoft.
2. Untuk informasi selengkapnya tentang.NET Core CLI, lihat [.NET Core CLI](#) di Panduan Pengembang. AWS Lambda

Instal Amazon.Lambda.Tools .NET Core Global Tool

1. Jalankan perintah berikut:

```
dotnet tool install -g Amazon.Lambda.Tools
```

2. Jika Anda sudah menginstal alat tersebut, Anda dapat memastikan alat tersebut menggunakan versi terbaru dengan perintah berikut.

```
dotnet tool update -g Amazon.Lambda.Tools
```

3. Untuk informasi selengkapnya tentang Amazon.Lambda.Tools .NET Core Global Tool, lihat [Extensions AWSfor .NET CLI](#) repositori.NET di. GitHub

Instal Docker

- Membangun dengan Native AOT, Docker harus diinstal. Untuk instruksi instalasi, lihat [Menginstal Docker untuk digunakan dengan AWS SAMCLI](#).

Tentukan fungsi.NET 7 Lambda di template Anda AWS SAM

Untuk menentukan fungsi.NET7 Lambda di AWS SAM template Anda, lakukan hal berikut:

- Tentukan runtime kustom dengan menyetel Runtime keprovided.al2. Kami mendefinisikan runtime khusus karena opsi runtime.NET 7 tidak tersedia.
- Tambahkan Metadata objek ke `AWS::Serverless::Function` sumber daya Anda dan tentukan `dotnet7` untuk `BuildMethod`.

```
Resources:
HelloWorldFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: function-source-folder
    Handler: app.handler
    Runtime: provided.al2
    Architectures:
      - x86_64
    Events:
      HelloWorld:
        Type: Api
        Properties:
          Path: /hello
          Method: get
  Metadata:
    BuildMethod: dotnet7
```

Note

Untuk `Architectures` properti, `x86_64` adalah satu-satunya opsi yang didukung karena `.NET 7` tidak dapat berjalan di `arm64`. Untuk informasi selengkapnya, lihat [GitHub masalah Reconsider .NET 7 build environment](#).

Membangun aplikasi Anda dengan AWS SAM CLI

Dari direktori root proyek Anda, jalankan `sam build` untuk mulai membangun aplikasi Anda. Jika `PublishAot` properti telah ditentukan dalam file proyek `.NET 7` Anda, AWS SAM CLI akan dibangun dengan kompilasi AOT Asli. Untuk mempelajari lebih lanjut tentang `PublishAot` properti, lihat [Penerapan AOT Asli](#) di dokumentasi `.NET` Microsoft.

Untuk membangun fungsi Anda, AWS SAM CLI memanggil CLI `.NET` yang menggunakan `Amazon.Lambda.Tools .NET Core Global Tool`.

Note

Saat membangun, jika `.sln` file ada di direktori yang sama atau induk proyek Anda, direktori yang berisi `.sln` file akan dipasang ke wadah. Jika `.sln` file tidak ditemukan, hanya folder proyek yang dipasang. Oleh karena itu, jika Anda sedang membangun aplikasi multi-proyek, pastikan `.sln` file tersebut berada di properti.

Pelajari selengkapnya

Untuk informasi selengkapnya tentang membangun fungsi `.NET 7` Lambda, lihat [Membangun aplikasi tanpa server di Lambda](#) menggunakan `.NET 7`.

Untuk referensi `sam build` perintah, lihat [sam build](#).

Membangun fungsi Lambda Karat dengan Cargo Lambda

Fitur ini dalam rilis pratinjau untuk AWS SAM dan dapat berubah.

Gunakan AWS Serverless Application Model Command Line Interface (AWS SAM CLI) dengan AWS Lambda fungsi Rust Anda.

Topik

- [Prasyarat](#)
- [Mengkonfigurasi AWS SAM untuk digunakan dengan fungsi Rust Lambda](#)
- [Contoh-contoh](#)

Prasyarat

Rustbahasa

Untuk menginstal Rust, lihat [Menginstal Rust](#) di situs web Rust bahasa.

Cargo Lambda

AWS SAM CLI memerlukan instalasi [Cargo Lambda](#), subperintah untuk Cargo. Untuk petunjuk penginstalan, lihat [Instalasi](#) dalam Cargo Lambda dokumentasi.

Docker

Membangun dan menguji fungsi Rust Lambda membutuhkan Docker. Untuk instruksi instalasi, lihat [Menginstal Docker](#).

Ikut serta dalam fitur AWS SAM CLI beta

Karena fitur ini dalam pratinjau, Anda harus memilih untuk menggunakan salah satu metode berikut:

1. Gunakan variabel lingkungan: `SAM_CLI_BETA_RUST_CARGO_LAMBDA=1`.
2. Tambahkan hal berikut ke file `samconfig.toml` Anda:

```
[default.build.parameters]
beta_features = true
[default.sync.parameters]
beta_features = true
```

3. Gunakan `--beta-features` opsi saat menggunakan AWS SAM CLI perintah yang didukung. Sebagai contoh:

```
$ sam build --beta-features
```

4. Pilih opsi `y` saat AWS SAM CLI meminta Anda untuk ikut serta. Berikut ini adalah contohnya:

```
$ sam build
```

```
Starting Build use cache
Build method "rust-cargolambda" is a beta feature.
Please confirm if you would like to proceed
You can also enable this beta feature with "sam build --beta-features". [y/N]: y
```

Mengkonfigurasi AWS SAM untuk digunakan dengan fungsi Rust Lambda

Langkah 1: Konfigurasi AWS SAM template Anda

Konfigurasi AWS SAM template Anda dengan yang berikut:

- `Biner` — Opsional. Tentukan kapan template Anda berisi beberapa fungsi Rust Lambda.
- `BuildMethod` – `rust-cargolambda`.
- `CodeUri`— jalur ke `Cargo.toml` file Anda.
- `Pawang` —`bootstrap`.
- `Runtime` —`provided.al2`.

Untuk mempelajari selengkapnya tentang runtime kustom, lihat [AWS Lambda Waktu proses kustom](#) di Panduan AWS Lambda Pengembang.

Berikut adalah contoh AWS SAM template yang dikonfigurasi:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Metadata:
      BuildMethod: rust-cargolambda
      BuildProperties: function_a
    Properties:
      CodeUri: ./rust_app
      Handler: bootstrap
      Runtime: provided.al2
  ...
```

Langkah 2: Gunakan AWS SAMCLI dengan fungsi Rust Lambda Anda

Gunakan AWS SAM CLI perintah apa pun dengan AWS SAM template Anda. Untuk informasi selengkapnya, lihat [Menggunakan AWS SAMCLI](#).

Contoh-contoh

Contoh Hello World

Dalam contoh ini, kita membangun contoh aplikasi Hello World menggunakan Rust sebagai runtime kita.

Pertama, kami menginisialisasi aplikasi tanpa server baru menggunakan `sam init` Selama aliran interaktif, kami memilih aplikasi Hello World dan memilih runtime Rust.

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

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

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

Which runtime would you like to use?
  1 - aot.dotnet7 (provided.al2)
  2 - dotnet6
  3 - dotnet5.0
  ...
 18 - python3.7
 19 - python3.10
 20 - ruby2.7
 21 - rust (provided.al2)
Runtime: 21

Based on your selections, the only Package type available is Zip.
```

We will proceed to selecting the Package type as Zip.

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

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

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

Project name [sam-app]: *hello-rust*

Generating application:

Name: hello-rust
Runtime: rust (provided.al2)
Architectures: x86_64
Dependency Manager: cargo
Application Template: hello-world
Output Directory: .
Configuration file: hello-rust/samconfig.toml

Next steps can be found in the README file at hello-rust/README.md

Commands you can use next

=====

[*] Create pipeline: `cd hello-rust && sam pipeline init --bootstrap`
[*] Validate SAM template: `cd hello-rust && sam validate`
[*] Test Function in the Cloud: `cd hello-rust && sam sync --stack-name {stack-name} --watch`

Berikut ini adalah struktur aplikasi Hello World kami:

```
hello-rust
### README.md
### events
#   ### event.json
### rust_app
#   ### Cargo.toml
```

```
#   ### src
#       ### main.rs
### samconfig.toml
### template.yaml
```

Dalam AWS SAM template kami, Rust fungsi kami didefinisikan sebagai berikut:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Metadata:
      BuildMethod: rust-cargolambda
    Properties:
      CodeUri: ./rust_app
      Handler: bootstrap
      Runtime: provided.al2
      Architectures:
        - x86_64
    Events:
      HelloWorld:
        Type: Api
        Path: /hello
        Method: get
```

Selanjutnya, kami menjalankan `sam build` untuk membangun aplikasi kami dan mempersiapkan penerapan. AWS SAMCLI membuat `.aws-sam` direktori dan mengatur artefak build kami di sana. Fungsi kami dibangun menggunakan Cargo Lambda dan disimpan sebagai biner yang dapat dieksekusi di `.aws-sam/build/HelloWorldFunction/bootstrap`

```
hello-rust$ sam build
Starting Build use cache
Build method "rust-cargolambda" is a beta feature.
Please confirm if you would like to proceed
You can also enable this beta feature with "sam build --beta-features". [y/N]: y

Experimental features are enabled for this session.
Visit the docs page to learn more about the AWS Beta terms https://aws.amazon.com/service-terms/.
```

```

Cache is invalid, running build and copying resources for following functions
(HelloWorldFunction)
Building codeuri: /Users/.../hello-rust/rust_app runtime: provided.al2 metadata:
{'BuildMethod': 'rust-cargolambda'} architecture: x86_64 functions: HelloWorldFunction
Running RustCargoLambdaBuilder:CargoLambdaBuild
Running RustCargoLambdaBuilder:RustCopyAndRename

Build Succeeded

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

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

```

Selanjutnya, kami menerapkan aplikasi kami menggunakan `sam deploy --guided`.

```

hello-rust$ sam deploy --guided

Configuring SAM deploy
=====

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

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

```

```
Save arguments to configuration file [Y/n]: ENTER
```

```
SAM configuration file [samconfig.toml]: ENTER
```

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

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

```
...
```

```
Uploading to hello-rust/56ba6585d80577dd82a7eaaee5945c0b 817973 / 817973
(100.00%)
```

```
Deploying with following values
```

```
=====
```

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

```
Initiating deployment
```

```
=====
```

```
Uploading to hello-rust/a4fc54cb6ab75dd0129e4cdb564b5e89.template 1239 / 1239
(100.00%)
```

```
Waiting for changeset to be created..
```

```
CloudFormation stack changeset
```

```
-----
```

Operation	LogicalResourceId	ResourceType	
Replacement			
+ Add	HelloWorldFunctionHelloW	AWS::Lambda::Permission	N/A
	orldPermissionProd		

```
-----
```

```
...
```

```
Changeset created successfully. arn:aws:cloudformation:us-
west-2:012345678910:changeSet/samcli-deploy1681427201/
f0ef1563-5ab6-4b07-9361-864ca3de6ad6
```

```
Previewing CloudFormation changeset before deployment
=====
```

```
Deploy this changeset? [y/N]: y
```

```
2023-04-13 13:07:17 - Waiting for stack create/update to complete
```

```
CloudFormation events from stack operations (refresh every 5.0 seconds)
```

```
-----
ResourceStatus      ResourceType      LogicalResourceId
ResourceStatusReason
-----
CREATE_IN_PROGRESS  AWS::IAM::Role   HelloWorldFunctionRole -
CREATE_IN_PROGRESS  AWS::IAM::Role   HelloWorldFunctionRole
Resource creation
...
-----
```

```
CloudFormation outputs from deployed stack
```

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

```
-----
Key                HelloWorldFunctionIamRole
Description        Implicit IAM Role created for Hello World function
Value              arn:aws:iam::012345678910:role/hello-rust-
HelloWorldFunctionRole-10II2P13AUDUY
Key                HelloWorldApi
Description        API Gateway endpoint URL for Prod stage for Hello World function
Value              https://ggdxec9le9.execute-api.us-west-2.amazonaws.com/Prod/hello/
Key                HelloWorldFunction
```



```

Description      Hello World Lambda Function ARN

Value           arn:aws:lambda:us-west-2:012345678910:function:hello-rust-
HelloWorldFunction-
yk4HzGzYeZBj

```

Successfully created/updated stack - hello-rust in us-west-2

Untuk menguji, kita dapat memanggil fungsi Lambda kita menggunakan titik akhir API.

```

$ curl https://ggdxec91e9.execute-api.us-west-2.amazonaws.com/Prod/hello/
Hello World!%

```

Untuk menguji fungsi kami secara lokal, pertama-tama kami memastikan Architectures properti fungsi kami cocok dengan mesin lokal kami.

```

...
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function # More info about Function Resource:
    https://github.com/awslabs/serverless-application-model/blob/master/
versions/2016-10-31.md#awsserverlessfunction
    Metadata:
      BuildMethod: rust-cargolambda # More info about Cargo Lambda: https://github.com/
cargo-lambda/cargo-lambda
    Properties:
      CodeUri: ./rust_app # Points to dir of Cargo.toml
      Handler: bootstrap # Do not change, as this is the default executable name
produced by Cargo Lambda
      Runtime: provided.al2
      Architectures:
        - arm64
...

```

Karena kami memodifikasi arsitektur kami dari x86_64 ke arm64 dalam contoh ini, kami menjalankan `sam build` untuk memperbarui artefak build kami. Kami kemudian menjalankan `sam local invoke` untuk memanggil fungsi kami secara lokal.

```

hello-rust$ sam local invoke

```

```

Invoking bootstrap (provided.al2)
Local image was not found.
Removing rapid images for repo public.ecr.aws/sam/emulation-provided.al2
Building
  image.....
Using local image: public.ecr.aws/lambda/provided:al2-rapid-arm64.

Mounting /Users/.../hello-rust/.aws-sam/build/HelloWorldFunction as /var/
task:ro,delegated, inside runtime container
START RequestId: fbc55e6e-0068-45f9-9f01-8e2276597fc6 Version: $LATEST
{"statusCode":200,"body":"Hello World!"}END RequestId:
fbc55e6e-0068-45f9-9f01-8e2276597fc6
REPORT RequestId: fbc55e6e-0068-45f9-9f01-8e2276597fc6  Init Duration: 0.68 ms
Duration: 130.63 ms    Billed Duration: 131 ms    Memory Size: 128 MB    Max Memory
Used: 128 MB

```

Proyek fungsi Lambda Tunggal

Berikut adalah contoh aplikasi tanpa server yang berisi satu fungsi Rust Lambda.

Struktur direktori proyek:

```

.
### Cargo.lock
### Cargo.toml
### src
#   ### main.rs
### template.yaml

```

AWS SAMtemplat:

```

AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Metadata:
      BuildMethod: rust-cargolambda
    Properties:
      CodeUri: ./
      Handler: bootstrap
      Runtime: provided.al2

```

```
...
```

Beberapa proyek fungsi Lambda

Berikut adalah contoh aplikasi tanpa server yang berisi beberapa fungsi Rust Lambda.

Struktur direktori proyek:

```
.  
### Cargo.lock  
### Cargo.toml  
### src  
#   ### function_a.rs  
#   ### function_b.rs  
### template.yaml
```

AWS SAMtemplat:

```
AWSTemplateFormatVersion: '2010-09-09'  
Transform: AWS::Serverless-2016-10-31  
...  
Resources:  
  FunctionA:  
    Type: AWS::Serverless::Function  
    Metadata:  
      BuildMethod: rust-cargolambda  
      BuildProperties:  
        Binary: function_a  
    Properties:  
      CodeUri: ./  
      Handler: bootstrap  
      Runtime: provided.al2  
  FunctionB:  
    Type: AWS::Serverless::Function  
    Metadata:  
      BuildMethod: rust-cargolambda  
      BuildProperties:  
        Binary: function_b  
    Properties:  
      CodeUri: ./  
      Handler: bootstrap  
      Runtime: provided.al2
```

Cargo.tomlberkas:

```
[package]
name = "test-handler"
version = "0.1.0"
edition = "2021"

[dependencies]
lambda_runtime = "0.6.0"
serde = "1.0.136"
tokio = { version = "1", features = ["macros"] }
tracing = { version = "0.1", features = ["log"] }
tracing-subscriber = { version = "0.3", default-features = false, features = ["fmt"] }

[[bin]]
name = "function_a"
path = "src/function_a.rs"

[[bin]]
name = "function_b"
path = "src/function_b.rs"
```

Membangun lapisan

Anda dapat menggunakan AWS SAM untuk membangun lapisan kustom. Untuk informasi tentang lapisan, lihat [Lapisan Lambda AWS](#) di Panduan Developer AWS Lambda.

Untuk membangun sebuah lapisan kustom, nyatakan dalam file templat AWS Serverless Application Model (AWS SAM) dan sertakan bagian atribut sumber daya Metadata dengan entri BuildMethod. Nilai yang valid untuk BuildMethod adalah pengidentifikasi pada [waktu aktif AWS Lambda](#), atau makefile. Sertakan BuildArchitecture entri untuk menentukan arsitektur set instruksi yang didukung lapisan Anda. Nilai yang valid untuk BuildArchitecture adalah arsitektur [set instruksi Lambda](#).

Jika Anda menentukan makefile, sediakan makefile kustom, tempat Anda menyatakan target bangunan formulir build-*layer-logical-id* yang berisi perintah pembangunan untuk lapisan Anda. Makefile Anda bertanggung jawab untuk menyusun lapisan jika diperlukan, dan menyalin artefak bangunan ke lokasi yang tepat yang diperlukan untuk langkah-langkah berikutnya dalam alur kerja Anda. Lokasi makefile ditentukan oleh properti ContentUri dari sumber daya lapisan dan harus diberi nama Makefile.

Note

Saat Anda membuat lapisan kustom, AWS Lambda bergantung pada variabel lingkungan untuk menemukan kode lapisan Anda. Waktu aktif Lambda termasuk jalur di direktori /opt tempat kode lapisan Anda disalin. Struktur folder artefak bangunan Anda harus sesuai dengan struktur folder yang diharapkan waktu aktif sehingga kode lapisan kustom Anda dapat ditemukan.

Contohnya, untuk Python, Anda dapat menempatkan kode di subdirektori `python/`. Untuk NodeJS, Anda dapat menempatkan kode Anda di subdirektori `nodejs/node_modules/`. Untuk informasi selengkapnya, lihat [Menyertakan dependensi pustaka dalam lapisan](#) di Panduan Developer AWS Lambda.

Berikut ini adalah contoh bagian atribut sumber daya Metadata.

Metadata:

```
BuildMethod: python3.8
BuildArchitecture: arm64
```

Note

Jika Anda tidak menyertakan bagian atribut sumber daya Metadata, AWS SAM tidak membangun lapisan. Sebaliknya, artefak bangunan akan disalin dari lokasi yang ditentukan dalam properti `CodeUri` sumber daya lapisan. Untuk informasi selengkapnya, lihat properti [ContentUri](#) tipe sumber daya `AWS::Serverless::LayerVersion`.

Ketika Anda menyertakan bagian atribut sumber daya Metadata, Anda dapat menggunakan perintah `sam build` untuk membangun lapisan, baik sebagai objek independen, atau sebagai dependensi fungsi AWS Lambda.

- Sebagai objek independen. Anda mungkin hanya ingin membangun objek lapisan, misalnya ketika Anda sedang menguji perubahan kode secara lokal pada lapisan dan tidak perlu membangun seluruh aplikasi Anda. Untuk membangun lapisan secara independen, tentukan sumber daya lapisan dengan perintah `sam build layer-logical-id`.
- Sebagai ketergantungan fungsi Lambda. Bila Anda menyertakan ID logis lapisan di properti `Layers` dari fungsi Lambda di file templat AWS SAM yang sama, lapisan merupakan dependensi

dari fungsi Lambda tersebut. Ketika lapisan tersebut juga menyertakan bagian atribut sumber daya Metadata dengan entri `BuildMethod`, Anda membangun lapisan baik dengan membangun seluruh aplikasi menggunakan perintah `sam build` atau dengan menentukan sumber daya fungsi menggunakan perintah `sam build function-logical-id`.

Contoh-contoh

Contoh template 1: Bangun layer terhadap lingkungan runtime Python 3.9

Contoh AWS SAM template berikut membangun lapisan terhadap lingkungan runtime Python 3.9.

```
Resources:
  MyLayer:
    Type: AWS::Serverless::LayerVersion
    Properties:
      ContentUri: my_layer
      CompatibleRuntimes:
        - python3.9
    Metadata:
      BuildMethod: python3.9 # Required to have AWS SAM build this layer
```

Contoh templat 2: Bangun lapisan menggunakan makefile kustom

Contoh templat AWS SAM berikut menggunakan `makefile` kustom untuk membangun lapisan.

```
Resources:
  MyLayer:
    Type: AWS::Serverless::LayerVersion
    Properties:
      ContentUri: my_layer
      CompatibleRuntimes:
        - python3.8
    Metadata:
      BuildMethod: makefile
```

Hal `makefile` berikut berisi target build dan perintah yang akan dieksekusi. Perhatikan bahwa properti `ContentUri` diatur ke `my_layer`, sehingga `makefile` harus terletak di root subdirektori `my_layer`, dan nama file harus `Makefile`. Perhatikan juga bahwa artefak bangunan disalin ke subdirektori `python/` sehingga AWS Lambda akan dapat menemukan kode lapisan.

```
build-MyLayer:
  mkdir -p "${ARTIFACTS_DIR}/python"
  cp *.py "${ARTIFACTS_DIR}/python"
  python -m pip install -r requirements.txt -t "${ARTIFACTS_DIR}/python"
```

Contoh perintah pembangunan sam

Perintah `sam build` berikut ini membangun lapisan yang mencakup bagian atribut sumber daya Metadata.

```
# 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
```

Membangun waktu aktif kustom

Anda dapat menggunakan perintah [sam build](#) untuk membangun waktu aktif kustom yang diperlukan untuk fungsi Lambda Anda. Anda menyatakan fungsi Lambda untuk menggunakan waktu aktif kustom dengan menentukan `Runtime: provided` pada fungsi tersebut.

Untuk membangun waktu aktif kustom, nyatakan atribut sumber daya Metadata dengan entri `BuildMethod: makefile`. Anda menyediakan makefile kustom, tempat Anda menyatakan target bangunan formulir `build-function-logical-id` yang mencakup perintah pembangunan untuk waktu aktif Anda. Makefile Anda bertanggung jawab untuk menyusun waktu aktif kustom jika diperlukan, dan menyalin artefak bangunan ke lokasi yang tepat yang diperlukan untuk langkah-langkah berikutnya dalam alur kerja Anda. Lokasi makefile ditentukan oleh properti `CodeUri` dari sumber daya fungsi dan harus bernama `Makefile`.

Contoh-contoh

Contoh 1: Waktu aktif kustom untuk fungsi yang ditulis dalam Rust

Note

Kami merekomendasikan membangun fungsi Lambda dengan Cargo Lambda Untuk mempelajari selengkapnya, lihat [Membangun fungsi Lambda Karat dengan Cargo Lambda](#).

Templat AWS SAM Berikut ini menyatakan fungsi yang menggunakan waktu aktif kustom untuk fungsi Lambda yang ditulis dalam Rust, dan menginstruksikan `sam build` agar mengeksekusi perintah untuk target bangunan `build-HelloRustFunction`.

```
Resources:
  HelloRustFunction:
    Type: AWS::Serverless::Function
    Properties:
      FunctionName: HelloRust
      Handler: bootstrap.is.real.handler
      Runtime: provided
      MemorySize: 512
      CodeUri: .
    Metadata:
      BuildMethod: makefile
```

Makefile berikut berisi target bangunan dan perintah yang akan dieksekusi. Perhatikan bahwa properti `CodeUri` diatur ke `.`, maka makefile harus terletak di direktori root proyek (yaitu, direktori yang sama dengan file templat AWS SAM aplikasi). Nama file tersebut harus `Makefile`.

```
build-HelloRustFunction:
  cargo build --release --target x86_64-unknown-linux-musl
  cp ./target/x86_64-unknown-linux-musl/release/bootstrap $(ARTIFACTS_DIR)
```

Untuk informasi selengkapnya tentang pengaturan lingkungan pengembangan Anda agar dapat mengeksekusi perintah `cargo build` di `makefile` sebelumnya, lihat postingan blog [Waktu aktif Rust untuk AWS Lambda](#).

Contoh 2: Pembangun Makefile untuk Python3.7 (alternatif untuk menggunakan pembangun yang dipaketkan bersama)

Anda mungkin ingin menggunakan pustaka atau modul yang tidak disertakan dalam pembangun yang dipaketkan bersama. Contoh ini menunjukkan templat AWS SAM untuk waktu aktif Python3.7 dengan pembangun makefile.

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello_world/
      Handler: app.lambda_handler
      Runtime: python3.7
    Metadata:
      BuildMethod: makefile
```

Makefile berikut berisi target bangunan dan perintah yang akan dieksekusi. Perhatikan bahwa properti CodeUri diatur ke hello_world, sehingga makefile harus terletak di root subdirektori hello_world, dan nama file harus 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
```

Pengujian dan debugging aplikasi nirserver

Dengan antarmuka baris perintah (CLI) AWS SAM, Anda dapat menguji secara lokal dan men-debug secara bertahap aplikasi nirserver Anda sebelum mengunggah aplikasi Anda ke AWS Cloud. Anda dapat memverifikasi apakah aplikasi Anda berperilaku sesuai yang diharapkan, men-debug kesalahan, dan memperbaiki masalah apa pun, sebelum melalui langkah-langkah pengemasan dan deployment aplikasi Anda.

Saat Anda memanggil fungsi Lambda secara lokal dalam mode debug di dalamnya AWS SAMCLI, Anda kemudian dapat melampirkan debugger ke dalamnya. Dengan debugger, Anda dapat menelusuri kode baris demi baris, melihat nilai-nilai dari berbagai variabel, dan memperbaiki masalah dengan cara yang sama seperti yang Anda lakukan untuk aplikasi lain.

Note

Jika aplikasi Anda memuat satu lapisan atau lebih, ketika Anda menjalankan dan men-debug aplikasi Anda secara lokal, paket lapisan akan diunduh dan di-cache pada host lokal Anda. Untuk informasi selengkapnya, lihat [Bagaimana lapisan di-cache secara lokal](#).

Topik

- [Memanggil fungsi Lambda secara lokal](#)
- [Menjalankan API Gateway lokal](#)
- [Mengintegrasikan dengan uji otomatis](#)
- [Membuat muatan peristiwa sampel](#)
- [Langkah-langkah debugging fungsi Lambda secara lokal](#)
- [Melewati argumen debug waktu aktif tambahan](#)
- [Validasi AWS SAM aplikasi Anda dengan Linter AWS CloudFormation](#)

Memanggil fungsi Lambda secara lokal

Anda dapat memanggil AWS Lambda fungsi Anda secara lokal dengan menggunakan [sam local invoke](#) AWS SAM CLI perintah dan memberikan ID logis fungsi dan file peristiwa. Dengan cara lain, sam local invoke juga menerima stdin sebagai sebuah peristiwa. Untuk informasi selengkapnya tentang peristiwa, lihat [Peristiwa](#) pada AWS Lambda Panduan Developer. Untuk informasi tentang

format pesan acara dari berbagai AWS layanan, lihat [Menggunakan AWS Lambda dengan layanan lain](#) di Panduan AWS Lambda Pengembang.

Note

sam local invokePerintah sesuai dengan perintah AWS Command Line Interface (AWS CLI) [aws lambda invoke](#). Anda dapat menggunakan salah satu perintah untuk menjalankan fungsi Lambda.

Anda harus menjalankan sam local invoke perintah di direktori proyek yang berisi fungsi yang ingin Anda panggil.

Contoh:

```
# 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
```

File variabel lingkungan

Untuk mendeklarasikan variabel lingkungan secara lokal yang mengesampingkan nilai yang ditentukan dalam template Anda, lakukan hal berikut:

1. Buat file JSON yang berisi variabel lingkungan untuk ditimpa.
2. Gunakan `--env-vars` argumen untuk mengganti nilai yang ditentukan dalam template Anda.

Mendeklarasikan variabel lingkungan

Untuk mendeklarasikan variabel lingkungan yang berlaku secara global ke semua sumber daya, tentukan `Parameters` objek seperti berikut:

```
{
  "Parameters": {
    "TABLE_NAME": "localtable",
```

```
    "BUCKET_NAME": "testBucket",
    "STAGE": "dev"
  }
}
```

Untuk mendeklarasikan variabel lingkungan yang berbeda untuk setiap sumber daya, tentukan objek untuk setiap sumber daya seperti berikut:

```
{
  "MyFunction1": {
    "TABLE_NAME": "localtable",
    "BUCKET_NAME": "testBucket",
  },
  "MyFunction2": {
    "TABLE_NAME": "localtable",
    "STAGE": "dev"
  }
}
```

Saat menentukan objek untuk setiap sumber daya, Anda dapat menggunakan pengidentifikasi berikut, yang terdaftar dalam urutan prioritas tertinggi hingga terendah:

1. `logical_id`
2. `function_id`
3. `function_name`
4. Pengidentifikasi jalur lengkap

Anda dapat menggunakan kedua metode sebelumnya untuk mendeklarasikan variabel lingkungan bersama-sama dalam satu file. Saat melakukannya, variabel lingkungan yang Anda sediakan untuk sumber daya tertentu lebih diutamakan daripada variabel lingkungan global.

Simpan variabel lingkungan Anda dalam file JSON, seperti `env.json`.

Mengesampingkan nilai variabel lingkungan

Untuk mengganti variabel lingkungan dengan yang ditentukan dalam file JSON Anda, gunakan `--env-vars` argumen dengan perintah `invoke orstart-api`. Sebagai contoh:

```
sam local invoke --env-vars env.json
```

Lapisan

Jika aplikasi Anda menyertakan lapisan, untuk informasi tentang cara men-debug masalah dengan lapisan pada host lokal Anda, lihat [Cara menggunakan lapisan](#).

Pelajari selengkapnya

Untuk contoh langsung menjalankan fungsi secara lokal, lihat [Modul 2 - Jalankan secara lokal di The Complete Workshop](#). AWS SAM

Menjalankan API Gateway lokal

Untuk memulai instance lokal Amazon API Gateway yang dapat Anda gunakan untuk menguji fungsionalitas permintaan/respons HTTP, gunakan perintah tersebut. [sam local start-api](#) AWS SAM CLI Fungsionalitas ini memiliki fitur hot reload sehingga Anda dapat dengan cepat mengembangkan dan mengulangi fungsi Anda.

Note

Hot reload adalah ketika hanya file yang diubah yang di-refresh, dan status aplikasi tetap sama. Sebaliknya, live reload adalah ketika seluruh aplikasi di-refresh, dan status aplikasi hilang.

Untuk petunjuk tentang penggunaan `sam local start-api` perintah, lihat [Menggunakan sam local start-api](#).

Secara default, AWS SAM menggunakan integrasi AWS Lambda proxy dan mendukung keduanya `HttpApi` dan jenis `Api` sumber daya. Untuk informasi selengkapnya tentang integrasi proxy untuk jenis `HttpApi` sumber daya, lihat [Bekerja dengan integrasi AWS Lambda proxy untuk API HTTP di Panduan](#) Pengembang API Gateway. Untuk informasi selengkapnya tentang integrasi proxy dengan tipe `Api` sumber daya, lihat [Memahami integrasi proxy API Gateway Lambda](#) di Panduan Pengembang API Gateway.

Contoh:

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

AWS SAM secara otomatis menemukan fungsi apa pun dalam templat AWS SAM yang memiliki sumber peristiwa `HttpApi` atau `Api` yang ditetapkan. Kemudian, ia memasang fungsi pada jalur HTTP yang ditentukan.

Pada contoh Api berikut, fungsi `Ratings` memasang `ratings.py:handler()` di `/ratings` untuk permintaan GET:

```
Ratings:
  Type: AWS::Serverless::Function
  Properties:
    Handler: ratings.handler
    Runtime: python3.9
    Events:
      Api:
        Type: Api
        Properties:
          Path: /ratings
          Method: get
```

Berikut adalah contoh respons Api:

```
// 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"
  });
}
```

Jika Anda memodifikasi kode fungsi Anda, jalankan `sam build` perintah `sam local start-api` untuk mendeteksi perubahan Anda.

File variabel lingkungan

Untuk mendeklarasikan variabel lingkungan secara lokal yang mengesampingkan nilai yang ditentukan dalam template Anda, lakukan hal berikut:

1. Buat file JSON yang berisi variabel lingkungan untuk diganti.
2. Gunakan `--env-vars` argumen untuk mengganti nilai yang ditentukan dalam template Anda.

Mendeklarasikan variabel lingkungan

Untuk mendeklarasikan variabel lingkungan yang berlaku secara global ke semua sumber daya, tentukan `Parameters` objek seperti berikut:

```
{
  "Parameters": {
    "TABLE_NAME": "localtable",
    "BUCKET_NAME": "testBucket",
    "STAGE": "dev"
  }
}
```

Untuk mendeklarasikan variabel lingkungan yang berbeda untuk setiap sumber daya, tentukan objek untuk setiap sumber daya seperti berikut:

```
{
  "MyFunction1": {
    "TABLE_NAME": "localtable",
    "BUCKET_NAME": "testBucket",
  },
  "MyFunction2": {
    "TABLE_NAME": "localtable",
    "STAGE": "dev"
  }
}
```

Saat menentukan objek untuk setiap sumber daya, Anda dapat menggunakan pengidentifikasi berikut, yang terdaftar dalam urutan prioritas tertinggi hingga terendah:

1. `logical_id`
2. `function_id`
3. `function_name`
4. Pengidentifikasi jalur lengkap

Anda dapat menggunakan kedua metode sebelumnya untuk mendeklarasikan variabel lingkungan bersama-sama dalam satu file. Saat melakukannya, variabel lingkungan yang Anda sediakan untuk sumber daya tertentu lebih diutamakan daripada variabel lingkungan global.

Simpan variabel lingkungan Anda dalam file JSON, seperti `env.json`.

Mengesampingkan nilai variabel lingkungan

Untuk mengganti variabel lingkungan dengan yang ditentukan dalam file JSON Anda, gunakan `--env-vars` argumen dengan perintah `invoke` `orstart-api`. Sebagai contoh:

```
$ sam local start-api --env-vars env.json
```

Lapisan

Jika aplikasi Anda menyertakan lapisan, untuk informasi tentang cara men-debug masalah dengan lapisan pada host lokal Anda, lihat [Cara menggunakan lapisan](#).

Mengintegrasikan dengan uji otomatis

Anda dapat menggunakan perintah `sam local invoke` untuk menguji kode Anda secara manual dengan menjalankan fungsi Lambda secara lokal. Dengan ini AWS SAMCLI, Anda dapat dengan mudah membuat pengujian integrasi otomatis dengan terlebih dahulu menjalankan pengujian terhadap fungsi Lambda lokal sebelum menerapkan ke Cloud. AWS

`sam local start-lambda` Perintah memulai titik akhir lokal yang mengemulasi titik akhir AWS Lambda pemanggilan. Anda dapat memanggilnya dari uji otomatis Anda. Karena titik akhir ini mengemulasi titik akhir AWS Lambda pemanggilan, Anda dapat menulis pengujian sekali, lalu menjalankannya (tanpa modifikasi apa pun) terhadap fungsi Lambda lokal, atau terhadap fungsi Lambda yang diterapkan. Anda juga dapat menjalankan pengujian yang sama terhadap AWS SAM tumpukan yang diterapkan di pipeline CI/CD Anda.

Beginilah cara kerjanya:

1. Mulai titik akhir Lambda lokal.

Mulai endpoint Lambda lokal dengan menjalankan perintah berikut di direktori yang berisi template Anda: AWS SAM

```
sam local start-lambda
```

Perintah ini memulai titik akhir lokal di `http://127.0.0.1:3001` yang mengemulasi AWS Lambda. Anda dapat menjalankan tes otomatis Anda pada titik akhir Lambda lokal ini. Saat Anda memanggil titik akhir ini menggunakan AWS CLI atau SDK, itu secara lokal mengeksekusi fungsi Lambda yang ditentukan dalam permintaan, dan menampilkan respons.

2. Jalankan uji integrasi terhadap titik akhir Lambda lokal.

Dalam pengujian integrasi, Anda dapat menggunakan AWS SDK untuk menjalankan fungsi Lambda dengan data pengujian, menunggu respons, dan memverifikasi bahwa respons sesuai dengan yang Anda harapkan. Untuk menjalankan uji integrasi secara lokal, Anda harus mengonfigurasi SDK AWS untuk mengirim Lambda lalu memanggil panggilan API guna meminta titik akhir Lambda lokal yang telah Anda mulai pada langkah sebelumnya.

Berikut ini adalah contoh Python (AWS SDK untuk bahasa lain memiliki konfigurasi yang serupa):

```
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"
```

Anda dapat menggunakan kode ini untuk menguji fungsi Lambda yang di-deploy dengan mengatur `running_locally` ke `False`. Ini mengatur AWS SDK untuk terhubung AWS Lambda di AWS Cloud.

Membuat muatan peristiwa sampel

Untuk mempermudah pengembangan dan pengujian lokal fungsi Lambda, Anda dapat membuat dan menyesuaikan muatan peristiwa untuk sejumlah layanan AWS seperti API Gateway, AWS CloudFormation, Amazon S3, dan seterusnya.

Untuk daftar lengkap layanan tempat Anda dapat membuat muatan peristiwa sampel, gunakan perintah ini:

```
sam local generate-event --help
```

Untuk daftar opsi yang dapat Anda gunakan untuk layanan tertentu, gunakan perintah ini:

```
sam local generate-event [SERVICE] --help
```

Contoh:

```
#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
```

Langkah-langkah debugging fungsi Lambda secara lokal

Anda dapat menggunakan AWS SAM dengan berbagai kit alat dan debugger AWS untuk menguji dan men-debug aplikasi nirserver Anda secara lokal.

Misalnya, Anda dapat melakukan debug bertahap lokal dari fungsi Lambda dengan mengatur titik putus, memeriksa variabel, dan menjalankan kode fungsi satu baris dalam satu waktu. Debug bertahap lokal memperketat putaran umpan balik dengan memungkinkan Anda menemukan dan memecahkan masalah yang mungkin Anda alami di cloud.

Menggunakan AWS Kit alat

AWS Kit alat adalah plugin lingkungan pengembangan terintegrasi (IDE) yang memberi Anda kemampuan untuk melakukan banyak tugas debug umum, seperti mengatur titik putus, memeriksa variabel, dan mengeksekusi kode fungsi satu baris dalam satu waktu. AWS Kit alat memudahkan Anda untuk mengembangkan, men-debug, dan men-deploy aplikasi nirserver yang dibuat menggunakan AWS SAM. Mereka memberikan pengalaman untuk membangun, menguji, men-debug, men-deploy, dan menjalankan fungsi Lambda yang terintegrasi ke dalam IDE Anda.

Untuk informasi selengkapnya tentang AWS Kit alat yang dapat Anda gunakan dengan AWS SAM, lihat hal berikut:

- [AWS Toolkit for Visual Studio Code](#)
- [AWS Cloud9](#)
- [AWS Toolkit for JetBrains](#)

Ada berbagai AWS Kit alat yang bekerja dengan kombinasi IDE dan waktu aktif yang berbeda. Tabel berikut mencantumkan kombinasi IDE/waktu aktif umum yang mendukung debugging bertahap aplikasi AWS SAM:

IDE	Waktu aktif	AWS Kit alat	Instruksi untuk debugging bertahap
Kode Studio Visual	<ul style="list-style-type: none"> • Node.Js • Python • .NET • Java • Go 	AWS Toolkit for Visual Studio Code	Bekerja dengan AWS Serverless Application dalam Panduan Pengguna AWS Toolkit for Visual Studio Code
AWS Cloud9	<ul style="list-style-type: none"> • Node.Js • Python 	AWS Cloud9, dengan AWS Kit alat diaktifkan ¹	Bekerja dengan aplikasi nirserver AWS menggunakan an AWS Kit alat di Panduan Pengguna AWS Cloud9.

IDE	Waktu aktif	AWS Kit alat	Instruksi untuk debugging bertahap
WebStorm	Node.Js	AWS Toolkit for JetBrains ²	Menjalankan (memanggil) atau debugging fungsi lokal di AWS Toolkit for JetBrains
PyCharm	Python	AWS Toolkit for JetBrains ²	Menjalankan (memanggil) atau debugging fungsi lokal di AWS Toolkit for JetBrains
Penunggang	.NET	AWS Toolkit for JetBrains ²	Menjalankan (memanggil) atau debugging fungsi lokal di AWS Toolkit for JetBrains
IntelliJ	Java	AWS Toolkit for JetBrains ²	Menjalankan (memanggil) atau debugging fungsi lokal di AWS Toolkit for JetBrains
GoLand	Go	AWS Toolkit for JetBrains ²	Menjalankan (memanggil) atau debugging fungsi lokal di AWS Toolkit for JetBrains

Catatan:

1. Untuk menggunakan AWS Cloud9 ke aplikasi debug AWS SAM secara bertahap, AWS Kit alat harus diaktifkan. Untuk informasi selengkapnya, lihat [Mengaktifkan AWS Kit alat](#) di Panduan Pengguna AWS Cloud9.
2. Untuk menggunakan AWS Toolkit for JetBrains ke aplikasi debug AWS SAM secara bertahap, Anda harus menginstal dan mengonfigurasinya terlebih dahulu dengan mengikuti petunjuk yang ada di [Menginstal AWS Toolkit for JetBrains](#) di AWS Toolkit for JetBrains.

Menjalakan AWS SAM secara lokal dalam mode debug

Selain berintegrasi dengan AWS Kit alat, Anda juga dapat menjalankan AWS SAM dalam "mode debug" untuk dilampirkan ke debugger pihak ke tiga seperti [ptvsd](#) atau [delve](#).

Untuk menjalankan AWS SAM dalam mode debug, gunakan perintah [sam local invoke](#) atau [sam local start-api](#) dengan opsi `--debug-port` atau `-d`.

Sebagai contoh:

```
# 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

Jika Anda menggunakan `sam local start-api`, instans API Gateway lokal memperlihatkan semua fungsi Lambda Anda. Namun, karena Anda dapat menentukan satu port debug, Anda hanya dapat men-debug satu fungsi dalam satu waktu. Anda perlu memanggil API Anda sebelum AWS SAM CLI mengikat ke port, yang memungkinkan debugger terhubung.

Melewati argumen debug waktu aktif tambahan

Untuk meneruskan argumen waktu aktif tambahan ketika Anda men-debug fungsi, gunakan variabel lingkungan `DEBUGGER_ARGS`. Ini melewati serangkaian argumen langsung ke perintah `run` yang AWS SAM CLI digunakan untuk memulai fungsi Anda.

Misalnya, jika Anda ingin memuat debugger seperti iKpdb pada waktu aktif dari fungsi Python, Anda dapat melewati sebagai berikut `DEBUGGER_ARGS`: `-m ikpdb --ikpdb-port=5858 --ikpdb-working-directory=/var/task/ --ikpdb-client-working-directory=/myApp --ikpdb-address=0.0.0.0`. Ini akan memuat iKpdb pada saat waktu aktif dengan argumen lain yang telah Anda tentukan.

Dalam hal ini, AWS SAM CLI perintah lengkap Anda adalah:

```
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
```

Anda dapat meneruskan argumen debugger ke fungsi dari semua waktu aktif.

Validasi AWS SAM aplikasi Anda dengan Linter AWS CloudFormation

AWS CloudFormationLinter (`cfn-lint`) adalah alat sumber terbuka yang dapat Anda gunakan untuk melakukan validasi terperinci pada templat Anda. AWS CloudFormation CFN-lint berisi aturan yang dipandu oleh spesifikasi sumber daya. AWS CloudFormation Gunakan `cfn-lint` untuk membandingkan sumber daya Anda dengan aturan tersebut untuk menerima pesan terperinci tentang kesalahan, peringatan, atau saran informasi. Atau, buat aturan kustom Anda sendiri untuk memvalidasi. Untuk mempelajari lebih lanjut tentang `cfn-lint`, lihat [cfn-lint](#) di repositori. AWS CloudFormation GitHub

Anda dapat menggunakan `cfn-lint` untuk memvalidasi template AWS Serverless Application Model (AWS SAM) Anda melalui AWS SAM Command Line Interface (AWS SAMCLI) dengan menjalankan `sam validate opsi. --lint`

```
sam validate --lint
```

Untuk menyesuaikan perilaku `cfn-lint`, seperti membuat aturan khusus atau menentukan opsi validasi, Anda dapat menentukan file konfigurasi. Untuk mempelajari lebih lanjut, lihat [File Config](#) di repositori AWS CloudFormation GitHub `cfn-lint`. Saat Anda menjalankansam validate --lint, perilaku `cfn-lint` yang ditentukan dalam file konfigurasi Anda akan diterapkan.

Contoh-contoh

Lakukan validasi cfn-lint pada template AWS SAM

```
sam validate --lint --template myTemplate.yaml
```

Pelajari selengkapnya

Untuk mempelajari lebih lanjut tentang sam validate, perintah lihat [sam validate](#).

Men-deploy aplikasi nirserver

AWS SAM menggunakan AWS CloudFormation sebagai mekanisme deployment utama. Untuk informasi selengkapnya, lihat [Apa yang dimaksud dengan AWS CloudFormation?](#) di Panduan Pengguna AWS CloudFormation. Input standar untuk men-deploy aplikasi nirserver adalah artefak bangunan yang dibuat dengan perintah [sam build](#). Untuk informasi selengkapnya tentang sam build, lihat [Membangun aplikasi nirserver](#).

Anda dapat men-deploy aplikasi Anda secara manual menggunakan perintah antarmuka baris perintah (CLI) AWS SAM. Anda juga dapat mengotomatisasi deployment aplikasi Anda menggunakan integrasi berkelanjutan serta sistem deployment berkelanjutan (CI/CD). [Anda dapat menggunakan banyak sistem CI/CD umum untuk menyebarkan AWS SAM aplikasi, termasuk, Jenkins, GitLab CI/CD AWS CodePipeline, dan Actions. GitHub](#)

Men-deploy menggunakan sistem CI/CD

AWS SAM membantu organisasi membuat alur untuk sistem CI/CD yang mereka pilih, sehingga organisasi tersebut dapat mencapai manfaat CI/CD dengan mudah, seperti mempercepat frekuensi deployment, memperpendek waktu tunggu untuk perubahan, dan mengurangi kesalahan deployment.

AWS SAM menyederhanakan tugas CI/CD untuk aplikasi nirserver dengan bantuan citra kontainer bangunan. Gambar yang AWS SAM disediakan menyertakan AWS SAM CLI dan membangun alat untuk sejumlah AWS Lambda runtime yang didukung. Ini membuatnya lebih mudah untuk membangun dan mengemas aplikasi tanpa server menggunakan file. AWS SAM CLI Citra ini juga mengurangi kebutuhan pada tim yang membuat dan mengelola citranya sendiri untuk sistem CI/CD. Untuk informasi selanjutnya tentang citra kontainer bangunan AWS SAM, lihat [Repositori citra](#).

Beberapa sistem CI/CD mendukung citra kontainer bangunan AWS SAM. Sistem CI/CD yang harus Anda gunakan tergantung pada beberapa faktor. Hal ini termasuk apakah aplikasi Anda menggunakan satu atau beberapa waktu aktif, atau apakah Anda ingin membangun aplikasi Anda dalam citra kontainer atau langsung pada mesin host, baik mesin virtual (VM) atau host bare metal.

AWS SAM juga menyediakan satu set templat alur default untuk beberapa sistem CI/CD yang merangkum praktik terbaik deployment AWS. Templat alur default ini menggunakan format konfigurasi alur JSON/YAML standar, dan praktik terbaik bawaan membantu melakukan deployment multi-akun dan multi-wilayah, serta memverifikasi bahwa alur tidak dapat membuat perubahan yang tidak diinginkan untuk infrastruktur.

Anda memiliki dua opsi utama AWS SAM untuk menggunakan aplikasi tanpa server: 1) Ubah konfigurasi pipeline yang ada untuk menggunakan AWS SAM CLI perintah, atau 2) Buat contoh konfigurasi pipeline CI/CD yang dapat Anda gunakan sebagai titik awal untuk aplikasi Anda sendiri.

Untuk informasi selengkapnya tentang opsi berikut, lihat topik berikut:

- [Mengubah alur CI/CD Anda yang sudah ada](#)
- [Menghasilkan alur awal CI/CD](#)

Men-deploy menggunakan AWS SAM CLI

Setelah Anda mengembangkan dan menguji aplikasi nirserver secara lokal, Anda dapat men-deploy aplikasi Anda dengan perintah [sam deploy](#).

Agar AWS SAM dapat memandu Anda melalui deployment dengan prompt, tentukan bendera `--guided`. Saat Anda menentukan flag ini, `sam deploy` perintah akan meressleting artefak aplikasi Anda, mengunggahnya ke Amazon Simple Storage Service (Amazon S3) (untuk arsip file.zip) atau ke Amazon Elastic Container Registry (Amazon ECR) (untuk gambar kontainer). Perintah tersebut kemudian men-deploy aplikasi Anda ke AWS Cloud.

Contoh:

```
# Deploy an application using prompts:
sam deploy --guided
```

Memecahkan masalah penerapan menggunakan AWS SAMCLI

AWS SAMCLIerror: “Kendala Keamanan Tidak Puas”

Saat menjalankan `sam deploy --guided`, Anda akan di-prompt dengan pertanyaan `HelloWorldFunction may not have authorization defined, Is this okay? [y/N]`. Jika Anda menanggapi prompt ini dengan **N** (respons default), Anda akan melihat kesalahan berikut:

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

Prompt tersebut menginformasikan bahwa aplikasi yang akan di-deploy mungkin memiliki API dari Amazon API Gateway yang dikonfigurasi tanpa otorisasi. Dengan memberikan tanggapan **N** untuk prompt ini, Anda memberitahu bahwa kesalahan mungkin terjadi.

Untuk memperbaikinya, Anda mempunyai opsi berikut:

- Konfigurasi aplikasi Anda dengan otorisasi. Untuk informasi tentang mengonfigurasi otorisasi, lihat [Pengendalian akses ke API dari API Gateway](#).
- Tanggapi pertanyaan ini dengan **Y** untuk menunjukkan bahwa Anda tidak memiliki masalah dengan men-deploy aplikasi yang memiliki API dari API Gateway yang dikonfigurasi tanpa otorisasi.

Deployment Gradual

Jika Anda ingin men-deploy aplikasi AWS SAM secara bertahap bukan sekaligus, Anda dapat menentukan konfigurasi deployment yang disediakan oleh AWS CodeDeploy. Untuk informasi selengkapnya, lihat [Bekerja dengan konfigurasi penerapan CodeDeploy di AWS CodeDeploy](#) Panduan Pengguna.

Untuk informasi tentang mengonfigurasi aplikasi AWS SAM untuk men-deploy secara bertahap, lihat [Men-deploy aplikasi nirserver secara bertahap](#).

Pelajari selengkapnya

Untuk contoh langsung penerapan aplikasi tanpa server, lihat berikut ini dari The Complete Workshop: AWS SAM

- [Modul 3 - Terapkan secara manual](#) - Pelajari cara membuat, mengemas, dan menerapkan aplikasi tanpa server menggunakan aplikasi. AWS SAM CLI
- [Modul 4 - CI/CD](#) - Pelajari cara mengotomatiskan fase pembuatan, paket, dan penerapan dengan membuat pipeline integrasi dan pengiriman berkelanjutan (CI/CD).

Mengubah alur CI/CD Anda yang sudah ada

Prosedur untuk alur CI/CD Anda yang sudah ada untuk men-deploy aplikasi nirserver menggunakan AWS SAM sedikit berbeda bergantung pada sistem CI/CD yang Anda gunakan.

Topik berikut memberikan contoh untuk mengonfigurasi sistem CI/CD Anda guna membangun aplikasi nirserver dalam citra kontainer build AWS SAM:

Topik

- [Men-deploy menggunakan AWS CodePipeline](#)
- [Menyebarkan menggunakan Bitbucket Pipelines](#)
- [Men-deploy menggunakan Jenkins](#)
- [Menyebarkan menggunakan CI/CD GitLab](#)
- [Menerapkan menggunakan Tindakan GitHub](#)

Men-deploy menggunakan AWS CodePipeline

Untuk mengonfigurasi alur [AWS CodePipeline](#) Anda guna mengotomatisasi build dan deployment aplikasi AWS SAM Anda, templat AWS CloudFormation dan file `buildspec.yml` Anda harus berisi baris yang melakukan hal berikut:

1. Referensikan citra kontainer build dengan waktu aktif yang diperlukan dari citra yang tersedia. Contoh berikut menggunakan citra kontainer build `public.ecr.aws/sam/build-nodejs14.x`.
2. Konfigurasi tahapan alur untuk menjalankan perintah antarmuka baris perintah (CLI) AWS SAM yang diperlukan. Contoh berikut menjalankan dua AWS SAM CLI perintah: `sam build` dan `sam deploy` (dengan opsi yang diperlukan).

Contoh ini mengasumsikan bahwa Anda telah mendeklarasikan semua fungsi dan lapisan dalam file templat AWS SAM Anda dengan `runtime: nodejs14.x`.

AWS CloudFormationcuplikan templat:

```
CodeBuildProject:
  Type: AWS::CodeBuild::Project
  Properties:
    Environment:
      ComputeType: BUILD_GENERAL1_SMALL
      Image: public.ecr.aws/sam/build-nodejs14.x
      Type: LINUX_CONTAINER
    ...
```

buildspec.ymlcuplikan:

```
version: 0.2
phases:
  build:
```

```
commands:
  - sam build
  - sam deploy --no-confirm-changeset --no-fail-on-empty-changeset
```

Untuk daftar citra kontainer build Amazon Elastic Container Registry (Amazon ECR) yang tersedia untuk waktu aktif yang berbeda, lihat [Repositori citra](#).

Menyebarkan menggunakan Bitbucket Pipelines

Untuk mengonfigurasi [Bitbucket Pipeline](#) untuk mengotomatiskan pembuatan dan penerapan AWS SAM aplikasi Anda, `bitbucket-pipelines.yml` file Anda harus berisi baris yang melakukan hal berikut:

1. Referensikan citra kontainer build dengan waktu aktif yang diperlukan dari citra yang tersedia. Contoh berikut menggunakan citra kontainer build `public.ecr.aws/sam/build-nodejs14.x`.
2. Konfigurasi tahapan alur untuk menjalankan perintah antarmuka baris perintah (CLI) AWS SAM yang diperlukan. Contoh berikut menjalankan dua AWS SAM CLI perintah: `sam build` dan `sam deploy` (dengan opsi yang diperlukan).

Contoh ini mengasumsikan bahwa Anda telah mendeklarasikan semua fungsi dan lapisan dalam file templat AWS SAM Anda dengan `runtime: nodejs14.x`.

```
image: public.ecr.aws/sam/build-nodejs14.x

pipelines:
  branches:
    main: # branch name
      - step:
          name: Build and Package
          script:
            - sam build
            - sam deploy --no-confirm-changeset --no-fail-on-empty-changeset
```

Untuk daftar citra kontainer build Amazon Elastic Container Registry (Amazon ECR) yang tersedia untuk waktu aktif yang berbeda, lihat [Repositori citra](#).

Men-deploy menggunakan Jenkins

Untuk mengonfigurasi alur [Jenkins](#) Anda guna mengotomatiskan build dan deployment aplikasi AWS SAM Anda, `Jenkinsfile` Anda harus berisi baris yang melakukan hal berikut:

1. Referensikan citra kontainer build dengan waktu aktif yang diperlukan dari citra yang tersedia. Contoh berikut menggunakan citra kontainer build `public.ecr.aws/sam/build-nodejs14.x`.
2. Konfigurasi tahapan alur untuk menjalankan perintah antarmuka baris perintah (CLI) AWS SAM yang diperlukan. Contoh berikut menjalankan dua AWS SAM CLI perintah: `sam build` dan `sam deploy` (dengan opsi yang diperlukan).

Contoh ini mengasumsikan bahwa Anda telah mendeklarasikan semua fungsi dan lapisan dalam file templat AWS SAM Anda dengan `runtime: nodejs14.x`.

```
pipeline {
  agent { docker { image 'public.ecr.aws/sam/build-nodejs14.x' } }
  stages {
    stage('build') {
      steps {
        sh 'sam build'
        sh 'sam deploy --no-confirm-changeset --no-fail-on-empty-changeset'
      }
    }
  }
}
```

Untuk daftar citra kontainer build Amazon Elastic Container Registry (Amazon ECR) yang tersedia untuk waktu aktif yang berbeda, lihat [Repositori citra](#).

Menyebarkan menggunakan CI/CD GitLab

Untuk mengonfigurasi [GitLab](#) pipeline agar mengotomatiskan pembuatan dan penerapan AWS SAM aplikasi, `gitlab-ci.yml` file harus berisi baris yang melakukan hal berikut:

1. Referensikan citra kontainer build dengan waktu aktif yang diperlukan dari citra yang tersedia. Contoh berikut menggunakan citra kontainer build `public.ecr.aws/sam/build-nodejs14.x`.
2. Konfigurasi tahapan alur untuk menjalankan perintah antarmuka baris perintah (CLI) AWS SAM yang diperlukan. Contoh berikut menjalankan dua AWS SAM CLI perintah: `sam build` dan `sam deploy` (dengan opsi yang diperlukan).

Contoh ini mengasumsikan bahwa Anda telah mendeklarasikan semua fungsi dan lapisan dalam file templat AWS SAM Anda dengan `runtime: nodejs14.x`.

```
image: public.ecr.aws/sam/build-nodejs14.x
deploy:
  script:
    - sam build
    - sam deploy --no-confirm-changeset --no-fail-on-empty-changeset
```

Untuk daftar citra kontainer build Amazon Elastic Container Registry (Amazon ECR) yang tersedia untuk waktu aktif yang berbeda, lihat [Repositori citra](#).

Menerapkan menggunakan Tindakan GitHub

Untuk mengonfigurasi [GitHub](#) pipeline Anda untuk mengotomatiskan pembuatan dan penerapan AWS SAM aplikasi Anda, Anda harus terlebih dahulu menginstal antarmuka baris AWS SAM perintah (CLI) di host Anda. Anda dapat menggunakan [GitHub Tindakan](#) dalam GitHub alur kerja Anda untuk membantu penyiapan ini.

Contoh GitHub alur kerja berikut menyiapkan host Ubuntu menggunakan serangkaian GitHub Tindakan, lalu menjalankan AWS SAM CLI perintah untuk membangun dan menyebarkan aplikasi AWS SAM:

```
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
```

Untuk daftar citra kontainer build Amazon Elastic Container Registry (Amazon ECR) yang tersedia untuk waktu aktif yang berbeda, lihat [Repositori citra](#).

Menghasilkan alur awal CI/CD

Saat Anda siap untuk men-deploy aplikasi nirserver secara otomatis, Anda dapat membuat alur deployment untuk sistem CI/CD pilihan Anda. AWS SAM menyediakan satu set templat alur awal yang dapat digunakan untuk menghasilkan alur dalam hitungan menit menggunakan perintah [sam pipeline init](#).

Templat alur awal menggunakan sintaks JSON/YAML yang sudah dikenal dari sistem CI/CD, dan menggabungkan praktik terbaik seperti mengelola artefak di beberapa akun dan wilayah, dan menggunakan jumlah minimum izin yang diperlukan untuk men-deploy aplikasi. [Saat ini, AWS SAM CLI mendukung pembuatan konfigurasi pipeline CI/CD starter untuk, Jenkins, CI/CD AWS CodePipeline, Actions, dan Bitbucket GitLab Pipelines. GitHub](#)

Berikut adalah tugas tingkat tinggi yang perlu Anda lakukan untuk menghasilkan konfigurasi alur awal:

1. Buat sumber daya infrastruktur – Alur Anda memerlukan sumber daya AWS tertentu, misalnya pengguna IAM dan IAM role dengan izin yang diperlukan, bucket Amazon S3, dan secara opsional repositori Amazon ECR.
2. Hubungkan repositori Git Anda dengan sistem CI/CD Anda – Sistem CI/CD Anda perlu mengetahui repositori Git mana yang akan memicu alur untuk dijalankan. Perhatikan bahwa langkah ini mungkin tidak diperlukan, tergantung pada kombinasi repositori Git dan sistem CI/CD yang Anda gunakan.
3. Hasilkan konfigurasi alur Anda – Langkah ini menghasilkan konfigurasi alur awal yang mencakup dua tahap deployment.
4. Lakukan konfigurasi alur ke repositori Git Anda – Langkah ini diperlukan guna memastikan bahwa sistem CI/CD Anda mengetahui konfigurasi alur Anda, dan akan berjalan saat perubahan dilakukan.

Setelah Anda membuat konfigurasi alur awal dan melakukannya ke repositori Git Anda, setiap kali seseorang melakukan perubahan kode ke repositori tersebut, alur Anda akan terpicu untuk berjalan secara otomatis.

Urutan langkah-langkah ini, dan detail setiap langkah, bervariasi berdasarkan sistem CI/CD Anda:

- Jika Anda menggunakan AWS CodePipeline, lihat [Menghasilkan alur awal untuk AWS CodePipeline](#).

- Jika Anda menggunakan Jenkins, GitLab CI/CD, GitHub Actions, atau Bitbucket Pipelines, lihat. [Menghasilkan saluran pipa starter untuk Jenkins, GitLab CI/CD, GitHub Actions, atau Bitbucket Pipelines](#)

Menghasilkan alur awal untuk AWS CodePipeline

Untuk membuat konfigurasi alur awal untuk AWS CodePipeline, lakukan tugas berikut dalam urutan ini:

1. Buat sumber daya infrastruktur
2. Hasilkan konfigurasi alur
3. Lakukan konfigurasi alur Anda ke Git
4. Hubungkan repositori Git dengan sistem CI/CD Anda

Note

Prosedur berikut menggunakan dua AWS SAM CLI perintah, [sam pipeline bootstrap](#) dan [sam pipeline init](#). Alasan ada dua perintah adalah untuk menangani kasus penggunaan tempat administrator (yaitu, pengguna yang memerlukan izin untuk menyiapkan sumber daya infrastruktur AWS, seperti pengguna IAM dan IAM role) memiliki lebih banyak izin daripada developer tersebut (yaitu, pengguna yang hanya memerlukan izin untuk mengatur alur individu, tetapi bukan sumber daya infrastruktur AWS yang diperlukan).

Langkah 1: Buat sumber daya infrastruktur

Alur yang menggunakan AWS SAM memerlukan sumber daya AWS tertentu, seperti pengguna IAM dan IAM role dengan izin yang diperlukan, bucket Amazon S3, dan secara opsional repositori Amazon ECR. Anda harus memiliki satu set sumber daya infrastruktur untuk setiap tahap deployment alur.

Anda dapat menjalankan perintah berikut guna membantu penyiapan ini:

```
sam pipeline bootstrap
```


Note

Jalankan perintah sebelumnya untuk setiap tahap penerapan pipeline Anda.

Langkah 2: Hasilkan konfigurasi alur

Untuk menghasilkan konfigurasi alur, jalankan perintah berikut:

```
sam pipeline init
```

Langkah 3: Lakukan konfigurasi alur Anda ke repositori Git

Langkah ini diperlukan guna memastikan bahwa sistem CI/CD mengetahui konfigurasi alur Anda, dan akan berjalan saat perubahan dilakukan.

Langkah 4: Hubungkan repositori Git dengan sistem CI/CD Anda

Untuk AWS CodePipeline sekarang Anda dapat membuat koneksi dengan menjalankan perintah berikut:

```
sam deploy -t codepipeline.yaml --stack-name <pipeline-stack-name> --  
capabilities=CAPABILITY_IAM --region <region-X>
```

Jika Anda menggunakan GitHub atau Bitbucket, setelah menjalankan `sam deploy` perintah sebelumnya, selesaikan koneksi dengan mengikuti langkah-langkah di bawah Untuk menyelesaikan koneksi yang ditemukan pada topik Perbarui koneksi [yang tertunda](#) di panduan pengguna konsol Alat Pengembang. Selain itu, simpan salinan `CodeStarConnectionArn` dari output perintah `sam deploy`, karena Anda akan membutuhkannya jika ingin menggunakan AWS CodePipeline dengan cabang selain `main`.

Mengonfigurasi cabang-cabang lain

Secara default, AWS CodePipeline menggunakan cabang `main` dengan AWS SAM. Jika ingin menggunakan cabang selain `main`, Anda harus menjalankan kembali perintah `sam deploy`. Perhatikan bahwa bergantung pada repositori Git mana yang digunakan, Anda mungkin juga perlu menyediakan `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>"
```

Pelajari selengkapnya

Untuk contoh langsung menyiapkan pipeline CI/CD, lihat [CI/CD](#) dengan di The Complete Workshop. [AWS CodePipeline AWS SAM](#)

Menghasilkan saluran pipa starter untuk Jenkins, GitLab CI/CD, GitHub Actions, atau Bitbucket Pipelines

Untuk menghasilkan konfigurasi pipeline starter untuk Jenkins, GitLab CI/CD, GitHub Actions, atau Bitbucket Pipelines, lakukan tugas-tugas berikut dalam urutan ini:

1. Buat sumber daya infrastruktur
2. Hubungkan repositori Git dengan sistem CI/CD Anda
3. Buat objek kredensial
4. Hasilkan konfigurasi alur
5. Lakukan konfigurasi alur Anda ke repositori Git

Note

Prosedur berikut menggunakan dua AWS SAM CLI perintah, [sam pipeline bootstrap](#) dan [sam pipeline init](#). Alasan ada dua perintah adalah untuk menangani kasus penggunaan tempat administrator (yaitu, pengguna yang memerlukan izin untuk menyiapkan sumber daya infrastruktur AWS, seperti pengguna IAM dan IAM role) memiliki lebih banyak izin daripada developer tersebut (yaitu, pengguna yang hanya memerlukan izin untuk mengatur alur individu, tetapi bukan sumber daya infrastruktur AWS yang diperlukan).

Langkah 1: Buat sumber daya infrastruktur

Alur yang menggunakan AWS SAM memerlukan sumber daya AWS tertentu, seperti pengguna IAM dan IAM role dengan izin yang diperlukan, bucket Amazon S3, dan secara opsional repositori Amazon ECR. Anda harus memiliki satu set sumber daya infrastruktur untuk setiap tahap deployment alur.

Anda dapat menjalankan perintah berikut guna membantu penyiapan ini:

```
sam pipeline bootstrap
```

Note

Jalankan perintah sebelumnya untuk setiap tahap penerapan pipeline Anda.

Anda harus menangkap kredensial AWS (id kunci dan kunci rahasia) untuk pengguna alur untuk setiap tahap deployment alur Anda, karena diperlukan untuk langkah selanjutnya.

Langkah 2: Hubungkan repositori Git dengan sistem CI/CD Anda

Menghubungkan repositori Git ke sistem CI/CD Anda diperlukan agar sistem CI/CD dapat mengakses kode sumber aplikasi Anda untuk build dan deployment.

Note

Anda dapat melewati langkah ini jika menggunakan salah satu kombinasi berikut, karena koneksi dibuat untuk Anda secara otomatis:

1. GitHub Tindakan dengan GitHub repositori
2. GitLab CI/CD dengan repositori GitLab
3. Bitbucket Pipelines dengan repositori Bitbucket

Untuk menghubungkan repositori Git dengan sistem CI/CD Anda, lakukan salah satu hal berikut:

- Jika Anda menggunakan Jenkins, lihat [dokumentasi Jenkins](#) untuk “Menambahkan sumber cabang.”

- Jika Anda menggunakan GitLab CI/CD dan repositori Git selain GitLab, lihat [GitLab dokumentasi](#) untuk “menghubungkan repositori eksternal.”

Langkah 3: Buat objek kredensial

Setiap sistem CI/CD memiliki caranya sendiri dalam mengelola kredensial yang diperlukan sistem CI/CD untuk mengakses repositori Git Anda.

Untuk membuat objek kredensial yang diperlukan, lakukan salah satu hal berikut:

- Jika Anda menggunakan Jenkins, buat satu "kredensial" yang menyimpan id kunci dan kunci rahasia. Ikuti instruksi di blog [Membangun Alur Jenkins dengan AWS SAM](#), di bagian Konfigurasi Jenkins. Anda akan membutuhkan “id Kredensial” untuk langkah berikutnya.
- Jika Anda menggunakan GitLab CI/CD, buat dua “variabel yang dilindungi”, satu untuk masing-masing id kunci dan kunci rahasia. Ikuti petunjuk dalam [GitLab dokumentasi](#) - Anda akan memerlukan dua “kunci variabel” untuk langkah berikutnya.
- Jika Anda menggunakan GitHub Actions, buat dua “rahasia terenkripsi”, satu untuk masing-masing kunci dan kunci rahasia. Ikuti instruksi dalam [GitHub dokumentasi](#) - Anda akan memerlukan dua “nama rahasia” untuk langkah selanjutnya.
- Jika Anda menggunakan Bitbucket Pipelines, buat dua “variabel aman”, satu untuk masing-masing id kunci dan kunci rahasia. Ikuti instruksi dalam [Variabel dan rahasia](#) - Anda akan memerlukan dua “nama rahasia” untuk langkah berikutnya.

Langkah 4: Hasilkan konfigurasi alur

Untuk menghasilkan konfigurasi alur, jalankan perintah berikut. Anda harus memasukkan objek kredensial yang dibuat pada langkah sebelumnya:

```
aws s3 cp sam pipeline init
```

Langkah 5: Lakukan konfigurasi alur Anda ke repositori Git

Langkah ini diperlukan untuk memastikan sistem CI/CD mengetahui konfigurasi alur Anda, dan akan berjalan saat perubahan dilakukan.

Pelajari selengkapnya

Untuk contoh langsung menyiapkan pipeline CI/CD menggunakan, lihat [CI/CD](#) dengan di The GitHub Actions Complete Workshop. GitHub AWS SAM

Menyesuaikan saluran pipa starter

Sebagai administrator CI/CD, Anda mungkin ingin menyesuaikan template pipeline pemula, dan petunjuk terpandu terkait, yang dapat digunakan pengembang di organisasi Anda untuk membuat konfigurasi pipeline.

AWS SAM CLI menggunakan template Cookiecutter saat membuat template pemula. Untuk detail tentang template pemotong kue, [Cookiecutter](#).

Anda juga dapat menyesuaikan prompt yang AWS SAM CLI ditampilkan kepada pengguna saat membuat konfigurasi pipeline menggunakan perintah `sam pipeline init` Untuk menyesuaikan permintaan pengguna, lakukan hal berikut:

1. Buat **questions.json** file — `questions.json` File harus berada di root repositori proyek. Ini adalah direktori yang sama dengan `cookiecutter.json` file. Untuk melihat skema `questions.json` file, lihat [questions.json.schema](#). Untuk melihat `questions.json` file contoh, lihat [questions.json](#).
2. Peta kunci pertanyaan dengan nama `cookiecutter` - Setiap objek dalam `questions.json` file membutuhkan kunci yang cocok dengan nama dalam template `cookiecutter`. Pencocokan kunci ini adalah bagaimana pengguna AWS SAM CLI memetakan tanggapan meminta ke template pemotong cookie. Untuk melihat contoh pencocokan kunci ini, lihat [Contoh file](#) bagian nanti dalam topik ini.
3. Buat **metadata.json** file — Deklarasikan jumlah tahapan yang akan dimiliki pipeline dalam file `metadata.json` Jumlah tahapan menginstruksikan `sam pipeline init` perintah berapa banyak tahapan untuk meminta informasi tentang, atau dalam kasus `--bootstrap` opsi, berapa banyak tahapan untuk membuat sumber daya infrastruktur untuk. Untuk melihat `metadata.json` file contoh yang mendeklarasikan pipeline dengan dua tahap, lihat [metadata.json](#).

Contoh proyek

Berikut adalah contoh proyek, yang masing-masing menyertakan template `Cookiecutter`, `questions.json` file, dan file: `metadata.json`

- Contoh Jenkins: Template pipa [Jenkins dua tahap](#)
- CodePipeline contoh: [Template CodePipeline pipa dua tahap](#)

Contoh file

Kumpulan file berikut menunjukkan bagaimana pertanyaan dalam `questions.json` file dikaitkan dengan entri dalam file template Cookiecutter. Perhatikan bahwa contoh ini adalah cuplikan file, bukan file lengkap. Untuk melihat contoh file lengkap, lihat [Contoh proyek](#) bagian sebelumnya dalam topik ini.

Contoh: `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"
  }, {
    ...
  }
```

Contoh: `cookiecutter.json`

```
{
  "outputDir": "aws-sam-pipeline",
  "pipeline_user_jenkins_credential_id": "",
  "sam_template": "",
  ...
}
```

Contoh: `Jenkinsfile`

```
pipeline {
```

```
agent any
environment {
  PIPELINE_USER_CREDENTIAL_ID =
'{{cookiecutter.pipeline_user_jenkins_credential_id}}'
  SAM_TEMPLATE = '{{cookiecutter.sam_template}}'
  ...
}
```

Menggunakan otentikasi OIDC dengan pipeline AWS SAM

AWS Serverless Application Model (AWS SAM) mendukung otentikasi pengguna OpenID Connect (OIDC) untuk platform Bitbucket, GitHub Actions, dan integrasi GitLab berkelanjutan dan pengiriman berkelanjutan (CI/CD). Dengan dukungan ini, Anda dapat menggunakan akun pengguna CI/CD resmi dari salah satu platform ini untuk mengelola saluran aplikasi tanpa server Anda. Jika tidak, Anda perlu membuat dan mengelola beberapa AWS Identity and Access Management (IAM) pengguna untuk mengontrol akses ke AWS SAM saluran pipa.

Siapkan OIDC dengan pipa AWS SAM

Selama proses `sam pipeline bootstrap` konfigurasi, lakukan hal berikut untuk mengatur OIDC dengan pipeline Anda AWS SAM.

1. Saat diminta untuk memilih penyedia identitas, pilih OIDC.
2. Selanjutnya, pilih penyedia OIDC yang didukung.
3. Masukkan URL penyedia OIDC, dimulai dengan **https://**

Note

AWS SAM merferensikan URL ini saat menghasilkan jenis `AWS::IAM::OIDCProvider` sumber daya.

4. Selanjutnya, ikuti petunjuknya dan masukkan informasi platform CI/CD yang diperlukan untuk mengakses platform yang dipilih. Detail ini bervariasi menurut platform dan dapat mencakup:
 - ID klien OIDC.
 - Nama repositori kode atau pengidentifikasi unik universal (UUID).
 - Nama grup atau organisasi yang terkait dengan repositori.
 - GitHub organisasi tempat repositori kode milik.
 - GitHub nama repositori.

- Cabang tempat penerapan akan terjadi.
5. AWS SAM menampilkan ringkasan konfigurasi OIDC yang dimasukkan. Masukkan nomor untuk pengaturan untuk mengeditnya, atau tekan Enter untuk melanjutkan.
 6. Ketika diminta untuk mengkonfirmasi pembuatan sumber daya yang diperlukan untuk mendukung koneksi OIDC yang dimasukkan, tekan Y untuk melanjutkan.

AWS SAM menghasilkan `AWS::IAM::OIDCProvider` AWS CloudFormation sumber daya dengan konfigurasi yang disediakan yang mengasumsikan peran eksekusi pipeline. Untuk mempelajari lebih lanjut tentang jenis AWS CloudFormation sumber daya ini, lihat [AWS::IAM::OIDCProvider](#) di Panduan Pengguna. AWS CloudFormation

Note

Jika sumber daya penyedia identitas (IDP) sudah ada di sumber daya Anda Akun AWS, AWS SAM merujuk ke itu alih-alih membuat sumber daya baru.

Contoh

Berikut ini adalah contoh pengaturan OIDC dengan AWS SAM pipeline.

```
Select a permissions provider:
  1 - IAM (default)
  2 - OpenID Connect (OIDC)
Choice (1, 2): 2
Select an OIDC provider:
  1 - GitHub Actions
  2 - GitLab
  3 - Bitbucket
Choice (1, 2, 3): 1
Enter the URL of the OIDC provider [https://token.actions.githubusercontent.com]:
Enter the OIDC client ID (sometimes called audience) [sts.amazonaws.com]:
Enter the GitHub organization that the code repository belongs to. If there is no
organization enter your username instead: my-org
Enter GitHub repository name: testing
Enter the name of the branch that deployments will occur from [main]:
```

[3] Reference application build resources


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

[4] Summary

Below is the summary of the answers:

- 1 - Account: 123456
- 2 - Stage configuration name: dev
- 3 - Region: us-east-1
- 4 - OIDC identity provider URL: https://token.actions.githubusercontent.com
- 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]:

Pelajari selengkapnya

Untuk informasi selengkapnya tentang penggunaan OIDC dengan AWS SAM pipeline, lihat [sam pipeline bootstrap](#)

Menggunakan AWS SAMCLI untuk mengunggah file lokal saat penerapan

Saat mengembangkan, Anda akan sering merasa bermanfaat untuk memecah kode aplikasi Anda menjadi file terpisah untuk mengatur dan mengelola aplikasi Anda dengan lebih baik. Contoh

dasar dari ini adalah memisahkan kode AWS Lambda fungsi Anda dari kode infrastruktur Anda. Anda melakukan ini dengan mengatur kode fungsi Lambda Anda di subdirektori proyek Anda dan mereferensikan jalur lokalnya dalam template () Anda AWS Serverless Application Model. AWS SAM

Saat Anda menerapkan aplikasi ke AWS Cloud, AWS CloudFormation mengharuskan file lokal Anda diunggah terlebih dahulu ke AWS layanan yang dapat diakses, seperti Amazon Simple Storage Service (Amazon S3). Anda dapat menggunakan AWS SAM CLI untuk memfasilitasi proses ini secara otomatis. Gunakan `sam package` perintah `sam deploy` or untuk melakukan hal berikut:

1. Unggah file lokal Anda secara otomatis ke AWS layanan yang dapat diakses.
2. Perbarui templat aplikasi Anda secara otomatis untuk mereferensikan jalur file baru.

Topik

- [Demo: Gunakan kode fungsi AWS SAM CLI untuk mengunggah Lambda](#)
- [Kasus penggunaan yang didukung](#)
- [Pelajari selengkapnya](#)

Demo: Gunakan kode fungsi AWS SAM CLI untuk mengunggah Lambda

Dalam demo ini, kami menginisialisasi contoh aplikasi Hello World menggunakan jenis paket.zip untuk fungsi Lambda kami. Kami menggunakan AWS SAM CLI untuk secara otomatis mengunggah kode fungsi Lambda kami ke Amazon S3 dan mereferensikan jalur barunya di templat aplikasi kami.

Pertama, kami menjalankan `sam init` untuk menginisialisasi aplikasi Hello World kami.

```
$ 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

...

```

Kode fungsi Lambda kami diatur dalam `hello_world` subdirektori proyek kami.

```

demo
### README.md
### hello_world
#   ### __init__.py
#   ### app.py
#   ### requirements.txt
### template.yaml
### tests

```

Dalam AWS SAM template kami, kami mereferensikan jalur lokal ke kode fungsi Lambda kami menggunakan properti `CodeUri`

```

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
  ...
```

Selanjutnya, kami menjalankan `sam build` untuk membangun aplikasi kami dan mempersiapkan penerapan.

```
$ 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
...
```

Selanjutnya, kami menjalankan `sam deploy --guided` untuk menyebarkan aplikasi kami.

```
$ sam deploy --guided

Configuring SAM deploy
=====

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

Setting default arguments for 'sam deploy'
=====
Stack Name [demo]: ENTER
```

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

Looking for resources needed for deployment:
...
Saved arguments to config file
Running 'sam deploy' for future deployments will use the parameters saved
above.

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

```
File with same data already exists at demo/da3c598813f1c2151579b73ad788cac8, skipping
upload
```

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

```
Initiating deployment
=====
...
Waiting for changeset to be created..
```

CloudFormation stack changeset

Operation	LogicalResourceId	ResourceType	Replacement
+ Add	HelloWorldFunctionHell oWorldPermissionProd	AWS::Lambda::Permissio n	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	ResourceType	LogicalResourceId	ResourceStatusReason
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

```

Selama penerapan, AWS SAM CLI secara otomatis mengunggah kode fungsi Lambda kami ke Amazon S3 dan memperbarui template kami. Template kami yang dimodifikasi di AWS CloudFormation konsol mencerminkan jalur bucket Amazon S3.

```

AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://aws-sam-cli-managed-default-samclisourcebucket-1a4x26zbcdkqr/demo/
da3c598813f1c2151579b73ad788cac8
      Handler: app.lambda_handler
      ...

```

Kasus penggunaan yang didukung

Secara otomatis AWS SAM CLI dapat memfasilitasi proses ini untuk sejumlah jenis file, jenis AWS CloudFormation sumber daya, dan AWS CloudFormation makro.

Jenis berkas

File dan Docker gambar aplikasi didukung.

Jenis sumber daya AWS CloudFormation

Berikut ini adalah daftar jenis sumber daya yang didukung dan propertinya:

Sumber daya	Properti
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

Sumber daya	Properti
<code>AWS::Glue::Job</code>	<code>Command.ScriptLocation</code>
<code>AWS::Lambda::Function</code>	Code <code>Code.ImageUri</code>
<code>AWS::Lambda::LayerVersion</code>	Content
<code>AWS::Serverless::Api</code>	<code>DefinitionUri</code>
<code>AWS::Serverless::Function</code>	<code>CodeUri</code> <code>ImageUri</code>
<code>AWS::Serverless::GraphQLApi</code>	<u>SchemaUri</u> <u>Function.CodeUri</u> <u>Resolver.CodeUri</u>
<code>AWS::Serverless::HttpApi</code>	<code>DefinitionUri</code>
<code>AWS::Serverless::LayerVersion</code>	<code>ContentUri</code>
<code>AWS::Serverless::StateMachine</code>	<code>DefinitionUri</code>
<code>AWS::StepFunctions::StateMachine</code>	<code>DefinitionS3Location</code>

AWS CloudFormation makro

File yang direferensikan menggunakan makro `AWS::Include` transformasi didukung.

Pelajari selengkapnya

Untuk mempelajari lebih lanjut tentang `AWS::Include` transformasi, lihat [AWS::Include transformasi](#) di Panduan AWS CloudFormation Pengguna.

Untuk melihat contoh penggunaan `AWS::Include` transformasi dalam AWS SAM template, lihat pola [API Gateway API ke SQS](#) di Serverless Land.

Pemantauan aplikasi nirserver

Konfigurasi dan lakukan pemantauan aplikasi AWS Serverless Application Model (AWS SAM) Anda.

Topik

- [Pantau aplikasi tanpa server Anda dengan CloudWatch Application Insights](#)
- [Cara menggunakan log](#)

Pantau aplikasi tanpa server Anda dengan CloudWatch Application Insights

Amazon CloudWatch Application Insights membantu Anda memantau AWS sumber daya dalam aplikasi Anda untuk membantu mengidentifikasi potensi masalah. Ini dapat menganalisis data AWS sumber daya untuk tanda-tanda masalah dan membangun dasbor otomatis untuk memvisualisasikannya. Anda dapat mengonfigurasi CloudWatch Application Insights untuk digunakan dengan AWS Serverless Application Model (AWS SAM) aplikasi Anda. Untuk mempelajari lebih lanjut tentang Wawasan CloudWatch Aplikasi, lihat [Wawasan CloudWatch Aplikasi Amazon](#) di CloudWatch Panduan Pengguna Amazon.

Topik

- [Mengkonfigurasi Wawasan CloudWatch Aplikasi dengan AWS SAM](#)
- [Langkah selanjutnya](#)

Mengkonfigurasi Wawasan CloudWatch Aplikasi dengan AWS SAM

Konfigurasi Wawasan CloudWatch Aplikasi untuk AWS SAM aplikasi Anda melalui AWS SAM Command Line Interface (AWS SAMCLI) atau melalui AWS SAM template Anda.

Konfigurasi melalui AWS SAMCLI

Saat menginisialisasi aplikasi Anda dengan `sam init`, aktifkan CloudWatch Application Insights melalui alur interaktif atau dengan menggunakan opsi `--application-insights`

Untuk mengaktifkan CloudWatch Application Insights melalui alur AWS SAM CLI interaktif, masukkan `y` saat diminta.

```
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]:
```

Untuk mengaktifkan CloudWatch Application Insights dengan `--application-insights` opsi, lakukan hal berikut.

```
sam init --application-insights
```

Untuk mempelajari lebih lanjut tentang menggunakan `sam init` perintah, lihat [sam init](#).

Konfigurasi melalui AWS SAM templat

Aktifkan CloudWatch Application Insights dengan mendefinisikan `AWS::ResourceGroups::Group` dan `AWS::ApplicationInsights::Application` sumber daya dalam template Anda AWS SAM.

```
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`— Membuat grup untuk mengatur AWS sumber daya Anda untuk mengelola dan mengotomatiskan tugas pada sejumlah besar sumber daya sekaligus. Di sini, Anda membuat grup sumber daya untuk digunakan dengan CloudWatch Application Insights. Untuk informasi selengkapnya tentang jenis sumber daya ini, lihat [AWS::ResourceGroups::Group](#) di Panduan AWS CloudFormation Pengguna.
- `AWS::ApplicationInsights::Application`— Mengkonfigurasi Wawasan CloudWatch Aplikasi untuk grup sumber daya. Untuk informasi selengkapnya tentang jenis sumber daya ini, lihat [AWS::ApplicationInsights::Application](#) di Panduan AWS CloudFormation Pengguna.

Kedua sumber daya secara otomatis diteruskan ke AWS CloudFormation saat penerapan aplikasi. Anda dapat menggunakan AWS CloudFormation sintaks dalam AWS SAM template Anda untuk mengkonfigurasi CloudWatch Application Insights lebih lanjut. Untuk informasi selengkapnya, lihat [Menggunakan AWS CloudFormation templat](#) di Panduan CloudWatch Pengguna Amazon.

Saat menggunakan `sam init --application-insights` perintah, kedua sumber daya ini secara otomatis dihasilkan di AWS SAM template Anda. Berikut adalah contoh template yang dihasilkan.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: >
  sam-app-test

  Sample SAM Template for sam-app-test

# More info about Globals: https://github.com/awslabs/serverless-application-model/blob/master/docs/globals.rst
Globals:
  Function:
    Timeout: 3
    MemorySize: 128

Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function # More info about Function Resource:
    https://github.com/awslabs/serverless-application-model/blob/master/versions/2016-10-31.md#awsserverlessfunction
    Properties:
      CodeUri: hello_world/
      Handler: app.lambda_handler
```

```

    Runtime: python3.9
    Architectures:
    - x86_64
    Events:
      HelloWorld:
        Type: Api # More info about API Event Source: https://github.com/awslabs/serverless-application-model/blob/master/versions/2016-10-31.md#api
        Properties:
          Path: /hello
          Method: get

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

Outputs:
  # ServerlessRestApi is an implicit API created out of Events key under
  Serverless::Function
  # Find out more about other implicit resources you can reference within SAM
  # https://github.com/awslabs/serverless-application-model/blob/master/docs/internals/generated_resources.rst#api
  HelloWorldApi:
    Description: API Gateway endpoint URL for Prod stage for Hello World function
    Value: !Sub "https://${ServerlessRestApi}.execute-api.${AWS::Region}.amazonaws.com/Prod/hello/"
  HelloWorldFunction:
    Description: Hello World Lambda Function ARN

```

```
Value: !GetAtt HelloWorldFunction.Arn
HelloWorldFunctionIamRole:
  Description: Implicit IAM Role created for Hello World function
  Value: !GetAtt HelloWorldFunctionRole.Arn
```

Langkah selanjutnya

Setelah mengonfigurasi CloudWatch Application Insights, gunakan `sam build` untuk membangun aplikasi Anda dan `sam deploy` untuk menyebarkan aplikasi Anda. Semua sumber daya yang didukung CloudWatch Application Insights akan dikonfigurasi untuk pemantauan.

- Untuk daftar sumber daya yang didukung, lihat [Log dan metrik yang didukung](#) di Panduan CloudWatch Pengguna Amazon.
- Untuk mempelajari cara mengakses Wawasan CloudWatch Aplikasi, lihat [Mengakses Wawasan CloudWatch Aplikasi](#) di CloudWatch Panduan Pengguna Amazon.

Cara menggunakan log

Untuk menyederhanakan pemecahan masalah, AWS SAM CLI memiliki perintah yang disebut. [sam logs](#) Perintah ini memungkinkan Anda untuk mengambil log yang dibuat oleh fungsi Lambda Anda dari baris perintah.

Note

Perintah `sam logs` ini bekerja untuk semua fungsi AWS Lambda, bukan hanya untuk yang sedang Anda deploy menggunakan AWS SAM.

Mengambil log dari tumpukan AWS CloudFormation

Ketika fungsi Anda adalah bagian dari sebuah tumpukan AWS CloudFormation, Anda dapat mengambil log dengan menggunakan ID logis fungsi:

```
sam logs -n HelloWorldFunction --stack-name mystack
```

Mengambil log dari nama fungsi Lambda

Atau, Anda dapat mengambil log dengan menggunakan nama fungsi:

```
sam logs -n mystack-HelloWorldFunction-1FJ8PD
```

Menunggu log

Tambahkan pilihan `--tail` untuk menunggu log baru dan lihat saat log tersebut masuk. Hal ini sangat membantu selama proses deployment atau saat Anda memecahkan masalah produksi.

```
sam logs -n HelloWorldFunction --stack-name mystack --tail
```

Melihat log pada rentang waktu tertentu

Anda dapat melihat log dalam rentang waktu tertentu dengan menggunakan pilihan `-s` dan `-e`:

```
sam logs -n HelloWorldFunction --stack-name mystack -s '10min ago' -e '2min ago'
```

Memfilter log

Gunakan pilihan `--filter` untuk menemukan log yang cocok dengan syarat, frasa, atau nilai dalam log acara Anda dengan cepat:

```
sam logs -n HelloWorldFunction --stack-name mystack --filter "error"
```

Dalam output, AWS SAM CLI menggarisbawahi semua kemunculan kata “kesalahan” sehingga Anda dapat dengan mudah menemukan kata kunci filter dalam output log.

Menyoroti error

Ketika fungsi Lambda Anda macet atau habis waktu, akan AWS SAM CLI menyoroti pesan batas waktu berwarna merah. Hal ini membantu Anda menemukan eksekusi tertentu yang habis waktu dalam pengaliran raksasa log output dengan mudah.

Pencetakan indah JSON

Jika pesan log Anda mencetak string JSON, AWS SAM CLI secara otomatis cukup mencetak JSON untuk membantu Anda mengurai dan memahami JSON secara visual.

Menerbitkan aplikasi tanpa server menggunakan AWS SAMCLI

Untuk membuat AWS SAM aplikasi Anda tersedia bagi orang lain untuk menemukan dan menyebarkan, Anda dapat menggunakan AWS SAM CLI untuk mempublikasikannya ke AWS Serverless Application Repository. Untuk mempublikasikan aplikasi Anda menggunakan AWS SAMCLI, Anda harus mendefinisikannya menggunakan AWS SAM template. Anda juga harus sudah mengujinya secara lokal atau di AWS Cloud.

Ikuti instruksi pada topik ini untuk membuat aplikasi baru, membuat versi baru dari aplikasi yang ada, atau memperbarui metadata aplikasi yang ada. (Apa yang Anda lakukan tergantung pada apakah aplikasi sudah ada di AWS Serverless Application Repository, dan apakah metadata aplikasi berubah.) Untuk informasi selengkapnya tentang metadata aplikasi, lihat [Properti bagian Metadata templat AWS SAM](#).

Prasyarat

Sebelum Anda mempublikasikan aplikasi untuk AWS Serverless Application Repository menggunakan AWS SAMCLI, Anda harus memiliki yang berikut:

- Yang AWS SAM CLI diinstal. Untuk informasi selengkapnya, lihat [Instalasi AWS SAMCLI](#). Untuk menentukan AWS SAM CLI apakah diinstal, jalankan perintah berikut:

```
sam --version
```

- Templat AWS SAM yang valid.
- Kode aplikasi Anda dan dependensi yang direferensikan oleh templat AWS SAM.
- Versi semantik, hanya diperlukan untuk membagikan aplikasi Anda secara publik. Nilai ini bisa hanya sebesar 1,0.
- URL yang memberi akses ke kode sumber aplikasi Anda.
- File README.md. File ini harus menjelaskan cara pelanggan dapat menggunakan aplikasi Anda dan cara mengonfigurasinya sebelum men-deploynya ke Akun AWS pelanggan sendiri.
- File LICENSE.txt, hanya diperlukan untuk membagikan aplikasi Anda secara publik.
- Jika aplikasi Anda mencakup aplikasi yang di-nest, Anda harus sudah memublikasikan aplikasi tersebut ke AWS Serverless Application Repository.

- Kebijakan valid bucket Amazon Simple Storage Service (Amazon S3) yang memberikan layanan izin baca untuk artefak yang Anda unggah ke Amazon S3 ketika Anda mengemas aplikasi Anda. Untuk menyiapkan kebijakan ini, lakukan hal berikut:
 1. Buka konsol Amazon S3 di <https://console.aws.amazon.com/s3/>.
 2. Pilih nama bucket Amazon S3 yang Anda gunakan untuk mengemas aplikasi Anda.
 3. Pilih Izin.
 4. Pada tab Izin, di bawah Kebijakan bucket, pilih Edit.
 5. Pada Halaman Edit kebijakan bucket, tempel pernyataan kebijakan berikut ke editor Kebijakan. Pada pernyataan kebijakan, pastikan untuk menggunakan nama bucket Anda di elemen Resource dan ID akun AWS di elemen Condition. Ekspresi pada elemen Condition memastikan bahwa AWS Serverless Application Repository memiliki izin untuk hanya mengakses aplikasi dari akun AWS yang ditentukan. Untuk informasi selengkapnya tentang pernyataan kebijakan, lihat [Referensi elemen kebijakan JSON IAM](#) di Panduan Pengguna IAM.

```
{
  "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. Pilih Simpan perubahan.

Memublikasikan aplikasi baru

Langkah 1: Tambahkan bagian **Metadata** ke templat AWS SAM

Pertama, tambahkan bagian Metadata ke templat AWS SAM. Berikan informasi aplikasi yang akan dipublikasikan ke AWS Serverless Application Repository.

Berikut ini adalah contoh bagian Metadata:

```
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
      ...
```

Untuk informasi selengkapnya tentang bagian Metadata dari templat AWS SAM, lihat [Properti bagian Metadata templat AWS SAM](#).

Langkah 2: Kemas aplikasi

Jalankan AWS SAM CLI perintah berikut, yang mengunggah artefak aplikasi ke Amazon S3 dan mengeluarkan file template baru yang disebut: `packaged.yaml`

```
sam package --output-template-file packaged.yaml --s3-bucket <your-bucket-name>
```

Anda menggunakan file templat `packaged.yaml` di langkah berikutnya untuk memublikasikan aplikasi ke AWS Serverless Application Repository. File ini mirip dengan file templat asli (`template.yaml`), tetapi memiliki perbedaan kunci— properti inti `CodeUri`, `LicenseUrl`, dan `ReadmeUrl` menunjuk pada bucket Amazon S3 dan objek yang mencakup artefaknya masing-masing.

Sinppet berikut dari contoh file templat `packaged.yaml` menunjukkan properti `CodeUri`:

```
MySampleFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: s3://bucketname/fb7d77a3647a4f47a352fc0bjectGUID
  ...
```

Langkah 3: Publikasikan aplikasi

Untuk memublikasikan versi pribadi AWS SAM aplikasi Anda ke AWS Serverless Application Repository, jalankan AWS SAM CLI perintah berikut:

```
sam publish --template packaged.yaml --region us-east-1
```

Output dari perintah `sam publish` memuat link ke aplikasi Anda pada AWS Serverless Application Repository. Anda juga dapat langsung membuka [Halaman arahan AWS Serverless Application Repository](#) dan mencari aplikasi Anda.

Langkah 4: Bagikan aplikasi (opsional)

Secara default, aplikasi Anda diatur pada pengaturan privat, sehingga tidak terlihat oleh akun AWS lain. Untuk berbagi aplikasi Anda dengan orang lain, Anda harus membuatnya publik atau memberikan izin untuk daftar Akun AWS tertentu.

Untuk informasi tentang berbagi aplikasi menggunakan AWS CLI, lihat [Contoh Kebijakan Berbasis Sumber Daya AWS Serverless Application Repository](#) di Panduan Developer AWS Serverless Application Repository. Untuk informasi tentang berbagi aplikasi menggunakan AWS Management Console, lihat [Berbagi Aplikasi](#) di Panduan Developer AWS Serverless Application Repository.

Memublikasikan versi baru aplikasi yang sudah ada

Setelah Anda memublikasikan aplikasi ke AWS Serverless Application Repository, Anda mungkin ingin memublikasikan versi baru dari aplikasi tersebut. Misalnya, Anda mungkin telah mengubah kode fungsi Lambda Anda atau menambahkan komponen baru ke arsitektur aplikasi Anda.

Untuk memperbarui aplikasi yang telah Anda publikasikan sebelumnya, Aplikasi dipublikasikan kembali menggunakan proses yang sama pada detail sebelumnya. Di bagian Metadata a file templat AWS SAM, berikan nama aplikasi yang sama dengan nama aplikasi yang sebelumnya Anda publikasikan, tetapi sertakan nilai `SemanticVersion` baru.

Misalnya, pertimbangkan aplikasi yang dipublikasikan dengan nama `SampleApp` dan `SemanticVersion` dari `1.0.0`. Untuk memperbarui aplikasi tersebut, templat AWS SAM harus memiliki nama aplikasi `SampleApp` dan `SemanticVersion` dari `1.0.1` (atau apa pun selain `1.0.0`).

Topik Tambahan

- [Properti bagian Metadata templat AWS SAM](#)

Properti bagian Metadata templat AWS SAM

`AWS::ServerlessRepo::Application` adalah kunci metadata yang dapat Anda gunakan untuk menentukan informasi aplikasi yang ingin Anda publikasikan ke AWS Serverless Application Repository.

Note

AWS CloudFormation [fungsi intrinsik](#) tidak didukung oleh kunci metadata `AWS::ServerlessRepo::Application`.

Properti

Tabel ini memberikan informasi tentang properti bagian Metadata dari templat AWS SAM. Bagian ini diperlukan untuk memublikasikan aplikasi untuk AWS Serverless Application Repository menggunakan AWS SAMCLI.

Properti	Tipe	Diperlukan	Deskripsi
Name	String	BENAR	Nama aplikasi. Panjang minimum=1. Panjang maksimum=140. Pola: "[a-zA-Z0-9\\-]+";
Description	String	BENAR	Deskripsi aplikasi. Panjang minimum=1. Panjang maksimum=256.
Author	String	BENAR	Nama penulis yang memublikasikan aplikasi. Panjang minimum=1. Panjang maksimum=127. Pola: "^([a-z0-9]([a-z0-9] -(?!-))*[a-z0-9])?\$";
SpdxLicenseId	String	SALAH	Pengidentifikasi lisensi yang valid. Untuk menampilkan daftar pengidentifikasi lisensi yang valid, lihat Daftar Lisensi SPDX pada situs web Software Package Data Exchange (SPDX).
LicenseUrl	String	SALAH	Referensi ke file lisensi lokal, atau tautan Amazon S3 ke file lisensi, yang cocok dengan nilai spdxLicenseId aplikasi Anda. File templat AWS SAM yang belum dikemas menggunakan perintah <code>sam package</code> dapat memiliki referensi ke file lokal untuk properti ini. Namun, untuk aplikasi yang akan dipublikasikan menggunakan perintah <code>sam publish</code> , properti ini harus menjadi referensi ke bucket Amazon S3. Ukuran maksimum: 5 MB. Anda harus memberikan nilai untuk properti ini agar dapat membuat aplikasi Anda publik. Perhatikan

Properti	Tipe	Diperlukan	Deskripsi
			bahwa Anda tidak dapat memperbarui properti ini setelah aplikasi Anda dipublikasikan. Jadi, untuk menambahkan lisensi ke aplikasi, Anda harus menghapusnya terlebih dahulu, atau memublikasikan aplikasi baru dengan nama yang berbeda.
ReadmeUrl	String	SALAH	<p>Referensi ke file readme lokal atau tautan Amazon S3 ke file readme yang berisi deskripsi lebih detail tentang aplikasi dan cara kerjanya.</p> <p>File templat AWS SAM yang belum dikemas menggunakan perintah <code>sam package</code> dapat memiliki referensi ke file lokal untuk properti ini. Namun, untuk dipublikasikan menggunakan perintah <code>sam publish</code>, properti ini harus menjadi referensi ke bucket Amazon S3.</p> <p>Ukuran maksimum: 5 MB.</p>
Labels	String	SALAH	<p>Label yang meningkatkan penemuan aplikasi dalam hasil pencarian.</p> <p>Panjang minimum=1. Panjang maksimum=127. Jumlah maksimum label: 10.</p> <p>Pola: <code>"^[a-zA-Z0-9+\\-_:\\/@]+\$"</code>;</p>
HomePageUrl	String	SALAH	URL dengan informasi lebih lanjut tentang aplikasi —misalnya, lokasi GitHub repositori Anda untuk aplikasi.
SemanticVersion	String	SALAH	<p>Versi semantik aplikasi. Untuk spesifikasi Versioning Semantik, lihat situs web Versioning Semantik.</p> <p>Anda harus memberikan nilai untuk properti ini agar dapat membuat aplikasi Anda publik.</p>

Properti	Tipe	Diperlukan	Deskripsi
SourceCodeUrl	String	SALAH	Tautan ke repositori publik untuk kode sumber aplikasi Anda.

Kasus penggunaan

Bagian ini mencantumkan kasus penggunaan untuk aplikasi pemublikasian, bersama dengan properti Metadata yang diproses untuk kasus penggunaan tersebut. Properti yang tidak dicantumkan untuk kasus penggunaan yang diberikan diabaikan.

- Membuat aplikasi baru – Aplikasi baru dibuat jika tidak ada aplikasi di AWS Serverless Application Repository dengan nama yang cocok untuk sebuah akun.
 - Name
 - SpdxLicenseId
 - LicenseUrl
 - Description
 - Author
 - ReadmeUrl
 - Labels
 - HomePageUrl
 - SourceCodeUrl
 - SemanticVersion
 - Isi dari templat AWS SAM (misalnya, setiap sumber peristiwa, sumber daya, dan kode fungsi Lambda)
- Membuat versi aplikasi — Versi aplikasi dibuat jika sudah ada aplikasi di AWS Serverless Application Repository dengan nama yang cocok untuk akun dan SemanticVersion sedang berubah.
 - Description
 - Author

- `ReadmeUrl`
 - `Labels`
 - `HomePageUrl`
 - `SourceCodeUrl`
 - `SemanticVersion`
 - Isi dari templat AWS SAM (misalnya, setiap sumber peristiwa, sumber daya, dan kode fungsi Lambda)
- Memperbarui aplikasi — Aplikasi diperbarui jika sudah ada aplikasi di AWS Serverless Application Repository dengan nama yang cocok untuk akun dan `SemanticVersion` tidak berubah.
- `Description`
 - `Author`
 - `ReadmeUrl`
 - `Labels`
 - `HomePageUrl`

Contoh

Berikut ini adalah contoh bagian Metadata:

```
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
```

Bekerja dengan layanan pihak ketiga

Bagian ini menyediakan dokumentasi tentang penggunaan AWS Serverless Application Model (AWS SAM) dengan layanan pihak ketiga.

Topik

- [AWS SAMCLITerraformdukungan](#)

AWS SAMCLITerraformdukungan

Bagian ini mencakup penggunaan AWS Serverless Application Model Command Line Interface (AWS SAMCLI) dengan Terraform proyek dan Terraform Cloud Anda.

Untuk memberikan umpan balik dan mengirimkan permintaan fitur, buat [GitHubMasalah](#).

Topik

- [Untuk apa AWS SAMCLI dukungan ituTerraform?](#)
- [Memulai dengan Terraform dukungan untuk AWS SAMCLI](#)
- [Menggunakan AWS SAMCLI with Terraform untuk debugging dan pengujian lokal](#)
- [Menggunakan AWS SAMCLI with ServerLess.tf untuk debugging dan pengujian lokal](#)
- [AWS SAMCLIdengan Terraform referensi](#)

Untuk apa AWS SAMCLI dukungan ituTerraform?

Gunakan AWS Serverless Application Model Command Line Interface (AWS SAMCLI) dengan Terraform proyek Anda atau Terraform Cloud untuk melakukan debugging lokal dan pengujian:

- AWS Lambdafungsi dan lapisan.
- API Gateway API Amazon.

Untuk pengantarTerraform, lihat [Apa ituTerraform?](#) di situs HashiCorpTerraformweb.

Untuk memberikan umpan balik dan mengirimkan permintaan fitur, buat [GitHubMasalah](#).

Topik

- [Apa itu AWS SAMCLI?](#)
- [Bagaimana cara menggunakan AWS SAMCLI dengan Terraform?](#)
- [Langkah selanjutnya](#)

Apa itu AWS SAMCLI?

AWS SAMCLI ini adalah alat baris perintah yang dapat Anda gunakan dengan AWS SAM templat dan integrasi pihak ketiga yang didukung, seperti Terraform, untuk membangun dan menjalankan aplikasi tanpa server Anda. Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#).

AWS SAMCLI mendukung perintah berikut untuk Terraform:

- `sam local invoke`— Memulai pemanggilan satu kali dari sumber daya fungsi secara lokal. AWS Lambda Untuk mempelajari lebih lanjut tentang perintah ini, lihat [Menggunakan sam local invoke](#).
- `sam local start-api`— Jalankan sumber daya Lambda Anda secara lokal dan uji melalui host server HTTP lokal. Jenis pengujian ini berguna untuk fungsi Lambda yang dipanggil oleh titik akhir API Gateway. Untuk mempelajari lebih lanjut tentang perintah ini, lihat [Menggunakan sam local start-api](#).
- `sam local start-lambda`— Mulai titik akhir lokal untuk fungsi Lambda Anda untuk menjalankan fungsi Anda secara lokal AWS Command Line Interface menggunakan AWS CLI () atau SDK. Untuk mempelajari lebih lanjut tentang perintah ini, lihat [Menggunakan sam local start-lambda](#).

Bagaimana cara menggunakan AWS SAMCLI dengan Terraform?

[Terraform Alur kerja inti](#) terdiri dari tiga tahap: Tulis, Rencana, dan Terapkan. Dengan AWS SAM CLI dukungan untuk Terraform, Anda dapat memanfaatkan AWS SAM CLI `sam local` serangkaian perintah sambil terus menggunakan Terraform alur kerja Anda untuk mengelola aplikasi Anda. AWS Secara umum, ini berarti sebagai berikut:

- Tulis — Buat infrastruktur Anda sebagai kode menggunakan Terraform.
- Uji dan debug — Gunakan AWS SAM CLI untuk menguji dan men-debug aplikasi Anda secara lokal.
- Rencana - Pratinjau perubahan sebelum mendaftar.
- Terapkan — Menyediakan infrastruktur Anda.

Untuk contoh menggunakan AWS SAM CLI with Terraform, lihat [Better together: AWS SAM CLI dan HashiCorp Terraform](#) di AWS Compute Blog.

Langkah selanjutnya

Untuk menyelesaikan semua prasyarat dan mengatur, lihat. Terraform [Memulai dengan Terraform dukungan untuk AWS SAM CLI](#)

Memulai dengan Terraform dukungan untuk AWS SAM CLI

Topik ini mencakup cara memulai dengan menggunakan AWS Serverless Application Model Command Line Interface (AWS SAM CLI) dengan Terraform.

Untuk memberikan umpan balik dan mengirimkan permintaan fitur, buat [GitHub Masalah](#).

Topik

- [AWS SAM CLI Terraform prasyarat](#)
- [Menggunakan AWS SAM CLI perintah dengan Terraform](#)
- [Siapkan untuk Terraform proyek](#)
- [Siapkan untuk Terraform Cloud](#)

AWS SAM CLI Terraform prasyarat

Lengkapi semua prasyarat untuk mulai menggunakan dengan proyek Anda. AWS SAM CLI Terraform

1. Instal atau tingkatkan AWS SAM CLI

Untuk memeriksa apakah Anda telah AWS SAM CLI menginstal, jalankan yang berikut ini:

```
$ sam --version
```

Jika AWS SAM CLI sudah diinstal, output akan menampilkan versi. Untuk meng-upgrade ke versi terbaru, lihat [Memutakhirkan AWS SAM CLI](#).

Untuk petunjuk tentang menginstal AWS SAM CLI bersama dengan semua prasyaratnya, lihat. [Instalasi AWS SAM CLI](#)

2. Instal Terraform

Untuk memeriksa apakah Anda telah Terraform menginstal, jalankan yang berikut ini:

```
$ terraform -version
```

Untuk menginstal Terraform, lihat [Instal Terraform](#) di Terraformregistri.

3. Instal Docker untuk pengujian lokal

AWS SAMCLIDockerKebutuhan untuk pengujian lokal. Untuk menginstal Docker, lihat [Menginstal Docker untuk digunakan dengan AWS SAMCLI](#).

Menggunakan AWS SAMCLI perintah dengan Terraform

Saat Anda menjalankan AWS SAM CLI perintah yang didukung, gunakan `--hook-name` opsi dan berikan `terraform` nilainya. Berikut ini adalah contohnya:

```
$ sam local invoke --hook-name terraform
```

Anda dapat mengonfigurasi opsi ini di file AWS SAM CLI konfigurasi Anda dengan yang berikut:

```
hook_name = "terraform"
```

Siapkan untuk Terraform proyek

Lengkapi langkah-langkah dalam topik ini untuk menggunakan Terraform proyek AWS SAM CLI with.

Tidak diperlukan pengaturan tambahan jika Anda membangun AWS Lambda artefak di luar Terraform proyek Anda. Lihat [Menggunakan AWS SAMCLI with Terraform untuk debugging dan pengujian lokal](#) untuk mulai menggunakan AWS SAMCLI.

Jika Anda membangun artefak Lambda Anda dalam Terraform proyek Anda, Anda harus melakukan hal berikut:

1. Instal Python 3.6 atau yang lebih baru
2. Instal Make alat.
3. Tentukan artefak Lambda membangun logika dalam proyek Anda. Terraform

4. Tentukan `sam` metadata sumber daya untuk menginformasikan logika build Anda. AWS SAM CLI
5. Gunakan AWS SAM CLI `sam build` perintah untuk membangun artefak Lambda Anda.

Instal Python 3.6 atau yang lebih baru

Python 3.6 atau yang lebih baru diperlukan untuk digunakan dengan AWS SAM CLI. Saat Anda menjalankan `sam build`, AWS SAM CLI kreasi `makefiles` yang berisi Python perintah untuk membangun artefak Lambda Anda.

Untuk petunjuk penginstalan, lihat [Mengunduh Python di Panduan Pemula Python](#).

Verifikasi bahwa Python 3.6 atau yang lebih baru ditambahkan ke jalur mesin Anda dengan menjalankan:

```
$ python --version
```

Output harus menampilkan versi Python yang 3.6 atau lebih baru.

Instal Make alat

GNU [Make](#) adalah alat yang mengontrol pembuatan file yang dapat dieksekusi dan file non-sumber lainnya untuk proyek Anda. AWS SAM CLI Ciptaan `makefiles` yang mengandalkan alat ini untuk membangun artefak Lambda Anda.

Jika Anda belum Make menginstal pada mesin lokal Anda, instal sebelum bergerak maju.

Untuk Windows, Anda dapat menginstal menggunakan [Chocolatey](#). Untuk petunjuk, lihat [Menggunakan Chocolatey](#) di Cara Menginstal dan Menggunakan “Make” di Windows

Tentukan logika membangun artefak Lambda

Gunakan jenis `null_resource` Terraform sumber daya untuk menentukan logika build Lambda Anda. Berikut ini adalah contoh yang menggunakan skrip build khusus untuk membangun fungsi Lambda.

```
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"
    }
}
}

```

Tentukan sumber sam metadata daya

sam metadataSumber daya adalah jenis `null_resource` Terraform sumber daya yang menyediakan informasi AWS SAM CLI yang dibutuhkan untuk menemukan artefak Lambda Anda. sam metadataSumber daya unik diperlukan untuk setiap fungsi atau lapisan Lambda dalam proyek Anda. Untuk mempelajari lebih lanjut tentang jenis sumber daya ini, lihat [null_resource](#) di registri.

Terraform

Untuk mendefinisikan sumber sam metadata daya

1. Beri nama sumber daya Anda mulai dengan `sam_metadata_` untuk mengidentifikasi sumber daya sebagai sam metadata sumber daya.
2. Tentukan properti artefak Lambda Anda di dalam `triggers` blok sumber daya Anda.
3. Tentukan `null_resource` yang berisi logika build Lambda Anda dengan argumen `depends_on`

Berikut ini adalah contoh template:

```

resource "null_resource" "sam_metadata_..." {
    triggers = {
        resource_name = resource_name
        resource_type = resource_type
        original_source_code = original_source_code
        built_output_path = built_output_path
    }
    depends_on = [
        null_resource.build_lambda_function # ref to your build logic
    ]
}

```

Berikut ini adalah sumber sam metadata daya contoh:

```
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
  ]
}
```

Isi `sam` metadata sumber daya Anda akan bervariasi berdasarkan jenis sumber daya Lambda (fungsi atau lapisan), dan jenis kemasan (ZIP atau gambar). Untuk informasi lebih lanjut, bersama dengan contoh, lihat [sumber daya metadata sam](#).

Ketika Anda mengkonfigurasi `sam` metadata sumber daya dan menggunakan AWS SAM CLI perintah yang didukung, AWS SAM CLI akan menghasilkan file metadata sebelum menjalankan perintah. AWS SAM CLI Setelah Anda membuat file ini, Anda dapat menggunakan `--skip-prepare-infra` opsi dengan AWS SAM CLI perintah `future` untuk melewati proses pembuatan metadata dan menghemat waktu. Opsi ini hanya boleh digunakan jika Anda belum membuat perubahan infrastruktur apa pun, seperti membuat fungsi Lambda baru atau titik akhir API baru.

Gunakan AWS SAMCLI untuk membangun artefak Lambda Anda

Gunakan AWS SAM CLI `sam build` perintah untuk membangun artefak Lambda Anda. Ketika Anda menjalankansam `build`, AWS SAM CLI melakukan hal berikut:

1. Mencari `sam` metadata sumber daya dalam Terraform proyek Anda untuk mempelajari dan menemukan sumber daya Lambda Anda.
2. Memulai logika build Lambda Anda untuk membangun artefak Lambda Anda.
3. Membuat `.aws-sam` direktori yang mengatur Terraform proyek Anda untuk digunakan dengan AWS SAM CLI `sam local` perintah.

Untuk membangun dengan `sam build`

1. Dari direktori yang berisi modul Terraform root Anda, jalankan yang berikut ini:


```
$ sam build --hook-name terraform
```

2. Untuk membangun fungsi atau layer Lambda tertentu, jalankan yang berikut

```
$ sam build --hook-name terraform lambda-resource-id
```

ID sumber daya Lambda dapat berupa nama fungsi Lambda atau alamat Terraform sumber daya lengkap, seperti atau `aws_lambda_function.list_books module.list_book_function.aws_lambda_function.this[0]`

Jika kode sumber fungsi Anda atau file Terraform konfigurasi lainnya terletak di luar direktori yang berisi modul Terraform root Anda, Anda perlu menentukan lokasi. Gunakan `--terraform-project-root-path` opsi untuk menentukan jalur absolut atau relatif ke direktori tingkat atas yang berisi file-file ini. Berikut ini adalah contohnya:

```
$ sam build --hook-name terraform --terraform-project-root-path ~/projects/terraform/demo
```

Bangun menggunakan wadah

Saat menjalankan AWS SAM CLI `sam build` perintah, Anda dapat mengonfigurasi AWS SAM CLI untuk membangun aplikasi Anda menggunakan Docker wadah lokal.

Note

Anda harus Docker menginstal dan mengkonfigurasi. Untuk petunjuk, lihat [Menginstal Docker untuk digunakan dengan AWS SAMCLI](#).

Untuk membangun menggunakan wadah

1. Buat `Dockerfile` yang berisi Terraform, Python, dan Make alat. Anda juga harus menyertakan runtime fungsi Lambda Anda.

Berikut ini adalah contoh `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"]
```

- Gunakan [docker build](#) untuk membangun Docker citra Anda.

Berikut ini adalah contohnya:

```
$ docker build --tag terraform-build:v1 <path-to-directory-containing-Dockerfile>
```

- Jalankan AWS SAM CLI `sam build` perintah dengan `--build-image` opsi `--use-container` dan.

Berikut ini adalah contohnya:

```
$ sam build --use-container --build-image terraform-build:v1
```

Langkah selanjutnya

Untuk mulai menggunakan AWS SAM CLI dengan Terraform proyek Anda, lihat [Menggunakan AWS SAMCLI with Terraform untuk debugging dan pengujian lokal](#).

Siapkan untuk Terraform Cloud

Kami menyarankan Anda menggunakan Terraform v1.6.0 atau yang lebih baru. Jika Anda menggunakan versi yang lebih lama, Anda harus membuat file Terraform paket secara lokal. File rencana lokal menyediakan informasi AWS SAM CLI yang dibutuhkan untuk melakukan pengujian dan debugging lokal.

Untuk menghasilkan file paket lokal

Note

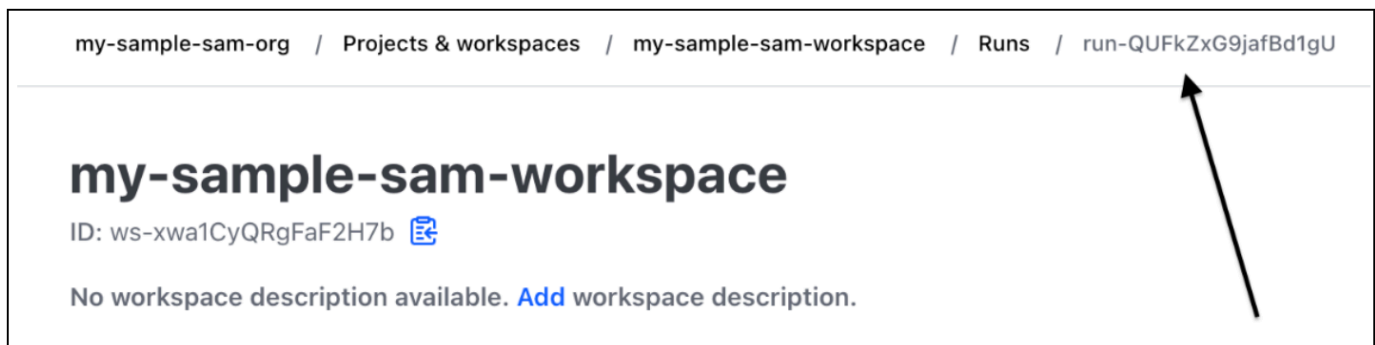
Langkah-langkah ini tidak diperlukan untuk Terraform v1.6.0 atau yang lebih baru. Untuk mulai menggunakan AWS SAM CLI with Terraform Cloud, lihat [Menggunakan AWS SAM CLI dengan Terraform](#).

1. Konfigurasi token API — Jenis token akan tergantung pada tingkat akses Anda. Untuk mempelajari selengkapnya, lihat [Token API](#) dalam Terraform Cloud dokumentasi.
2. Tetapkan variabel lingkungan token API Anda - Berikut ini adalah contoh dari baris perintah:

```
$ export TOKEN="<api-token-value>"
```

3. Dapatkan ID run Anda — Dari Terraform Cloud konsol, cari ID Terraform run untuk proses yang ingin Anda gunakan dengan AWS SAMCLI.

ID run terletak di jalur breadcrumb run Anda.



4. Ambil file paket — Menggunakan token API Anda, dapatkan file paket lokal Anda. Berikut ini adalah contoh dari baris perintah:

```
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
```

Anda sekarang siap untuk menggunakan AWS SAM CLI dengan Terraform Cloud. Saat menggunakan AWS SAM CLI perintah yang didukung, gunakan `--terraform-plan-file` opsi untuk menentukan nama dan jalur file paket lokal Anda. Berikut ini adalah contohnya:

```
$ sam local invoke --hook-name terraform --terraform-plan-file custom-plan.json
```

Berikut ini adalah contoh, menggunakan `sam local start-api` perintah:

```
$ sam local start-api --hook-name terraform --terraform-plan-file custom-plan.json
```

Untuk contoh aplikasi yang dapat Anda gunakan dengan contoh ini, lihat [api_gateway_v2_tf_cloud](#) di repositori `aws-samples`. GitHub

Langkah selanjutnya

Untuk mulai menggunakan AWS SAM CLI with Terraform Cloud, lihat [Menggunakan AWS SAM CLI with Terraform untuk debugging dan pengujian lokal](#).

Menggunakan AWS SAM CLI with Terraform untuk debugging dan pengujian lokal

Topik ini mencakup cara menggunakan AWS Serverless Application Model perintah Command Line Interface (AWS SAM CLI) yang didukung dengan Terraform proyek Anda dan Terraform Cloud.

Untuk memberikan umpan balik dan mengirimkan permintaan fitur, buat [GitHub Masalah](#).

Topik

- [Pengujian lokal dengan `sam local invoke`](#)

- [Penguujian lokal dengan sam local start-api](#)
- [Penguujian lokal dengan sam local start-lambda](#)
- [Batasan Terraform](#)

Penguujian lokal dengan sam local invoke

Note

Untuk menggunakan AWS SAM CLI to test secara lokal, Anda harus memiliki Docker intalled dan dikonfigurasi. Untuk petunjuk, lihat [Menginstal Docker untuk digunakan dengan AWS SAMCLI](#).

Berikut ini adalah contoh penguujian fungsi Lambda Anda secara lokal dengan meneruskan suatu peristiwa:

```
$ sam local invoke --hook-name terraform hello_world_function -e events/event.json -
```

Untuk mempelajari lebih lanjut tentang menggunakan perintah ini, lihat [Menggunakan sam local invoke](#).

Penguujian lokal dengan sam local start-api

Untuk menggunakannya sam local start-api Terraform, jalankan yang berikut ini:

```
$ sam local start-api --hook-name terraform
```

Berikut ini adalah contohnya:

```
$ 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)
```

Untuk mempelajari lebih lanjut tentang perintah ini, lihat [Menggunakan sam local start-api](#).

Fungsi Lambda yang menggunakan otorisasi Lambda

Untuk fungsi Lambda yang dikonfigurasi untuk menggunakan otorisasi Lambda, Lambda akan AWS SAM CLI secara otomatis memanggil otorisasi Lambda Anda sebelum menjalankan titik akhir fungsi Lambda Anda.

- Untuk mempelajari lebih lanjut tentang fitur ini di AWS SAMCLI, lihat [Fungsi Lambda yang menggunakan otorisasi Lambda](#).
- Untuk informasi selengkapnya tentang penggunaan otorisasi Lambda di Terraform, lihat [Resource: aws_api_gateway_authorizer](#) di registri Terraform

Pengujian lokal dengan sam local start-lambda

Berikut ini adalah contoh pengujian fungsi Lambda Anda secara lokal dengan (): AWS Command Line Interface AWS CLI

1. Gunakan AWS SAM CLI untuk membuat lingkungan pengujian lokal:

```
$ sam local start-lambda --hook-name terraform hello_world_function
```

2. Gunakan tombol AWS CLI untuk menjalankan fungsi Anda secara lokal:

```
$ 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
```

Untuk mempelajari lebih lanjut tentang perintah ini, lihat [Menggunakan sam local start-lambda](#).

Batasan Terraform

Berikut ini adalah batasan saat menggunakan AWS SAM CLI with Terraform:

- Fungsi Lambda ditautkan ke beberapa lapisan.
- Terraform variabel lokal yang mendefinisikan hubungan antar sumber daya.
- Mereferensikan fungsi Lambda yang belum dibuat. Ini termasuk fungsi yang didefinisikan dalam atribut body dari sumber daya REST API.

Untuk menghindari keterbatasan ini, Anda dapat menjalankan `terraform apply` ketika sumber daya baru ditambahkan.

Menggunakan AWS SAMCLI with ServerLess.tf untuk debugging dan pengujian lokal

AWS Serverless Application Model Command Line Interface (AWS SAMCLI) dapat digunakan dengan modul ServerLess.tf untuk debugging lokal dan pengujian fungsi dan lapisan Anda. AWS Lambda AWS SAMCLI Perintah berikut didukung:

- `sam build`
- `sam local invoke`
- `sam local start-api`
- `sam local start-lambda`

Note

ServerLess.tf versi 4.6.0 dan yang lebih baru mendukung integrasi. AWS SAM CLI

Untuk mulai menggunakan modul AWS SAM CLI dengan ServerLess.tf Anda, perbarui ke versi terbaru ServerLess.tf dan file. AWS SAM CLI

Mulai dari serverless.tf versi 6.0.0, Anda harus mengatur parameter sebagai.

`create_sam_metadata true` Ini menghasilkan sumber daya metadata yang dibutuhkan AWS SAM CLI `sam build` perintah.

Untuk mempelajari selengkapnya tentang Serverless.tf, lihat [terraform-aws-lambda-module](#).

AWS SAMCLIdengan Terraform referensi

Bagian ini adalah referensi untuk menggunakan AWS Serverless Application Model Command Line Interface (AWS SAMCLI) dengan Terraform untuk debugging dan pengujian lokal.

Untuk memberikan umpan balik dan mengirimkan permintaan fitur, buat [GitHubMasalah](#).

AWS SAMreferensi fitur yang didukung

Dokumentasi referensi untuk AWS SAM CLI fitur yang didukung untuk digunakan dengan Terraform dapat ditemukan di sini:

- [sam build](#)
- [sam local invoke](#)
- [sam local start-api](#)
- [sam local start-lambda](#)

Terraformreferensi khusus

Dokumentasi referensi khusus untuk digunakan AWS SAM CLI dengan Terraform dapat ditemukan di sini:

- [sumber daya metadata sam](#)

sumber daya metadata sam

Halaman ini berisi informasi referensi untuk jenis sam metadata resource sumber daya yang digunakan dengan Terraform proyek.

- Untuk pengenalan menggunakan AWS Serverless Application Model Command Line Interface (AWS SAMCLI) denganTerraform, lihat[Untuk apa AWS SAMCLI dukungan ituTerraform?](#).
- Untuk menggunakan AWS SAM CLI denganTerraform, lihat[Menggunakan AWS SAMCLI with Terraform untuk debugging dan pengujian lokal.](#)

Topik

- [Argumen](#)
- [Contoh-contoh](#)

Argumen

Pendapat	Deskripsi
built_output_path	Jalan menuju artefak yang dibangun AWS Lambda fungsi Anda.

Pendapat	Deskripsi
<code>docker_build_args</code>	String decoded dari objek JSON argumen build Docker. Argumen ini opsional.
<code>docker_context</code>	Jalur ke direktori yang berisi konteks build image Docker.
<code>docker_file</code>	Jalur ke file Docker. Jalan ini relatif terhadap <code>docker_context</code> jalan. Argumen ini opsional. Nilai default-nya adalah <code>Dockerfile</code> .
<code>docker_tag</code>	Nilai tag gambar Docker yang dibuat. Nilai ini bersifat opsional.
<code>depends_on</code>	Jalur ke sumber daya bangunan untuk fungsi atau lapisan Lambda Anda. Untuk mempelajari lebih lanjut, lihat depends_onArgumen dalam Terraformregistri.
<code>original_source_code</code>	Jalur ke tempat fungsi Lambda Anda ditentukan. Nilai ini bisa berupa string, array string, atau objek JSON decoded sebagai string. <ul style="list-style-type: none"> • Untuk array string, hanya nilai pertama yang digunakan karena beberapa jalur kode tidak didukung. • Untuk objek JSON, <code>source_code_property</code> harus juga didefinisikan.
<code>resource_name</code>	Nama fungsi Lambda.
<code>resource_type</code>	Format jenis paket fungsi Lambda Anda. Nilai yang diterima adalah: <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>	Jalur ke kode sumber daya Lambda di objek JSON. Tentukan properti ini ketika <code>original_source_code</code> adalah objek JSON.

Contoh-contoh

sumber daya metadata sam yang mereferensikan fungsi Lambda menggunakan jenis paket ZIP

```
# 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
  ]
}
```

sumber daya metadata sam yang mereferensikan fungsi Lambda menggunakan jenis paket gambar

```
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"
  }
}
```

sumber daya metadata sam yang mereferensikan lapisan Lambda

```
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]
}
```

Contoh aplikasi nirserver

Contoh berikut menunjukkan cara mengunduh, menguji, dan men-deploy sejumlah aplikasi nirserver tambahan—termasuk cara mengonfigurasi sumber peristiwa dan sumber daya AWS.

Topik

- [Proses peristiwa DynamoDB](#)
- [Proses peristiwa Amazon S3](#)

Proses peristiwa DynamoDB

Dengan aplikasi contoh ini, Anda membangun hal yang Anda pelajari dalam gambaran umum dan panduan Quick Start, serta menginstal aplikasi contoh lain. Aplikasi ini terdiri dari fungsi Lambda yang dipanggil oleh sumber peristiwa tabel DynamoDB. Fungsi Lambda sangat sederhana, fungsi Lambda mencatat data yang diteruskan melalui pesan sumber peristiwa.

Latihan ini menunjukkan cara meniru pesan sumber peristiwa yang diteruskan ke fungsi Lambda saat dipanggil.

Sebelum Anda memulai

Pastikan bahwa Anda telah menyelesaikan penyiapan yang diperlukan di dalam [Instalasi AWS SAMCLI](#).

Langkah 1: Inisialisasi aplikasi

Di bagian ini, Anda mengunduh paket aplikasi, yang terdiri dari templat AWS SAM dan kode aplikasi.

Untuk menginisialisasi aplikasi

1. Jalankan perintah berikut pada AWS SAM CLI command prompt.

```
sam init \  
--location gh:aws-samples/cookiecutter-aws-sam-dynamodb-python \  
--no-input
```

Perhatikan bahwa `gh:` dalam perintah di atas akan diperluas ke GitHub url `https://github.com/`.

2. Tinjau konten direktori yang dibuat oleh perintah (`dynamodb_event_reader/`):
 - `template.yaml` – Mendefinisikan dua sumber daya AWS yang dibutuhkan aplikasi Read DynamoDB: fungsi Lambda dan tabel DynamoDB. Templat juga menentukan pemetaan antara dua sumber daya.
 - Direktori `read_dynamodb_event/` - Berisi kode aplikasi DynamoDB.

Langkah 2: Uji aplikasi secara lokal

Untuk pengujian lokal, gunakan AWS SAM CLI untuk menghasilkan contoh peristiwa DynamoDB dan menjalankan fungsi Lambda:

```
sam local generate-event dynamodb update | sam local invoke --event - ReadDynamoDBEvent
```

Perintah `generate-event` membuat pesan sumber peristiwa pengujian seperti pesan yang dibuat saat semua komponen di-deploy ke Cloud AWS. Pesan sumber peristiwa ini disalurkan ke fungsi Lambda `ReadDynamoDbEvent`.

Verifikasi bahwa pesan yang diharapkan, dicetak ke konsol tersebut, berdasarkan kode sumber di `app.py`.

Langkah 3: Buat paket aplikasi

Setelah menguji aplikasi Anda secara lokal, Anda menggunakan AWS SAM CLI untuk membuat paket penyebaran, yang Anda gunakan untuk menyebarkan aplikasi ke Cloud. AWS

Untuk membuat paket deployment Lambda

1. Buat bucket S3 di lokasi tempat Anda ingin menyimpan kode yang dipaketkan. Jika Anda ingin menggunakan bucket S3 yang sudah ada, lewati langkah ini.

```
aws s3 mb s3://bucketname
```

2. Buat paket deployment dengan menjalankan perintah CLI package berikut pada prompt perintah.

```
sam package \  
  --template-file template.yaml \  
  --output-template-file packaged.yaml \  
  --s3-bucket bucketname
```

Anda tentukan file templat baru, `packaged.yaml`, saat Anda men-deploy aplikasi di langkah berikutnya.

Langkah 4: Deploy aplikasi

Sekarang setelah Anda membuat paket deployment, Anda gunakan paket tersebut untuk men-deploy aplikasi ke Cloud AWS. Lalu Anda uji aplikasi tersebut.

Untuk men-deploy aplikasi nirserver ke Cloud AWS

- Dalam AWS SAMCLI, gunakan perintah `deploy` CLI untuk menyebarkan semua sumber daya yang Anda tentukan dalam template.

```
sam deploy \  
  --template-file packaged.yaml \  
  --stack-name sam-app \  
  --capabilities CAPABILITY_IAM \  
  --region us-east-1
```

Dalam perintah tersebut, parameter `--capabilities` memungkinkan AWS CloudFormation untuk membuat IAM role.

AWS CloudFormation membuat sumber daya AWS yang ditentukan di dalam templat. Anda dapat mengakses nama sumber daya ini di dalam konsol AWS CloudFormation.

Untuk menguji aplikasi nirserver di dalam Cloud AWS

1. Buka konsol DynamoDB.
2. Masukkan catatan ke dalam tabel yang baru saja Anda buat.
3. Buka tab Metrik tabel, dan pilih Lihat semua CloudWatch metrik. Di CloudWatch konsol, pilih Log untuk dapat melihat output log.

Langkah selanjutnya

AWS SAM GitHub Repositori berisi contoh aplikasi tambahan bagi Anda untuk men-download dan bereksperimen dengan. Untuk mengakses repositori ini, lihat [contoh aplikasi AWS SAM](#).

Proses peristiwa Amazon S3

Dengan aplikasi contoh ini, Anda membangun hal yang sudah Anda pelajari di contoh sebelumnya, dan memasang aplikasi yang lebih kompleks. Aplikasi ini terdiri dari fungsi Lambda yang dipanggil oleh sumber peristiwa unggah objek Amazon S3. Latihan ini menunjukkan cara mengakses sumber daya AWS dan membuat layanan AWS menjalankan panggilan melalui fungsi Lambda.

Contoh aplikasi nirserver ini memproses peristiwa pembuatan-objek di Amazon S3. Untuk setiap citra yang diunggah ke bucket, Amazon S3 mendeteksi peristiwa yang dibuat oleh objek dan memanggil fungsi Lambda. Fungsi Lambda memanggil Amazon Rekognition untuk mendeteksi teks yang ada di dalam citra. Fungsi Lambda lalu menyimpan hasil yang dikembalikan oleh Amazon Rekognition dalam tabel DynamoDB.

Note

Dengan aplikasi contoh ini, Anda melakukan langkah-langkah dengan urutan yang sedikit berbeda daripada contoh sebelumnya. Alasannya adalah karena contoh ini mengharuskan sumber daya AWS dibuat dan izin IAM dikonfigurasi sebelum Anda dapat menguji fungsi Lambda secara lokal. Kami akan memanfaatkan AWS CloudFormation untuk membuat sumber daya dan mengonfigurasi izin untuk Anda. Jika tidak, Anda perlu melakukan tindakan ini secara manual sebelum Anda dapat menguji fungsi Lambda lokal.

Karena contoh ini lebih rumit, pastikan bahwa Anda sudah terbiasa menginstal aplikasi pada contoh sebelumnya sebelum menjalankan yang satu ini.

Sebelum Anda memulai

Pastikan bahwa Anda telah menyelesaikan penyiapan yang diperlukan di dalam [Instalasi AWS SAMCLI](#).

Langkah 1: Inisialisasikan aplikasi

Dalam bagian ini, Anda mengunduh aplikasi contoh, yang terdiri dari templat AWS SAM dan kode aplikasi.

Untuk menginisialisasi aplikasi

1. Jalankan perintah berikut pada AWS SAM CLI command prompt.

```
sam init \  
--location https://github.com/aws-samples/cookiecutter-aws-sam-s3-rekognition-  
dynamodb-python \  
--no-input
```

2. Tinjau isi direktori yang dibuat oleh perintah (`aws_sam_ocr/`):
 - `template.yaml` – Menentukan tiga sumber daya AWS yang dibutuhkan aplikasi Amazon S3: fungsi Lambda, bucket Amazon S3, dan tabel DynamoDB. Templat tersebut juga menentukan pemetaan dan izin antara sumber daya tersebut.
 - Direktori `src/` - Berisi kode aplikasi Amazon S3.
 - `SampleEvent.json` - Sampel sumber peristiwa, yang digunakan untuk pengujian lokal.

Langkah 2: Paketkan aplikasi

Sebelum Anda dapat menguji aplikasi ini secara lokal, Anda harus menggunakan AWS SAM CLI untuk membuat paket penyebaran, yang Anda gunakan untuk menyebarkan aplikasi ke Cloud. AWS Deployment ini membuat sumber daya AWS dan izin yang diperlukan untuk menguji aplikasi lokal.

Untuk membuat paket deployment Lambda

1. Buat bucket S3 di lokasi tempat Anda ingin menyimpan kode yang dipaketkan. Jika Anda ingin menggunakan bucket S3 yang sudah ada, lewati langkah ini.

```
aws s3 mb s3://bucketname
```

2. Buat paket deployment dengan menjalankan perintah CLI package berikut pada prompt perintah.

```
sam package \  
--template-file template.yaml \  
--output-template-file packaged.yaml \  
--s3-bucket bucketname
```

Anda tentukan file templat baru, `packaged.yaml`, saat men-deploy aplikasi pada langkah berikutnya.

Langkah 3: Deploy aplikasi

Setelah Anda membuat paket deployment tersebut, Anda menggunakannya untuk men-deploy aplikasi ke Cloud AWS. Lalu Anda uji aplikasi dengan memanggilnya di dalam Cloud AWS.

Untuk men-deploy aplikasi nirserver ke Cloud AWS

- Dalam AWS SAMCLI, gunakan `deploy` perintah untuk menyebarkan semua sumber daya yang Anda tentukan dalam template.

```
sam deploy \  
  --template-file packaged.yaml \  
  --stack-name aws-sam-ocr \  
  --capabilities CAPABILITY_IAM \  
  --region us-east-1
```

Dalam perintah tersebut, parameter `--capabilities` memungkinkan AWS CloudFormation untuk membuat IAM role.

AWS CloudFormation membuat sumber daya AWS yang ditentukan di dalam templat. Anda dapat mengakses nama sumber daya ini di dalam konsol AWS CloudFormation.

Untuk menguji aplikasi nirserver di dalam Cloud AWS

1. Unggah citra ke bucket Amazon S3 yang Anda buat untuk aplikasi contoh ini.
2. Buka konsol DynamoDB dan cari tabel yang dibuat. Lihat tabel untuk hasil yang dikembalikan oleh Amazon Rekognition.
3. Verifikasi bahwa Daftar Tabel DynamoDB berisi catatan baru yang berisi teks ketika Amazon Rekognition menemukan catatan tersebut pada citra yang diunggah.

Langkah 4: Uji aplikasi secara lokal

Anda dapat menguji aplikasi secara lokal setelah Anda mengambil nama-nama dari sumber daya AWS yang dibuat oleh AWS CloudFormation.

- Ambil nama kunci Amazon S3 dan nama bucket dari AWS CloudFormation. Ubah file `SampleEvent.json` dengan cara mengganti nilai-nilai untuk objek kunci, nama bucket, dan ARN bucket.
- Ambil nama tabel DynamoDB. Nama ini digunakan untuk perintah `sam local invoke` berikut.

Gunakan AWS SAM CLI untuk membuat contoh peristiwa Amazon S3 dan menjalankan fungsi Lambda:

```
TABLE_NAME=Table name obtained from AWS CloudFormation console sam local invoke --event SampleEvent.json
```

Bagian `TABLE_NAME=` mengatur nama tabel DynamoDB. Parameter `--event` menentukan file yang berisi pesan peristiwa uji untuk diteruskan ke fungsi Lambda.

Anda sekarang dapat memverifikasi bahwa catatan DynamoDB yang diharapkan telah dibuat, berdasarkan hasil yang dikembalikan oleh Amazon Rekognition.

Langkah selanjutnya

AWS SAM GitHub Repositori berisi contoh aplikasi tambahan bagi Anda untuk men-download dan bereksperimen dengan. Untuk mengakses repositori ini, lihat [contoh aplikasi AWS SAM](#).

AWS Cloud Development Kit (AWS CDK)

Anda dapat menggunakan AWS SAM CLI untuk menguji dan membangun aplikasi tanpa server secara lokal yang ditentukan menggunakan file. AWS Cloud Development Kit (AWS CDK) Karena AWS SAM CLI pekerjaan dalam struktur AWS CDK proyek, Anda masih dapat menggunakan [AWS CDKToolkit](#) untuk membuat, memodifikasi, dan menyebarkan aplikasi Anda. AWS CDK

Untuk informasi tentang cara menginstal dan mengonfigurasi AWS CDK, lihat [Memulai dengan AWS CDK](#) di Panduan Developer AWS Cloud Development Kit (AWS CDK).

Note

AWS SAMCLIDukung AWS CDK v1 mulai dari versi 1.135.0 dan AWS CDK v2 mulai dari versi 2.0.0.

Topik

- [Memulai dengan AWS SAM dan AWS CDK](#)
- [Pengujian aplikasi AWS CDK secara lokal](#)
- [Membangun aplikasi AWS CDK](#)
- [Men-deploy aplikasi AWS CDK](#)

Memulai dengan AWS SAM dan AWS CDK

Topik ini menjelaskan apa yang perlu Anda gunakan AWS SAMCLI dengan AWS CDK aplikasi, dan memberikan instruksi untuk membangun dan menguji AWS CDK aplikasi sederhana secara lokal.

Prasyarat

Untuk menggunakan AWS SAMCLI dengan AWS CDK, Anda harus menginstal AWS CDK, dan AWS SAMCLI.

- Untuk informasi tentang menginstal AWS CDK, lihat [Memulai AWS CDK](#) di PanduanAWS Cloud Development Kit (AWS CDK) Pengembang.
- Untuk informasi tentang menginstal AWS SAMCLI, lihat [Instalasi AWS SAMCLI](#).

Membuat dan menguji aplikasi secara lokal AWS CDK

Untuk menguji AWS CDK aplikasi secara lokal menggunakan AWS SAMCLI, Anda harus memiliki AWS CDK aplikasi yang berisi fungsi Lambda. Gunakan langkah-langkah berikut untuk membuat AWS CDK aplikasi dasar dengan fungsi Lambda. Untuk informasi selengkapnya, lihat [Membuat aplikasi tanpa server menggunakan Panduan AWS CDK](#) [AWS Cloud Development Kit \(AWS CDK\) Pengembang](#).

Note

AWS SAMCLIDukung AWS CDK v1 mulai dari versi 1.135.0 dan AWS CDK v2 mulai dari versi 2.0.0.

Langkah 1: Buat AWS CDK aplikasi

Untuk tutorial ini, inisialisasi AWS CDK aplikasi yang menggunakan TypeScript.

Perintah untuk menjalankan:

AWS CDK v2

```
mkdir cdk-sam-example
cd cdk-sam-example
cdk init app --language typescript
```

AWS CDK v1

```
mkdir cdk-sam-example
cd cdk-sam-example
cdk init app --language typescript
npm install @aws-cdk/aws-lambda
```

Langkah 2: Tambahkan fungsi Lambda ke aplikasi Anda

Ganti kode `lib/cdk-sam-example-stack.ts` dengan yang berikut ini:

AWS CDK v2

```
import { Stack, StackProps } from 'aws-cdk-lib';
```

```
import { Construct } from 'constructs';
import * as lambda from 'aws-cdk-lib/aws-lambda';

export class CdkSamExampleStack extends Stack {
  constructor(scope: Construct, id: string, props?: StackProps) {
    super(scope, id, props);

    new lambda.Function(this, 'MyFunction', {
      runtime: lambda.Runtime.PYTHON_3_9,
      handler: 'app.lambda_handler',
      code: lambda.Code.fromAsset('./my_function'),
    });
  }
}
```

AWS CDK v1

```
import * as cdk from '@aws-cdk/core';
import * as lambda from '@aws-cdk/aws-lambda';

export class CdkSamExampleStack extends cdk.Stack {
  constructor(scope: Construct, id: string, props?: StackProps) {
    super(scope, id, props);

    new lambda.Function(this, 'MyFunction', {
      runtime: lambda.Runtime.PYTHON_3_9,
      handler: 'app.lambda_handler',
      code: lambda.Code.fromAsset('./my_function'),
    });
  }
}
```

Langkah 3: Tambahkan kode fungsi Lambda Anda

Membuat sebuah direktori bernama `my_function`. Di direktori itu, buat file bernama `app.py`.

Perintah untuk menjalankan:

```
mkdir my_function
cd my_function
touch app.py
```

Tambahkan kode berikut ke `app.py`:

```
def lambda_handler(event, context):  
    return "Hello from SAM and the CDK!"
```

Langkah 4: Uji fungsi Lambda Anda

Anda dapat menggunakan AWS SAM CLI untuk memanggil fungsi Lambda secara lokal yang Anda tentukan dalam aplikasi. AWS CDK Untuk melakukan ini, Anda memerlukan pengenalan konstruksi fungsi dan jalur ke template yang disintesis AWS CloudFormation .

Perintah untuk menjalankan:

```
cdk synth --no-staging
```

```
sam local invoke MyFunction --no-event -t ./cdk.out/CdkSamExampleStack.template.json
```

Contoh keluaran:

```
Invoking app.lambda_handler (python3.9)
```

```
START RequestId: 5434c093-7182-4012-9b06-635011cac4f2 Version: $LATEST
```

```
"Hello from SAM and the CDK!"
```

```
END RequestId: 5434c093-7182-4012-9b06-635011cac4f2
```

```
REPORT RequestId: 5434c093-7182-4012-9b06-635011cac4f2 Init Duration: 0.32 ms Duration:  
177.47 ms Billed Duration: 178 ms Memory Size: 128 MB Max Memory Used: 128 MB
```

Untuk informasi selengkapnya tentang opsi yang tersedia untuk menguji AWS CDK aplikasi menggunakan AWS SAM CLI, lihat [Pengujian aplikasi AWS CDK secara lokal](#)

Pengujian aplikasi AWS CDK secara lokal

Anda dapat menggunakan AWS SAM CLI untuk menguji AWS CDK aplikasi Anda secara lokal dengan menjalankan perintah berikut dari direktori root proyek AWS CDK aplikasi Anda:

- [sam local invoke](#)
- [sam local start-api](#)
- [sam local start-lambda](#)

Sebelum Anda menjalankan salah satu sam local perintah dengan AWS CDK aplikasi, Anda harus menjalankannya `cdk synth`.

Saat menjalankan, sam local invoke Anda memerlukan pengenalan konstruksi fungsi yang ingin Anda panggil, dan jalur ke templat yang disintesis. AWS CloudFormation Jika aplikasi Anda menggunakan tumpukan bersarang, untuk menyelesaikan konflik penamaan, Anda juga memerlukan nama tumpukan tempat fungsi didefinisikan.

Pemakaian:

```
# Invoke the function FUNCTION_IDENTIFIER declared in the stack STACK_NAME
sam local invoke [OPTIONS] [STACK_NAME/FUNCTION_IDENTIFIER]

# Start all APIs declared in the AWS CDK application
sam local start-api -t ./cdk.out/CdkSamExampleStack.template.json [OPTIONS]

# Start a local endpoint that emulates AWS Lambda
sam local start-lambda -t ./cdk.out/CdkSamExampleStack.template.json [OPTIONS]
```

Contoh

Pertimbangkan tumpukan dan fungsi yang dideklarasikan dengan sampel berikut:

```
app = new HelloCdkStack(app, "HelloCdkStack",
    ...
)
class HelloCdkStack extends cdk.Stack {
    constructor(scope: Construct, id: string, props?: cdk.StackProps) {
        ...
        new lambda.Function(this, 'MyFunction', {
            ...
        });

        new HelloCdkNestedStack(this, 'HelloNestedStack' ,{
            ...
        });
    }
}

class HelloCdkNestedStack extends cdk.NestedStack {
    constructor(scope: Construct, id: string, props?: cdk.NestedStackProps) {
        ...
        new lambda.Function(this, 'MyFunction', {
```



```
    ...
  });
  new lambda.Function(this, 'MyNestedFunction', {
    ...
  });
}
```

Perintah berikut secara lokal memanggil fungsi Lambda yang didefinisikan dalam contoh yang disajikan di atas:

```
# Invoke MyFunction from the HelloCdkStack
sam local invoke -t ./cdk.out/HelloCdkStack.template.json MyFunction
```

```
# Invoke MyNestedFunction from the HelloCdkNestedStack
sam local invoke -t ./cdk.out/HelloCdkStack.template.json MyNestedFunction
```

```
# Invoke MyFunction from the HelloCdkNestedStack
sam local invoke -t ./cdk.out/HelloCdkStack.template.json HelloNestedStack/MyFunction
```

Membangun aplikasi AWS CDK

AWS SAM CLI ini menyediakan dukungan untuk membangun fungsi dan lapisan Lambda yang ditentukan dalam AWS CDK aplikasi Anda. [sam build](#)

Untuk fungsi Lambda yang menggunakan artefak zip, jalankan `cdk synth` sebelum Anda menjalankan perintah. `sam local` `sam build` tidak diperlukan.

Jika AWS CDK aplikasi Anda menggunakan fungsi dengan tipe gambar, jalankan `cdk synth` dan kemudian jalankan `sam build` sebelum Anda menjalankan `sam local` perintah. Saat Anda menjalankan `sam build`, AWS SAM tidak membangun fungsi atau lapisan Lambda yang menggunakan konstruksi khusus runtime, misalnya, [NodejsFunction](#) `sam build` tidak mendukung [aset yang dibundel](#).

Contoh

Menjalankan perintah berikut dari direktori root AWS CDK proyek membangun aplikasi.

```
sam build -t ./cdk.out/CdkSamExampleStack.template.json
```

Men-deploy aplikasi AWS CDK

AWS SAM CLI itu tidak mendukung penerapan AWS CDK aplikasi. Gunakan `cdk deploy` untuk menyebarkan aplikasi Anda. Untuk informasi selengkapnya, lihat [AWS CDK Toolkit \(perintah `cdk`\)](#) di Panduan Pengembang AWS Cloud Development Kit (AWS CDK)

Referensi AWS SAM

Spesifikasi AWS SAM

Spesifikasi AWS SAM adalah spesifikasi sumber terbuka berdasarkan lisensi Apache 2.0. Versi spesifikasi AWS SAM saat ini tersedia di [AWS Serverless Application Model\(AWS SAM\) spesifikasi](#).

Templat AWS SAM adalah ekstensi dari templat AWS CloudFormation. Untuk referensi lengkap untuk AWS CloudFormation template, lihat [Referensi template](#) di Panduan AWS CloudFormation Pengguna.

AWS SAM CLI referensi perintah

AWS Serverless Application Model Command Line Interface (AWS SAM CLI) adalah alat baris perintah yang dapat Anda gunakan dengan AWS SAM templat dan integrasi pihak ketiga yang didukung untuk membangun dan menjalankan aplikasi tanpa server Anda.

Anda dapat menggunakan AWS SAM CLI perintah untuk mengembangkan, menguji, dan menyebarkan aplikasi tanpa server Anda ke file. AWS Cloud Berikut ini adalah beberapa contoh AWS SAM CLI perintah:

- `sam init`— Jika Anda AWS SAM CLI pengguna pertama kali, Anda dapat menjalankan `sam init` perintah tanpa parameter apa pun untuk membuat aplikasi Hello World. Perintah tersebut membuat templat AWS SAM yang telah dikonfigurasi dan contoh kode aplikasi dengan bahasa yang Anda pilih.
- `sam local invokedan sam local start-api` — Gunakan perintah ini untuk menguji kode aplikasi Anda secara lokal, sebelum menerapkannya ke file. AWS Cloud
- `sam logs`— Gunakan perintah ini untuk mengambil log yang dihasilkan oleh fungsi Lambda Anda. Ini dapat membantu Anda menguji dan men-debug aplikasi Anda setelah Anda menerapkannya ke file. AWS Cloud
- `sam package`— Gunakan perintah ini untuk menggabungkan kode aplikasi dan dependensi Anda ke dalam paket deployment. Anda memerlukan paket penyebaran untuk mengunggah aplikasi Anda ke file. AWS Cloud
- `sam deploy`— Gunakan perintah ini untuk menyebarkan aplikasi tanpa server Anda ke file. AWS Cloud Perintah tersebut membuat sumber daya AWS dan menetapkan izin serta konfigurasi lain yang ditentukan dalam templat AWS SAM.

Untuk petunjuk tentang menginstal AWS SAMCLI, lihat [Instalasi AWS SAMCLI](#).

Templat kebijakan AWS SAM

Dengan AWS SAM, Anda dapat memilih dari daftar templat kebijakan untuk cakupan izin AWS Lambda fungsi Anda ke sumber daya yang digunakan aplikasi Anda.

Topik

- [AWS Serverless Application Model \(AWS SAM\) spesifikasi](#)
- [AWS SAMCLI referensi perintah](#)
- [AWS SAMCLI berkas konfigurasi](#)
- [AWS SAM referensi konektor](#)
- [AWS SAM templat kebijakan](#)
- [Repositori citra](#)
- [Telemetri di AWS SAMCLI](#)
- [Mengelola akses dan izin sumber daya](#)

AWS SAMCLI referensi

Bagian ini adalah referensi untuk AWS Serverless Application Model Command Line Interface (AWS SAMCLI).

Topik

- [AWS SAMCLI referensi perintah](#)
- [AWS SAMCLI berkas konfigurasi](#)
- [Mengelola AWS SAMCLI versi](#)
- [Verifikasi integritas AWS SAMCLI penginstal](#)
- [Menyiapkan kredensial AWS](#)
- [AWS SAMCLI pemecahan masalah](#)

AWS SAMCLIREferensi perintah

Bagian ini adalah referensi untuk AWS SAM CLI perintah. Untuk petunjuk tentang menginstal AWS SAMCLI, lihat [Instalasi AWS SAMCLI](#).

Topik

- [sam build](#)
- [sam delete](#)
- [sam deploy](#)
- [sam init](#)
- [sam list](#)
- [sam local generate-event](#)
- [sam local invoke](#)
- [sam local start-api](#)
- [sam local start-lambda](#)
- [sam logs](#)
- [sam package](#)
- [sam pipeline bootstrap](#)
- [sam pipeline init](#)
- [sam publish](#)
- [sam remote invoke](#)
- [sam remote test-event](#)
- [sam sync](#)
- [sam traces](#)
- [sam validate](#)

sam build

Halaman ini memberikan informasi referensi untuk AWS Serverless Application Model perintah Command Line Interface (AWS SAMCLI) `sam build`.

- Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#).

- Untuk dokumentasi tentang penggunaan AWS SAMCLI `sam build` perintah, lihat [Menggunakan sam build](#).

`sam build` Perintah menyiapkan aplikasi untuk langkah-langkah selanjutnya dalam alur kerja pengembang, seperti pengujian lokal atau penyebaran ke file. AWS Cloud

Penggunaan

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

Argumen

ID Sumber Daya

Opsional. Menginstruksikan AWS SAM untuk membangun sumber daya tunggal yang dideklarasikan dalam [AWS SAM template](#). Artefak build untuk sumber daya yang ditentukan akan menjadi satu-satunya yang tersedia untuk perintah berikutnya dalam alur kerja, mis. `sam package` dan `sam deploy`.

Opsi

`--base-dir, -s DIRECTORY`

Selesaikan jalur relatif ke fungsi atau kode sumber lapisan sehubungan dengan direktori ini. Gunakan opsi ini jika Anda ingin mengubah cara jalur relatif ke folder kode sumber diselesaikan. Secara default, jalur relatif diselesaikan sehubungan dengan lokasi templat AWS SAM .

Selain sumber daya dalam aplikasi root atau tumpukan yang sedang Anda bangun, opsi ini juga memberlakukan aplikasi atau tumpukan nest.

Opsi ini berlaku untuk tipe sumber daya dan properti berikut:

- Tipe sumber daya: Properti `AWS::Serverless::Function: CodeUri`
- Tipe sumber daya: Atribut Sumber Daya `AWS::Serverless::Function: Entry Metadata: DockerContext`
- Tipe sumber daya: Properti `AWS::Serverless::LayerVersion: ContentUri`
- Tipe sumber daya: Properti `AWS::Lambda::Function: Code`
- Tipe sumber daya: Properti `AWS::Lambda::LayerVersion: Content`

`--beta-features` | `--no-beta-features`

Izinkan atau tolak fitur beta.

`--build-dir`, `-b` *DIRECTORY*

Jalur ke direktori tempat artefak build disimpan. Direktori ini dan semua isinya akan dihapus dengan opsi ini.

`--build-image` *TEXT*

URI dari citra kontainer yang ingin Anda tarik untuk build. Secara default, AWS SAM menarik citra kontainer dari Amazon ECR Publik. Gunakan opsi ini untuk menarik citra dari lokasi lain.

Anda dapat menentukan opsi ini beberapa kali. Setiap instans dari opsi ini dapat mengambil baik string atau pasangan kunci-nilai. Jika Anda menentukan string, string adalah URI citra kontainer untuk digunakan untuk semua sumber daya dalam aplikasi Anda. Misalnya, `sam build --use-container --build-image amazon/aws-sam-cli-build-image-python3.8`. Jika Anda menentukan pasangan kunci-nilai, kuncinya adalah nama sumber daya, dan nilainya adalah URI citra kontainer untuk digunakan untuk sumber daya tersebut. Sebagai contoh, `sam build --use-container --build-image Function1=amazon/aws-sam-cli-build-image-python3.8`. Dengan pasangan kunci-nilai, Anda dapat menentukan citra kontainer yang berbeda untuk sumber daya yang berbeda.

Opsi ini hanya berlaku jika opsi `--use-container` ditentukan, jika tidak akan terjadi kesalahan.

`--build-in-source` | `--no-build-in-source`

Berikan `--build-in-source` untuk membangun proyek Anda langsung di folder sumber.

`--build-in-source`Opsi ini mendukung runtime dan metode build berikut:

- Runtime - Setiap Node.js runtime yang didukung oleh opsi. [sam init --runtime](#)
- Membangun metode `—Makefile`,`esbuild`.


`--build-in-source`Opsi ini tidak kompatibel dengan opsi berikut:

- `--hook-name`
- `--use-container`

Default: `--no-build-in-source`

`--cached` | `--no-cached`

Aktifkan atau nonaktifkan build yang di-cache. Gunakan opsi ini untuk menggunakan kembali artefak build yang belum berubah dari build sebelumnya. AWS SAM mengevaluasi apakah Anda telah mengubah file apa pun di direktori proyek Anda. Secara default, build tidak di-cache. Jika `--no-cached` opsi dipanggil, itu akan mengganti `cached = true` pengaturan di `samconfig.toml`.

 Note

AWS SAM tidak mengevaluasi apakah Anda telah mengubah modul pihak ketiga yang bergantung pada proyek Anda, di mana Anda belum menyediakan versi tertentu. Misalnya, jika fungsi Python Anda menyertakan `requirements.txt` file dengan entri `requests=1.x`, dan versi modul permintaan terbaru berubah dari 1.1 ke 1.2, maka AWS SAM tidak menarik versi terbaru hingga Anda menjalankan build yang tidak di-cache.

`--cache-dir`

Direktori tempat artefak cache disimpan saat `--cached` ditentukan. Direktori cache default adalah `.aws-sam/cache`.

`--config-env` *TEXT*

Nama lingkungan yang menentukan nilai parameter default dalam file konfigurasi yang akan digunakan. Nilai default-nya adalah "default". Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAM CLI berkas konfigurasi](#).

`--config-file` *PATH*

Jalur dan nama file dari file konfigurasi yang berisi nilai parameter default yang akan digunakan. Nilai default-nya adalah "samconfig.toml" di root direktori proyek. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAM CLI berkas konfigurasi](#).

`--container-env-var`, `-e` *TEXT*

Variabel lingkungan yang akan diteruskan ke kontainer build. Anda dapat menentukan opsi ini beberapa kali. Setiap instans opsi ini mengambil pasangan kunci-nilai, dengan penjelasan kunci adalah variabel sumber daya dan lingkungan, dan nilai adalah nilai variabel lingkungan. Misalnya: `--container-env-var Function1.GITHUB_TOKEN=TOKEN1 --container-env-var Function2.GITHUB_TOKEN=TOKEN2`.

Opsi ini hanya berlaku jika opsi `--use-container` ditentukan, jika tidak akan terjadi kesalahan.

`--container-env-var-file, -ef PATH`

Nama jalur dan file dari file JSON yang berisi nilai-nilai untuk variabel lingkungan kontainer ini. Untuk informasi selengkapnya tentang file variabel lingkungan kontainer, lihat [File variabel lingkungan kontainer](#).

Opsi ini hanya berlaku jika opsi `--use-container` ditentukan, jika tidak akan terjadi kesalahan.

`--debug`

Mengaktifkan logging debug untuk mencetak pesan debug yang AWS SAMCLI dihasilkan, dan untuk menampilkan stempel waktu.

`--docker-network TEXT`

Menentukan nama atau ID dari Docker jaringan yang ada yang kontainer Docker Lambda harus terhubung ke, bersama dengan jaringan jembatan default. Jika tidak ditentukan, kontainer Lambda hanya terhubung ke jaringan jembatan Docker default.

`--exclude, -x`

Nama sumber daya untuk dikecualikan darisam `build`. Misalnya, jika template Anda berisi `Function1,Function2,Function3` dan Anda menjalankansam `build --exclude Function2`, hanya `Function1` dan `Function3` akan dibangun.

`--help`

Menunjukkan pesan ini dan keluar.

`--hook-name TEXT`

Nama hook yang digunakan untuk memperluas AWS SAMCLI fungsionalitas.

Nilai yang diterima:`terraform`.

`--manifest , -m PATH`

Jalur ke file manifest dependensi kustom (misalnya, `package.json`) yang akan digunakan bukan default.

`--parallel`

Pengaktifan build paralel. Gunakan opsi ini untuk membangun fungsi dan lapisan AWS SAM template Anda secara paralel. Secara default, fungsi dan lapisan dibangun secara berurutan.

--parameter-overrides

(Opsional) String yang berisi penggantian AWS CloudFormation parameter yang dikodekan sebagai pasangan nilai kunci. Menggunakan format yang sama dengan AWS Command Line Interface (AWS CLI). Misalnya: 'ParameterKey=KeyPairName, ParameterValue = MyKey ParameterKey =InstanceType, ParameterValue =t1.micro'. Opsi ini tidak kompatibel dengan --hook-name.

--profile *TEXT*

Profil spesifik dari file kredensialmu yang mendapat AWS kredensialnya.

--region *TEXT*

Wilayah AWS Untuk menyebarkan ke. Misalnya, us-east-1.

--save-params

Simpan parameter yang Anda berikan pada baris perintah ke file AWS SAM konfigurasi.

--skip-prepare-infra

Melewati tahap persiapan jika tidak ada perubahan infrastruktur yang dilakukan. Gunakan dengan --hook-name opsi.

--skip-pull-image

Menentukan apakah perintah harus melewati menarik ke bawah citra Docker terbaru untuk waktu aktif Lambda.

--template-file, --template, -t *PATH*

Path dan nama file file AWS SAM template[default: template.[yaml|yml]]. Opsi ini tidak kompatibel dengan --hook-name.

--terraform-project-root-path

Jalur relatif atau absolut ke direktori tingkat atas yang berisi file Terraform konfigurasi atau kode sumber fungsi Anda. Jika file-file ini terletak di luar direktori yang berisi modul Terraform root Anda, gunakan opsi ini untuk menentukan jalur absolut atau relatifnya. Opsi ini mengharuskan --hook-name diatur keterraform.

--use-container, -u

Jika fungsi Anda tergantung pada paket yang memiliki dependensi terkompilasi secara native, gunakan opsi ini untuk membangun fungsi Anda di dalam kontainer Docker mirip Lambda.

sam delete

Halaman ini memberikan informasi referensi untuk AWS Serverless Application Model perintah Command Line Interface (AWS SAMCLI) `sam delete`.

Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#).

`sam delete` Perintah menghapus AWS SAM aplikasi dengan menghapus AWS CloudFormation tumpukan, artefak yang dikemas dan digunakan ke Amazon S3 dan Amazon ECR, dan file template. **AWS SAM**

Perintah ini juga memeriksa apakah ada tumpukan pendamping Amazon ECR yang diterapkan, dan jika demikian meminta pengguna untuk menghapus tumpukan itu dan repositori Amazon ECR. Jika `--no-prompts` ditentukan, maka tumpukan pendamping dan repositori Amazon ECR dihapus secara default.

Penggunaan

```
$ sam delete <options>
```

Opsi

`--config-env` *TEXT*

Nama lingkungan yang menentukan nilai parameter default dalam file konfigurasi yang akan digunakan. Nilai default-nya adalah `default`. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--config-file` *PATH*

Jalur dan nama file dari file konfigurasi yang berisi nilai parameter default yang akan digunakan. Nilai default-nya adalah `samconfig.toml` di root direktori proyek. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--debug`

Mengaktifkan logging debug untuk mencetak pesan debug yang AWS SAM CLI dihasilkan dan untuk menampilkan stempel waktu.

`--help`

Menunjukkan pesan ini dan keluar.

--no-prompts

Tentukan opsi ini agar AWS SAM beroperasi dalam mode non-interaktif. Nama tumpukan harus disediakan, baik dengan `--stack-name` opsi, atau dalam `toml` file konfigurasi.

--profile *TEXT*

Profil spesifik dari file kredensial Anda yang mendapatkan kredensial AWS.

--region *TEXT*

Wilayah AWS tempat men-deploy. Misalnya, `us-east-1`.

--s3-bucket

Jalur bucket Amazon S3 yang ingin Anda hapus.

--s3-prefix

Awalan bucket Amazon S3 yang ingin Anda hapus.

--save-params

Simpan parameter yang Anda berikan pada baris perintah ke file AWS SAM konfigurasi.

--stack-name *TEXT*

Nama AWS CloudFormation tumpukan yang ingin Anda hapus.

sam deploy

Halaman ini memberikan informasi referensi untuk AWS Serverless Application Model perintah Command Line Interface (AWS SAMCLI) `sam deploy`.

- Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#).
- Untuk dokumentasi tentang penggunaan AWS SAM CLI `sam deploy` perintah, lihat [Menggunakan sam deploy](#).

`sam deploy` Perintah menyebarkan aplikasi untuk AWS Cloud menggunakan AWS CloudFormation.

Penggunaan

```
$ <environment variables> sam deploy <options>
```

Variabel-variabel lingkungan

SAM_CLI_POLL_DELAY

Tentukan penundaan, dalam hitungan detik, di antara panggilan DescribeStack API.

Berikut ini adalah contohnya:

```
$ SAM_CLI_POLL_DELAY=5 sam deploy
```

Opsi

`--capabilities` *LIST*

Daftar kemampuan yang harus Anda tentukan untuk mengizinkan AWS CloudFormation membuat tumpukan tertentu. Beberapa templat tumpukan mungkin menyertakan sumber daya yang memengaruhi izin di AndaAkun AWS, misalnya, dengan membuat pengguna baru AWS Identity and Access Management (IAM). Untuk tumpukan tersebut, Anda harus secara eksplisit menerima kemampuannya dengan menentukan opsi ini. Nilai yang valid adalah hanya `CAPABILITY_IAM` dan `CAPABILITY_NAMED_IAM`. Jika Anda memiliki sumber daya IAM, maka Anda dapat menentukan salah satu kemampuan. Jika Anda memiliki sumber daya IAM dengan nama khusus, maka Anda harus menentukan `CAPABILITY_NAMED_IAM`. Jika Anda tidak menentukan opsi ini, maka operasi mengembalikan `InsufficientCapabilities` kesalahan.

`--config-env` *TEXT*

Nama lingkungan yang menentukan nilai parameter default dalam file konfigurasi yang akan digunakan. Nilai default-nya adalah `default`. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--config-file` *PATH*

Jalur dan nama file dari file konfigurasi yang berisi nilai parameter default yang akan digunakan. Nilai default-nya adalah `samconfig.toml` di root direktori proyek. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--confirm-changeset` | `--no-confirm-changeset`

Prompt untuk mengonfirmasi apakah AWS SAM CLI penerapan set perubahan yang dihitung.

--debug

Aktifkan logging debug untuk mencetak pesan debug yang AWS SAM CLI dihasilkan dan untuk menampilkan stempel waktu.

--disable-rollback | --no-disable-rollback

Tentukan apakah akan memutar kembali AWS CloudFormation tumpukan Anda jika terjadi kesalahan selama penerapan. Secara default, jika ada kesalahan selama penerapan, AWS CloudFormation tumpukan Anda akan kembali ke status stabil terakhir. Jika Anda menentukan `--disable-rollback` dan terjadi kesalahan selama penerapan, maka sumber daya yang dibuat atau diperbarui sebelum kesalahan terjadi tidak dibatalkan.

--fail-on-empty-changeset | --no-fail-on-empty-changeset

Tentukan apakah akan mengembalikan kode keluar bukan nol jika tidak ada perubahan yang harus dilakukan pada tumpukan. Perilaku default-nya adalah mengembalikan kode keluar nonkosong.

--force-upload

Tentukan opsi ini untuk mengunggah artefak bahkan jika artefak itu cocok dengan artefak yang ada di bucket Amazon S3. Mencocokkan artefak dibatalkan.

--guided, -g

Tentukan opsi ini untuk memiliki petunjuk AWS SAM CLI penggunaan untuk memandu Anda melalui penerapan.

--help

Tampilkan pesan ini dan keluar.

--image-repositories *TEXT*

Pemetaan fungsi ke URI repositori Amazon ECR mereka. Fungsi referensi dengan ID logisnya. Berikut ini adalah contohnya:

```
$ sam deploy --image-repositories Function1=123456789012.dkr.ecr.us-east-1.amazonaws.com/my-repo
```

Anda dapat menentukan opsi ini beberapa kali dalam satu perintah.

`--image-repository` *TEXT*

Nama repositori Amazon ECR tempat perintah ini mengunggah gambar fungsi Anda. Opsi ini diperlukan untuk fungsi yang dideklarasikan dengan jenis Image paket.

`--kms-key-id` *TEXT*

ID dari kunci AWS Key Management Service (AWS KMS) yang digunakan untuk mengenkripsi artefak yang nonaktif di bucket Amazon S3. Jika Anda tidak menentukan opsi ini, AWS SAM gunakan kunci enkripsi yang dikelola Amazon S3.

`--metadata`

Peta metadata untuk melampirkan ke semua artefak yang direferensikan dalam templat Anda.

`--no-execute-changeset`

Menunjukkan apakah akan menerapkan changeset. Tentukan opsi ini jika Anda ingin melihat perubahan tumpukan Anda sebelum menerapkan set perubahan. Perintah ini membuat AWS CloudFormation changeset dan kemudian keluar tanpa menerapkan changeset. Untuk menerapkan changeset, jalankan perintah yang sama tanpa opsi ini.

`--no-progressbar`

Jangan tampilkan baris progres saat mengunggah artefak ke Amazon S3.

`--notification-arns` *LIST*

Daftar ARN topik Amazon Simple Notification Service (Amazon SNS) yang AWS CloudFormation hubungkan dengan tumpukan.

`--on-failure` [ROLLBACK | DELETE | DO_NOTHING]

Tentukan tindakan yang akan diambil saat tumpukan gagal dibuat.

Pilihan berikut tersedia:

- ROLLBACK— Menggulung kembali tumpukan ke keadaan baik yang diketahui sebelumnya.
- DELETE— Menggulung kembali tumpukan ke keadaan baik yang diketahui sebelumnya, jika ada. Jika tidak, menghapus tumpukan.
- DO_NOTHING— Tidak memutar kembali atau menghapus tumpukan. Efeknya sama dengan efek `--disable-rollback`.

Perilaku bawaannya adalah ROLLBACK.

Note

Anda dapat menentukan `--disable-rollback` opsi atau `--on-failure` opsi, tetapi tidak keduanya.

--parameter-overrides

Sebuah string yang berisi parameter AWS CloudFormation membatalkan pengkodean sebagai pasangan nilai kunci. Gunakan format yang sama seperti AWS Command Line Interface (AWS CLI). Misalnya, `ParameterKey=ParameterValue InstanceType=t1.micro`.

--profile *TEXT*

Profil spesifik dari file kredensial Anda yang mendapatkan kredensial AWS.

--region *TEXT*

Wilayah AWS Untuk menyebarkan ke. Misalnya, `us-east-1`.

--resolve-image-repos

Secara otomatis membuat repositori Amazon ECR untuk digunakan untuk pengemasan dan penerapan untuk penerapan yang tidak dipandu. Opsi ini hanya berlaku untuk fungsi dan lapisan dengan `PackageType: Image` yang ditentukan. Jika Anda menentukan `--guided` opsi, maka AWS SAM CLI mengabaikan `--resolve-image-repos`.

Note

Jika AWS SAM secara otomatis membuat repositori ECR Amazon untuk fungsi atau lapisan dengan opsi ini, dan Anda kemudian menghapus fungsi atau lapisan tersebut dari AWS SAM templat Anda, maka repositori ECR Amazon yang sesuai akan dihapus secara otomatis.

--resolve-s3

Buat bucket Amazon S3 secara otomatis untuk digunakan untuk pengemasan dan penerapan untuk penerapan yang tidak dipandu. Jika Anda menentukan `--guided` opsi, maka AWS SAM CLI mengabaikan `--resolve-s3`. Jika Anda menentukan opsi `--s3-bucket` dan `--resolve-s3` opsi, maka terjadi kesalahan.

`--role-arn` *TEXT*

Nama Sumber Daya Amazon (ARN) dari peran IAM yang AWS CloudFormation diasumsikan saat menerapkan kumpulan perubahan.

`--s3-bucket` *TEXT*

Nama bucket Amazon S3 tempat perintah ini mengunggah template Anda. AWS CloudFormation Jika template Anda lebih besar dari 51.200 byte, maka `--s3-bucket` opsi atau opsi diperlukan `--resolve-s3`. Jika Anda menentukan opsi `--s3-bucket` dan `--resolve-s3` opsi, maka terjadi kesalahan.

`--s3-prefix` *TEXT*

Prefiks ditambahkan ke nama artefak yang diunggah ke bucket Amazon S3. Nama prefiks adalah nama jalur (nama folder) untuk bucket Amazon S3.

`--save-params`

Simpan parameter yang Anda berikan pada baris perintah ke file AWS SAM konfigurasi.

`--signing-profiles` *LIST*

Daftar profil penandatanganan untuk menandatangani paket deployment Anda. Opsi ini mengambil daftar pasangan kunci-nilai, dengan penjelasan kunci adalah nama fungsi atau lapisan untuk menandatangani, dan nilai adalah profil penandatanganan, dengan pemilik profil opsional yang dibatasi dengan `:`. Misalnya, `FunctionNameToSign=SigningProfileName1 LayerNameToSign=SigningProfileName2:SigningProfileOwner`.

`--stack-name` *TEXT*

(Diperlukan) Nama tumpukan AWS CloudFormation tempat Anda men-deploy. Jika Anda menentukan tumpukan yang ada, maka perintah memperbarui tumpukan. Jika Anda menentukan tumpukan baru, maka perintah membuatnya.

`--tags` *LIST*

Daftar tanda untuk menghubungkan dengan tumpukan yang dibuat atau diperbarui. AWS CloudFormation juga menyebarkan tanda ini ke sumber daya dalam tumpukan yang mendukungnya.

`--template-file`, `--template`, `-t` *PATH*

Nama jalur dan file tempat templat AWS SAM Anda berada.

Note

Jika Anda menentukan opsi ini, maka AWS SAM gunakan hanya templat dan sumber daya lokal yang ditunjukkannya.

`--use-json`

Output JSON untuk templat AWS CloudFormation. Output default-nya adalah YAML.

sam init

Halaman ini memberikan informasi referensi untuk AWS Serverless Application Model perintah Command Line Interface (AWS SAMCLI) `sam init`.

- Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#).
- Untuk dokumentasi tentang penggunaan AWS SAMCLI `sam init` perintah, lihat [Menggunakan sam init](#).

`sam init` Perintah ini menyediakan opsi untuk menginisialisasi aplikasi tanpa server baru.

Penggunaan

```
$ sam init <options>
```

Opsi

`--app-template` *TEXT*

Pengenal templat aplikasi terkelola yang ingin Anda gunakan. Jika Anda tidak yakin, hubungi `sam init` tanpa opsi untuk alur kerja interaktif.

Parameter ini diperlukan jika `--no-interactive` ditentukan dan `--location` tidak disediakan.

Parameter ini hanya tersedia dalam AWS SAMCLI versi 0.30.0 dan yang lebih baru. Menentukan parameter ini dengan versi sebelumnya menyebabkan kesalahan.

`--application-insights | --no-application-insights`

Aktifkan pemantauan Amazon CloudWatch Application Insights untuk aplikasi Anda. Untuk mempelajari selengkapnya, lihat [Pantau aplikasi tanpa server Anda dengan CloudWatch Application Insights](#).

Opsi default-nya adalah `--no-application-insights`.

`--architecture, -a [x86_64 | arm64]`

Arsitektur set instruksi untuk fungsi Lambda aplikasi Anda. Tentukan salah satu `x86_64` atau `arm64`.

`--base-image [amazon/dotnet8-base | amazon/dotnet6-base | amazon/dotnetcore3.1-base | amazon/go1.x-base | amazon/java21-base | amazon/java17-base | amazon/java11-base | amazon/java8.al2-base | amazon/java8-base | amazon/nodejs20.x-base | amazon/nodejs18.x-base | amazon/nodejs16.x-base | amazon/python3.12-base | amazon/python3.11-base | amazon/python3.10-base | amazon/python3.9-base | amazon/python3.8-base | amazon/ruby3.3-base | amazon/ruby3.2-base]`

Citra dasar dari aplikasi Anda. Opsi ini hanya berlaku jika tipe pakatnya adalah Image.

Parameter ini diperlukan jika `--no-interactive` ditentukan, `--package-type` ditentukan sebagai Image, dan `--location` tidak ditentukan.

`--config-env TEXT`

Nama lingkungan yang menentukan nilai parameter default dalam file konfigurasi yang akan digunakan. Nilai default-nya adalah "default". Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--config-file PATH`

Jalur dan nama file dari file konfigurasi yang berisi nilai parameter default yang akan digunakan. Nilai default-nya adalah "samconfig.toml" di root direktori proyek. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--debug`

Mengaktifkan logging debug untuk mencetak pesan debug yang AWS SAMCLI dihasilkan, dan untuk menampilkan stempel waktu.

`--dependency-manager, -d [gradle | mod | maven | bundler | npm | cli-package | pip]`

Manajer dependensi waktu aktif Lambda Anda.

`--extra-content`

Ganti parameter kustom apa pun dalam `cookiecutter.json` konfigurasi template, misalnya, `{"customParam1": "customValue1", "customParam2": "customValue2"}`.

`--help, -h`

Menunjukkan pesan ini dan keluar.

`--location, -l TEXT`

Lokasi templat atau aplikasi (Git, Mercurial, HTTP/HTTPS, file .zip, jalur).

Parameter ini diperlukan jika `--no-interactive` ditentukan dan `--runtime`, `--name`, dan `--app-template` tidak disediakan.

Untuk repositori Git, Anda harus menggunakan lokasi root repositori.

Untuk jalur lokal, templatnya harus dalam bentuk antara file .zip atau format [Cookiecutter](#).

`--name, -n TEXT`

Nama proyek Anda yang akan dihasilkan sebagai direktori.

Parameter ini diperlukan jika `--no-interactive` ditentukan dan `--location` tidak disediakan.

`--no-input`

Menonaktifkan prompt Cookiecutter dan menerima nilai `vcfdefault` yang ditentukan dalam konfigurasi templat.

`--no-interactive`

Nonaktifkan prompt interaktif untuk parameter `init`, dan gagal jika nilai-nilai yang diperlukan tidak ada.

`--output-dir, -o PATH`

Lokasi tempat aplikasi diinisialisasi adalah `output`.

`--package-type [Zip | Image]`

Tipe paket aplikasi contoh. Zip membuat arsip file .zip, dan Image membuat citra kontainer.

`--runtime, -r [dotnet8 | dotnet6 | dotnetcore3.1 | go1.x | java21 | java17 | java11 | java8 | java8.al2 | nodejs20.x | nodejs18.x | nodejs16.x | python3.12 | python3.11 | python3.10 | python3.9 | python3.8 | ruby3.3 | ruby3.2]`

Waktu aktif Lambda aplikasi Anda. Opsi ini hanya berlaku jika tipe paketnya adalah Zip.

Parameter ini diperlukan jika `--no-interactive` ditentukan, `--package-type` ditentukan sebagai Zip, dan `--location` tidak ditentukan.

`--save-params`

Simpan parameter yang Anda berikan pada baris perintah ke file AWS SAM konfigurasi.

`--tracing | --no-tracing`

Aktifkan AWS X-Ray penelusuran untuk fungsi Lambda Anda.

sam list

Halaman ini memberikan informasi referensi untuk AWS Serverless Application Model perintah Command Line Interface (AWS SAMCLI) `sam list`.

Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#)

`sam list` Perintah tersebut mengeluarkan informasi penting tentang sumber daya dalam aplikasi tanpa server Anda dan status aplikasi tanpa server Anda. Gunakan `sam list` sebelum dan sesudah penerapan untuk membantu selama pengembangan lokal dan cloud.

Penggunaan

```
$ sam list <options> <subcommand>
```

Opsi

`--help, -h`

Tampilkan pesan ini dan keluar.

Subperintah

endpoints

Menampilkan daftar cloud dan titik akhir lokal dari AWS CloudFormation tumpukan Anda. Untuk informasi selengkapnya, lihat [sam list endpoints](#).

resources

Menampilkan sumber daya dalam template AWS Serverless Application Model (AWS SAM) Anda yang dibuat pada AWS CloudFormation saat penerapan. Untuk informasi selengkapnya, lihat [sam list resources](#).

stack-outputs

Menampilkan output AWS CloudFormation tumpukan Anda dari AWS CloudFormation template AWS SAM atau. Lihat informasi yang lebih lengkap di [sam list stack-outputs](#).

sam list endpoints

Halaman ini memberikan informasi referensi untuk AWS Serverless Application Model `sam list endpoints` subperintah Command Line Interface (AWS SAMCLI).

Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#).

`sam list endpoints` Subperintah menampilkan daftar cloud dan endpoint lokal dari tumpukan Anda AWS CloudFormation. Anda dapat berinteraksi dengan sumber daya ini melalui `sam sync` perintah `sam local` dan.

AWS Lambda dan jenis sumber daya Amazon API Gateway didukung dengan perintah ini.

Note

Domain khusus didukung saat dikonfigurasi untuk sumber daya Amazon API Gateway Anda. Perintah ini akan menampilkan domain kustom alih-alih titik akhir default.

Penggunaan

```
$ sam list endpoints <options>
```

Opsi

`--config-env` *TEXT*

Nama lingkungan yang menentukan nilai parameter default dalam file konfigurasi yang akan digunakan.

Nilai default: `default`

Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAM CLI berkas konfigurasi](#).

`--config-file` *TEXT*

Jalur dan nama file dari file konfigurasi yang berisi nilai parameter default yang akan digunakan.

Nilai default: `samconfig.toml` di direktori kerja saat ini.

Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAM CLI berkas konfigurasi](#).

`--debug`

Aktifkan pencatatan debug untuk mencetak pesan debug yang dihasilkan oleh stempel waktu AWS SAM CLI with.

`--help`, `-h`

Tampilkan pesan ini dan keluar.

`--output` [`json`|`table`]

Tentukan format untuk hasil keluaran.

Nilai default: `table`

`--profile` *TEXT*

Pilih profil tertentu dari file kredensi Anda untuk mendapatkan AWS kredensial.

`--region` *TEXT*

Atur AWS wilayah layanan. Misalnya, `us-east-1`.

`--save-params`

Simpan parameter yang Anda berikan pada baris perintah ke file AWS SAM konfigurasi.

`--stack-name` *TEXT*

Nama AWS CloudFormation tumpukan yang digunakan. Nama tumpukan dapat ditemukan di file aplikasi Anda atau `samconfig.toml` file konfigurasi yang ditunjuk.

Ketika opsi ini tidak ditentukan, sumber daya lokal yang ditentukan dalam template Anda akan ditampilkan.

`--template-file`, `--template`, `-t` *PATH*

AWS SAMberkas templat.

Nilai default: `template.[yaml|yml|json]`

Contoh-contoh

Menampilkan output, dalam format json, dari titik akhir sumber daya yang diterapkan dari tumpukan Anda AWS CloudFormation bernama. `test-stack`

```
$ sam list endpoints --stack-name test-stack --output json

[
  {
    "LogicalResourceId": "HelloWorldFunction",
    "PhysicalResourceId": "sam-app-test-list-HelloWorldFunction-H85Y7yIV7ZLq",
    "CloudEndpoint": "https://zt55oi7kbljxjmcoahsj3cknwu0rposq.lambda-url.us-east-1.on.aws/",
    "Methods": "-"
  },
  {
    "LogicalResourceId": "ServerlessRestApi",
    "PhysicalResourceId": "uj80uoe2o2",
    "CloudEndpoint": [
      "https://uj80uoe2o2.execute-api.us-east-1.amazonaws.com/Prod",
      "https://uj80uoe2o2.execute-api.us-east-1.amazonaws.com/Stage"
    ],
    "Methods": [
      "/hello['get']"
    ]
  }
]
```

sam list resources

Halaman ini memberikan informasi referensi untuk AWS Serverless Application Model `sam list resources` subperintah Command Line Interface (AWS SAMCLI).

Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#).

`aws sam list resources` Subperintah menampilkan sumber daya dalam template AWS Serverless Application Model (AWS SAM) Anda yang dibuat AWS CloudFormation oleh AWS SAM transformasi saat penerapan.

Gunakan `aws sam list resources` dengan AWS SAM template sebelum penerapan untuk melihat sumber daya yang akan dibuat. Berikan nama AWS CloudFormation tumpukan untuk melihat daftar konsolidasi yang mencakup sumber daya yang digunakan.

Note

Untuk menghasilkan daftar sumber daya dari AWS SAM template Anda, transformasi lokal template Anda dilakukan. Sumber daya yang akan digunakan dengan kondisi, seperti dalam wilayah tertentu, termasuk dalam daftar ini.

Penggunaan

```
$ aws sam list resources <options>
```

Opsi

`--config-env` *TEXT*

Nama lingkungan yang menentukan nilai parameter default dalam file konfigurasi yang akan digunakan.

Nilai default: `default`

Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAM CLI berkas konfigurasi](#).

`--config-file` *TEXT*

Jalur dan nama file dari file konfigurasi yang berisi nilai parameter default yang akan digunakan.

Nilai default: `samconfig.toml` di direktori kerja saat ini.

Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAM CLI berkas konfigurasi](#).

`--debug`

Aktifkan pencatatan debug untuk mencetak pesan debug yang dihasilkan oleh stempel waktu AWS SAM CLI with.

`--help, -h`

Tampilkan pesan ini dan keluar.

`--output [json|table]`

Tentukan format untuk hasil keluaran.

Nilai default: `table`

`--profile TEXT`

Pilih profil tertentu dari file kredensialmu untuk mendapatkan AWS kredensialnya.

`--region TEXT`

Atur AWS wilayah layanan. Misalnya, `us-east-1`.

`--save-params`

Simpan parameter yang Anda berikan pada baris perintah ke file AWS SAM konfigurasi.

`--stack-name TEXT`

Nama AWS CloudFormation tumpukan yang digunakan. Nama tumpukan dapat ditemukan di file aplikasi Anda atau `samconfig.toml` file konfigurasi yang ditunjuk.

Ketika disediakan, ID logis sumber daya dari template Anda akan dipetakan ke ID fisik yang sesuai di AWS CloudFormation. Untuk mempelajari lebih lanjut tentang ID fisik, lihat [Bidang sumber daya](#) di Panduan AWS CloudFormation Pengguna.

Ketika opsi ini tidak ditentukan, sumber daya lokal yang ditentukan dalam template Anda akan ditampilkan.

`--template-file, --template, -t PATH`

AWS SAMberkas templat.

Nilai default: `template.[yaml|yml|json]`

Contoh-contoh

Menampilkan output, dalam format tabel, sumber daya lokal dari AWS SAM template Anda dan sumber daya yang digunakan dari AWS CloudFormation tumpukan Anda bernama `test-stack`. Jalankan dari direktori yang sama dengan template lokal Anda.

```
$ 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-1QH7CP0CBL2IK
HelloWorldFunctionRole                                     sam-app-test-list-
HelloWorldFunctionRole-SRJDMJ6F7F41
ServerlessRestApi                                           uj80uoe2o2
ServerlessRestApiDeployment47fc2d5f9d                       pncw5f
ServerlessRestApiProdStage                                  Prod
ServerlessRestApiDeploymentf5716dc08b                       -
-----
```

sam list stack-outputs

Halaman ini memberikan informasi referensi untuk AWS Serverless Application Model `sam list stack-outputs` subperintah Command Line Interface (AWS SAMCLI).

Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#).

`sam list stack-outputs` Subperintah menampilkan output AWS CloudFormation tumpukan Anda dari AWS Serverless Application Model (AWS SAM) atau AWS CloudFormation template. Untuk informasi selengkapnya `Outputs`, lihat [Output](#) di Panduan AWS CloudFormation Pengguna.

Penggunaan

```
$ sam list stack-outputs <options>
```

Opsi

```
--config-env TEXT
```

Nama lingkungan yang menentukan nilai parameter default dalam file konfigurasi yang akan digunakan.

Nilai default: `default`

Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLIberkas konfigurasi](#).

`--config-file` *TEXT*

Jalur dan nama file dari file konfigurasi yang berisi nilai parameter default yang akan digunakan.

Nilai default: `samconfig.toml` di direktori kerja saat ini.

Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLIberkas konfigurasi](#).

`--debug`

Aktifkan pencatatan debug untuk mencetak pesan debug yang dihasilkan oleh stempel waktu AWS SAM CLI with.

`--help`, `-h`

Tampilkan pesan ini dan keluar.

`--output` [`json`|`table`]

Tentukan format untuk hasil keluaran.

Nilai default: `table`

`--profile` *TEXT*

Pilih profil tertentu dari file kredensi Anda untuk mendapatkan AWS kredensial.

`--region` *TEXT*

Atur AWS wilayah layanan. Misalnya, `us-east-1`.

`--save-params`

Simpan parameter yang Anda berikan pada baris perintah ke file AWS SAM konfigurasi.

`--stack-name` *TEXT*

Nama AWS CloudFormation tumpukan yang digunakan. Nama tumpukan dapat ditemukan di file aplikasi Anda atau `samconfig.toml` file konfigurasi yang ditunjuk.

Opsi ini diperlukan.

Contoh-contoh

Menampilkan output, dalam format tabel, sumber daya dalam AWS CloudFormation tumpukan Anda bernama `test-stack`.

```
$ sam list stack-outputs --stack-name test-stack --output table
```

```
-----
OutputKey                               OutputValue
Description
-----
HelloWorldFunctionIamRole               arn:aws:iam::<account-number>:role/sam-
Implicit IAM Role created for Hello
function                                app-test-list-HelloWorldFunctionRole-   World
                                         SRJDMJ6F7F41
HelloWorldApi                           https://uj80ue2o2.execute-api.us-      API
Gateway endpoint URL for Prod
                                         east-1.amazonaws.com/Prod/hello/       stage
for Hello World function
HelloWorldFunction                       arn:aws:lambda:us-                      Hello
World Lambda Function ARN
                                         east-1:<account-number>:function:sam-app-
test-list-
HelloWorldFunction-H85Y7yIV7ZLq
-----
```

sam local generate-event

Halaman ini memberikan informasi referensi untuk AWS Serverless Application Model `sam local generate-event` subperintah Command Line Interface (AWS SAMCLI).

- Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#).
- Untuk dokumentasi tentang penggunaan AWS SAM CLI `sam local generate-event` perintah, lihat [Menggunakan sam local generate-event](#).

`sam local generate-event` Subperintah menghasilkan sampel payload peristiwa untuk didukung. Layanan AWS

Penggunaan

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

Opsi

`--config-env TEXT`

Nama lingkungan yang menentukan nilai parameter default dalam file konfigurasi yang akan digunakan. Nilai default-nya adalah “default”. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--config-file PATH`

Jalur dan nama file dari file konfigurasi yang berisi nilai parameter default yang akan digunakan. Nilai default-nya adalah `samconfig.toml` di root direktori proyek. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--help`

Menunjukkan pesan ini dan keluar.

Layanan

Untuk melihat daftar layanan yang didukung, jalankan yang berikut ini:

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

Peristiwa

Untuk melihat daftar peristiwa yang didukung yang dapat dihasilkan untuk setiap layanan, jalankan yang berikut ini:

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

Opsi acara

Untuk melihat daftar opsi acara yang didukung yang dapat Anda ubah, jalankan yang berikut ini:

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

sam local invoke

Halaman ini memberikan informasi referensi untuk AWS Serverless Application Model `sam local invoke` subperintah Command Line Interface (AWS SAMCLI).

- Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#).
- Untuk dokumentasi tentang penggunaan AWS SAMCLI `sam local invoke` subperintah, lihat [Menggunakan sam local invoke](#).

`sam local invoke` Subperintah memulai pemanggilan satu kali fungsi secara lokal. AWS Lambda

Penggunaan

```
$ sam local invoke <arguments> <options>
```

Note

Jika Anda memiliki lebih dari satu fungsi yang ditentukan dalam AWS SAM template Anda, berikan ID logis fungsi yang ingin Anda panggil.

Argumen

ID Sumber Daya

ID dari fungsi Lambda untuk dipanggil.

Argumen ini opsional. Jika aplikasi Anda berisi satu fungsi Lambda, CLI akan AWS SAM memanggilnya. Jika aplikasi Anda berisi beberapa fungsi, berikan ID fungsi yang akan dipanggil.

Nilai yang valid: ID logis sumber daya atau ARN sumber daya.

Opsi

`--add-host` *LIST*

Meneruskan nama host ke pemetaan alamat IP ke file host wadah Docker. Parameter ini dapat dilewatkan beberapa kali.

Example

Contoh: `--add-host` *example.com:127.0.0.1*

`--beta-features` | `--no-beta-features`

Izinkan atau tolak fitur beta.

`--config-env` *TEXT*

Nama lingkungan yang menentukan nilai parameter default dalam file konfigurasi yang akan digunakan. Nilai default-nya adalah "default". Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--config-file` *PATH*

Jalur dan nama file dari file konfigurasi yang berisi nilai parameter default yang akan digunakan. Nilai default-nya adalah "samconfig.toml" di root direktori proyek. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--container-env-vars`

(Opsional) Teruskan variabel lingkungan untuk kontainer citra fungsi Lambda ketika debugging lokal.

`--container-host` *TEXT*

Host kontainer Lambda yang teremulasi secara lokal. Nilai default-nya adalah localhost. Jika Anda ingin menjalankan AWS SAMCLI penampung Docker di macOS, Anda dapat menentukan host.docker.internal Jika Anda ingin menjalankan wadah pada host yang berbeda dari AWS SAMCLI, Anda dapat menentukan alamat IP dari host jarak jauh.

`--container-host-interface` *TEXT*

Alamat IP dari antarmuka jaringan host tempat kontainer port harus terikat. Nilai default-nya adalah 127.0.0.1. Gunakan 0.0.0.0 untuk mengikat ke semua antarmuka.

`--debug`

Mengaktifkan logging debug untuk mencetak pesan debug yang AWS SAMCLI dihasilkan, dan untuk menampilkan stempel waktu.

`--debug-args` *TEXT*

Argumen tambahan untuk meneruskan ke debugger.

`--debug-port, -d` *TEXT*

Ketika ditentukan, mulai kontainer fungsi Lambda dalam modus debug dan ekspos port ini pada host lokal.

`--debugger-path` *TEXT*

Jalur host ke debugger yang dipasang ke kontainer Lambda.

`--docker-network` *TEXT*

Nama atau ID jaringan Docker yang ada yang harus terhubung dengan kontainer Docker Lambda, bersama dengan jaringan bridge default. Jika tidak ditentukan, kontainer Lambda hanya terhubung ke jaringan Docker bridge default.

`--docker-volume-basedir`, `-v` *TEXT*

Lokasi direktori dasar tempat AWS SAM file ada. Jika Docker berjalan pada mesin jarak jauh, Anda harus memasang jalur tempat AWS SAM file ada di mesin Docker dan memodifikasi nilai ini agar sesuai dengan mesin jarak jauh.

`--env-vars`, `-n` *PATH*

File JSON yang berisi nilai-nilai untuk variabel lingkungan fungsi Lambda ini. Untuk informasi selengkapnya tentang file variabel lingkungan, lihat [File variabel lingkungan](#).

`--event`, `-e` *PATH*

File JSON yang berisi data peristiwa yang diteruskan ke fungsi Lambda ketika dipanggil. Jika Anda tidak menentukan opsi ini, tidak ada peristiwa yang diterima. Untuk meng-input JSON dari `stdin`, Anda harus meneruskan nilai `'-'`. Untuk detail tentang format pesan acara dari berbagai AWS layanan, lihat [Bekerja dengan layanan lain](#) di Panduan AWS Lambda Pengembang.

`--force-image-build`

Menentukan apakah AWS SAMCLI harus membangun kembali gambar yang digunakan untuk memanggil fungsi Lambda dengan lapisan.

`--help`

Menunjukkan pesan ini dan keluar.

`--hook-name` *TEXT*

Nama hook yang digunakan untuk memperluas AWS SAMCLI fungsionalitas.

Nilai yang diterima: `terraform`.

`--invoke-image` *TEXT*

URI dari gambar kontainer yang ingin Anda gunakan untuk pemanggilan fungsi lokal. Secara default, AWS SAM menarik citra kontainer dari Amazon ECR Publik. Gunakan opsi ini untuk menarik citra dari lokasi lain.

Misalnya, `sam local invoke MyFunction --invoke-image amazon/aws-sam-cli-emulation-image-python3.8`.

`--layer-cache-basedir` *DIRECTORY*

Menentukan lokasi direktori dasar tempat layer yang digunakan templat Anda diunduh.

`--log-file, -l` *TEXT*

Berkas log untuk mengirim log waktu aktif.

`--no-event`

Memanggil fungsi dengan peristiwa kosong.

`--parameter-overrides`

(Opsional) String yang berisi penggantian AWS CloudFormation parameter yang dikodekan sebagai pasangan nilai kunci. Menggunakan format yang sama dengan AWS Command Line Interface (AWS CLI). Misalnya: `'ParameterKey=KeyPairName, ParameterValue = MyKey ParameterKey =InstanceType, ParameterValue =t1.micro'`.

Opsi ini tidak kompatibel dengan `--hook-name`.

`--profile` *TEXT*

Profil spesifik dari file kredensialmu yang mendapat AWS kredensialnya.

`--region` *TEXT*

AWS Wilayah untuk dikerahkan ke. Misalnya, `us-east-1`.

`--save-params`

Simpan parameter yang Anda berikan pada baris perintah ke file AWS SAM konfigurasi.

`--shutdown`

Mengemulasi peristiwa shutdown setelah pemanggilan selesai, untuk menguji penanganan ekstensi perilaku shutdown.

`--skip-prepare-infra`

Melewati tahap persiapan jika tidak ada perubahan infrastruktur yang dilakukan. Gunakan dengan `--hook-name` opsi.

--skip-pull-image

Secara default, Lambda AWS SAMCLI memeriksa lingkungan runtime jarak jauh terbaru Lambda dan memperbarui gambar lokal Anda secara otomatis agar tetap sinkron.

Tentukan opsi ini untuk melewati menarik Docker gambar terbaru untuk lingkungan runtime Lambda Anda.

--template, -t *PATH*

File AWS SAM template.

Opsi ini tidak kompatibel dengan --hook-name.

Note

Jika Anda menentukan opsi ini, hanya AWS SAM memuat template dan sumber daya lokal yang ditunjuknya.

--terraform-plan-file

Jalur relatif atau absolut ke file Terraform paket lokal Anda saat menggunakan AWS SAMCLI with Terraform Cloud. Opsi ini mengharuskan --hook-name diatur keterraform.

sam local start-api

Halaman ini memberikan informasi referensi untuk AWS Serverless Application Model `sam local start-api` subperintah Command Line Interface (AWS SAMCLI).

- Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#).
- Untuk dokumentasi tentang penggunaan AWS SAMCLI `sam local start-api` subperintah, lihat [Menggunakan sam local start-api](#).

`sam local start-api` Subperintah menjalankan AWS Lambda fungsi Anda secara lokal untuk menguji melalui host server HTTP lokal.

Penggunaan

```
$ sam local start-api <options>
```

Opsi

`--add-host` *LIST*

Meneruskan nama host ke pemetaan alamat IP ke file host wadah Docker. Parameter ini dapat dilewatkan beberapa kali.

Example

Contoh: `--add-host` *example.com:127.0.0.1*

`--beta-features` | `--no-beta-features`

Izinkan atau tolak fitur beta.

`--config-env` *TEXT*

Nama lingkungan yang menentukan nilai parameter default dalam file konfigurasi yang akan digunakan. Nilai default-nya adalah “default”. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--config-file` *PATH*

Jalur dan nama file dari file konfigurasi yang berisi nilai parameter default yang akan digunakan. Nilai default-nya adalah “samconfig.toml” di root direktori proyek. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--container-env-vars`

Opsional. Teruskan variabel lingkungan ke kontainer citra saat debugging lokal.

`--container-host` *TEXT*

Host kontainer Lambda yang teremulasi secara lokal. Nilai default-nya adalah `localhost`. Jika Anda ingin menjalankan AWS SAMCLI penampung Docker di macOS, Anda dapat menentukan `host.docker.internal` Jika Anda ingin menjalankan wadah pada host yang berbeda dari AWS SAMCLI, Anda dapat menentukan alamat IP dari host jarak jauh.

`--container-host-interface` *TEXT*

Alamat IP dari antarmuka jaringan host tempat kontainer port harus terikat. Nilai default-nya adalah `127.0.0.1`. Gunakan `0.0.0.0` untuk mengikat ke semua antarmuka.

`--debug`

Mengaktifkan logging debug untuk mencetak pesan debug yang dihasilkan oleh AWS SAMCLI dan menampilkan stempel waktu.

`--debug-args` *TEXT*

Argumen tambahan untuk diteruskan ke debugger.

`--debug-function`

Opsional. Menentukan fungsi Lambda untuk diberlakukan opsi debug ketika `--warm-containers` ditentukan. Parameter ini berlaku untuk `--debug-port`, `--debugger-path`, dan `--debug-args`.

`--debug-port`, `-d` *TEXT*

Ketika ditentukan, mulai kontainer fungsi Lambda dalam modus debug dan ekspos port ini pada host lokal.

`--debugger-path` *TEXT*

Jalan host ke debugger yang akan dipasang ke kontainer Lambda.

`--docker-network` *TEXT*

Nama atau ID dari jaringan Docker yang ada tempat Lambda Docker terhubung, bersama dengan jaringan bridge default. Jika tidak ditentukan, kontainer Lambda hanya terhubung ke jaringan Docker bridge default.

`--docker-volume-basedir`, `-v` *TEXT*

Lokasi direktori dasar tempat AWS SAM file ada. Jika Docker berjalan pada mesin jarak jauh, Anda harus memasang jalur tempat AWS SAM file ada di mesin Docker, dan memodifikasi nilai ini agar sesuai dengan mesin jarak jauh.

`--env-vars`, `-n` *PATH*

File JSON yang berisi nilai-nilai untuk variabel lingkungan fungsi Lambda ini.

`--force-image-build`

Menentukan apakah AWS SAM CLI harus membangun kembali gambar yang digunakan untuk memanggil fungsi dengan lapisan.

`--help`

Menunjukkan pesan ini dan keluar.

`--hook-name` *TEXT*

Nama hook yang digunakan untuk memperluas AWS SAMCLI fungsionalitas.

Nilai yang diterima: `terraform`.

`--host` *TEXT*

Nama host lokal atau alamat IP untuk diikatkan (default: '127.0.0.1').

`--invoke-image` *TEXT*

URI gambar kontainer yang ingin Anda gunakan untuk fungsi Lambda Anda. Secara default, AWS SAM tarik gambar kontainer dari Amazon ECR Public. Gunakan opsi ini untuk menarik citra dari lokasi lain.

Anda dapat menentukan opsi ini beberapa kali. Setiap instans dari opsi ini dapat mengambil baik string atau pasangan kunci-nilai. Jika Anda menentukan string, itu adalah URI dari gambar kontainer yang akan digunakan untuk semua fungsi dalam aplikasi Anda. Misalnya, `sam local start-api --invoke-image public.ecr.aws/sam/emu-python3.8`. Jika Anda menentukan pasangan kunci-nilai, kuncinya adalah nama sumber daya, dan nilainya adalah URI citra kontainer untuk digunakan untuk sumber daya tersebut. Sebagai contoh, `sam local start-api --invoke-image public.ecr.aws/sam/emu-python3.8 --invoke-image Function1=amazon/aws-sam-cli-emulation-image-python3.8`. Dengan pasangan kunci-nilai, Anda dapat menentukan citra kontainer yang berbeda untuk sumber daya yang berbeda.

`--layer-cache-basedir` *DIRECTORY*

Menentukan lokasi basedir tempat Layer yang digunakan templat Anda terunduh.

`--log-file, -l` *TEXT*

Berkas log untuk mengirim log waktu aktif.

`--parameter-overrides`

Opsional. String yang berisi penggantian AWS CloudFormation parameter yang dikodekan sebagai pasangan kunci-nilai. Gunakan format yang sama dengan AWS CLI—misalnya, 'ParameterKey=, ParameterValue MyKey ParameterKey = KeyPairNameInstanceType, ParameterValue =t1.micro'.

`--port, -p` *INTEGER*

Nomor port lokal untuk didengarkan (default: '3000').

`--profile` *TEXT*

Profil spesifik dari file kredensial Anda yang mendapat AWS kredensial.

`--region` *TEXT*

AWS Wilayah untuk disebarkan. Misalnya, us-east-1.

`--save-params`

Simpan parameter yang Anda berikan pada baris perintah ke file AWS SAM konfigurasi.

`--shutdown`

Mengemulasi peristiwa shutdown setelah pemanggilan selesai, untuk menguji penanganan ekstensi perilaku shutdown.

`--skip-prepare-infra`

Melewati tahap persiapan jika tidak ada perubahan infrastruktur yang dilakukan. Gunakan dengan `--hook-name` opsi.

`--skip-pull-image`

Menentukan apakah CLI harus melewati menarik ke bawah citra Docker terbaru untuk waktu aktif Lambda.

`--ssl-cert-file` *PATH*

Jalur ke file sertifikat SSL (default: Tidak ada). Saat menggunakan opsi ini, `--ssl-key-file` opsi juga harus digunakan.

`--ssl-key-file` *PATH*

Jalur ke file kunci SSL (default: Tidak ada). Saat menggunakan opsi ini, `--ssl-cert-file` opsi juga harus digunakan.

`--static-dir, -s` *TEXT*

Setiap aset statis (misalnya, JavaScript CSS//HTML) file yang terletak di direktori ini disajikan di. /

`--template, -t` *PATH*

File AWS SAM template.

 Note

Jika Anda menentukan opsi ini, hanya AWS SAM memuat template dan sumber daya lokal yang ditunjukkannya.

--terraform-plan-file

Jalur relatif atau absolut ke file Terraform paket lokal Anda saat menggunakan AWS SAMCLI with Terraform Cloud. Opsi ini mengharuskan --hook-name diatur keterraform.

--warm-containers *[EAGER | LAZY]*

Opsional. Menentukan bagaimana AWS SAMCLI mengelola kontainer untuk setiap fungsi.

Tersedia dua opsi:

EAGER: Kontainer untuk semua fungsi dimuat saat pertama dimulai dan terus ada di antara pemanggilan.

LAZY: Kontainer hanya dimuat ketika setiap fungsi pertama kali dipanggil. Kontainer tersebut terus ada untuk pemanggilan tambahan.

sam local start-lambda

Halaman ini memberikan informasi referensi untuk AWS Serverless Application Model `sam local start-lambda` subperintah Command Line Interface (AWS SAMCLI).

- Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#).
- Untuk dokumentasi tentang penggunaan AWS SAMCLI `sam local start-lambda` subperintah, lihat [Menggunakan sam local start-lambda](#).

`sam local start-lambda` Subperintah memulai endpoint lokal untuk ditiru. AWS Lambda

Penggunaan

```
$ sam local start-lambda <options>
```

Opsi

--add-host *LIST*

Meneruskan nama host ke pemetaan alamat IP ke file host wadah Docker. Parameter ini dapat dilewatkan beberapa kali.

Example

Contoh: --add-host *example.com:127.0.0.1*

`--beta-features` | `--no-beta-features`

Izinkan atau tolak fitur beta.

`--config-env` *TEXT*

Nama lingkungan yang menentukan nilai parameter default dalam file konfigurasi yang akan digunakan. Nilai default-nya adalah “default”. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--config-file` *PATH*

Jalur dan nama file dari file konfigurasi yang berisi nilai parameter default yang akan digunakan. Nilai default-nya adalah “samconfig.toml” di root direktori proyek. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--container-env-vars`

Opsional. Teruskan variabel lingkungan ke kontainer citra saat debugging lokal.

`--container-host` *TEXT*

Host kontainer Lambda yang teremulasi secara lokal. Nilai default-nya adalah `localhost`. Jika Anda ingin menjalankan AWS SAMCLI penampung Docker di macOS, Anda dapat menentukan `host.docker.internal` Jika Anda ingin menjalankan wadah pada host yang berbeda dari AWS SAMCLI, Anda dapat menentukan alamat IP dari host jarak jauh.

`--container-host-interface` *TEXT*

Alamat IP dari antarmuka jaringan host tempat kontainer port harus terikat. Nilai default-nya adalah `127.0.0.1`. Gunakan `0.0.0.0` untuk mengikat ke semua antarmuka.

`--debug`

Mengaktifkan logging debug untuk mencetak pesan debug yang dihasilkan oleh AWS SAMCLI dan menampilkan stempel waktu.

`--debug-args` *TEXT*

Argumen tambahan untuk diteruskan ke debugger.

`--debug-function`

Opsional. Menentukan fungsi Lambda untuk diberlakukan opsi debug ketika `--warm-containers` ditentukan. Parameter ini berlaku untuk `--debug-port`, `--debugger-path`, dan `--debug-args`.

`--debug-port, -d TEXT`

Ketika ditentukan, mulai kontainer fungsi Lambda dalam modus debug, dan ekspos port ini pada host lokal.

`--debugger-path TEXT`

Jalur host ke debugger yang akan dipasang ke dalam kontainer Lambda.

`--docker-network TEXT`

Nama atau ID jaringan Docker yang ada tempat Lambda Docker kontainer harus terhubung, bersama dengan jaringan bridge default. Jika ini ditentukan, kontainer Lambda hanya terhubung ke jaringan Docker bridge default.

`--docker-volume-basedir, -v TEXT`

Lokasi direktori dasar tempat AWS SAM file ada. Jika Docker berjalan pada mesin jarak jauh, Anda harus memasang jalur tempat AWS SAM file ada di mesin Docker, dan memodifikasi nilai ini agar sesuai dengan mesin jarak jauh.

`--env-vars, -n PATH`

File JSON yang berisi nilai-nilai untuk variabel lingkungan fungsi Lambda ini.

`--force-image-build`

Tentukan apakah CLI harus membangun kembali gambar yang digunakan untuk menjalankan fungsi dengan lapisan.

`--help`

Menunjukkan pesan ini dan keluar.

`--hook-name TEXT`

Nama hook yang digunakan untuk memperluas AWS SAMCLI fungsionalitas.

Nilai yang diterima: `terraform`.

`--host TEXT`

Nama host lokal atau alamat IP untuk diikatkan (default: '127.0.0.1').

`--invoke-image TEXT`

URI dari gambar kontainer yang ingin Anda gunakan untuk pemanggilan fungsi lokal. Secara default, AWS SAM tarik gambar kontainer dari Amazon ECR Public. Gunakan opsi ini untuk menarik citra dari lokasi lain.

Misalnya, `local start-lambda MyFunction --invoke-image amazon/aws-sam-cli-emulation-image-python3.8.`

`--layer-cache-basedir` *DIRECTORY*

Menentukan lokasi basedir tempat layer yang digunakan templat Anda terunduh.

`--log-file, -l` *TEXT*

Berkas log untuk mengirim log waktu aktif.

`--parameter-overrides`

Opsional. String yang berisi penggantian AWS CloudFormation parameter yang dikodekan sebagai pasangan kunci-nilai. Gunakan format yang sama dengan AWS CLI—misalnya, 'ParameterKey=, ParameterValue MyKey ParameterKey = KeyPairNameInstanceType, ParameterValue =t1.micro'. Opsi ini tidak kompatibel dengan `--hook-name`.

`--port, -p` *INTEGER*

Nomor port lokal untuk didengarkan (default: '3001').

`--profile` *TEXT*

Profil spesifik dari file kredensial Anda yang mendapat AWS kredensial.

`--region` *TEXT*

AWS Wilayah untuk dikerahkan ke. Misalnya, us-east-1.

`--save-params`

Simpan parameter yang Anda berikan pada baris perintah ke file AWS SAM konfigurasi.

`--shutdown`

Mengemulasi peristiwa shutdown setelah pemanggilan selesai, untuk menguji penanganan ekstensi perilaku shutdown.

`--skip-prepare-infra`

Melewati tahap persiapan jika tidak ada perubahan infrastruktur yang dilakukan. Gunakan dengan `--hook-name` opsi.

`--skip-pull-image`

Menentukan apakah CLI harus melewati menarik gambar Docker terbaru untuk runtime Lambda.

`--template, -t PATH`

File AWS SAM template.

Note

Jika Anda menentukan opsi ini, hanya AWS SAM memuat template dan sumber daya lokal yang ditunjukkannya. Opsi ini tidak kompatibel dengan `--hook-name`.

`--terraform-plan-file`

Jalur relatif atau absolut ke file Terraform paket lokal Anda saat menggunakan AWS SAMCLI with Terraform Cloud. Opsi ini mengharuskan `--hook-name` diatur ke `terraform`.

`--warm-containers [EAGER | LAZY]`

Opsional. Menentukan bagaimana AWS SAMCLI mengelola kontainer untuk setiap fungsi.

Tersedia dua opsi:

- **EAGER:** Kontainer untuk semua fungsi dimuat saat pertama dimulai dan terus ada di antara pemanggilan.
- **LAZY:** Kontainer hanya dimuat ketika setiap fungsi pertama kali dipanggil. Kontainer tersebut terus ada untuk pemanggilan tambahan.

sam logs

Halaman ini memberikan informasi referensi untuk AWS Serverless Application Model perintah Command Line Interface (AWS SAMCLI) `sam logs`.

Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#)

`sam logs` Perintah mengambil log yang dihasilkan oleh AWS Lambda fungsi Anda.

Penggunaan

```
$ sam logs <options>
```

Opsi

`--config-env` *TEXT*

Nama lingkungan yang menentukan nilai parameter default dalam file konfigurasi yang akan digunakan. Nilai default-nya adalah “default”. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--config-file` *PATH*

Jalur dan nama file dari file konfigurasi yang berisi nilai parameter default yang akan digunakan. Nilai default-nya adalah “samconfig.toml” di root direktori proyek. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--cw-log-group` *LIST*

Termasuk log dari grup CloudWatch log Log yang Anda tentukan. Jika Anda menentukan opsi ini bersama denganname, AWS SAM termasuk log dari grup log yang ditentukan selain log dari sumber daya bernama.

`--debug`

Mengaktifkan logging debug untuk mencetak pesan debug yang dihasilkan oleh AWS SAM CLI dan menampilkan stempel waktu.

`---end-time, e` *TEXT*

Mengambil log sampai saat ini. Waktu dapat berupa nilai-nilai relatif seperti '5 menit yang lalu', 'besok', atau stempel waktu yang diformat seperti '01-01-2018 10.10.10'.

`--filter` *TEXT*

Memungkinkan Anda menentukan ekspresi untuk menemukan log yang cocok dengan istilah, frasa, atau nilai-nilai dalam peristiwa log Anda dengan cepat. Ini bisa berupa kata kunci sederhana (misalnya, “error”) atau pola yang didukung oleh Amazon CloudWatch Logs. Untuk sintaksnya, lihat [dokumentasi Amazon CloudWatch Logs](#).

`--help`

Menunjukkan pesan ini dan keluar.

`--include-traces`

Termasuk jejak X-Ray dalam output log.

`--name, -n TEXT`

Nama sumber daya untuk mengambil log. Jika sumber daya ini adalah bagian dari AWS CloudFormation tumpukan, ini bisa menjadi ID logis dari sumber daya fungsi di AWS SAM template AWS CloudFormation/. Beberapa nama dapat diberikan dengan mengulangi parameter lagi. Jika sumber daya berada dalam tumpukan bersarang, nama dapat ditambahkan dengan nama tumpukan bersarang untuk menarik log dari sumber daya tersebut (/). NestedStackLogicalId ResourceLogicalId Jika nama sumber daya tidak diberikan, tumpukan yang diberikan akan dipindai dan informasi log akan ditarik untuk semua sumber daya yang didukung. Jika Anda tidak menentukan opsi ini, AWS SAM mengambil log untuk semua sumber daya di tumpukan yang Anda tentukan. Jenis sumber daya berikut didukung:

- `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`

Menentukan format output untuk log. Untuk mencetak log yang diformat, tentukan `text`. Untuk mencetak log sebagai JSON, tentukan `json`.

`--profile TEXT`

Profil spesifik dari file kredensial Anda yang mendapatkan kredensial AWS.

`--region TEXT`

Wilayah AWS tempat men-deploy. Misalnya, `us-east-1`.

`--save-params`

Simpan parameter yang Anda berikan pada baris perintah ke file AWS SAM konfigurasi.

`--stack-name TEXT`

Nama AWS CloudFormation tumpukan yang merupakan bagian dari sumber daya.

`--start-time, -s TEXT`

Mengambil log mulai saat ini. Waktu dapat berupa nilai-nilai relatif seperti '5 menit lalu', 'kemarin', atau stempel waktu yang diformat seperti '01-01-2018 10.10.10'. Default ke '10 menit yang lalu'.

`--tail, -t`

Mengikuti output log. Ini mengabaikan argumen waktu akhir dan terus mengambil log setelah log tersedia.

Contoh-contoh

Ketika fungsi Anda adalah bagian dari tumpukan AWS CloudFormation, Anda dapat mengambil log dengan menggunakan ID logis fungsi ketika Anda menentukan nama tumpukan.

```
$ sam logs -n HelloWorldFunction --stack-name myStack
```

Lihat log untuk rentang waktu tertentu menggunakan opsi `-s` (`--start-time`) dan `-e` (`--end-time`).

```
$ sam logs -n HelloWorldFunction --stack-name myStack -s '10min ago' -e '2min ago'
```

Anda juga dapat menambahkan `--tail` opsi untuk menunggu log baru dan melihatnya saat mereka tiba.

```
$ sam logs -n HelloWorldFunction --stack-name myStack --tail
```

Gunakan `--filter` opsi untuk menemukan log yang cocok dengan istilah, frasa, atau nilai dengan cepat dalam peristiwa log Anda.

```
$ sam logs -n HelloWorldFunction --stack-name myStack --filter "error"
```

Lihat log untuk sumber daya di tumpukan anak.

```
$ sam logs --stack-name myStack -n childStack/HelloWorldFunction
```

Log ekor untuk semua sumber daya yang didukung dalam aplikasi Anda.

```
$ sam logs --stack-name sam-app --tail
```

Ambil log untuk fungsi Lambda tertentu dan API Gateway API di aplikasi Anda.

```
$ sam logs --stack-name sam-app --name HelloWorldFunction --name HelloWorldRestApi
```

Ambil log untuk semua sumber daya yang didukung dalam aplikasi Anda, dan juga dari grup log yang ditentukan.

```
$ sam logs --cw-log-group /aws/lambda/myfunction-123 --cw-log-group /aws/lambda/myfunction-456
```

sam package

AWS Serverless Application Model Command Line Interface (AWS SAM CLI) mengemas AWS SAM aplikasi.

Perintah ini membuat .zip file kode dan dependensi Anda, dan mengunggah file ke Amazon Simple Storage Service (Amazon S3). AWS SAM memungkinkan enkripsi untuk semua file yang disimpan di Amazon S3. Kemudian mengembalikan salinan AWS SAM template Anda, menggantikan referensi ke artefak lokal dengan lokasi Amazon S3 tempat perintah mengunggah artefak.

Secara default ketika Anda menggunakan perintah ini, AWS SAMCLI mengasumsikan bahwa direktori kerja Anda saat ini adalah direktori root proyek Anda. Yang AWS SAMCLI pertama mencoba menemukan file template yang dibangun menggunakan [sam build](#) perintah, yang terletak di .aws-sam subfolder, dan diberi namatemplate.yaml. Selanjutnya, AWS SAMCLI mencoba untuk menemukan file template bernama template.yaml atau template.yml di direktori kerja saat ini. Jika Anda menentukan --template opsi, AWS SAMCLI perilaku default diganti, dan hanya akan mengemas AWS SAM templat itu dan sumber daya lokal yang ditunjuknya.

Note

[sam deploy](#) sekarang secara implisit melakukan fungsionalitas dari sam package. Anda dapat menggunakan perintah [sam deploy](#) langsung ke paket dan men-deploy aplikasi Anda.

Penggunaan

```
$ sam package <arguments> <options>
```


Argumen

ID Sumber Daya

ID fungsi Lambda ke paket.

Argumen ini opsional. Jika aplikasi Anda berisi satu fungsi Lambda, AWS SAM CLI akan mengemasnya. Jika aplikasi Anda berisi beberapa fungsi, berikan ID fungsi untuk mengemas satu fungsi.

Nilai yang valid: ID logis sumber daya atau ARN sumber daya.

Opsi

`--config-env` *TEXT*

Nama lingkungan yang menentukan nilai parameter default dalam file konfigurasi yang akan digunakan. Nilai default-nya adalah “default”. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--config-file` *PATH*

Jalur dan nama file dari file konfigurasi yang berisi nilai parameter default yang akan digunakan. Nilai default-nya adalah “samconfig.toml” di root direktori proyek. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--debug`

Mengaktifkan logging debug untuk mencetak pesan debug yang dihasilkan oleh AWS SAMCLI dan menampilkan stempel waktu.

`--force-upload`

Membatalkan file yang ada di bucket Amazon S3. Tentukan bendera ini untuk mengunggah artefak bahkan jika artefak tersebut cocok dengan artefak yang ada di bucket Amazon S3.

`--help`

Menunjukkan pesan ini dan keluar.

`--image-repository` *TEXT*

URI repositori Amazon Elastic Container Registry (Amazon ECR) tempat perintah ini mengunggah citra fungsi Anda. Diperlukan untuk fungsi dideklarasikan dengan tipe paket Image.

`--kms-key-id` *TEXT*

ID kunci AWS Key Management Service (AWS KMS) yang digunakan untuk mengenkripsi artefak yang diam di bucket Amazon S3. Jika opsi ini tidak ditentukan, maka AWS SAM gunakan kunci enkripsi yang dikelola Amazon S3.

`--metadata`

(Opsional) Peta metadata yang akan dilampirkan ke semua artefak yang direferensikan dalam templat Anda.

`--no-progressbar`

Jangan tampilkan baris progres saat mengunggah artefak ke Amazon S3.

`--output-template-file` *PATH*

Jalur ke file tempat perintah menulis templat terkemas. Jika Anda tidak menentukan jalur, perintah menulis templat ke output standar.

`--profile` *TEXT*

Profil spesifik dari file kredensialmu yang mendapat AWS kredensialnya.

`--region` *TEXT*

AWS Wilayah untuk dikerahkan ke. Misalnya, us-east-1.

`--resolve-s3`

Buat bucket Amazon S3 secara otomatis untuk digunakan untuk pengemasan. Jika Anda menentukan `--resolve-s3` opsi `--s3-bucket` dan opsi, maka kesalahan akan terjadi.

`--s3-bucket` *TEXT*

Nama bucket Amazon S3 tempat perintah ini mengunggah artefak Anda. Jika artefak Anda lebih besar dari 51.200 byte, maka opsi `--s3-bucket` atau opsi diperlukan. `--resolve-s3` Jika Anda menentukan `--resolve-s3` opsi `--s3-bucket` dan opsi, maka kesalahan akan terjadi.

`--s3-prefix` *TEXT*

Prefiks yang ditambahkan ke nama artefak diunggah ke bucket Amazon S3. Nama prefiks adalah nama jalur (nama folder) untuk bucket Amazon S3. Ini hanya berlaku untuk fungsi yang dilaporkan dengan tipe paket Zip.

`--save-params`


Simpan parameter yang Anda berikan pada baris perintah ke file AWS SAM konfigurasi.

`--signing-profiles` *LIST*

(Opsional) Daftar profil penandatanganan untuk menandatangani paket deployment Anda. Parameter ini mengambil daftar pasangan kunci-nilai, dengan penjeasan kunci adalah nama fungsi atau lapisan untuk menandatangani, dan nilai adalah profil penandatanganan, dengan pemilik profil opsional yang terbatas `:`. Misalnya, `FunctionNameToSign=SigningProfileName1`
`LayerNameToSign=SigningProfileName2:SigningProfileOwner`.

`--template-file`, `--template`, `-t` *PATH*

Path dan nama file tempat AWS SAM template Anda berada.

 Note

Jika Anda menentukan opsi ini, AWS SAM paket hanya template dan sumber daya lokal yang ditunjuknya.

`--use-json`

Output JSON untuk AWS CloudFormation template. YAML digunakan secara default.

sam pipeline bootstrap

Halaman ini memberikan informasi referensi untuk AWS Serverless Application Model `sam local pipeline bootstrap` subperintah Command Line Interface (AWS SAMCLI).

Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#).

`sam pipeline bootstrap` subperintah menghasilkan sumber daya AWS infrastruktur yang diperlukan untuk terhubung ke sistem CI/CD Anda. Langkah ini harus dijalankan untuk setiap tahap penerapan di pipeline Anda sebelum menjalankan `sam pipeline init` perintah.

Subperintah ini menyiapkan sumber daya AWS infrastruktur berikut:

- Opsi untuk mengonfigurasi izin pipa melalui:
 - Pengguna IAM alur dengan access key ID dan kredensial akses kunci rahasia untuk dibagi dengan sistem CI/CD.

Note

Kami merekomendasikan memutar kunci akses secara teratur. Untuk informasi selengkapnya, lihat [Memutar kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensial jangka panjang](#) di Panduan Pengguna IAM.

- Platform CI/CD yang didukung melalui OIDC. Untuk pengenalan tentang penggunaan OIDC dengan AWS SAM pipeline, buka. [Menggunakan otentikasi OIDC dengan pipeline AWS SAM](#)
- IAM role eksekusi AWS CloudFormation diterima oleh AWS CloudFormation untuk menerapkan aplikasi AWS SAM.
- Bucket Amazon S3 untuk menahan artefak AWS SAM.
- Opsional, repositori citra Amazon ECR untuk menahan paket deployment Lambda citra kontainer (jika Anda memiliki sumber daya dari tipe paket Image).

Penggunaan

```
$ sam pipeline bootstrap <options>
```

Opsi

```
--bitbucket-repo-uuid TEXT
```

UUID dari repositori Bitbucket. Opsi ini khusus untuk menggunakan Bitbucket OIDC untuk izin.

Note

Nilai ini dapat ditemukan di <https://bitbucket.org/workspace/repository/admin/addon/admin/pipelines/openid-connect>

```
--bucket TEXT
```

ARN dari ember Amazon S3 yang menampung artefak. AWS SAM

```
--ci-cd-provider TEXT
```

Platform CI/CD untuk pipa. AWS SAM

`--cloudformation-execution-role` *TEXT*

ARN dari peran IAM yang akan diasumsikan oleh AWS CloudFormation saat menerapkan tumpukan aplikasi. Berikan hanya jika Anda ingin menggunakan peran Anda sendiri. Jika tidak, perintah akan membuat peran baru.

`--config-env` *TEXT*

Nama lingkungan yang menentukan nilai parameter default dalam file konfigurasi yang akan digunakan. Nilai default-nya adalah **default**. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--config-file` *PATH*

Path dan nama file dari file konfigurasi yang berisi nilai parameter default yang akan digunakan. Nilai default-nya adalah `samconfig.toml` di root direktori proyek. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--confirm-changeset` | `--no-confirm-changeset`

Prompt untuk mengonfirmasi penyebaran sumber daya Anda.

`--create-image-repository` | `--no-create-image-repository`

Tentukan apakah akan membuat repositori citra Amazon ECR jika tidak ada yang disediakan. Repositori Amazon ECR menyimpan gambar kontainer fungsi Lambda, atau lapisan yang memiliki jenis paket. Image Default-nya adalah `--no-create-image-repository`.

`--debug`

Mengaktifkan logging debug dan mencetak pesan debug yang AWS SAM CLI dihasilkan, dan untuk menampilkan stempel waktu.

`--deployment-branch` *TEXT*

Nama cabang tempat penerapan akan terjadi. Opsi ini khusus untuk menggunakan GitHub Actions OIDC untuk izin.

`--github-org` *TEXT*

GitHub Organisasi tempat repositori milik. Jika tidak ada organisasi, masukkan nama pengguna pemilik repositori. Opsi ini khusus untuk menggunakan GitHub Actions OIDC untuk izin.

`--github-repo` *TEXT*

Nama GitHub repositori tempat penerapan akan terjadi. Opsi ini khusus untuk menggunakan GitHub Actions OIDC untuk izin.

`--gitlab-group` *TEXT*

GitLab Grup tempat repositori milik. Opsi ini khusus untuk menggunakan GitLab OIDC untuk izin.

`--gitlab-project` *TEXT*

Nama GitLab proyek. Opsi ini khusus untuk menggunakan GitLab OIDC untuk izin.

`--help`, `-h`


Menunjukkan pesan ini dan keluar.

`--image-repository` *TEXT*

ARN dari repositori gambar Amazon ECR yang menyimpan gambar kontainer fungsi Lambda, atau lapisan yang memiliki jenis paket. Image Jika tersedia, opsi `--create-image-repository` diabaikan. Jika tidak disediakan dan `--create-image-repository` ditentukan, perintah membuat satu.

`--interactive` | `--no-interactive`

Nonaktifkan prompt interaktif untuk parameter bootstrap dan gagal jika ada parameter yang diperlukan yang hilang. Nilai default-nya adalah `--interactive`. Untuk perintah ini, `--stage` adalah satu-satunya parameter yang diperlukan.

 Note

Jika `--no-interactive` ditentukan bersama dengan `--use-oidc-provider`, semua parameter yang diperlukan untuk penyedia OIDC Anda harus disertakan.

`--oidc-client-id` *TEXT*

ID klien dikonfigurasi untuk digunakan dengan penyedia OIDC Anda.

`--oidc-provider` [*github-actions* | *gitlab* | *bitbucket-pipelines*]

Nama penyedia CI/CD yang akan digunakan untuk izin OIDC. GitLab, GitHub, dan Bitbucket didukung.

`--oidc-provider-url` *TEXT*

URL untuk penyedia OIDC. Nilai harus dimulai dengan **https://**.

`--permissions-provider` [*oidc* | *iam*]

Pilih penyedia izin untuk mengambil peran eksekusi pipeline. Nilai default-nya adalah **iam**.

`--pipeline-execution-role` *TEXT*

ARN IAM role yang akan diterima oleh pengguna alur untuk beroperasi pada tahap ini. Berikan hanya jika Anda ingin menggunakan peran Anda sendiri. Jika tidak disediakan, perintah ini akan membuat peran baru.

`--pipeline-user` *TEXT*

Amazon Resource Name (ARN) dari pengguna IAM memiliki access key ID dan secret access key yang dibagi dengan sistem CI/CD. Hal ini digunakan untuk memberikan izin pengguna IAM ini untuk mengakses akun AWS yang sesuai. Jika tidak disediakan, perintah akan membuat pengguna IAM bersama dengan ID kunci akses dan kredensial kunci akses rahasia.

`--profile` *TEXT*

Profil spesifik dari file kredensial Anda yang mendapatkan kredensial AWS.

`--region` *TEXT*

Wilayah AWS tempat men-deploy. Misalnya, `us-east-1`.

`--save-params`

Simpan parameter yang Anda berikan pada baris perintah ke file AWS SAM konfigurasi.

`--stage` *TEXT*

Nama tahap deployment yang sesuai. Hal ini digunakan sebagai sufiks untuk sumber daya infrastruktur AWS yang dibuat.

Memecahkan masalah

Kesalahan: Parameter yang diperlukan tidak ada

Kapan `--no-interactive` ditentukan bersama dengan `--use-oidc-provider` dan salah satu parameter yang diperlukan tidak disediakan, pesan kesalahan ini akan ditampilkan bersama dengan deskripsi parameter yang hilang.

sam pipeline init

Halaman ini memberikan informasi referensi untuk AWS Serverless Application Model `sam local pipeline init` subperintah Command Line Interface (AWS SAMCLI).

Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#).

`aws sam pipeline init` menghasilkan file konfigurasi pipeline yang dapat digunakan oleh sistem CI/CD Anda untuk menyebarkan aplikasi tanpa server. AWS SAM

Sebelum menggunakan `aws sam pipeline init`, Anda harus mem-bootstrap sumber daya yang diperlukan untuk setiap tahap dalam alur Anda. Anda dapat melakukan ini dengan menjalankan `aws sam pipeline init --bootstrap` untuk dipandu melalui proses penyiapan dan pembuatan file konfigurasi, atau merujuk ke sumber daya yang telah Anda buat sebelumnya dengan perintah `aws sam pipeline bootstrap`.

Penggunaan

```
$ aws sam pipeline init <options>
```

Opsi

`--bootstrap`

Aktifkan mode interaktif yang memandu pengguna membuat Sumber daya infrastruktur AWS yang diperlukan.

`--config-env` *TEXT*

Nama lingkungan yang menentukan nilai parameter default dalam file konfigurasi yang akan digunakan. Nilai default-nya adalah `default`. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAM CLI berkas konfigurasi](#).

`--config-file` *TEXT*

Jalur dan nama file dari file konfigurasi yang berisi nilai parameter default yang akan digunakan. Nilai default-nya adalah `samconfig.toml` dalam direktori root proyek. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAM CLI berkas konfigurasi](#).

`--debug`

Menghidupkan logging debug untuk mencetak pesan debug yang AWS SAM CLI dihasilkan, dan untuk menampilkan stempel waktu.

`--help`, `-h`

Menunjukkan pesan ini dan keluar.

`--save-params`

Simpan parameter yang Anda berikan pada baris perintah ke file AWS SAM konfigurasi.

sam publish

Halaman ini memberikan informasi referensi untuk AWS Serverless Application Model perintah Command Line Interface (AWS SAMCLI) `sam publish`.

Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#).

`sam publish` Perintah menerbitkan AWS SAM aplikasi ke file. AWS Serverless Application Repository Perintah ini mengambil AWS SAM template yang dikemas dan menerbitkan aplikasi ke Wilayah yang ditentukan AWS.

Perintah `sam publish` mengharapkan templat AWS SAM menyertakan bagian Metadata yang berisi metadata aplikasi yang diperlukan untuk penerbitan. Di bagian Metadata, properti `LicenseUrl` dan `ReadmeUrl` harus merujuk ke bucket Amazon Simple Storage Service (Amazon S3), bukan file lokal. Untuk informasi lebih lanjut tentang bagian Metadata dari templat AWS SAM, lihat [Menerbitkan aplikasi tanpa server menggunakan AWS SAMCLI](#).

Secara default, `sam publish` membuat aplikasi sebagai privat. Sebelum akun AWS lainnya diizinkan untuk melihat dan men-deploy aplikasi Anda, Anda harus membaginya. Untuk informasi tentang berbagi aplikasi, lihat [Contoh Kebijakan Berbasis Sumber Daya AWS Serverless Application Repository](#) di Panduan Developer AWS Serverless Application Repository.

Note

Saat ini `sam publish` tidak mendukung penerbitan aplikasi yang di-nest yang ditentukan secara lokal. Jika aplikasi Anda berisi aplikasi yang di-nest, Anda harus menerbitkannya secara terpisah ke AWS Serverless Application Repository sebelum menerbitkan aplikasi induk Anda.

Penggunaan

```
$ sam publish <options>
```

Opsi

`--config-env` *TEXT*

Nama lingkungan yang menentukan nilai parameter default dalam file konfigurasi yang akan digunakan. Nilai default-nya adalah "default". Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLIberkas konfigurasi](#).

`--config-file` *PATH*

Jalur dan nama file dari file konfigurasi yang berisi nilai parameter default yang akan digunakan. Nilai default-nya adalah "samconfig.toml" di root direktori proyek. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLIberkas konfigurasi](#).

`--debug`

Menghidupkan logging debug untuk mencetak pesan debug yang AWS SAM CLI dihasilkan, dan untuk menampilkan stempel waktu.

`--help`

Menunjukkan pesan ini dan keluar.

`--profile` *TEXT*

Profil spesifik dari file kredensial Anda yang mendapatkan kredensial AWS.

`--region` *TEXT*

Wilayah AWS tempat men-deploy. Misalnya, us-east-1.

`--save-params`

Simpan parameter yang Anda berikan pada baris perintah ke file AWS SAM konfigurasi.

`--semantic-version` *TEXT*

(Opsional) Gunakan opsi ini untuk menyediakan versi semantik aplikasi Anda yang membatalkan properti `SemanticVersion` di bagian Metadata dari file templat. Untuk informasi selengkapnya tentang versioning semantik, lihat [Spesifikasi Versioning semantik](#).

`--template, -t` *PATH*

Jalur file templat AWS SAM [default: `template.[yaml|yml]`].

Contoh-contoh

Untuk mempublikasikan aplikasi:

```
$ sam publish --template packaged.yaml --region us-east-1
```

sam remote invoke

Halaman ini memberikan informasi referensi untuk AWS Serverless Application Model perintah Command Line Interface (AWS SAMCLI) `sam remote invoke`.

- Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#).
- Untuk dokumentasi tentang penggunaan AWS SAM CLI `sam remote invoke` perintah, lihat [Menggunakan sam remote invoke](#).

`sam remote invoke` Perintah memanggil sumber daya yang didukung di file. AWS Cloud

Penggunaan

```
$ sam remote invoke <arguments> <options>
```

Argumen

ID Sumber Daya

ID dari sumber daya yang seharusnya dipanggil.

Argumen ini menerima nilai-nilai berikut:

- Nama Sumber Daya Amazon (ARN) — ARN sumber daya.

Tip

Gunakan `sam list stack-outputs --stack-name <stack-name>` untuk mendapatkan ARN sumber daya Anda.

- Logical ID — ID logis dari sumber daya. Anda juga harus memberikan nama AWS CloudFormation tumpukan menggunakan `--stack-name` opsi.
- ID Fisik — ID fisik sumber daya. ID ini akan dibuat saat Anda menerapkan sumber daya menggunakan AWS CloudFormation.

i Tip

Gunakan `aws sam list resources --stack-name <stack-name>` untuk mendapatkan ID fisik sumber daya Anda.

Saat Anda memberikan ARN atau ID fisik:

Jika Anda memberikan ARN atau ID fisik, jangan berikan nama tumpukan. Ketika nama tumpukan disediakan menggunakan `--stack-name` opsi, atau ketika nama tumpukan didefinisikan dalam file konfigurasi Anda, secara otomatis AWS SAM CLI akan memproses ID sumber daya Anda sebagai nilai ID logis dari AWS CloudFormation tumpukan.

Bila Anda tidak memberikan ID sumber daya:

Jika Anda tidak memberikan ID sumber daya, tetapi memberikan nama tumpukan dengan `--stack-name` opsi, AWS SAM CLI akan mencoba untuk secara otomatis memanggil sumber daya di AWS CloudFormation tumpukan Anda menggunakan logika berikut:

1. Ini AWS SAM CLI akan mengidentifikasi jenis sumber daya dalam urutan berikut dan pindah ke langkah berikutnya setelah jenis sumber daya ditemukan di tumpukan Anda:
 - a. Lambda
 - b. Step Functions
 - c. Amazon SQS
 - d. Kinesis Data Streams
2. Jika jenis sumber daya memiliki sumber daya tunggal di tumpukan Anda, AWS SAM CLI maka akan memanggilnya. Jika beberapa sumber daya dari jenis sumber daya ada di tumpukan Anda, AWS SAM CLI akan mengembalikan kesalahan.

Berikut ini adalah contoh dari apa yang AWS SAM CLI akan dilakukan:

- Tumpukan yang berisi dua fungsi Lambda dan antrian Amazon SQS — Ini AWS SAM CLI akan menemukan jenis sumber daya Lambda dan pengembalian dan kesalahan karena tumpukan berisi lebih dari satu fungsi Lambda.
- Tumpukan yang berisi fungsi Lambda dan dua aplikasi Amazon Kinesis Data Streams — The AWS SAM CLI akan menemukan fungsi Lambda dan memanggilnya karena tumpukan berisi sumber daya Lambda tunggal.

- Tumpukan yang berisi satu antrian Amazon SQS dan dua aplikasi Kinesis Data Streams — The akan AWS SAM CLI menemukan antrian Amazon SQS dan memanggilnya karena tumpukan berisi satu antrian Amazon SQS.

Opsi

`--beta-features` | `--no-beta-features`

Izinkan atau tolak fitur beta.

`--config-env` *TEXT*

Tentukan lingkungan yang akan digunakan dari file AWS SAM CLI konfigurasi Anda.

Default: `default`

`--config-file` *FILENAME*

Tentukan jalur dan nama file konfigurasi Anda.

Untuk informasi selengkapnya tentang file konfigurasi, lihat [Mengkonfigurasi AWS SAMCLI](#).

Default: `samconfig.toml` di root direktori proyek Anda.

`--debug`

Aktifkan logging debug. Ini mencetak pesan debug dan stempel waktu yang dihasilkan oleh file. AWS SAM CLI

`--event`, `-e` *TEXT*

Acara untuk dikirim ke sumber daya target.

`--event-file` *FILENAME*

Path ke file yang berisi acara untuk dikirim ke sumber daya target.

`--help`, `-h`

Tampilkan pesan bantuan dan keluar.

`--output` [*text* | *json*]

Output hasil pemanggilan Anda dalam format output tertentu.

`json`— Metadata permintaan dan respons sumber daya dikembalikan dalam struktur JSON. Respons berisi output SDK lengkap.

`text`— Metadata permintaan dikembalikan dalam struktur teks. Respons sumber daya dikembalikan dalam format output dari sumber daya yang dipanggil.

`--parameter`

[Boto3](#) Parameter tambahan yang dapat Anda berikan ke sumber daya yang dipanggil.

Amazon Kinesis Data Streams

Parameter tambahan berikut dapat digunakan untuk menempatkan catatan dalam aliran data Kinesis:

- `ExplicitHashKey='string'`
- `PartitionKey='string'`
- `SequenceNumberForOrdering='string'`
- `StreamARN='string'`

Untuk deskripsi setiap parameter, lihat [Kinesis.client.PUT_RECORD](#).

AWS Lambda

Parameter tambahan berikut dapat digunakan untuk memanggil sumber daya Lambda dan menerima respons buffer:

- `ClientContext='base64-encoded string'`
- `InvocationType='[DryRun | Event | RequestResponse]'`
- `LogType='[None | Tail]'`
- `Qualifier='string'`

Parameter tambahan berikut dapat digunakan untuk memanggil sumber daya Lambda dengan streaming respons:

- `ClientContext='base64-encoded string'`
- `InvocationType='[DryRun | RequestResponse]'`
- `LogType='[None | Tail]'`
- `Qualifier='string'`

Untuk deskripsi setiap parameter, lihat yang berikut ini:

- [Lambda dengan respon buffer - Lambda.client.Invoke](#)
- [Lambda dengan streaming respons - Lambda.client.invoke_with_response_stream](#)

Amazon Simple Queue Service (Amazon SQS)

Parameter tambahan berikut dapat digunakan untuk mengirim pesan ke antrean Amazon SQS:

- DelaySeconds=*integer*
- MessageAttributes=*'json string'*
- MessageDeduplicationId=*'string'*
- MessageGroupId=*'string'*
- MessageSystemAttributes=*'json string'*

Untuk deskripsi setiap parameter, lihat [Sqs.client.send_message](#).

AWS Step Functions

Parameter tambahan berikut dapat digunakan untuk memulai eksekusi mesin negara:

- name=*'string'*
- traceHeader=*'string'*

Untuk deskripsi setiap parameter, lihat [SFN.Client.START_EXECUTION](#).

--profile *TEXT*

Profil spesifik dari file kredensialmu untuk mendapatkan AWS kredensialnya.

--region *TEXT*

Wilayah AWSSumber daya. Misalnya, us-east-1.

--stack-name *TEXT*

Nama AWS CloudFormation tumpukan yang dimiliki sumber daya.

--test-event-name *NAME*

Nama acara pengujian yang dapat dibagikan untuk diteruskan ke fungsi Lambda Anda.

Note

Opsi ini hanya mendukung fungsi Lambda.

sam remote test-event

Halaman ini memberikan informasi referensi untuk AWS Serverless Application Model perintah Command Line Interface (AWS SAMCLI) `sam remote test-event`.

- Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#).
- Untuk dokumentasi tentang penggunaan AWS SAM CLI `sam remote test-event` perintah, lihat [Menggunakan sam remote test-event](#).

`sam remote test-event` Perintah berinteraksi dengan peristiwa pengujian yang dapat dibagikan di registri EventBridge skema Amazon.

Penggunaan

```
$ sam remote test-event <options> <subcommand>
```

Opsi

`--help`, `-h`

Tampilkan pesan bantuan dan keluar.

Subperintah

delete

Hapus peristiwa pengujian yang dapat dibagikan dari registri EventBridge skema. Untuk informasi referensi lebih lanjut, lihat [sam remote test-event delete](#).

get

Dapatkan acara pengujian yang dapat dibagikan dari registri EventBridge skema. Untuk informasi referensi lebih lanjut, lihat [sam remote test-event get](#).

list

Buat daftar peristiwa pengujian yang dapat dibagikan untuk suatu AWS Lambda fungsi. Untuk informasi referensi lebih lanjut, lihat [sam remote test-event list](#).

put

Simpan acara dari file lokal ke registri EventBridge skema. Untuk informasi referensi lebih lanjut, lihat [sam remote test-event put](#).

sam remote test-event delete

Halaman ini memberikan informasi referensi untuk AWS Serverless Application Model `sam remote test-event delete` subperintah Command Line Interface (AWS SAMCLI).

- Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#).
- Untuk dokumentasi tentang penggunaan AWS SAM CLI `sam remote test-event` perintah, lihat [Menggunakan sam remote test-event](#).

`sam remote test-event delete` Subperintah menghapus peristiwa pengujian yang dapat dibagikan dari registri skema Amazon EventBridge .

Penggunaan

```
$ sam remote test-event delete <arguments> <options>
```

Argumen

ID Sumber Daya

ID AWS Lambda fungsi yang terkait dengan peristiwa pengujian yang dapat dibagikan.

Jika Anda memberikan ID logis, Anda juga harus memberikan nilai untuk AWS CloudFormation tumpukan yang terkait dengan fungsi Lambda menggunakan opsi. `--stack-name`

Nilai yang valid: ID logis sumber daya atau sumber dayaARN.

Opsi

```
--config-env TEXT
```

Nama lingkungan yang menentukan nilai parameter default dalam file konfigurasi yang akan digunakan. Nilai default-nya adalah “default”. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--config-file` *PATH*

Jalur dan nama file dari file konfigurasi yang berisi nilai parameter default yang akan digunakan. Nilai default-nya adalah "samconfig.toml" di root direktori proyek. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--help`, `-h`

Tampilkan pesan bantuan dan keluar.

`--name` *TEXT*

Nama acara uji yang dapat dibagikan untuk dihapus.

`--stack-name` *TEXT*

Nama AWS CloudFormation tumpukan yang terkait dengan fungsi Lambda.

Opsi ini diperlukan jika Anda memberikan ID logis fungsi Lambda sebagai argumen.

`sam remote test-event get`

Halaman ini memberikan informasi referensi untuk AWS Serverless Application Model `sam remote test-event get` subperintah Command Line Interface (AWS SAMCLI).

- Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#).
- Untuk dokumentasi tentang penggunaan AWS SAM CLI `sam remote test-event` perintah, lihat [Menggunakan sam remote test-event](#).

`sam remote test-event get` Subperintah mendapatkan acara pengujian yang dapat dibagikan dari registri EventBridge skema Amazon.

Penggunaan

```
$ sam remote test-event get <arguments> <options>
```

Argumen

ID Sumber Daya

ID AWS Lambda fungsi yang terkait dengan peristiwa pengujian yang dapat dibagikan untuk mendapatkan.

Jika Anda memberikan ID logis, Anda juga harus memberikan nilai untuk AWS CloudFormation tumpukan yang terkait dengan fungsi Lambda menggunakan opsi. `--stack-name`

Nilai yang valid: ID logis sumber daya atau sumber dayaARN.

Opsi

`--config-env` *TEXT*

Nama lingkungan yang menentukan nilai parameter default dalam file konfigurasi yang akan digunakan. Nilai default-nya adalah "default". Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--config-file` *PATH*

Jalur dan nama file dari file konfigurasi yang berisi nilai parameter default yang akan digunakan. Nilai default-nya adalah "samconfig.toml" di root direktori proyek. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--help`, `-h`

Tampilkan pesan bantuan dan keluar.

`--name` *TEXT*

Nama acara uji yang dapat dibagikan untuk didapatkan.

`--output-file` *FILENAME*

Jalur file dan nama untuk menyimpan acara di mesin lokal Anda.

Jika Anda tidak memberikan opsi ini, AWS SAM CLI akan menampilkan konten acara pengujian yang dapat dibagikan ke konsol Anda.

`--stack-name` *TEXT*

Nama AWS CloudFormation tumpukan yang terkait dengan fungsi Lambda.

Opsi ini diperlukan jika Anda memberikan ID logis fungsi Lambda sebagai argumen.

`sam remote test-event list`

Halaman ini memberikan informasi referensi untuk AWS Serverless Application Model `sam remote test-event list` subperintah Command Line Interface (AWS SAMCLI).

- Untuk pengantar AWS SAM CLI, lihat [Apa itu AWS SAM CLI?](#).
- Untuk dokumentasi tentang penggunaan AWS SAM CLI `sam remote test-event` perintah, lihat [Menggunakan sam remote test-event](#).

`sam remote test-event list`Subperintah mencantumkan peristiwa pengujian yang dapat dibagikan untuk AWS Lambda fungsi tertentu dari registri EventBridge skema Amazon.

Penggunaan

```
$ sam remote test-event list <arguments> <options>
```

Argumen

ID Sumber Daya

ID fungsi Lambda yang terkait dengan peristiwa pengujian yang dapat dibagikan.

Jika Anda memberikan ID logis, Anda juga harus memberikan nilai untuk AWS CloudFormation tumpukan yang terkait dengan fungsi Lambda menggunakan opsi. `--stack-name`

Nilai yang valid: ID logis sumber daya atau sumber dayaARN.

Opsi

`--config-env` *TEXT*

Nama lingkungan yang menentukan nilai parameter default dalam file konfigurasi yang akan digunakan. Nilai default-nya adalah "default". Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAM CLI berkas konfigurasi](#).

`--config-file` *PATH*

Jalur dan nama file dari file konfigurasi yang berisi nilai parameter default yang akan digunakan. Nilai default-nya adalah "samconfig.toml" di root direktori proyek. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAM CLI berkas konfigurasi](#).

`--help`, `-h`

Tampilkan pesan bantuan dan keluar.

`--stack-name` *TEXT*

Nama AWS CloudFormation tumpukan yang terkait dengan fungsi Lambda.

Opsi ini diperlukan jika Anda memberikan ID logis fungsi Lambda sebagai argumen.

`sam remote test-event put`

Halaman ini memberikan informasi referensi untuk AWS Serverless Application Model `sam remote test-event put` subperintah Command Line Interface (AWS SAMCLI).

- Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#).
- Untuk dokumentasi tentang penggunaan AWS SAM CLI `sam remote test-event` perintah, lihat [Menggunakan sam remote test-event](#).

`sam remote test-event put` Subperintah menyimpan peristiwa pengujian yang dapat dibagikan dari mesin lokal Anda ke registri EventBridge skema Amazon.

Penggunaan

```
$ sam remote test-event put <arguments> <options>
```

Argumen

ID Sumber Daya

ID AWS Lambda fungsi yang terkait dengan peristiwa pengujian yang dapat dibagikan.

Jika Anda memberikan ID logis, Anda juga harus memberikan nilai untuk AWS CloudFormation tumpukan yang terkait dengan fungsi Lambda menggunakan opsi. `--stack-name`

Nilai yang valid: ID logis sumber daya atau sumber dayaARN.

Opsi

`--config-env` *TEXT*

Nama lingkungan yang menentukan nilai parameter default dalam file konfigurasi yang akan digunakan. Nilai default-nya adalah “default”. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--config-file` *PATH*

Jalur dan nama file dari file konfigurasi yang berisi nilai parameter default yang akan digunakan. Nilai default-nya adalah "samconfig.toml" di root direktori proyek. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--file` *FILENAME*

Jalur file dan nama acara ke mesin lokal Anda.

Berikan - sebagai nilai nama file untuk dibaca stdin.

Opsi ini diperlukan.

`--force, -f`

Menimpa acara pengujian yang dapat dibagikan dengan nama yang sama.

`--help, -h`

Tampilkan pesan bantuan dan keluar.

`--name` *TEXT*

Nama untuk menyimpan acara pengujian yang dapat dibagikan sebagai.

Jika peristiwa pengujian yang dapat dibagikan dengan nama yang sama ada di registri EventBridge skema, AWS SAM CLI maka tidak akan menimpa. Untuk menimpa, tambahkan `--force` opsi.

`--output-file` *FILENAME*

Jalur file dan nama untuk menyimpan acara di mesin lokal Anda.

Jika Anda tidak memberikan opsi ini, AWS SAM CLI akan menampilkan konten acara pengujian yang dapat dibagikan ke konsol Anda.

`--stack-name` *TEXT*

Nama AWS CloudFormation tumpukan yang terkait dengan fungsi Lambda.

Opsi ini diperlukan jika Anda memberikan ID logis fungsi Lambda sebagai argumen.

sam sync

Halaman ini memberikan informasi referensi untuk AWS Serverless Application Model perintah Command Line Interface (AWS SAMCLI) `sam sync`.

- Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#).
- Untuk dokumentasi tentang penggunaan AWS SAMCLI, lihat [Menggunakan AWS SAMCLI](#).

`sam sync` Perintah menyinkronkan perubahan aplikasi lokal ke file. AWS Cloud

Penggunaan

```
$ sam sync <options>
```

Opsi

`--base-dir, -s` *DIRECTORY*

Selesaikan jalur relatif ke fungsi atau kode sumber lapisan sehubungan dengan direktori ini. Gunakan opsi ini untuk mengubah cara jalur relatif ke folder kode sumber diselesaikan. Secara default, jalur relatif diselesaikan sehubungan dengan lokasi templat AWS SAM.

Selain sumber daya dalam aplikasi root atau tumpukan yang Anda buat, opsi ini juga berlaku untuk aplikasi atau tumpukan bersarang. Selain itu, opsi ini berlaku untuk jenis dan properti sumber daya berikut:

- Tipe sumber daya: Properti AWS::Serverless::Function: CodeUri
- Tipe sumber daya: Atribut Sumber Daya AWS::Serverless::Function: Entri Metadata: DockerContext
- Tipe sumber daya: Properti AWS::Serverless::LayerVersion: ContentUri
- Tipe sumber daya: Properti AWS::Lambda::Function: Code
- Tipe sumber daya: Properti AWS::Lambda::LayerVersion: Content

`--build-image` *TEXT*

URI untuk [gambar kontainer](#) yang ingin Anda gunakan saat membangun aplikasi Anda. Secara default, AWS SAM menggunakan URI repositori gambar kontainer dari Amazon [Elastic Container Registry \(Amazon ECR\) Registry ECR\) Public](#). Tentukan opsi ini untuk menggunakan gambar yang berbeda.

Anda dapat menggunakan opsi ini beberapa kali dalam satu perintah. Setiap opsi menerima string atau pasangan kunci-nilai.

- String - Tentukan URI gambar kontainer yang akan digunakan semua sumber daya dalam aplikasi Anda. Berikut ini adalah contohnya:

```
$ sam sync --build-image amazon/aws-sam-cli-build-image-python3.8
```

- Pasangan kunci-nilai - Tentukan nama sumber daya sebagai kunci dan URI gambar kontainer yang akan digunakan dengan sumber daya itu sebagai nilai. Gunakan format ini untuk menentukan URI gambar kontainer yang berbeda untuk setiap sumber daya dalam aplikasi Anda. Berikut ini adalah contohnya:

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

Opsi ini hanya berlaku jika opsi `--use-container` ditentukan, jika tidak akan terjadi kesalahan.

`--build-in-source` | `--no-build-in-source`

Menyediakan `--build-in-source` untuk membangun proyek Anda langsung di folder sumber.

`--build-in-source` Opsi ini mendukung runtime dan metode build berikut:

- Runtime — Setiap Node.js runtime yang didukung oleh opsi. [sam init --runtime](#)
- Membangun metode —Makefile, esbuild.

`--build-in-source` Opsi ini tidak kompatibel dengan opsi berikut:

- `--use-container`

Default: `--no-build-in-source`

`--capabilities` *LIST*

Daftar kemampuan yang Anda tentukan untuk memungkinkan AWS CloudFormation untuk membuat tumpukan tertentu. Beberapa templat tumpukan mungkin menyertakan sumber daya yang dapat memengaruhi izin di Anda Akun AWS. Misalnya, dengan membuat pengguna baru AWS Identity and Access Management (IAM). Tentukan opsi ini untuk mengganti nilai default. Nilai-nilai yang valid meliputi:

- KEMAMPUAN_IAM
- CAPABILITY_NAMED_IAM
- CAPABILITY_RESOURCE_POLICY
- CAPABILITY_AUTO_EXPAND

Default: `CAPABILITY_NAMED_IAM` dan `CAPABILITY_AUTO_EXPAND`

--code

Secara default, AWS SAM sinkronisasi semua sumber daya dalam aplikasi Anda. Tentukan opsi ini untuk menyinkronkan hanya sumber daya kode, yang meliputi yang berikut ini:

- `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`

Untuk menyinkronkan sumber daya kode, AWS SAM gunakan API AWS layanan secara langsung, alih-alih AWS CloudFormation menerapkannya. Untuk memperbarui AWS CloudFormation tumpukan Anda, jalankan `sam sync --watch` atau `sam deploy`.

--config-env *TEXT*

Nama lingkungan yang menentukan nilai parameter default dalam file konfigurasi yang akan digunakan. Nilai default-nya adalah "default". Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAM CLI berkas konfigurasi](#).

--config-file *PATH*

Jalur dan nama file dari file konfigurasi yang berisi nilai parameter default yang akan digunakan. Nilai default-nya adalah "samconfig.toml" di root direktori proyek. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAM CLI berkas konfigurasi](#).

--dependency-layer | --no-dependency-layer

Menentukan apakah untuk memisahkan dependensi fungsi individu ke lapisan lain untuk mempercepat proses sinkronisasi.

Default: `--dependency-layer`

`--image-repository` *TEXT*

Nama repositori Amazon Elastic Container Registry (Amazon ECR) tempat perintah ini mengunggah citra fungsi Anda. Diperlukan untuk fungsi yang dilaporkan dengan tipe paket Image.

`--image-repositories` *TEXT*

Pemetaan fungsi ke URI repositori Amazon ECR mereka. Fungsi referensi dengan ID logisnya. Berikut ini adalah contohnya:

```
$ sam sync --image-repositories Function1=123456789012.dkr.ecr.us-east-1.amazonaws.com/my-repo
```

Anda dapat menentukan opsi ini beberapa kali dalam satu perintah.

`--kms-key-id` *TEXT*

ID dari kunci AWS Key Management Service (AWS KMS) yang digunakan untuk mengenkripsi artefak yang nonaktif di bucket Amazon S3. Jika Anda tidak menentukan opsi ini, AWS SAM gunakan kunci enkripsi yang dikelola Amazon S3.

`--metadata`

Peta metadata untuk dilampirkan ke semua artefak yang Anda referensikan di template Anda.

`--notification-arns` *LIST*

Daftar ARN topik Amazon Simple Notification Service (Amazon SNS) yang AWS CloudFormation hubungkan dengan tumpukan.

`--parameter-overrides`

Sebuah string yang berisi parameter AWS CloudFormation membatalkan pengkodean sebagai pasangan nilai kunci. Gunakan format yang sama seperti AWS Command Line Interface (AWS CLI). Misalnya, `ParameterKey=ParameterValue InstanceType=t1.micro`.

`--resource` *TEXT*

Menentukan jenis sumber daya untuk sinkronisasi. Untuk menyinkronkan beberapa sumber daya, Anda dapat menentukan opsi ini beberapa kali. Opsi ini didukung dengan `--code` opsi. Nilai harus menjadi salah satu sumber daya yang terdaftar di bawah `--code`. Misalnya, `--resource AWS::Serverless::Function --resource AWS::Serverless::LayerVersion`.

`--resource-id` *TEXT*

Menentukan ID sumber daya untuk disinkronkan. Untuk menyinkronkan beberapa sumber daya, Anda dapat menentukan opsi ini beberapa kali. Opsi ini didukung dengan `--code` opsi. Misalnya, `--resource-id Function1 --resource-id Function2`.

`--role-arn` *TEXT*

Nama Sumber Daya Amazon (ARN) dari peran IAM yang AWS CloudFormation diasumsikan saat menerapkan kumpulan perubahan.

`--s3-bucket` *TEXT*

Nama bucket Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3) tempat perintah ini mengunggah template Anda. AWS CloudFormation Jika template Anda lebih besar dari 51.200 byte, maka opsi `--s3-bucket` atau `--resolve-s3` opsi diperlukan. Jika Anda menentukan opsi `--s3-bucket` dan `--resolve-s3` opsi, maka terjadi kesalahan.

`--s3-prefix` *TEXT*

Awalan ditambahkan ke nama artefak yang Anda unggah ke bucket Amazon S3. Nama prefiks adalah nama jalur (nama folder) untuk bucket Amazon S3. Ini hanya berlaku untuk fungsi yang dideklarasikan dengan jenis Zip paket.

`--save-params`

Menyimpan parameter yang Anda berikan pada baris perintah ke file AWS SAM konfigurasi.

`--skip-deploy-sync` | `--no-skip-deploy-sync`

Menentukan `--skip-deploy-sync` untuk melewati sinkronisasi infrastruktur awal jika tidak diperlukan. Ini AWS SAM CLI akan membandingkan AWS SAM template lokal Anda dengan AWS CloudFormation template yang diterapkan dan melakukan penerapan hanya jika perubahan terdeteksi.

Menentukan `--no-skip-deploy-sync` untuk melakukan AWS CloudFormation penyebaran setiap kali `aws sam sync` dijalankan.

Untuk mempelajari selengkapnya, lihat [Lewati AWS CloudFormation penerapan awal](#).

Default: `--skip-deploy-sync`

`--stack-name` *TEXT*

Nama AWS CloudFormation tumpukan untuk aplikasi Anda.


Opsi ini diperlukan.

--tags *LIST*

Daftar tanda untuk menghubungkan dengan tumpukan yang dibuat atau diperbarui. AWS CloudFormation juga menyebarkan tanda ini ke sumber daya dalam tumpukan yang mendukungnya.

--template-file, --template, -t *PATH*


Nama jalur dan file tempat templat AWS SAM Anda berada.

 Note

Jika Anda menentukan opsi ini, maka AWS SAM gunakan hanya templat dan sumber daya lokal yang ditunjukkannya.

--use-container, -u

Jika fungsi Anda bergantung pada paket yang memiliki dependensi yang dikompilasi secara native, gunakan opsi ini untuk membangun fungsi Anda di dalam wadah -like. AWS Lambda Docker

 Note

Saat ini, opsi ini tidak kompatibel dengan --dependency-layer. Jika Anda menggunakan --use-container dengan --dependency-layer, AWS SAM CLI menginformasikan Anda dan melanjutkan dengan --no-dependency-layer.

--watch

Memulai proses yang mengawasi aplikasi lokal Anda untuk perubahan dan secara otomatis menyinkronkannya ke file. AWS Cloud Secara default, ketika Anda menentukan opsi ini, AWS SAM menyinkronkan semua sumber daya dalam aplikasi Anda saat Anda memperbaruinya. Dengan opsi ini, AWS SAM melakukan AWS CloudFormation penyebaran awal. Kemudian, AWS SAM gunakan API AWS layanan untuk memperbarui sumber daya kode. AWS SAM digunakan AWS CloudFormation untuk memperbarui sumber daya infrastruktur saat Anda memperbarui AWS SAM template Anda.

`--watch-exclude` *TEXT*

Mengecualikan file atau folder agar tidak diamati untuk perubahan file. Untuk menggunakan opsi ini, juga `--watch` harus disediakan.

Opsi ini menerima pasangan kunci-nilai:

- Kunci — ID logis dari fungsi Lambda dalam aplikasi Anda.
- Nilai - Nama file atau folder terkait untuk dikecualikan.

Saat Anda memperbarui file atau folder apa pun yang ditentukan dengan `--watch-exclude` opsi, tidak AWS SAM CLI akan memulai sinkronisasi. Namun, ketika pembaruan ke file atau folder lain memulai sinkronisasi, file atau folder ini akan disertakan dalam sinkronisasi itu.

Anda dapat memberikan opsi ini beberapa kali dalam satu perintah.

sam traces

Halaman ini memberikan informasi referensi untuk AWS Serverless Application Model perintah Command Line Interface (AWS SAMCLI) `sam traces`.

Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#).

`sam traces` Perintah mengambil AWS X-Ray jejak di Akun AWS di Wilayah AWS

Penggunaan

```
$ sam traces <options>
```

Opsi

`--config-env` *TEXT*

Nama lingkungan yang menentukan nilai parameter default dalam file konfigurasi yang akan digunakan. Nilai default-nya adalah "default". Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--config-file` *PATH*

Jalur dan nama file dari file konfigurasi yang berisi nilai parameter default yang akan digunakan. Nilai default-nya adalah "samconfig.toml" di root direktori proyek. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--end-time` *TEXT*

Mengambil jejak hingga saat ini. Waktu dapat berupa nilai-nilai relatif seperti '5 menit yang lalu', 'besok', atau stempel waktu yang diformat seperti '01-01-2018 10.10.10'.

`--output` *TEXT*

Menentukan format output untuk log. Untuk mencetak log yang diformat, tentukan `text`. Untuk mencetak log sebagai JSON, tentukan `json`.

`--save-params`

Simpan parameter yang Anda berikan pada baris perintah ke file AWS SAM konfigurasi.

`--start-time` *TEXT*

Mengambil jejak mulai saat ini. Waktu dapat berupa nilai-nilai relatif seperti '5 menit lalu', 'kemarin', atau stempel waktu yang diformat seperti '01-01-2018 10.10.10'. Default ke '10 menit yang lalu'.

`--tail`

Ekor output jejak. Ini mengabaikan argumen waktu akhir dan terus menampilkan jejak saat tersedia.

`--trace-id` *TEXT*

Pengidentifikasi unik untuk jejak X-Ray.

Contoh-contoh

Jalankan perintah berikut untuk mengambil jejak X-Ray dengan ID.

```
$ sam traces --trace-id tracing-id-1 --trace-id tracing-id-2
```

Jalankan perintah berikut untuk mengekor jejak X-Ray saat tersedia.

```
$ sam traces --tail
```

sam validate

Halaman ini memberikan informasi referensi untuk AWS Serverless Application Model perintah Command Line Interface (AWS SAMCLI) `sam validate`.

Untuk pengantar AWS SAMCLI, lihat [Apa itu AWS SAMCLI?](#).

`sam validate` Perintah memverifikasi apakah file AWS SAM template valid.

Penggunaan

```
$ sam validate <options>
```

Opsi

`--config-env` *TEXT*

Nama lingkungan yang menentukan nilai parameter default dalam file konfigurasi yang akan digunakan. Nilai default-nya adalah “default”. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--config-file` *PATH*

Jalur dan nama file dari file konfigurasi yang berisi nilai parameter default yang akan digunakan. Nilai default-nya adalah “samconfig.toml” di root direktori proyek. Untuk informasi selengkapnya tentang file konfigurasi, lihat [AWS SAMCLiberkas konfigurasi](#).

`--debug`

Mengaktifkan logging debug untuk mencetak pesan debug yang dihasilkan oleh AWS SAM CLI dan menampilkan stempel waktu.

`--lint`

Jalankan validasi linting pada template melalui `cfn-lint`. Buat file `cfnlintrc` konfigurasi untuk menentukan parameter tambahan. Untuk informasi selengkapnya, lihat [cfn-lint](#) di repositori. [AWS CloudFormation GitHub](#)

`--profile` *TEXT*

Profil spesifik dari file kredensial Anda yang mendapatkan kredensial AWS.

`--region` *TEXT*

Wilayah AWS tempat men-deploy. Misalnya, `us-east-1`.

`--save-params`

Simpan parameter yang Anda berikan pada baris perintah ke file AWS SAM konfigurasi.

`--template-file`, `--template`, `-t` *PATH*

File templat AWS SAM. Nilai default-nya adalah `template.[yaml|yml]`.

Jika `template` Anda ada di direktori kerja Anda saat ini dan diberi nama `template.[yaml|yml|json]`, opsi ini tidak diperlukan.

Jika Anda baru saja berlarisam build, opsi ini tidak diperlukan.

AWS SAM CLI berkas konfigurasi

AWS Serverless Application Model Command Line Interface (AWS SAM CLI) mendukung file konfigurasi tingkat proyek yang dapat Anda gunakan untuk mengkonfigurasi nilai parameter AWS SAM CLI perintah.

Untuk dokumentasi tentang membuat dan menggunakan file konfigurasi, lihat [Mengkonfigurasi AWS SAM CLI](#).

Topik

- [Pengaturan file konfigurasi default](#)
- [Format file konfigurasi yang didukung](#)
- [Tentukan file konfigurasi](#)
- [Dasar-dasar file konfigurasi](#)
- [Aturan nilai parameter](#)
- [Konfigurasi diutamakan](#)
- [Membuat dan memodifikasi file konfigurasi](#)

Pengaturan file konfigurasi default

AWS SAM menggunakan pengaturan file konfigurasi default berikut:

- Nama — `samconfig`.
- Lokasi — Di akar proyek Anda. Ini adalah lokasi yang sama dengan `template.yaml` file Anda.
- Format — TOML. Untuk mempelajari lebih lanjut, lihat [TOMB](#) di TOML dokumentasi.

Berikut ini adalah contoh struktur proyek yang mencakup nama file konfigurasi default dan lokasi:


```
sam-app
### README.md
### __init__.py
### events
### hello_world
### samconfig.toml
### template.yaml
### tests
```

Berikut ini adalah `samconfig.toml` file contoh:

```
...
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
```

Format file konfigurasi yang didukung

TOML dan [YAML | YML] format yang didukung. Lihat sintaks dasar berikut:

TOML

```
version = 0.1
[environment]
[environment.command]
[environment.command.parameters]
option = parameter value
```

YAML

```
version: 0.1
environment:
  command:
    parameters:
      option: parameter value
```

Tentukan file konfigurasi

Secara default, AWS SAM CLI mencari file konfigurasi dalam urutan sebagai berikut:

1. File konfigurasi kustom - Jika Anda menggunakan `--config-file` opsi untuk menentukan nama dan lokasi file, AWS SAM CLI cari file ini terlebih dahulu.
2. **samconfig.toml** File default — Ini adalah nama dan format file konfigurasi default, yang terletak di root proyek Anda. Jika Anda tidak menentukan file konfigurasi kustom, AWS SAM CLI cari file ini selanjutnya.
3. **samconfig.[yaml|yml]** file — Jika `samconfig.toml` tidak ada di root proyek Anda, AWS SAM CLI mencari file ini.

Berikut ini adalah contoh menentukan file konfigurasi kustom menggunakan `--config-file` opsi:

```
$ sam deploy --config-file myconfig.yaml
```

Dasar-dasar file konfigurasi

Environment

Lingkungan adalah pengenalan bernama yang berisi seperangkat pengaturan konfigurasi yang unik. Anda dapat memiliki beberapa lingkungan dalam satu AWS SAM aplikasi.

Nama lingkungan default adalah `default`.

Gunakan AWS SAM CLI `--config-env` opsi untuk menentukan lingkungan yang akan digunakan.

Perintah

Perintah adalah AWS SAM CLI perintah untuk menentukan nilai parameter untuk.

Untuk menentukan nilai parameter untuk semua perintah, gunakan `global` pengenalan.

Saat mereferensikan AWS SAM CLI perintah, ganti spasi () dan tanda hubung (-) dengan garis bawah (_). Lihat contoh berikut:

- `build`
- `local_invoke`
- `local_start_api`

Parameter

Parameter ditentukan sebagai pasangan kunci-nilai.

- Kuncinya adalah nama opsi AWS SAM CLI perintah.
- Nilai adalah nilai yang akan ditentukan.

Saat menentukan kunci, gunakan nama opsi perintah bentuk panjang dan ganti tanda hubung (-) dengan garis bawah (_). Berikut ini adalah beberapa contohnya:

- `region`
- `stack_name`
- `template_file`

Aturan nilai parameter

TOML

- Nilai Boolean bisa `true` atau `false`. Misalnya, `confirm_changeset = true`.
- Untuk nilai string, gunakan tanda kutip ("). Misalnya, `region = "us-west-2"`.
- Untuk nilai daftar, gunakan tanda kutip (") dan pisahkan setiap nilai menggunakan spasi ().
Misalnya: `capabilities = "CAPABILITY_IAM CAPABILITY_NAMED_IAM"`.

- Untuk nilai yang berisi daftar pasangan kunci-nilai, pasangan adalah spasi-delimited () dan nilai setiap pasangan dikelilingi oleh tanda kutip yang dikodekan (). \ " \ " Misalnya, tags = "project=\"my-application\" stage=\"production\"".
- Untuk nilai parameter yang dapat ditentukan beberapa kali, nilainya adalah array argumen. Misalnya: image_repositories = ["my-function-1=image-repo-1", "my-function-2=image-repo-2"].

YAML

- Nilai Boolean bisa true atau false. Misalnya, confirm_changeset: true.
- Untuk entri yang berisi nilai string tunggal, tanda kutip (") bersifat opsional. Misalnya, region: us-west-2. Ini termasuk entri yang berisi beberapa pasangan kunci-nilai yang disediakan sebagai string tunggal. Berikut ini adalah contohnya:

```
$ sam deploy --tags "foo=bar hello=world"
```

```
default:
  deploy:
    parameters:
      tags: foo=bar hello=world
```

- Untuk entri yang berisi daftar nilai, atau entri yang dapat digunakan beberapa kali dalam satu perintah, tentukan sebagai daftar string.

Berikut ini adalah contohnya:

```
$ sam remote invoke --parameter "InvocationType=Event" --parameter "LogType=None"
```

```
default:
  remote_invoke:
    parameter:
      - InvocationType=Event
      - LogType=None
```

Konfigurasi diutamakan

Saat mengonfigurasi nilai, prioritas berikut terjadi:

- Nilai parameter yang Anda berikan pada baris perintah lebih diutamakan daripada nilai yang sesuai dalam file konfigurasi dan Parameters bagian file template.
- Jika `--parameter-overrides` opsi digunakan pada baris perintah atau dalam file konfigurasi Anda dengan `parameter_overrides` kunci, nilainya lebih diutamakan daripada nilai di Parameters bagian file template.
- Dalam file konfigurasi Anda, entri yang disediakan untuk perintah tertentu lebih diutamakan daripada entri global. Dalam contoh berikut, `sam deploy` perintah akan menggunakan nama `stackmy-app-stack`.

MAKAM

```
[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
```

Membuat dan memodifikasi file konfigurasi

Membuat file konfigurasi

Saat Anda membuat aplikasi menggunakan `sam init`, `samconfig.toml` file default dibuat. Anda juga dapat membuat file konfigurasi secara manual.

Memodifikasi file konfigurasi

Anda dapat memodifikasi file konfigurasi secara manual. Juga, selama aliran AWS SAM CLI interaktif apa pun, nilai yang dikonfigurasi akan ditampilkan dalam tanda kurung (`[]`). Jika Anda memodifikasi nilai-nilai ini, AWS SAM CLI akan memperbarui file konfigurasi Anda.

Berikut ini adalah contoh aliran interaktif menggunakan `sam deploy --guided` perintah:

```
$ 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
```

Saat memodifikasi file konfigurasi Anda, AWS SAM CLI menangani nilai global sebagai berikut:

- Jika nilai parameter ada di `global` bagian file konfigurasi Anda, nilai AWS SAM CLI tidak akan menuliskan nilai ke bagian perintah tertentu.
- Jika nilai parameter ada di kedua bagian perintah `global` dan spesifik, AWS SAM CLI menghapus entri spesifik yang mendukung nilai global.

Mengelola AWS SAMCLI versi

Kelola versi AWS Serverless Application Model Command Line Interface (AWS SAMCLI) Anda melalui `upgrade`, `downgrade`, dan `uninstall`. Secara opsional, Anda dapat mengunduh dan menginstal build AWS SAM CLI malam.

Topik

- [Memutakhirkan AWS SAMCLI](#)
- [Menghapus instalasi AWS SAMCLI](#)
- [Beralih dari menggunakan Homebrew untuk mengelola AWS SAMCLI](#)
- [Mengelola AWS SAMCLI pembangunan malam](#)
- [Memasang AWS SAMCLI ke dalam lingkungan virtual menggunakan pip](#)
- [Mengelola AWS SAMCLI dengan Homebrew](#)
- [Memecahkan masalah](#)

Memutakhirkan AWS SAMCLI

Linux

Untuk memutakhirkan AWS SAM CLI di Linux, ikuti petunjuk penginstalan di [Instalasi AWS SAMCLI](#), tetapi tambahkan `--update` opsi ke perintah instal, sebagai berikut:

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

macOS

AWS SAMCLI harus ditingkatkan melalui metode yang sama yang digunakan untuk menginstalnya. Kami menyarankan Anda menggunakan penginstal paket untuk menginstal dan memutakhirkan AWS SAM CLI.

Untuk meng-upgrade AWS SAM CLI menggunakan installer paket, instal versi paket terbaru. Untuk petunjuk, lihat [Instalasi AWS SAMCLI](#).

Windows

Untuk memutakhirkan AWS SAMCLI, ulangi langkah instalasi Windows [Instalasi AWS SAMCLI](#) lagi.

Menghapus instalasi AWS SAMCLI

Linux

Untuk menghapus instalasi AWS SAM CLI di Linux, Anda harus menghapus direktori symlink dan instalasi dengan menjalankan perintah berikut:

1. Temukan symlink dan instal jalur.

- Temukan symlink menggunakan perintah `which`:

```
which sam
```

Output menunjukkan jalur tempat binari AWS SAM berada, contohnya:

```
/usr/local/bin/sam
```

- Temukan direktori yang diarahkan symlink untuk menggunakan perintah `ls`:

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

Pada contoh berikut, direktori instalasi adalah `/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. Hapus symlink.

```
sudo rm /usr/local/bin/sam
```

3. Hapus direktori instalasi.

```
sudo rm -rf /usr/local/aws-sam-cli
```

macOS

Copot pemasangan AWS SAM CLI melalui metode yang sama yang digunakan untuk menginstalnya. Kami menyarankan Anda menggunakan penginstal paket untuk menginstal AWS SAM CLI

Jika Anda menginstal AWS SAM CLI menggunakan penginstal paket, ikuti langkah-langkah ini untuk menghapus instalasi.

Untuk mencopot pemasangan AWS SAMCLI

1. Hapus AWS SAM CLI program dengan memodifikasi dan menjalankan yang berikut ini:

```
$ sudo rm -rf /path-to/aws-sam-cli
```


- a. ***sudo*** — Jika pengguna Anda memiliki izin menulis ke tempat AWS SAM CLI program diinstal, tidak sudo diperlukan. Sebaliknya, sudo diperlukan.
 - b. ***/path-to*** — Jalur ke tempat Anda menginstal program. AWS SAM CLI Lokasi default adalah `/usr/local`.
2. Hapus AWS SAM CLI \$PATH dengan memodifikasi dan menjalankan yang berikut ini:

```
$ sudo rm -rf /path-to-symlink-directory/sam
```

- a. ***sudo*** — Jika pengguna Anda memiliki izin menulis \$PATH, tidak sudo diperlukan. Sebaliknya, sudo diperlukan.
 - b. ***path-to-symlink-directory***- Variabel \$PATH lingkungan Anda. Lokasi default adalah `/usr/local/bin`.
3. Verifikasi bahwa telah AWS SAM CLI dihapus instalasinya dengan menjalankan yang berikut:

```
$ sam --version  
command not found: sam
```

Windows

Untuk menghapus instalasi AWS SAM CLI menggunakan Pengaturan Windows, ikuti langkah-langkah ini:

1. Dari menu Mulai, cari "Tambah atau hapus program".
2. Pilih hasil bernama AWS SAMCommand Line Interface, lalu pilih Uninstall untuk meluncurkan uninstaller.
3. Konfirmasikan bahwa Anda ingin menghapus instalasi. AWS SAM CLI

Beralih dari menggunakan Homebrew untuk mengelola AWS SAMCLI

Jika Anda menggunakan Homebrew untuk menginstal dan meningkatkan AWS SAMCLI, kami sarankan menggunakan metode yang AWS didukung. Ikuti petunjuk ini untuk beralih ke metode yang didukung.

Untuk beralih dari menggunakan Homebrew

1. Ikuti instruksi di [Menghapus instalasi CLI yang Homebrew diinstal AWS SAM](#) untuk menghapus instalasi versi Homebrew terkelola.
2. Ikuti petunjuk di [Instalasi AWS SAMCLI](#) untuk menginstal AWS SAM CLI menggunakan metode yang didukung.

Mengelola AWS SAMCLI pembangunan malam

Anda dapat mengunduh dan menginstal AWS SAM CLI nightly build. Ini berisi versi pra-rilis AWS SAM CLI kode yang mungkin kurang stabil daripada versi produksi. Saat diinstal, Anda dapat menggunakan nightly build dengan perintah. `aws-sam-cli install --nightly` Anda dapat menginstal dan menggunakan versi produksi dan pembuatan malam secara AWS SAM CLI bersamaan.

Note

Nightly build tidak berisi versi prarilis dari image build. Karena itu, membangun aplikasi tanpa server Anda dengan `--use-container` opsi menggunakan versi produksi terbaru dari image build.

Memasang AWS SAMCLI build malam

Untuk menginstal build AWS SAM CLI malam, ikuti petunjuk ini.

Linux

Anda dapat menginstal versi nightly build AWS SAM CLI pada platform Linux x86_64 menggunakan penginstal paket.

Untuk menginstal AWS SAMCLI build malam

1. Unduh penginstal paket dari [sam-cli-nightly](#) dalam aws-sam-cli GitHub repositori.
2. Ikuti langkah-langkah untuk [menginstal paket AWS SAMCLI](#) nightly build.

macOS

Anda dapat menginstal versi nightly build dari AWS SAM CLI on macOS, menggunakan penginstal paket nightly build.

Untuk menginstal AWS SAMCLI build malam

1. Unduh penginstal paket untuk platform Anda dari [sam-cli-nightly](#) dalam aws-sam-cli GitHubrepository.
2. Ikuti langkah-langkah untuk [menginstal paket AWS SAMCLI](#) nightly build.

Windows

Versi build malam tersedia dengan tautan unduhan ini: [AWS SAMCLInightly](#) build. AWS SAM CLI Untuk menginstal nightly build di Windows, lakukan langkah yang sama seperti di [Instalasi AWS SAMCLI](#), tetapi gunakan tautan unduhan build malam sebagai gantinya.

Untuk memverifikasi bahwa Anda telah menginstal versi nightly build, jalankan perintah. sam-nightly --version Output dari perintah ini ada dalam bentuk 1.X.Y.dev<YYYYMMDDHHmm>, misalnya:

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

Beralih dari Homebrew ke penginstal paket

Jika Anda menggunakan Homebrew untuk menginstal dan memutakhirkan build AWS SAM CLI malam dan ingin beralih menggunakan penginstal paket, ikuti langkah-langkah ini.

Untuk beralih dari Homebrew ke installer paket

1. Copot pemasangan build AWS SAM CLI malam yang Homebrew diinstal.

```
$ brew uninstall aws-sam-cli-nightly
```

2. Verifikasi bahwa AWS SAM CLI nightly build dihapus instalasinya dengan menjalankan yang berikut:


```
$ sam-nightly --version  
zsh: command not found: sam-nightly
```

3. Ikuti langkah-langkah di bagian sebelumnya untuk menginstal AWS SAM CLI nightly build.

Memasang AWS SAMCLI ke dalam lingkungan virtual menggunakan pip

Kami merekomendasikan menggunakan installer paket asli untuk menginstal file. AWS SAM CLI Jika Anda harus menggunakan pip, kami sarankan Anda menginstal AWS SAM CLI ke dalam lingkungan

virtual. Ini memastikan lingkungan instalasi yang bersih dan lingkungan yang terisolasi jika terjadi kesalahan.

 Note

Pada 24 Oktober 2023, AWS SAM CLI sedang menghentikan dukungan untuk Python 3.7. Untuk mempelajari selengkapnya, lihat [AWS SAM CLI menghentikan dukungan untuk Python 3.7](#).

Untuk menginstal AWS SAM CLI ke dalam lingkungan virtual

1. Dari direktori awal pilihan Anda, buat lingkungan virtual dan beri nama.

Linux / macOS

```
$ mkdir project
$ cd project
$ python3 -m venv venv
```

Windows

```
> mkdir project
> cd project
> py -3 -m venv venv
```

2. Aktifkan lingkungan virtual

Linux / macOS

```
$ . venv/bin/activate
```

Perubahan prompt untuk menunjukkan kepada Anda bahwa lingkungan virtual Anda aktif.

```
(venv) $
```

Windows

```
> venv\Scripts\activate
```

Perubahan prompt untuk menunjukkan kepada Anda bahwa lingkungan virtual Anda aktif.

```
(venv) >
```

3. Instal AWS SAM CLI ke dalam lingkungan virtual Anda.

```
(venv) $ pip install --upgrade aws-sam-cli
```

4. Verifikasi bahwa AWS SAM CLI sudah diinstal dengan benar.

```
(venv) $ sam --version  
SAM CLI, version 1.94.0
```

5. Anda dapat menggunakan perintah `deactivate` untuk keluar dari lingkungan virtual. Setiap kali Anda memulai sesi baru, Anda harus mengaktifkan kembali lingkungan.

Mengelola AWS SAMCLI dengan Homebrew

Note

Mulai September 2023, tidak AWS akan lagi mempertahankan Homebrew installer AWS dikelola untuk AWS SAM CLI (`aws/tap/aws-sam-cli`). Untuk terus menggunakan Homebrew, Anda dapat menggunakan penginstal dikelola komunitas (`aws-sam-cli`). Mulai September 2023, Homebrew perintah apa pun yang `aws/tap/aws-sam-cli` akan dialihkan ke referensi `aws-sam-cli`. Kami menyarankan Anda menggunakan metode [instalasi](#) dan [peningkatan](#) yang didukung kami.

Instalasi AWS SAMCLI menggunakan Homebrew

Note

Instruksi ini menggunakan AWS SAM CLI Homebrew penginstal yang dikelola komunitas. Untuk dukungan lebih lanjut, lihat repositori [homebrew-core](#).

Untuk menginstal AWS SAMCLI

1. Jalankan hal berikut:

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

2. Verifikasi instalasi:

```
$ sam --version
```

Setelah instalasi berhasil AWS SAMCLI, Anda akan melihat output seperti berikut:

```
SAM CLI, version 1.94.0
```

Memutakhirkan penggunaan AWS SAMCLHomebrew

Untuk memutakhirkan AWS SAM CLI penggunaanHomebrew, jalankan perintah berikut:

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

Menghapus instalasi CLI yang Homebrew diinstal AWS SAM

Jika diinstal menggunakanHomebrew, ikuti langkah-langkah ini untuk menghapus instalannya. AWS SAM CLI

Untuk mencopot pemasangan AWS SAMCLI

1. Jalankan hal berikut:

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

2. Verifikasi bahwa telah AWS SAM CLI dihapus instalasinya dengan menjalankan yang berikut:

```
$ sam --version  
command not found: sam
```

Beralih ke Homebrew penginstal terkelola komunitas

Jika Anda menggunakan Homebrew installer AWS terkelola (`aws/tap/aws-sam-cli`) dan lebih memilih untuk terus menggunakan Homebrew, sebaiknya beralih ke Homebrew installer terkelola komunitas (`aws-sam-cli`).

Untuk beralih dalam satu perintah, jalankan yang berikut ini:

```
$ brew uninstall aws-sam-cli && brew untap aws/tap && brew cleanup aws/tap && brew update && brew install aws-sam-cli
```

Ikuti petunjuk ini untuk menjalankan setiap perintah satu per satu.

Untuk beralih ke Homebrew penginstal terkelola komunitas

1. Copot pemasangan Homebrew versi AWS terkelola dari AWS SAMCLI:

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

2. Verifikasi bahwa AWS SAM CLI telah dihapus instalasinya:

```
$ which sam  
sam not found
```

3. Hapus AWS SAM CLI ketukan AWS terkelola:

```
$ brew untap aws/tap
```

Jika Anda menerima kesalahan seperti berikut ini, tambahkan `--force` opsi dan coba lagi.

```
Error: Refusing to untap aws/tap because it contains the following installed  
formulae or casks:  
aws-sam-cli-nightly
```

4. Hapus file cache untuk installer AWS terkelola:

```
$ brew cleanup aws/tap
```

5. Perbarui Homebrew dan semua rumus:

```
$ brew update
```

6. Instal versi terkelola komunitas dari AWS SAMCLI:

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

7. Verifikasi AWS SAM CLI bahwa berhasil diinstal:

```
$ sam --version  
SAM CLI, version 1.94.0
```

Memecahkan masalah

Jika Anda menemukan kesalahan saat menginstal atau menggunakan AWS SAMCLI, lihat [AWS SAMCLI pemecahan masalah](#).

Verifikasi integritas AWS SAMCLI penginstal

Saat menginstal AWS Serverless Application Model Command Line Interface (AWS SAMCLI) menggunakan penginstal paket, Anda dapat memverifikasi integritasnya sebelum instalasi. Ini adalah langkah opsional, tetapi sangat disarankan.

Dua opsi verifikasi yang tersedia untuk Anda adalah:

- Verifikasi file tanda tangan penginstal paket.
- Verifikasi nilai hash penginstal paket.

Jika tersedia untuk platform Anda, kami sarankan untuk memverifikasi opsi file tanda tangan. Opsi ini menawarkan lapisan keamanan ekstra karena nilai kunci diterbitkan di sini dan dikelola secara terpisah dari GitHub repositori kami.

Topik

- [Verifikasi file tanda tangan penginstal](#)
- [Verifikasi nilai hash](#)

Verifikasi file tanda tangan penginstal

Linux

arm64 - penginstal baris perintah

AWS SAM menggunakan [GnuPG](#) untuk menandatangani AWS SAM CLI installer.zip. Verifikasi dilakukan dalam langkah-langkah berikut:

1. Gunakan kunci publik utama untuk memverifikasi kunci publik penandatanganan.
2. Gunakan kunci publik penandatanganan untuk memverifikasi penginstal AWS SAM CLI paket.

Untuk memverifikasi integritas kunci publik penandatanganan

1. Salin kunci publik utama dan simpan ke mesin lokal Anda sebagai `.txt` file. Misalnya, *primary-public-key.txt*.

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
```

```
Version: GnuPG v2.0.22 (GNU/Linux)
```

```
mQINBGRuSzMBEADsqiw0y78w7F4+sshaMFRIwRGNRm94p5Qey2KMZBxekFtoryVD
D9jE0nvupx4tvhfBHz5EcUHCE0d14MTqdBy6vVAshozgxVb9RE8JpECn5lw7XC69
4Y7Gy1TKKQMEwtDXElkGxIFdUwWjSnPlzfnoXwQYGeE93CUS3h5dImp22Yk1Ct6
eGgHlcbg1X4L8EpFMj7GvcsU8f7ziVI/PyC1Xwy39Q8/I67ip5eU5ddx0/xHqrbL
YC7+8pJPbRMej2twT2LrcpWYAbprMtRoa6WfE0/thoo3xhHpIMHdPFAA86ZNGIN
kRLjGUg7jnPTRW40in3pCc8nT4Tfc1QERkHm641gTC/jUvpmQsM6h/FUVP2i5iE/
JHpJcMuL2Mg6zDo3x+3gTCf+Wqz3rZzxB+wQT3yryZs6efcQy7nR0iRxYBxCSXX0
2cNYzsYLB/bYaW8yqWIHD5IqKhW269gp2E5Khs60zgs3CorMb5/xHgXjUCVgcu8a
a8ncdf9fj13WS5p0ohetPb02ZjWv+MaqrZ0mUIgKbA4RpWZ/fu97P5BW9y1wmIDB
sWy0cMxg8M1vSdLytPieogaM0qMg3u5qXRGBr6Wmevkty0qgnmpGGc5zPiUbt0E8
CnFFqyxBpj5IOnG0KZGVihvn+iRrxrv6G07WW092+Dc6m94U0EEiBR7Qi0wARAQAB
tDRBV1MgU0FNIENSSSBQcmLtYXJ5IDxhd3Mtc2FtLWNsaS1wcm1tYXJ5QGFtYXpv
bi5jb20+iQI/BBMBCQApBQJkbksZAhsvBQkHhM4ABwsJCAcDAgEGFQgCCQoLBBYC
AwEChgECF4AACgkQQv1fen0tiFqTuhAAzi5+ju5UV0WqHkev0JS008T4QB8HcqAE
SV03mY6/j29knkcL8ubZP/DbpV7QpHPi2PB5qSXsiDTP3IYPbeY78zHSDjljaIK3
njJLMScFeGPyfPpwMsuY4nzrRIgAtXShPA8N/k4ZJcafnpNqKj7QnPxIC1KaIQWm
p0tvb8msUF3/s0UTa5Ys/1NRhVC0eGg32ogXGdojZA2kHZWdm9udLo4CDrDcrQT7
NtDcJASapXSQL63XfAS3snEc4e1941YxcjFYZ33rel8K9juyDZfi1s1WR/L3AviI
QFIaqSHzy0tP1oinUkoVwL8ThevKD3Ag9CZf1ZLzNCV7yq1F8R1hEZ4zce/3s9E1
WzCFsozb5HfE1AZonmrDh3Sy0EIBMCS6vG5dWnvJrAuSYv2rX38++K5Pr/MIAf0X
D0I1rtA+XDshNv91SwSy01t+iClawZAN09IXCiN1r0YcVQ1wzDFwCNWDgkwd0qS0
g0A2f8NF91E5nBbeEuYquo011Vy8+ICbg0Fs9LoWZlnVh7/RyY6ssowiU9vGUnHI
```

```
L8f9jqRspIz/Fm3JD86ntZxLVGkeZUz62FqErdohYfkFIVcv7G0NTEyrz5HL1npv
FJ0MR0HjrMrZrn0VZnwBKhpLocTsH+3t5It4ReYEX0f1DIOL/KRwPvjMvBVkXY5
hb1RVDQo0Wc=
=d9oG
-----END PGP PUBLIC KEY BLOCK-----
```

2. Impor kunci publik utama ke keyring Anda.

```
$ 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. Salin kunci publik penandatanganan dan simpan ke mesin lokal Anda sebagai .txt file. Misalnya, *signer-public-key.txt*.

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

```
mQINBGRtS20BEAC7GjaAwverrB1zNEu2q3EGI6HC37WzwL5dy30f4LirZOWS3piK
oKfTqPjXPrlCf1GL2mMqUSgSnpEbPNXuvWTW1CfSnnjwuH8ZqbvvUQyHJwQyYpKm
KMwb+8V0bzzQkMzDVqolYQCi5XyGpAuo3wroxXSzG6r/mIhbiq3aRnL+21o4X0Yk
r7q9bhBqbJhzjkm7N62PhPWmi/+EGdEBakA1pReE+cKjP2UAp5L6CPSHQ12fRKL
9BumitNfFHHs1JZgZSCCruiWny3XkUaXUEMfyoE9nNbfqNvuqV2KjWguZCXASgz2
ZSPF4DTVIBMfP+xrZGQSWdGU/67QdysDQW81TbF0jK9ZsRwwGC4kbg/K98IsCNHT
ril5RZbyr8pw3fw7jYjjI2E1AacRWp53iRzvutm5AruPpLfoKDQ/tKzBUYItBwlu
Z/diKgcqtW7xDlyqNyTN8xPPFqM02I8IsZ2Pd1131htdFiZMiin1RQG9pV9p2vHS
eQVY2uKcNvnA6vFCQYKXP7p0IwReuPNzDvECUsidw8VTakTqZsANT/bU17e4KuKn
+JgbNrK0asJX37sDb/9ruysozLv78ozYKJDLmC3yoRQ8DhEjviT4cnjORgNmvnZ
0a5AA/DJPQW4buRrXdxu+fITzBxQn2+G0/iDNCxtJaq5SYVBKjTmTWPUJwARAQAB
tDBBV1MgU0FNIENMSSBUZWFtIDxhd3Mtc2FtLWNsaS1zaWduZXJAYW1hem9uLmNv
bT6JAj8EEwEJACKfAMrtS20CGy8FCQPCZwAHCwkIBwMCAQYVCAIJCgsEFgIDAQIe
AQIXgAAKCRDHoF9D/grd+1E4D/4kJW65He2LNsBLTta71cGfsEXCf4zgIvkytS7U
3R36zMD8IEyWJjlZ+aPkIP8/jFJrF14pVHbU7vX85Iut1vV7m+8BgWt25mJhnoJ9
```

```

KPjXGra9mYP+Cj8zFACjvtl3NBAPodyfcfCTWsU3umF9ArOFICcrGCzHX2SS7wX5
h9n0vYRZxk5Qj5FsgskKAQLq33CKFAMlaqZnL5gWRvTeycSIxsyus+stX+8YBPC0
J64f7+y+MPIP1+m2nj1VXg1xLEMMVa08oWccOMiakgzDev3LCrPy+wdwn7Ut7oA
pna3DNy9aYND2lh6vUCJeJ+Yi1B12jYpzLcCLKrHUmLn9/rRSz70rbg8P181kfPu
G/M7CD5FwhxP3p4+0XoGwxQefrV2jqpSnbLae7xbYJiJAhbpbjWDQhuNGUbPcDmqk
aH0Q3XU8AonJ8YqaQ/q3VZ3JBiH3TbBr0Xsvd59cwxYyf83aJ/WLCb2P8y75zDad
ln0P713ThF5J/Afj9Hj09waFV0Z2WZZe4rU20JTAiXEtM8xsFMrc7TCUacJtJGs
u4kdBmXREcVpSz65h9ImSy2ner9qktnVVCW4mZPj63IhB37YtoLAMyz3a3R2RFNk
viEX8fo0TUg1FmwHoftxZ9P91QwLoTajkDhrh26ueIe45sG6Uxua2AP4Vo37cFfCj
ryV80okCHAQQAQkABgUCZG5MWAACKRBC/V96c62IWmg1D/9idU43kW8Zy8Af1j81
Am31I4d9ks0leeKRZqxo/SZ5rovF32D02nw7XRXq1+EbhgJaI3Qww0i0U0pfAMVT
4b9TdxH+n+tzqCHh3jZqmo9sw+c9WFXyJN1hU9bLzcHXS8h0TbyoE2EuXx56ds9
L/BWCcd+LIvapw0lggFfavVx/QF4C7nBKjnJ66+xxwfgVIKR7oG1qDiHMfp9ZWh5
HhEqZo/nrNhdY0h3sczEdqC2N6eIa8mgHffHZdKudDMXIXHbgdhW9pcZXDIktVf7
j9wehsW0yYXiRgR0dz7DI26AUG4JLh5FTtx9XuSBdEsI69Jd4dJuibmgtImzbZjn
7un8DJWIyqi7Ckk96Tr4oXB9mYAXaWLR4C9j5XJhMNZgk0ycuY2DADnbGmSb+1kA
ju77H4ff84+vMDwUzUt2Wwb+GjzXu2g6Wh+bWhGSirY1e1+6xYrI6beu1BDCFLq+
VZFE8WggjJHpwcl7CiqadfVIQaw4HY0jQFTSdwzPWhJvYjXF0hMkyCcjsbBtmB+z
/otfgySyQqThrD48RWS5GuyqCA+pK3UNmEJ11c1AXMdTn2VWInR1N0JNALQ2du3y
q8t1vMsErV0J7pkZ50F4ef17PE6DKrXX8ilwGFyVuX5ddyT/t9J5pC3sRwHWXVZx
GXwoX75FwIEHA3n5Q7rZ69Ea6Q==
=ZI07
-----END PGP PUBLIC KEY BLOCK-----

```

4. Impor kunci publik penandatanganan ke keyring Anda.

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

```

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

```

Perhatikan nilai kunci dari output. Misalnya, *FE0ADDFA*.

5. Gunakan nilai kunci untuk mendapatkan dan memverifikasi sidik jari kunci publik penandatanganan.

```
$ gpg --fingerprint FE0ADDFA
```

```

pub 4096R/FE0ADDFA 2023-05-23 [expires: 2025-05-22]
Key fingerprint = 37D8 BE16 0355 2DA7 BD6A 04D8 C7A0 5F43 FE0A DDFA

```

```
uid                AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>
```

Sidik jari harus sesuai dengan yang berikut:

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

Jika string sidik jari tidak cocok, jangan gunakan AWS SAM CLI penginstal. Eskalasi ke AWS SAM tim dengan [membuat masalah](#) di aws-sam-cli GitHub repositori.

6. Verifikasi tanda tangan kunci publik penandatanganan:

```
$ gpg --check-sigs FE0ADDFA

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

Jika Anda melihatnya `1 signature not checked due to a missing key`, ulangi langkah-langkah sebelumnya untuk mengimpor kunci publik utama dan penandatanganan ke keyring Anda.

Anda harus melihat nilai kunci untuk kunci publik primer dan kunci publik penandatanganan yang terdaftar.

Sekarang setelah Anda memverifikasi integritas kunci publik penandatanganan, Anda dapat menggunakan kunci publik penandatanganan untuk memverifikasi penginstal AWS SAM CLI paket.

Untuk memverifikasi integritas penginstal AWS SAMCLI paket

1. Dapatkan file tanda tangan AWS SAM CLI paket - Unduh file tanda tangan untuk penginstal AWS SAM CLI paket dengan menggunakan perintah berikut:

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

2. Verifikasi file tanda tangan — Berikan file yang diunduh `.sig` dan `.zip` file sebagai parameter ke `gpg` perintah. Berikut ini adalah contohnya:

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

Outputnya akan serupa dengan yang berikut ini:

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

- **WARNING: This key is not certified with a trusted signature!** Pesannya bisa diabaikan. Itu terjadi karena tidak ada rantai kepercayaan antara kunci PGP pribadi Anda (jika Anda memilikinya) dan kunci AWS SAM CLI PGP. Untuk informasi lebih lanjut, lihat [Web of trust](#).
- Jika output berisi frasa `BAD signature`, periksa apakah Anda melakukan prosedur dengan benar. Jika Anda terus mendapatkan respons ini, tingkatkan ke AWS SAM tim dengan [membuat masalah](#) di aws-sam-cli GitHub repositori dan hindari menggunakan file yang diunduh.

Good signature from "AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>" Pesan berarti bahwa tanda tangan diverifikasi dan Anda dapat melanjutkan dengan instalasi.

x86_64 - penginstal baris perintah

AWS SAM menggunakan [GnuPG](#) untuk menandatangani AWS SAM CLI installer.zip. Verifikasi dilakukan dalam langkah-langkah berikut:

1. Gunakan kunci publik utama untuk memverifikasi kunci publik penandatanganan.
2. Gunakan kunci publik penandatanganan untuk memverifikasi penginstal AWS SAM CLI paket.

Untuk memverifikasi integritas kunci publik penandatanganan

1. Salin kunci publik utama dan simpan ke mesin lokal Anda sebagai `.txt` file. Misalnya, *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+8pJPbRMej2twT2LrcpWYAbprMtRoa6WfE0/thoo3xhHpIMhdPFAA86ZNGIN
kRLjGUg7jnPTRW40in3pCc8nT4Tfc1QERkHm641gTC/jUvpmQsM6h/FUVP2i5iE/
JHpJcMuL2Mg6zDo3x+3gTCf+Wqz3rZzxB+wQT3yryZs6efcQy7nR0iRxYBxCSXX0
2cNYzsYLB/bYaW8yqWIHD5IqKhW269gp2E5Khs60zgS3CoRmb5/xHgXjUCVgcu8a
a8ncdf9fj13WS5p0ohetPb0Z2jWv+MaqrZ0mUIgKbA4RpWZ/fU97P5BW9y1wmIDB
sWy0cMxg8M1vSdLytPieogaM0qMg3u5qXRGBr6Wmevkty0qgnmpGGc5zPiUbtOE8
CnFFqyxBpj5I0nG0KZGVihvn+iRrxv6G07WW092+Dc6m94U0EEiBR7Qi0wARAQAB
tDRBV1MgU0FNIENMSSBQcm1tYXJ5IDxhd3Mtc2FtLWNsaS1wcm1tYXJ5QGFTYXpv
bi5jb20+iQI/BBMBCQApBQJkbksZAhsvBQkHhM4ABwsJCAcDAgEGFQgCCQoLBBYC
AwEChgECF4AACgkQQv1fen0tiFqTuhAAzi5+ju5UV0WqHKeV0JS008T4QB8HcqAE
SV03mY6/j29knkcL8ubZP/DbpV7QpHPI2PB5qSXsiDTP3IYPbeY78zHSDjljaIK3
njJLMScFeGPYfPpwMsuY4nzrRIgAtXShPA8N/k4ZJcafnpNqKj7QnPxIC1KaIQWm
p0tvb8msUF3/s0UTa5Ys/1NRhVC0eGg32ogXGdojZA2kHZWdm9udLo4CDrDcrQT7
NtDcJASapXSQL63XfAS3snEc4e1941YxcjFYZ33rel8K9juyDZfi1s1WR/L3AviI
QFIaqSHzy0tP1oinUkoVwL8ThevKD3Ag9CZf1ZLzNCV7yq1F8R1hEz4zcE/3s9E1
WzCFsozb5HfE1AZonmrDh3Sy0EIBMCS6vG5dWnvJrAuSYv2rX38++K5Pr/MIAf0X
DOI1rtA+XDshNv91SwSy0lt+iClawZAN09IXCiN1r0YcVQlwDFwCNWDgkwd0qS0
g0A2f8NF91E5nBbeEuYquo011Vy8+ICbg0F9LoWZlnVh7/RyY6ssowiU9vGUNHI
L8f9jqRspIz/Fm3JD86ntZxLVGkeZuZ62FqErdohYfkFIVcv7GONTEyrz5HL1npv
FJ0MR0HjrMrZrn0VZnwBKhpLocTsH+3t5It4ReYEX0f1DIOL/KRwPvjMvBVkXY5
hb1RVDQo0Wc=
=d9oG
-----END PGP PUBLIC KEY BLOCK-----

```

2. Impor kunci publik utama ke keyring Anda.

```
$ 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. Salin kunci publik penandatanganan dan simpan ke mesin lokal Anda sebagai .txt file. Misalnya, *signer-public-key.txt*.

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

mQINBGRtS20BEAC7GjaAwverrB1zNEu2q3EGI6HC37WzwL5dy30f4LirZ0WS3piK
oKfTqPjXPrLCf1GL2mMqUSgSnpEbPNXuvWTW1CfSnnjwuH8ZqbvvUQyHJwQyYpKm
KMwb+8V0bzzQkMzDVqo1YQCi5XyGpAuo3wroxXSzG6r/mIhbiq3aRnL+21o4X0Yk
r7q9bhBqbJhzjkm7N62PhPWmi/+EGdEBakA1pReE+cKjP2UAp5L6CPSHQ12fRKL
9BumitNfFHHs1JZgZSCCruiWny3XkUaXUEMfyoE9nNbfqNvuqV2KjWguZCXASgz2
ZSPF4DTVIBMfP+xrZGQSWdGU/67QdysDQW81TbF0jK9ZsRwwGC4kbg/K98IsCNHT
ril5RZbyr8pw3fw7jYjjI2E1AacRWp53iRzvtm5AruPpLfoKDQ/tKzBUYItBwLu
Z/diKgcqtW7xDlyqNyTN8xFPFqM02I8IsZ2Pd1131htdFiZMiin1RQG9pV9p2vHS
eQVY2uKCnvnA6vFCQYKXP7p0IwReuPNzDvECUsidw8VTakTqZsANT/bU17e4KuKn
+JgbNrk0asJX37sDb/9ruysozLvy78ozYKJDLmC3yoRQ8DhEjviT4cnjORgNmvnZ
0a5AA/DJPQW4buRrXdxu+fITzBxQn2+G0/iDNCxtJaq5SYVBKjTmTWPUJwARAQAB
tDBBV1MgU0FNIENMSSBUZWFtIDxhd3Mtc2FtLWNsaS1zaWduZXJAYW1hem9uLmNv
bT6JAJ8EEwEJACKfAMRtS20CGy8FCQPCZwAHCwkIBwMCAQYVCAIJCgsEFgIDAQIe
AQIXgAAKCRDHoF9D/grd+1E4D/4kJW65He2LNSbLTta71cGfsEXCf4zgIvkytS7U
3R36zMD8IEyWJj1Z+aPkIP8/jFJrF14pVHbU7vX85Iut1vV7m+8BgWt25mJhnoJ9
KPjXGra9mYP+Cj8zFACjvtl3NBAPodyfcfCTWsU3umF9Ar0FICcrGCzHX2SS7wX5
h9n0vYRZxk5Qj5FsgskKAQLq33CKFAM1aqZnL5gWRvTeycSIxsius+stX+8YBPC0
J64f7+y+MPIP1+m2nj1VXg1xLEMMVa08oWcc0MiakgzDev3LCrPy+wdwdn7Ut7oA
pna3DNy9aYnd21h6vUCJeJ+Yi1B12jYpzLcCLKrHUm1n9/rRSz70rbg8P181kfPu
G/M7CD5FwhxP3p4+0XoGwxQefrV2jqpSnbLae7xbYJiJAhbpiWDQhuNGUbPcDmqk
aH0Q3XU8AonJ8YqaQ/q3VZ3JBiH3TbBr0Xsvd59cwxYyf83aJ/WLCb2P8y75zDad
ln0P713ThF5J/Afj9Hj09waFV0Z2W2ZZe4rU20JTAiXEtM8xsFMrc7TCUacJtJGs
u4kdBmXREcVpSz65h9ImSy2ner9qktnVVCW4mZPj63IhB37YtoLAMyz3a3R2RFNk
viEX8fo0TUg1FmwHoftxZ9P91QwLoTajkDrh26ueIe45sG6Uxua2AP4Vo37cFfcj
ryV80okCHAQQAQkABgUCZG5MWAACKRBC/V96c62IWmg1D/9idU43kW8Zy8Af1j81
Am31I4d9ks0leeKRZqxo/SZ5rovF32D02nw7XRXq1+EbhgJaI3Qww0i0U0pfAMVT
4b9TdxH+n+tzqCHh3jZqmo9sw+c9WFXyJN1hU9bLzCHXs8h0TbyoE2EuXx56ds9
L/BWCcd+LIvawp0l9gFfavVx/QF4C7nBKjnJ66+xxwfgVIKR7oGlqDiHMfp9ZWh5
HhEqZo/nrNhdy0h3sczEdqC2N6eIa8mgHffHZdKudDMXIXHbgdhW9pcZXDiktVf7
j9wehsW0yYXiRgR0dz7DI26AUG4JLh5FTtx9XuSbDesI69Jd4dJuibmgtImzbZjn
7un8DJWIyqi7Ckk96Tr4oXB9mYAXaW1R4C9j5XJhMNZgk0ycuY2DADnbGmSb+1kA
ju77H4ff84+vMDwUzUt2Wwb+GjzXu2g6Wh+bWhGSirYle1+6xYrI6beu1BDCFLq+
```

```
VZFE8WggjJHpwcL7CiqadfVIQaw4HY0jQFTSdwzPWhJvYjXF0hMkyCcjsbtmB+z
/otfgySyQqThrD48RWS5GuyqCA+pK3UNmEJ11c1AXMdTn2VWInR1N0JNALQ2du3y
q8t1vMsErVOJ7pkZ50F4ef17PE6DKrXX8ilwGFyVuX5ddyT/t9J5pC3sRwHWXVZx
GXwoX75FwIEHA3n5Q7rZ69Ea6Q==
=ZI07
-----END PGP PUBLIC KEY BLOCK-----
```

4. Impor kunci publik penandatanganan ke keyring Anda.

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

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

Perhatikan nilai kunci dari output. Misalnya, **FE0ADDFA**.

5. Gunakan nilai kunci untuk mendapatkan dan memverifikasi sidik jari kunci publik penandatanganan.

```
$ gpg --fingerprint FE0ADDFA
```

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

Sidik jari harus sesuai dengan yang berikut:

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

Jika string sidik jari tidak cocok, jangan gunakan AWS SAM CLI penginstal. Eskalasi ke AWS SAM tim dengan [membuat masalah](#) di aws-sam-cli GitHub repositori.

6. Verifikasi tanda tangan kunci publik penandatanganan:

```
$ gpg --check-sigs FE0ADDFA
```

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



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

Jika Anda melihatnya `signature not checked due to a missing key`, ulangi langkah-langkah sebelumnya untuk mengimpor kunci publik utama dan penandatanganan ke keyring Anda.

Anda harus melihat nilai kunci untuk kunci publik primer dan kunci publik penandatanganan yang terdaftar.

Sekarang setelah Anda memverifikasi integritas kunci publik penandatanganan, Anda dapat menggunakan kunci publik penandatanganan untuk memverifikasi penginstal AWS SAM CLI paket.

Untuk memverifikasi integritas penginstal AWS SAM CLI paket

1. Dapatkan file tanda tangan AWS SAM CLI paket - Unduh file tanda tangan untuk penginstal AWS SAM CLI paket dengan menggunakan perintah berikut:

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

2. Verifikasi file tanda tangan — Berikan file yang diunduh `.sig` dan `.zip` file sebagai parameter ke `gpg` perintah. Berikut ini adalah contohnya:

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

Outputnya akan serupa dengan yang berikut ini:

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

- **WARNING: This key is not certified with a trusted signature!** Pesannya bisa diabaikan. Itu terjadi karena tidak ada rantai kepercayaan antara kunci PGP pribadi Anda (jika Anda memilikinya) dan kunci AWS SAM CLI PGP. Untuk informasi lebih lanjut, lihat [Web of trust](#).

- Jika output berisi frasa `BAD signature`, periksa apakah Anda melakukan prosedur dengan benar. Jika Anda terus mendapatkan respons ini, tingkatkan ke AWS SAM tim dengan [membuat masalah](#) di `aws-sam-cli` GitHub repositori dan hindari menggunakan file yang diunduh.

Good signature from "AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>" Pesan berarti bahwa tanda tangan diverifikasi dan Anda dapat melanjutkan dengan instalasi.

macOS

GUI dan penginstal baris perintah

Anda dapat memverifikasi integritas file tanda tangan penginstal AWS SAM CLI paket dengan menggunakan `pkgutil` alat atau secara manual.

Untuk memverifikasi menggunakan `pkgutil`

1. Jalankan perintah berikut, berikan jalur ke installer yang diunduh di mesin lokal Anda:

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

Berikut ini adalah contohnya:

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

2. Dari output, cari SHA256 fingerprint untuk Developer ID Installer: AMZN Mobile LLC. Berikut ini adalah contohnya:

```
Package "aws-sam-cli-macos-arm64.pkg":
  Status: signed by a developer certificate issued by Apple for distribution
  Notarization: trusted by the Apple notary service
  Signed with a trusted timestamp on: 2023-05-16 20:29:29 +0000
  Certificate Chain:
    1. Developer ID Installer: AMZN Mobile LLC (94KV3E626L)
       Expires: 2027-06-28 22:57:06 +0000
       SHA256 Fingerprint:
         49 68 39 4A BA 83 3B F0 CC 5E 98 3B E7 C1 72 AC 85 97 65 18 B9 4C
         BA 34 62 BF E9 23 76 98 C5 DA
       -----
```

```

2. Developer ID Certification Authority
Expires: 2031-09-17 00:00:00 +0000
SHA256 Fingerprint:
    F1 6C D3 C5 4C 7F 83 CE A4 BF 1A 3E 6A 08 19 C8 AA A8 E4 A1 52 8F
    D1 44 71 5F 35 06 43 D2 DF 3A
-----
3. Apple Root CA
Expires: 2035-02-09 21:40:36 +0000
SHA256 Fingerprint:
    B0 B1 73 0E CB C7 FF 45 05 14 2C 49 F1 29 5E 6E DA 6B CA ED 7E 2C
    68 C5 BE 91 B5 A1 10 01 F0 24

```

3. Developer ID Installer: AMZN Mobile LLC SHA256 fingerprintHarus cocok dengan nilai berikut:

```

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

```

Jika string sidik jari tidak cocok, jangan gunakan AWS SAM CLI penginstal. Eskalasi ke AWS SAM tim dengan [membuat masalah](#) di aws-sam-cli GitHub repositori. Jika string sidik jari tidak cocok, Anda dapat bergerak maju dengan menggunakan penginstal paket.

Untuk memverifikasi penginstal paket secara manual

- Lihat [Cara memverifikasi keaslian pembaruan perangkat lunak Apple yang diunduh secara manual](#) di situs web dukungan Apple.

Windows

AWS SAMCLInstaller dikemas sebagai MSI file untuk Windows sistem operasi.

Untuk memverifikasi integritas penginstal

1. Klik kanan pada installer dan buka jendela Properties.
2. Pilih tab Tanda Tangan Digital.
3. Dari Daftar Tanda Tangan, pilih Amazon Web Services, Inc., lalu pilih Detail.
4. Pilih tab Umum, jika belum dipilih, lalu pilih Lihat Sertifikat.
5. Pilih tab Detail, dan kemudian pilih Semua di Tampilkan daftar tarik turun, jika belum dipilih.
6. Gulir ke bawah sampai Anda melihat bidang Cap Jempol lalu pilih Cap Jempol. Ini menampilkan seluruh nilai cap jempol di jendela bawah.

7. Cocokkan nilai sidik jari dengan nilai berikut. Jika nilainya cocok, lanjutkan dengan instalasi. Jika tidak, eskalasi ke AWS SAM tim dengan [membuat masalah](#) di aws-sam-cli GitHub repositori.

```
c011d416e99a1142c0e0235118ef64c2681f3db9
```

Verifikasi nilai hash

Linux

x86_64 - penginstal baris perintah

Verifikasi integritas dan keaslian file penginstal yang diunduh dengan membuat nilai hash menggunakan perintah berikut:

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

Output tersebut akan terlihat seperti contoh berikut:

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

[Bandingkan nilai hash SHA-256 64 karakter dengan yang untuk AWS SAMCLI versi yang Anda inginkan dalam catatan rilis pada AWS SAMCLI.](#) GitHub

macOS

GUI dan penginstal baris perintah

Verifikasi integritas dan keaslian installer yang diunduh dengan menghasilkan nilai hash menggunakan perintah berikut:

```
$ 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
```

[Bandingkan nilai hash SHA-256 64 karakter Anda dengan nilai yang sesuai di repositori catatan rilis.](#) [AWS SAMCLI](#) GitHub

Menyiapkan kredensial AWS

Antarmuka baris perintah AWS SAM (CLI) mengharuskan Anda untuk mengatur kredensial AWS sehingga dapat melakukan panggilan ke AWS atas nama Anda. Misalnya, AWS SAM CLI melakukan panggilan ke Amazon S3 dan AWS CloudFormation

Anda mungkin telah menetapkan kredensial AWS agar dapat bekerja dengan AWS, seperti salah satu SDK AWS atau AWS CLI. Jika belum, topik ini menunjukkan pendekatan yang disarankan untuk mengatur kredensial AWS.

Untuk mengatur kredensial AWS, Anda harus memiliki access key ID dan secret access key untuk pengguna IAM yang ingin Anda konfigurasi. Untuk informasi tentang access key ID dan secret access key, lihat [Mengelola Access Keys untuk Pengguna IAM](#) dalam Panduan Pengguna IAM.

Berikutnya, tentukan apakah Anda sudah menginstal AWS CLI. Lalu ikuti petunjuk di salah satu bagian berikut:

Menggunakan AWS CLI

Jika Anda sudah menginstal AWS CLI, gunakan perintah `aws configure` dan ikuti petunjuk-petunjuk tersebut:

```
$ 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]:
```

Untuk informasi tentang perintah `aws configure`, lihat [Mengonfigurasi AWS CLI dengan Cepat](#) dalam Panduan Pengguna AWS Command Line Interface.

Tidak menggunakan AWS CLI

Jika Anda belum menginstal AWS CLI, Anda dapat membuat file kredensial atau mengatur variabel lingkungan:

- File kredensial — Anda dapat mengatur kredensial di dalam file kredensial AWS pada sistem lokal Anda. File ini harus terletak di salah satu lokasi berikut:
 - `~/.aws/credentials` di Linux atau macOS

- C:\Users*USERNAME*\.aws\credentials di Windows

File ini harus berisi baris dalam format berikut:

```
[default]
aws_access_key_id = your_access_key_id
aws_secret_access_key = your_secret_access_key
```

- Variabel lingkungan — Anda dapat mengatur variabel lingkungan `AWS_ACCESS_KEY_ID` dan `AWS_SECRET_ACCESS_KEY`.

Untuk mengatur variabel ini di Linux atau macOS, gunakan perintah ekspor:

```
export AWS_ACCESS_KEY_ID=your_access_key_id
export AWS_SECRET_ACCESS_KEY=your_secret_access_key
```

Untuk mengatur variabel ini pada Windows, gunakan perintah atur:

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

AWS SAMCLI pemecahan masalah

Memecahkan masalah pesan kesalahan saat menggunakan, menginstal, dan mengelola Antarmuka Baris AWS Serverless Application Model Perintah (AWS SAMCLI).

Topik

- [Kesalahan penginstalan](#)
- [Pesan kesalahan](#)
- [Pesan peringatan](#)

Kesalahan penginstalan

Linux

Kesalahan Docker: "Tidak dapat terhubung ke daemon Docker. Apakah daemon docker berjalan pada host ini?"

Di beberapa kasus, Anda mungkin perlu mem-boot ulang instans Anda agar memberikan izin kepada `ec2-user` untuk mengakses Docker daemon. Jika Anda menerima pesan kesalahan ini, coba boot ulang instans Anda.

Kesalahan Shell: "perintah tidak ditemukan"

Jika Anda menerima kesalahan ini, shell Anda tidak dapat menemukan AWS SAMCLI executable di jalur. Verifikasi lokasi direktori tempat Anda menginstal AWS SAMCLI executable, dan kemudian verifikasi bahwa direktori ada di jalur Anda.

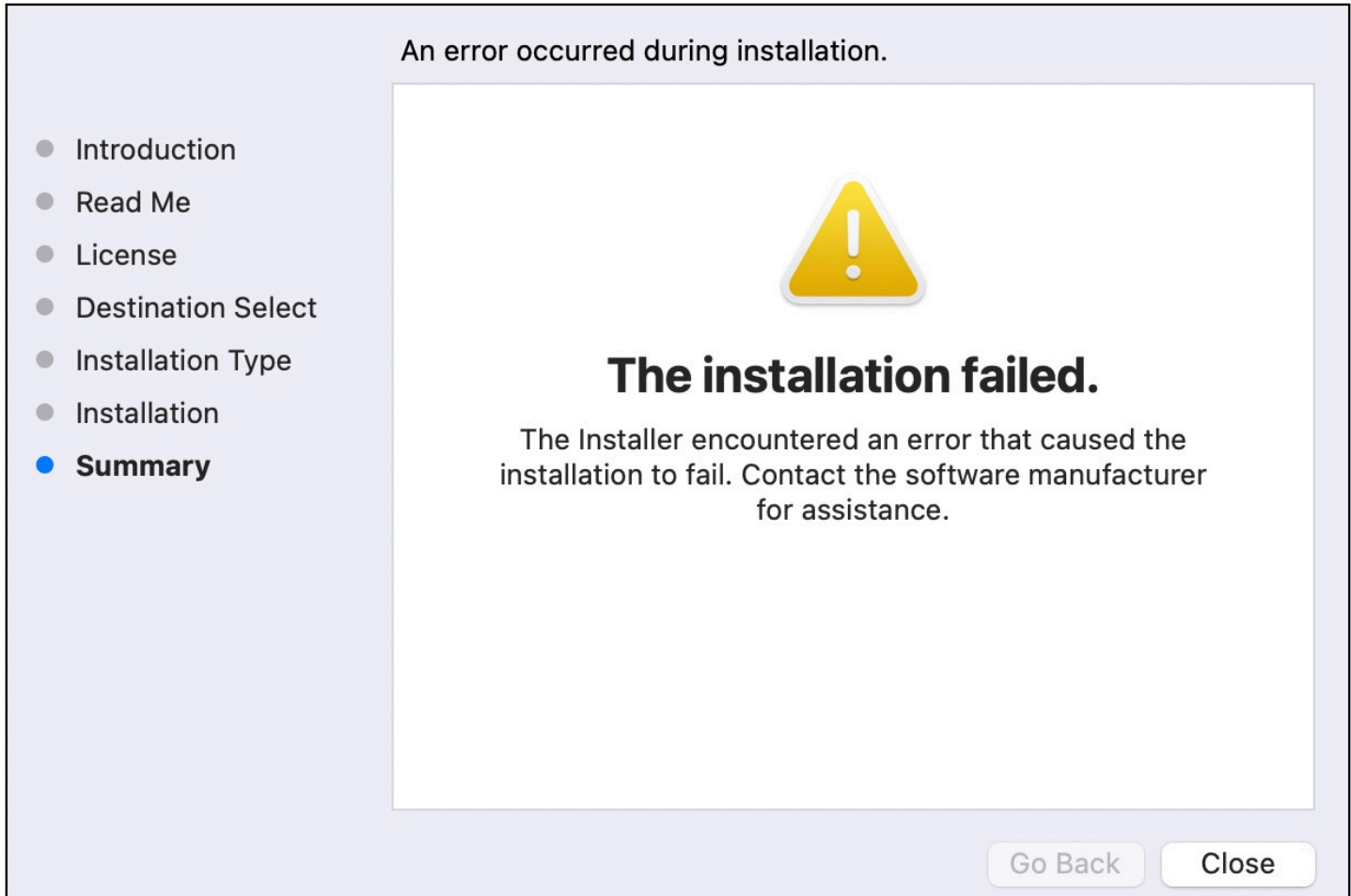
AWS SAMCLIferror: "/lib64/libc.so.6: versi `GLIBC_2.14' tidak ditemukan (diperlukan oleh /usr/local/ / dist/libz.so.1)" aws-sam-cli

Jika Anda menerima kesalahan ini, Anda menggunakan versi Linux yang tidak didukung, dan versi glibc bawaan yang sudah kedaluwarsa. Coba salah satu dari langkah berikut:

- Tingkatkan host Linux Anda ke versi 64-bit dari distribusi CentOS, Fedora, Ubuntu, atau Amazon Linux 2 terbaru.
- Ikuti petunjuk untuk [Instalasi AWS SAMCLI](#).

macOS

Instalasi gagal



Jika Anda menginstal AWS SAMCLI untuk pengguna Anda dan memilih direktori instalasi yang Anda tidak memiliki izin menulis untuk, kesalahan ini dapat terjadi. Coba salah satu dari langkah berikut:

1. Pilih direktori instalasi lain yang Anda memiliki izin menulis untuk.
2. Hapus penginstal. Kemudian, unduh dan jalankan lagi.

Pesan kesalahan

Kesalahan curl: "curl: (6) Tidak dapat menyelesaikan: ..."

Ketika mencoba untuk memanggil titik akhir API Gateway, Anda akan melihat kesalahan berikut ini:

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


Ini berarti Anda telah mencoba mengirim permintaan ke domain yang tidak valid. Hal ini dapat terjadi jika aplikasi nirserver Anda gagal untuk men-deploy, atau jika Anda memiliki kesalahan ketik dalam perintah curl Anda. Verifikasi bahwa aplikasi berhasil digunakan dengan menggunakan AWS CloudFormation konsol atau AWS CLI, dan verifikasi bahwa curl perintah Anda benar.

Kesalahan: Tidak dapat menemukan informasi sumber daya yang tepat dengan nama tumpukan yang diberikan

Saat menjalankan `sam remote invoke` perintah pada aplikasi yang berisi sumber daya fungsi Lambda tunggal, Anda melihat kesalahan berikut:

```
Error: Can't find exact resource information with given <stack-name>. Please provide full resource ARN or --stack-name to resolve the ambiguity.
```

Kemungkinan penyebabnya: Anda tidak memberikan `--stack-name` opsi.

Jika fungsi ARN tidak disediakan sebagai argumen, `sam remote invoke` perintah mengharuskan `--stack-name` opsi disediakan.

Solusi: Berikan `--stack-name` opsi.

Berikut ini adalah contohnya:

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

Kesalahan: Tidak dapat menemukan informasi sumber daya dari nama tumpukan

Saat menjalankan `sam remote invoke` perintah dan meneruskan ARN fungsi Lambda sebagai argumen, Anda melihat kesalahan berikut:

```
Error: Can't find resource information from stack name (<stack-name>) and resource id (<function-id>)
```

Kemungkinan penyebabnya: Anda memiliki nilai nama tumpukan yang ditentukan dalam **samconfig.toml** file Anda.

Yang AWS SAMCLI pertama memeriksa `samconfig.toml` file Anda untuk nama tumpukan. Jika ditentukan, argumen dilewatkan sebagai nilai ID logis.

Solusi: Lulus ID logis fungsi sebagai gantinya.

Anda dapat meneruskan ID logis fungsi sebagai argumen, bukan ARN fungsi.

Solusi: Hapus nilai nama tumpukan dari file konfigurasi Anda.

Anda dapat menghapus nilai nama tumpukan dari file konfigurasi Anda. Ini AWS SAMCLI mencegah agar tidak meneruskan ARN fungsi Anda sebagai nilai ID logis.

Jalankan `sam build` setelah memodifikasi file konfigurasi Anda.

Galat: Gagal membuat sumber daya terkelola: Tidak dapat menemukan kredensial

Saat menjalankan `sam deploy` perintah, Anda melihat kesalahan berikut:

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

Ini berarti Anda belum menyiapkan AWS kredensial untuk mengaktifkan panggilan AWS SAMCLI AWS layanan. Untuk memperbaikinya, Anda harus mengatur AWS kredensial. Untuk informasi selengkapnya, lihat [Menyiapkan kredensial AWS](#).

Kesalahan: `FileNotFoundException` di Windows

Saat menjalankan perintah AWS SAMCLI di Windows, Anda mungkin melihat kesalahan berikut:

```
Error: FileNotFoundException
```

Kemungkinan penyebabnya: AWS SAMCLI Mungkin berinteraksi dengan jalur file yang melebihi batasan jalur maks Windows.

Solusi: Untuk mengatasi masalah ini, perilaku jalur panjang yang baru harus diaktifkan. Untuk melakukannya, lihat [Aktifkan Jalur Panjang di Windows 10, Versi 1607, dan Nanti](#) di Dokumentasi Pengembangan Aplikasi Microsoft Windows.

Kesalahan: penyelesaian ketergantungan pip...

Contoh teks kesalahan:

```
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.
```

Kemungkinan penyebabnya: Jika Anda menggunakan pip untuk menginstal paket, dependensi antar paket mungkin bertentangan.

Setiap versi `aws-sam-cli` paket tergantung pada versi `aws-sam-translator` paket. Misalnya, `aws-sam-cli v1.58.0` mungkin bergantung pada `v1.51.0`. `aws-sam-translator`

Jika Anda menginstal AWS SAMCLI using pip, kemudian menginstal paket lain yang tergantung pada versi yang lebih baru dari `aws-sam-translator`, berikut ini akan terjadi:

- Versi yang lebih baru `aws-sam-translator` akan diinstal.
- Versi saat ini `aws-sam-cli` dan versi yang lebih baru `aws-sam-translator` mungkin tidak kompatibel.
- Saat Anda menggunakan AWS SAMCLI, kesalahan penyelesaian ketergantungan akan terjadi.

Solusi:

1. Gunakan penginstal paket AWS SAMCLI asli.
 - a. Copot pemasangan AWS SAMCLI menggunakan pip. Untuk petunjuk, lihat [Menghapus instalasi AWS SAMCLI](#).
 - b. Instal AWS SAMCLI menggunakan installer paket asli. Untuk petunjuk, lihat [Instalasi AWS SAMCLI](#).
 - c. Bila perlu, tingkatkan AWS SAMCLI menggunakan penginstal paket asli. Untuk petunjuk, lihat [Memutakhirkan AWS SAMCLI](#).
2. Jika Anda harus menggunakan pip, kami sarankan Anda menginstal AWS SAM CLI ke lingkungan virtual. Ini memastikan lingkungan instalasi yang bersih dan lingkungan yang terisolasi jika terjadi kesalahan. Untuk petunjuk, lihat [Memasang AWS SAMCLI ke dalam lingkungan virtual menggunakan pip](#).

Kesalahan: Tidak ada perintah 'remote' seperti itu

Saat menjalankan `sam remote invoke` perintah, Anda melihat kesalahan berikut:

```
$ 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'.
```

Kemungkinan penyebabnya: Versi Anda AWS SAMCLI sudah ketinggalan zaman.

AWS SAMCLIsam remote invokePerintah ini dirilis dengan AWS SAMCLI versi 1.88.0. Anda dapat memeriksa versi Anda dengan menjalankan `sam --version` perintah.

Solusi: Tingkatkan versi Anda AWS SAMCLI ke versi terbaru.

Untuk petunjuk, lihat [Memutakhirkan AWS SAMCLI](#).

Kesalahan: Menjalankan proyek AWS SAM secara lokal membutuhkan Docker. Sudahkah Anda menginstalnya?

Saat menjalankan `sam local start-api` perintah, Anda melihat kesalahan berikut:

```
Error: Running AWS SAM projects locally requires Docker. Have you got it installed?
```

Ini berarti Anda belum menginstal Docker dengan benar. Docker diperlukan untuk menguji aplikasi Anda secara lokal. Untuk memperbaiki masalah ini, ikuti petunjuk untuk menginstal Docker untuk host pengembangan Anda. Untuk informasi selengkapnya, lihat [Menginstal Docker](#).

Kesalahan: Kendala Keamanan Tidak Puas

Saat menjalankan `sam deploy --guided`, Anda akan di-prompt dengan pertanyaan *Function* may not have authorization defined, Is this okay? [y/N]. Jika Anda menanggapi prompt ini dengan **N** (respons default), Anda akan melihat kesalahan berikut:

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

Prompt ini memberi tahu Anda bahwa aplikasi yang akan Anda terapkan mungkin memiliki API Amazon API Gateway API yang dapat diakses publik yang dikonfigurasi tanpa otorisasi. Dengan memberikan tanggapan **N** untuk prompt ini, Anda memberitahu bahwa kesalahan mungkin terjadi.

Untuk memperbaikinya, Anda mempunyai opsi berikut:

- Konfigurasi aplikasi Anda dengan otorisasi. Untuk informasi tentang mengonfigurasi otorisasi, lihat [Pengendalian akses ke API dari API Gateway](#).
- Jika niat Anda adalah memiliki titik akhir API yang dapat diakses publik tanpa otorisasi, mulai ulang penerapan Anda dan tanggap pertanyaan ini **Y** untuk menunjukkan bahwa Anda setuju dengan penerapan.

pesan: Token Otentikasi Hilang

Ketika mencoba untuk memanggil titik akhir API Gateway, Anda akan melihat kesalahan berikut ini:

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

Ini berarti Anda telah mencoba mengirim permintaan ke domain yang benar, namun URI tidak dapat dikenali. Untuk memperbaikinya, verifikasi URL lengkap, dan perbarui perintah curl dengan URL yang benar.

Pesan peringatan

Peringatan:... AWS tidak akan lagi mempertahankan Homebrew installer untuk AWS SAM ...

Saat menginstal AWS SAMCLI using Homebrew, Anda melihat pesan peringatan berikut:

```
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 ...
```

Penyebab potensial: AWS tidak lagi mempertahankan Homebrew dukungan.

Mulai September 2023, tidak AWS akan lagi mempertahankan Homebrew installer untuk AWS SAMCLI

Solusi: Gunakan metode instalasi yang AWS didukung.

- Anda dapat menemukan metode instalasi yang AWS didukung di [Instalasi AWS SAMCLI](#).

Solusi: Untuk terus menggunakan Homebrew, gunakan penginstal yang dikelola komunitas.

- Anda dapat menggunakan Homebrew penginstal yang dikelola komunitas sesuai kebijaksanaan Anda. Untuk petunjuk, lihat [Mengelola AWS SAM CLI dengan Homebrew](#).

AWS SAM referensi konektor

Bagian ini berisi informasi referensi untuk tipe sumber daya konektor AWS Serverless Application Model (AWS SAM). Untuk pengenalan konektor, lihat [Mengelola izin sumber daya dengan konektor AWS SAM](#).

Jenis sumber daya dan tujuan yang didukung untuk konektor

Jenis `AWS::Serverless::Connector` sumber daya mendukung sejumlah koneksi tertentu antara sumber dan sumber daya tujuan. Saat mengonfigurasi konektor di AWS SAM templat Anda, gunakan tabel berikut untuk mereferensikan koneksi yang didukung dan properti yang perlu ditentukan untuk setiap jenis sumber daya sumber dan tujuan. Untuk informasi selengkapnya tentang mengonfigurasi konektor di templat Anda, lihat [AWS::Serverless::Connector](#).

Untuk sumber daya sumber dan tujuan, ketika didefinisikan dalam template yang sama, gunakan `Id` properti. Secara opsional, a `Qualifier` dapat ditambahkan untuk mempersempit ruang lingkup sumber daya yang Anda tentukan. Ketika sumber daya tidak berada dalam template yang sama, gunakan kombinasi properti yang didukung.

Untuk meminta koneksi baru, [kirimkan masalah baru](#) di `serverless-application-model` AWS GitHub repositori.

Jenis sumber	Jenis tujuan	Izin	Properti sumber	Properti tujuan
<code>AWS::ApiGateway::RestApi</code>	<code>AWS::Lambda::Function</code>	Write	<code>Id</code> atau <code>Qualifier, ResourceId</code> , dan <code>Type</code>	<code>Id</code> atau <code>Arn</code> dan <code>Type</code>
<code>AWS::ApiGateway::RestApi</code>	<code>AWS::Serverless::Function</code>	Write	<code>Id</code> atau <code>Qualifier, ResourceId</code> , dan <code>Type</code>	<code>Id</code> atau <code>Arn</code> dan <code>Type</code>

Jenis sumber	Jenis tujuan	Izin	Properti sumber	Properti tujuan
AWS::ApiGatewayV2::Api	AWS::Lambda::Function	Write	IdatauQualifier, ResourceId, dan Type	Idatau Arn dan Type
AWS::ApiGatewayV2::Api	AWS::Serverless::Function	Write	IdatauQualifier, ResourceId, dan Type	Idatau Arn dan Type
AWS::AppSync::DataSource	AWS::DynamoDB::Table	Read	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::AppSync::DataSource	AWS::DynamoDB::Table	Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::AppSync::DataSource	AWS::Events::EventBus	Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::AppSync::DataSource	AWS::Lambda::Function	Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::AppSync::DataSource	AWS::Serverless::Function	Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::AppSync::DataSource	AWS::Serverless::SimpleTable	Read	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::AppSync::DataSource	AWS::Serverless::SimpleTable	Write	Idatau RoleName dan Type	Idatau Arn dan Type

Jenis sumber	Jenis tujuan	Izin	Properti sumber	Properti tujuan
AWS::AppSync::GraphQLApi	AWS::Lambda::Function	Write	Idatau ResourceId dan Type	Idatau Arn dan Type
AWS::AppSync::GraphQLApi	AWS::Serverless::Function	Write	Idatau ResourceId dan Type	Idatau Arn dan Type
AWS::DynamoDB::Table	AWS::Lambda::Function	Read	Idatau Arn dan Type	Idatau RoleName dan Type
AWS::DynamoDB::Table	AWS::Serverless::Function	Read	Idatau Arn dan Type	Idatau RoleName dan Type
AWS::Events::Rule	AWS::Events::Event Bus	Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Events::Rule	AWS::Lambda::Function	Write	Idatau Arn dan Type	Idatau Arn dan Type
AWS::Events::Rule	AWS::Serverless::Function	Write	Idatau Arn dan Type	Idatau Arn dan Type
AWS::Events::Rule	AWS::Serverless::StateMachine	Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Events::Rule	AWS::SNS::Topic	Write	Idatau Arn dan Type	Idatau Arn dan Type

Jenis sumber	Jenis tujuan	Izin	Properti sumber	Properti tujuan
AWS::Events::Rule	AWS::SQS::Queue	Write	Idatau Arn dan Type	IdatauArn,QueueUrl, dan Type
AWS::Events::Rule	AWS::StepFunctions::StateMachine	Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Lambda::Function	AWS::DynamoDB::Table	Read, Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Lambda::Function	AWS::Events::EventBus	Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Lambda::Function	AWS::Lambda::Function	Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Lambda::Function	AWS::Location::PlaceIndex	Read	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Lambda::Function	AWS::S3::Bucket	Read, Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Lambda::Function	AWS::Serverless::Function	Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Lambda::Function	AWS::Serverless::SimpleTable	Read, Write	Idatau RoleName dan Type	Idatau Arn dan Type

Jenis sumber	Jenis tujuan	Izin	Properti sumber	Properti tujuan
AWS::Lambda::Function	AWS::Serverless::StateMachine	Read, Write	Idatau RoleName dan Type	Idatau Arn, Name, dan Type
AWS::Lambda::Function	AWS::SNS::Topic	Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Lambda::Function	AWS::SQS::Queue	Read, Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Lambda::Function	AWS::StepFunctions::StateMachine	Read, Write	Idatau RoleName dan Type	Idatau Arn, Name, dan Type
AWS::S3::Bucket	AWS::Lambda::Function	Write	Idatau Arn dan Type	Idatau Arn dan Type
AWS::S3::Bucket	AWS::Serverless::Function	Write	Idatau Arn dan Type	Idatau Arn dan Type
AWS::Serverless::Api	AWS::Lambda::Function	Write	IdatauQualifier, ResourceId, dan Type	Idatau Arn dan Type
AWS::Serverless::Api	AWS::Serverless::Function	Write	IdatauQualifier, ResourceId, dan Type	Idatau Arn dan Type

Jenis sumber	Jenis tujuan	Izin	Properti sumber	Properti tujuan
AWS::Serverless::Function	AWS::DynamoDB::Table	Read, Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Serverless::Function	AWS::Events::Event Bus	Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Serverless::Function	AWS::Lambda::Function	Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Serverless::Function	AWS::S3::Bucket	Read, Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Serverless::Function	AWS::Serverless::Function	Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Serverless::Function	AWS::Serverless::SimpleTable	Read, Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Serverless::Function	AWS::Serverless::StateMachine	Read, Write	Idatau RoleName dan Type	Idatau Arn, Name, dan Type
AWS::Serverless::Function	AWS::SNS::Topic	Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Serverless::Function	AWS::SQS::Queue	Read, Write	Idatau RoleName dan Type	Idatau Arn dan Type

Jenis sumber	Jenis tujuan	Izin	Properti sumber	Properti tujuan
AWS::Serverless::Function	AWS::StepFunctions::StateMachine	Read, Write	Idatau RoleName dan Type	Idatau Arn, Name, dan Type
AWS::Serverless::HttpApi	AWS::Lambda::Function	Write	IdatauQualifier, ResourceId, dan Type	Idatau Arn dan Type
AWS::Serverless::HttpApi	AWS::Serverless::Function	Write	IdatauQualifier, ResourceId, dan Type	Idatau Arn dan Type
AWS::Serverless::SimpleTable	AWS::Lambda::Function	Read	Idatau Arn dan Type	Idatau RoleName dan Type
AWS::Serverless::SimpleTable	AWS::Serverless::Function	Read	Idatau Arn dan Type	Idatau RoleName dan Type
AWS::Serverless::StateMachine	AWS::DynamoDB::Table	Read, Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Serverless::StateMachine	AWS::Events::EventBus	Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Serverless::StateMachine	AWS::Lambda::Function	Write	Idatau RoleName dan Type	Idatau Arn dan Type

Jenis sumber	Jenis tujuan	Izin	Properti sumber	Properti tujuan
AWS::Serverless::StateMachine	AWS::S3::Bucket	Read, Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Serverless::StateMachine	AWS::Serverless::Function	Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Serverless::StateMachine	AWS::Serverless::SimpleTable	Read, Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Serverless::StateMachine	AWS::Serverless::StateMachine	Read, Write	Idatau RoleName dan Type	Idatau Arn, Name, dan Type
AWS::Serverless::StateMachine	AWS::SNS::Topic	Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Serverless::StateMachine	AWS::SQS::Queue	Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Serverless::StateMachine	AWS::StepFunctions::StateMachine	Read, Write	Idatau RoleName dan Type	Idatau Arn, Name, dan Type

Jenis sumber	Jenis tujuan	Izin	Properti sumber	Properti tujuan
AWS::SNS: :Topic	AWS::Lamb da::Funct ion	Write	Idatau Arn dan Type	Idatau Arn dan Type
AWS::SNS: :Topic	AWS::Serv erless::F unction	Write	Idatau Arn dan Type	Idatau Arn dan Type
AWS::SNS: :Topic	AWS::SQS: :Queue	Write	Idatau Arn dan Type	IdatauArn,QueueUrl, dan Type
AWS::SQS: :Queue	AWS::Lamb da::Funct ion	Read, Write	Idatau Arn dan Type	Idatau RoleName dan Type
AWS::SQS: :Queue	AWS::Serv erless::F unction	Read, Write	Idatau Arn dan Type	Idatau RoleName dan Type
AWS::Step Functions ::StateMa chine	AWS::Dyna moDB::Tab le	Read, Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Step Functions ::StateMa chine	AWS::Even ts::Event Bus	Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Step Functions ::StateMa chine	AWS::Lamb da::Funct ion	Write	Idatau RoleName dan Type	Idatau Arn dan Type

Jenis sumber	Jenis tujuan	Izin	Properti sumber	Properti tujuan
AWS::Step Functions::StateMachine	AWS::S3::Bucket	Read, Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Step Functions::StateMachine	AWS::Serverless::Function	Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Step Functions::StateMachine	AWS::Serverless::SimpleTable	Read, Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Step Functions::StateMachine	AWS::Serverless::StateMachine	Read, Write	Idatau RoleName dan Type	IdatauArn,Name, dan Type
AWS::Step Functions::StateMachine	AWS::SNS::Topic	Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Step Functions::StateMachine	AWS::SQS::Queue	Write	Idatau RoleName dan Type	Idatau Arn dan Type
AWS::Step Functions::StateMachine	AWS::Step Functions::StateMachine	Read, Write	Idatau RoleName dan Type	IdatauArn,Name, dan Type

Kebijakan IAM yang dibuat oleh konektor

Bagian ini mendokumentasikan kebijakan AWS Identity and Access Management (IAM) yang dibuat oleh AWS SAM saat menggunakan konektor.

`AWS::DynamoDB::Table` untuk `AWS::Lambda::Function`

Jenis kebijakan

[Kebijakan yang dikelola pelanggan](#) melekat pada `AWS::Lambda::Function` peran tersebut.

Kategori akses

Read

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:DescribeStream",
        "dynamodb:GetRecords",
        "dynamodb:GetShardIterator",
        "dynamodb:ListStreams"
      ],
      "Resource": [
        "%{Source.Arn}/stream/*"
      ]
    }
  ]
}
```

`AWS::Events::Rule` untuk `AWS::SNS::Topic`

Jenis kebijakan

[AWS::SNS::TopicPolicy](#) melekat pada `AWS::SNS::Topic`.

Kategori akses

Write

```
{
```



```

"Statement": [
  {
    "Effect": "Allow",
    "Principal": {
      "Service": "events.amazonaws.com"
    },
    "Resource": "%{Destination.Arn}",
    "Action": "sns:Publish",
    "Condition": {
      "ArnEquals": {
        "aws:SourceArn": "%{Source.Arn}"
      }
    }
  }
]
}

```

AWS::Events::Rule untuk AWS::Events::EventBus

Jenis kebijakan

[Kebijakan yang dikelola pelanggan](#) melekat pada AWS::Events::Rule peran tersebut.

Kategori akses

Write

```

{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}

```

AWS::Events::Rule untuk AWS::StepFunctions::StateMachine

Jenis kebijakan

[Kebijakan yang dikelola pelanggan](#) melekat pada `AWS::Events::Rule` peran tersebut.

Kategori akses

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

`AWS::Events::Rule` untuk `AWS::Lambda::Function`

Jenis kebijakan

[AWS::Lambda::Permission](#) melekat pada `AWS::Lambda::Function`.

Kategori akses

Write

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "events.amazonaws.com",
  "SourceArn": "%{Source.Arn}"
}
```

`AWS::Events::Rule` untuk `AWS::SQS::Queue`

Jenis kebijakan

[AWS::SQS::QueuePolicy](#) melekat pada `AWS::SQS::Queue`.

Kategori akses

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "events.amazonaws.com"
      },
      "Resource": "%{Destination.Arn}",
      "Action": "sqs:SendMessage",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "%{Source.Arn}"
        }
      }
    }
  ]
}
```

AWS::Lambda::Function untuk AWS::Lambda::Function

Jenis kebijakan

[Kebijakan yang dikelola pelanggan](#) melekat pada AWS::Lambda::Function peran tersebut.

Kategori akses

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeAsync",
        "lambda:InvokeFunction"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

```
}
```

AWS::Lambda::Function untuk AWS::S3::Bucket

Jenis kebijakan

[Kebijakan yang dikelola pelanggan](#) melekat pada AWS::Lambda::Function peran tersebut.

Kategori akses

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 untuk AWS::DynamoDB::Table

Jenis kebijakan

[Kebijakan yang dikelola pelanggan](#) melekat pada AWS::Lambda::Function peran tersebut.

Kategori akses

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 untuk AWS::SQS::Queue

Jenis kebijakan

[Kebijakan yang dikelola pelanggan](#) melekat pada AWS::Lambda::Function peran tersebut.

Kategori akses

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 untuk AWS::SNS::Topic

Jenis kebijakan

[Kebijakan yang dikelola pelanggan](#) melekat pada AWS::Lambda::Function peran tersebut.

Kategori akses

Write

```

{
  "Statement": [
    {

```

```

    "Effect": "Allow",
    "Action": [
      "sns:Publish"
    ],
    "Resource": [
      "%{Destination.Arn}"
    ]
  }
]
}

```

AWS::Lambda::Function untuk AWS::StepFunctions::StateMachine

Jenis kebijakan

[Kebijakan yang dikelola pelanggan](#) melekat pada AWS::Lambda::Function peran tersebut.

Kategori akses

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 untuk AWS::Events::EventBus

Jenis kebijakan

[Kebijakan yang dikelola pelanggan](#) melekat pada AWS::Lambda::Function peran tersebut.

Kategori akses

Write

```
{
  "Statement": [

```

```

    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}

```

AWS::Lambda::Function untuk AWS::Location::PlaceIndex

Jenis kebijakan

[Kebijakan yang dikelola pelanggan](#) melekat pada AWS::Lambda::Function peran tersebut.

Kategori akses

Read

```

{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "geo:DescribePlaceIndex",
        "geo:GetPlace",
        "geo:SearchPlaceIndexForPosition",
        "geo:SearchPlaceIndexForSuggestions",
        "geo:SearchPlaceIndexForText"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}

```

AWS::ApiGatewayV2::Api untuk AWS::Lambda::Function

Jenis kebijakan

[AWS::Lambda::Permission](#) melekat pada `AWS::Lambda::Function`.

Kategori akses

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` untuk `AWS::Lambda::Function`

Jenis kebijakan

[AWS::Lambda::Permission](#) melekat pada `AWS::Lambda::Function`.

Kategori akses

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` untuk `AWS::SQS::Queue`

Jenis kebijakan

[AWS::SQS::QueuePolicy](#) melekat pada `AWS::SQS::Queue`.

Kategori akses

Write

```
{
  "Statement": [
```

```

{
  "Effect": "Allow",
  "Principal": {
    "Service": "sns.amazonaws.com"
  },
  "Resource": "%{Destination.Arn}",
  "Action": "sqs:SendMessage",
  "Condition": {
    "ArnEquals": {
      "aws:SourceArn": "%{Source.Arn}"
    }
  }
}
]
}

```

AWS::SNS::Topic untuk **AWS::Lambda::Function**

Jenis kebijakan

[AWS::Lambda::Permission](#) melekat pada **AWS::Lambda::Function**.

Kategori akses

Write

```

{
  "Action": "lambda:InvokeFunction",
  "Principal": "sns.amazonaws.com",
  "SourceArn": "%{Source.Arn}"
}

```

AWS::SQS::Queue untuk **AWS::Lambda::Function**

Jenis kebijakan

[Kebijakan yang dikelola pelanggan](#) melekat pada **AWS::Lambda::Function** peran tersebut.

Kategori akses

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 untuk AWS::Lambda::Function

Jenis kebijakan

[AWS::Lambda::Permission](#) melekat pada `AWS::Lambda::Function`.

Kategori akses

Write

```

{
  "Action": "lambda:InvokeFunction",
  "Principal": "s3.amazonaws.com",

```

```
"SourceArn": "%{Source.Arn}",
"SourceAccount": "${AWS::AccountId}"
}
```

AWS::StepFunctions::StateMachine untuk AWS::Lambda::Function

Jenis kebijakan

[Kebijakan yang dikelola pelanggan](#) melekat pada AWS::StepFunctions::StateMachine peran tersebut.

Kategori akses

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeAsync",
        "lambda:InvokeFunction"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::StepFunctions::StateMachine untuk AWS::SNS::Topic

Jenis kebijakan

[Kebijakan yang dikelola pelanggan](#) melekat pada AWS::StepFunctions::StateMachine peran tersebut.

Kategori akses

Write

```
{
  "Statement": [
```

```
{
  "Effect": "Allow",
  "Action": [
    "sns:Publish"
  ],
  "Resource": [
    "%{Destination.Arn}"
  ]
}
```

AWS::StepFunctions::StateMachine untuk AWS::SQS::Queue

Jenis kebijakan

[Kebijakan yang dikelola pelanggan](#) melekat pada AWS::StepFunctions::StateMachine peran tersebut.

Kategori akses

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:SendMessage"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::StepFunctions::StateMachine untuk AWS::S3::Bucket

Jenis kebijakan

[Kebijakan yang dikelola pelanggan](#) melekat pada AWS::StepFunctions::StateMachine peran tersebut.

Kategori akses

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 untuk AWS::DynamoDB::Table

Jenis kebijakan

[Kebijakan yang dikelola pelanggan](#) melekat pada AWS::StepFunctions::StateMachine peran tersebut.

Kategori akses

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 untuk AWS::StepFunctions::StateMachine

Jenis kebijakan

[Kebijakan yang dikelola pelanggan](#) melekat pada AWS::StepFunctions::StateMachine peran tersebut.

Kategori akses

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 untuk AWS::Events::EventBus

Jenis kebijakan

[Kebijakan yang dikelola pelanggan](#) melekat pada AWS::StepFunctions::StateMachine peran tersebut.

Kategori akses

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::AppSync::DataSource untuk AWS::DynamoDB::Table

Jenis kebijakan

[Kebijakan yang dikelola pelanggan](#) melekat pada AWS::AppSync::DataSource peran tersebut.

Kategori akses

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 untuk AWS::Lambda::Function

Jenis kebijakan

[Kebijakan yang dikelola pelanggan](#) melekat pada AWS::AppSync::DataSource peran tersebut.

Kategori akses

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeAsync",
        "lambda:InvokeFunction"
      ],
      "Resource": [
        "%{Destination.Arn}",
        "%{Destination.Arn}:*"
      ]
    }
  ]
}
```

AWS::AppSync::DataSource untuk AWS::Events::EventBus

Jenis kebijakan

[Kebijakan yang dikelola pelanggan](#) melekat pada AWS::AppSync::DataSource peran tersebut.

Kategori akses

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ]
    }
  ]
}
```

```

    ],
    "Resource": [
      "%{Destination.Arn}"
    ]
  }
]
}

```

`AWS::AppSync::GraphQLApi` untuk `AWS::Lambda::Function`

Jenis kebijakan

[AWS::Lambda::Permission](#) melekat pada `AWS::Lambda::Function`.

Kategori akses

Write

```

{
  "Action": "lambda:InvokeFunction",
  "Principal": "appsync.amazonaws.com",
  "SourceArn": "arn:${AWS::Partition}:appsync:${AWS::Region}:${AWS::AccountId}:apis/
%{Source.ResourceId}"
}

```

Menginstal Docker untuk digunakan dengan AWS SAMCLI

Docker adalah aplikasi yang menjalankan kontainer di mesin Anda. Dengan Docker, AWS SAM dapat menyediakan lingkungan lokal yang mirip dengan AWS Lambda sebagai wadah untuk membangun, menguji, dan men-debug aplikasi tanpa server Anda.

Note

Docker diperlukan hanya untuk menguji aplikasi Anda secara lokal dan untuk membangun paket penerapan menggunakan opsi `--use-container`

Topik

- [Menginstal Docker](#)

- [Langkah selanjutnya](#)

Menginstal Docker

Ikuti petunjuk ini untuk menginstal Docker pada sistem operasi Anda.

Linux

Docker tersedia di banyak sistem operasi yang berbeda, termasuk sebagian besar distribusi Linux modern, seperti, CentOSDebian, dan. Ubuntu Untuk informasi tentang menginstal Docker pada sistem operasi tertentu, lihat [Mendapatkan Docker di situs web Docker Docs](#).

Untuk menginstal Docker di Amazon Linux 2 atau Amazon Linux 2023

1. Perbarui paket yang diinstal dan paket cache pada instans Anda.

```
$ sudo yum update -y
```

2. Instal paket Edisi Docker Komunitas terbaru.

- Untuk Amazon Linux 2, jalankan yang berikut ini:

```
$ sudo amazon-linux-extras install docker
```

- Untuk Amazon Linux 2023, jalankan yang berikut ini:

```
$ sudo yum install -y docker
```

3. Mulai layanan Docker.

```
$ sudo service docker start
```

4. Tambahkan `ec2-user` ke `docker` grup sehingga Anda dapat menjalankan Docker perintah tanpa menggunakan `sudo`.

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

5. Keluar dan masuk kembali untuk mendapatkan izin grup `docker` yang baru. Untuk melakukannya, tutup jendela terminal SSH Anda saat ini dan hubungkan kembali ke instans Anda di sesi yang baru. Sesi SSH baru Anda akan memiliki izin grup `docker` yang sesuai.

6. Verifikasi bahwa `ec2-user` dapat menjalankan perintah Docker tanpa menggunakan `sudo`.

```
$ docker ps
```

Anda akan melihat output berikut, mengonfirmasi jika Docker sudah diinstal dan dapat berjalan:

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS	NAMES	

Note

Di Linux, untuk membangun dan menjalankan fungsi Lambda dengan arsitektur set instruksi yang berbeda dari mesin host Anda, ada langkah-langkah tambahan untuk mengkonfigurasi Docker. Misalnya, untuk menjalankan `arm64` fungsi pada `x86_64` mesin, Anda dapat menjalankan perintah berikut untuk mengkonfigurasi Docker daemon: `docker run --rm --privileged multiarch/qemu-user-static --reset -p yes`

Jika Anda mengalami masalah saat menginstal Docker, lihat [Kesalahan penginstalan](#). Atau, lihat bagian [Pemecahan Masalah](#) dari Langkah pasca-instalasi untuk Linux di situs web Docker Docs.

macOS

Note

Docker Desktop secara resmi didukung, tetapi dimulai dengan AWS SAM CLI versi 1.47.0, Anda dapat menggunakan alternatif selama mereka menggunakan runtime Docker.

1. Instal Docker

AWS SAM CLI mendukung Docker berjalan di macOS Sierra 10.12 atau yang lebih baru. Untuk cara menginstal Docker, lihat [Menginstal Docker Desktop untuk Mac](#) di situs web Docker Docs.

2. Konfigurasi drive berbagi Anda

AWS SAM CLI mengharuskan direktori proyek, atau direktori induk apa pun, terdaftar dalam drive bersama. Ini mungkin memerlukan berbagi file. Untuk informasi selengkapnya, lihat topik Pemecahan masalah [berbagi file yang memerlukan pemasangan volume](#) di Docker dokumen.

3. Verifikasi instalasi.

Setelah Docker diinstal, verifikasi bahwa itu berfungsi. Juga konfirmasi bahwa Anda dapat menjalankan Docker perintah dari baris perintah (misalnya, `docker ps`). Anda tidak perlu menginstal, mengambil, atau menarik kontainer apa AWS SAM CLI pun—melakukan ini secara otomatis sesuai kebutuhan.

Jika Anda mengalami masalah saat menginstal Docker, untuk tips pemecahan masalah lainnya, lihat bagian [Pemecahan Masalah dan diagnosis](#) di situs web Dokumen. Docker

Windows

Note

AWS SAM secara resmi mendukung Docker Desktop. Namun, dimulai dengan AWS SAM CLI versi 1.47.0, Anda dapat menggunakan alternatif selama mereka menggunakan runtime. Docker

1. Instal Docker.

Docker Desktop mendukung sistem operasi Windows terbaru. Untuk versi lama Windows, Docker Toolbox tersedia. Pilih versi Windows Anda untuk langkah-langkah Docker instalasi yang benar:

- Untuk menginstal Docker untuk Windows 10, lihat [Menginstal Docker Desktop untuk Windows](#) di situs web Docker Dokumen.
- Docker Untuk menginstal versi Windows yang lebih lama, lihat [The Docker Toolbox](#) pada repositori Docker Toolbox GitHub .

2. Konfigurasi drive berbagi Anda.

AWS SAM CLI mengharuskan direktori proyek, atau direktori induk apa pun, terdaftar dalam drive bersama. Dalam beberapa kasus, Anda harus berbagi drive Anda Docker agar berfungsi dengan baik.

3. Verifikasi instalasi.

Setelah Docker diinstal, verifikasi bahwa itu berfungsi. Juga konfirmasi bahwa Anda dapat menjalankan Docker perintah dari baris perintah (misalnya, `docker ps`). Anda tidak perlu

menginstal, mengambil, atau menarik kontainer apa AWS SAM CLI pun—melakukan ini secara otomatis sesuai kebutuhan.

Jika Anda mengalami masalah saat menginstal Docker, untuk tips pemecahan masalah lainnya, lihat bagian [Pemecahan Masalah dan diagnosis](#) di situs web Dokumen. Docker

Langkah selanjutnya

Untuk cara menginstal AWS SAM CLI, lihat [Instalasi AWS SAM CLI](#).

Repositori citra

AWS SAM menyederhanakan integrasi berkelanjutan dan tugas pengiriman berkelanjutan (CI/CD) untuk aplikasi tanpa server dengan bantuan gambar kontainer build. Gambar yang AWS SAM disediakan termasuk antarmuka baris AWS SAM perintah (CLI) dan alat build untuk sejumlah runtime yang didukung AWS Lambda. Ini membuatnya lebih mudah untuk membangun dan mengemas aplikasi tanpa server menggunakan file. AWS SAM CLI Anda dapat menggunakan gambar-gambar ini dengan sistem CI/CD untuk mengotomatiskan pembangunan dan penyebaran aplikasi. AWS SAM Sebagai contoh, lihat [Men-deploy menggunakan sistem CI/CD](#).

AWS SAM URI image container build ditandai dengan versi yang AWS SAM CLI disertakan dalam gambar itu. Jika Anda menentukan URI yang tidak ditandai, maka versi terbaru digunakan. Misalnya, `public.ecr.aws/sam/build-nodejs14.x` menggunakan citra terbaru. Namun, `public.ecr.aws/sam/build-nodejs14.x:1.24.1` menggunakan citra yang berisi AWS SAM CLI versi 1.24.1.

Dimulai dengan versi 1.33.0 AWS SAM CLI, keduanya x86_64 dan gambar arm64 kontainer tersedia untuk runtime yang didukung. Untuk informasi selengkapnya, lihat [runtime Lambda di Panduan Pengembang AWS Lambda](#)

Note

Sebelum versi 1.22.0 AWS SAM CLI, DockerHub adalah repositori default tempat gambar kontainer AWS SAM CLI ditarik. Memulai dengan versi 1.22.0, repositori default berubah menjadi Amazon Elastic Container Registry Public (Amazon ECR Public). Untuk menarik citra kontainer dari repositori selain default saat ini, Anda dapat menggunakan perintah [sam build](#) dengan opsi `--build-image`. Contoh di akhir topik ini menunjukkan cara membangun aplikasi menggunakan gambar DockerHub repositori.

URI repositori citra

Tabel berikut mencantumkan URI image container build [Amazon ECR Public](#) yang dapat Anda gunakan untuk membangun dan mengemas aplikasi tanpa server. AWS SAM

Note

Amazon ECR Public diganti DockerHub dimulai dengan AWS SAMCLI versi 1.22.0. Jika Anda menggunakan versi sebelumnya AWS SAMCLI, kami sarankan Anda meningkatkan.

Waktu Aktif	Amazon ECR Public
Runtime kustom (AL2023)	public.ecr.aws/sam/build-provided.al2023
Waktu aktif kustom (AL2)	public.ecr.aws/sam/build-provided.al2
Waktu aktif kustom	public.ecr.aws/sam/build-disediakan
Go 1.x	public.ecr.aws/sam/build-go1.x
Jawa 21	public.ecr.aws/sam/build-java21
Jawa 17	public.ecr.aws/sam/build-java17
Java 11	public.ecr.aws/sam/build-java11
Java 8 (AL2)	public.ecr.aws/sam/build-java8.al2
Java 8	public.ecr.aws/sam/build-java8
.NET 8	public.ecr.aws/sam/build-dotnet8
.NET 7	public.ecr.aws/sam/build-dotnet7
.NET 6	public.ecr.aws/sam/build-dotnet6
Node.js 20	public.ecr.aws/sam/build-nodejs20.x
Node.js 18	public.ecr.aws/sam/build-nodejs18.x

Waktu Aktif	Amazon ECR Public
Node.js 16	public.ecr.aws/sam/build-nodejs16.x
Python 3.12	public.ecr.aws/sam/build-python3.12
Python 3.11	public.ecr.aws/sam/build-python3.11
Python 3.10	public.ecr.aws/sam/build-python3.10
Python 3.9	public.ecr.aws/sam/build-python3.9
Python 3.8	public.ecr.aws/sam/build-python3.8
Ruby 3.3	public.ecr.aws/sam/build-ruby3.3
Ruby 3.2	public.ecr.aws/sam/build-ruby3.2

Contoh-contoh

Berikut dua contoh perintah membangun aplikasi menggunakan gambar kontainer dari DockerHub repositori:

Buat Node.js 12 aplikasi menggunakan gambar kontainer yang ditarik dari DockerHub:

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

Buat sumber daya fungsi menggunakan gambar Python 3.8 kontainer yang ditarik dari DockerHub:

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

Men-deploy aplikasi nirserver secara bertahap

AWS Serverless Application Model(AWS SAM) dilengkapi dengan [CodeDeploy](#) untuk menyediakan AWS Lambda penerapan bertahap. Hanya dengan beberapa baris konfigurasi, AWS SAM melakukan hal berikut untuk Anda:

- Men-deploy versi baru fungsi Lambda Anda, dan secara otomatis membuat alias yang mengarah ke versi baru.
- Secara bertahap menggeser lalu lintas pelanggan ke versi baru sampai Anda puas bahwa itu berfungsi seperti yang diharapkan. Jika pembaruan tidak berfungsi dengan benar, Anda dapat memutar kembali perubahan.
- Mendefinisikan fungsi uji pra-lalu lintas dan pasca-lalu lintas untuk memverifikasi bahwa kode yang baru digunakan dikonfigurasi dengan benar dan bahwa aplikasi Anda beroperasi seperti yang diharapkan.
- Secara otomatis memutar kembali penerapan jika CloudWatch alarm dipicu.

Note

Jika Anda mengaktifkan penerapan bertahap melalui AWS SAM template Anda, CodeDeploy sumber daya secara otomatis dibuat untuk Anda. Anda dapat melihat CodeDeploy sumber daya secara langsung melalui AWS Management Console.

Contoh

Contoh berikut menunjukkan penggunaan CodeDeploy untuk secara bertahap mengalihkan pelanggan ke versi fungsi Lambda yang baru Anda gunakan:

```
Resources:
  MyLambdaFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: nodejs12.x
      CodeUri: s3://bucket/code.zip

      AutoPublishAlias: live

    DeploymentPreference:
      Type: Canary10Percent10Minutes
    Alarms:
      # A list of alarms that you want to monitor
      - !Ref AliasErrorMetricGreaterThanZeroAlarm
      - !Ref LatestVersionErrorMetricGreaterThanZeroAlarm
    Hooks:
```

```
# Validation Lambda functions that are run before & after traffic shifting
PreTraffic: !Ref PreTrafficLambdaFunction
PostTraffic: !Ref PostTrafficLambdaFunction
```

Revisi ke templat AWS SAM ini melakukan hal berikut:

- **AutoPublishAlias:** Dengan menambahkan properti ini dan menentukan nama alias, AWS SAM
 - Mendeteksi saat kode baru di-deploy, berdasarkan perubahan pada fungsi Lambda URI Amazon S3.
 - Membuat dan memublikasikan versi terbaru dari fungsi tersebut dengan kode terbaru.
 - Membuat alias dengan nama yang Anda sediakan (kecuali alias sudah ada), dan menunjuk ke versi terbaru dari fungsi Lambda. Pemanggilan fungsi harus menggunakan pengualifikasi alias untuk memanfaatkan hal ini. [Jika Anda tidak terbiasa dengan versi dan alias fungsi Lambda, lihat AWS Lambda Pembuatan Versi Fungsi dan Alias.](#)
- **Deployment Preference Type** Pada contoh sebelumnya, 10 persen lalu lintas pelanggan Anda segera bergeser ke versi baru Anda. Setelah 10 menit, semua lalu lintas dialihkan ke versi baru. Namun, jika tes pra-lalu lintas atau pasca-lalu lintas Anda gagal, atau jika CloudWatch alarm dipicu, putar CodeDeploy kembali penerapan Anda. Anda dapat menentukan bagaimana lalu lintas harus digeser antar versi dengan cara berikut:
 - **Canary:** Lalu lintas digeser dalam dua peningkatan. Anda dapat memilih dari opsi canary yang telah ditentukan sebelumnya. Opsi menentukan persentase lalu lintas yang dialihkan ke versi fungsi Lambda Anda yang diperbarui pada kenaikan pertama, dan interval, dalam menit, sebelum lalu lintas yang tersisa dialihkan pada kenaikan kedua.
 - **Linear:** Lalu lintas digeser dalam peningkatan yang sama dengan jumlah menit yang sama antara setiap kenaikan. Anda dapat memilih dari opsi linier yang telah ditentukan sebelumnya yang menentukan persentase lalu lintas yang dialihkan di setiap kenaikan dan jumlah menit di antara setiap kenaikan.
 - **AllAtOnce:** Semua lalu lintas digeser dari fungsi Lambda asli ke versi fungsi Lambda yang diperbarui sekaligus.

Tabel berikut menguraikan opsi pergeseran lalu lintas lainnya yang tersedia di luar yang digunakan dalam contoh.

Tipe Preferensi Deployment

Canary10Percent30Minutes

Tipe Preferensi Deployment

Canary10Percent5Minutes

Canary10Percent10Minutes

Canary10Percent15Minutes

Linear10 10 PercentEvery Menit

Linear10 1 PercentEvery Menit

Linear10 PercentEvery 2Menit

Linear10 PercentEvery 3Menit

AllAtOnce

- **Alarms:** Ini adalah CloudWatch alarm yang dipicu oleh kesalahan apa pun yang ditimbulkan oleh penerapan. Saat ditemui, mereka secara otomatis memutar kembali penerapan Anda. Misalnya, jika kode yang diperbarui yang Anda gunakan menyebabkan kesalahan dalam aplikasi. Contoh lain adalah jika ada [AWS Lambda](#) atau CloudWatch metrik khusus yang Anda tentukan telah melanggar ambang alarm.
- **Hooks:** Ini adalah fungsi uji pra-lalu lintas dan pasca-lalu lintas yang menjalankan pemeriksaan sebelum pergeseran lalu lintas mulai ke versi baru, dan setelah pergeseran lalu lintas selesai.
 - **PreTraffic:** Sebelum perpindahan lalu lintas dimulai, CodeDeploy memanggil fungsi Lambda kait pra-lalu lintas. Fungsi Lambda ini harus memanggil kembali CodeDeploy dan menunjukkan keberhasilan atau kegagalan. Jika fungsi gagal, fungsi akan dibatalkan dan melaporkan kegagalan kembali ke AWS CloudFormation. Jika fungsi berhasil, CodeDeploy lanjutkan ke perpindahan lalu lintas.
 - **PostTraffic:** Setelah perpindahan lalu lintas selesai, CodeDeploy memanggil fungsi Lambda kait pasca-lalu lintas. Ini mirip dengan hook pra-lalu lintas, di mana fungsi harus menelepon kembali CodeDeploy untuk melaporkan keberhasilan atau kegagalan. Gunakan hook pasca-lalu lintas untuk menjalankan pengujian integrasi atau tindakan validasi lainnya.

Untuk informasi selengkapnya, lihat [Referensi SAM untuk Deployment yang Aman](#).

Secara bertahap menerapkan fungsi Lambda untuk pertama kalinya

Saat menerapkan fungsi Lambda secara bertahap CodeDeploy, memerlukan versi fungsi yang digunakan sebelumnya untuk mengalihkan lalu lintas dari. Oleh karena itu, penerapan pertama Anda harus diselesaikan dalam dua langkah:

- Langkah 1: Menyebarkan fungsi Lambda Anda dan secara otomatis membuat alias dengan `AutoPublishAlias`
- Langkah 2: Lakukan penyebaran bertahap Anda dengan `DeploymentPreference`

Melakukan penerapan bertahap pertama Anda dalam dua langkah memberikan CodeDeploy versi fungsi Lambda sebelumnya untuk mengalihkan lalu lintas dari.

Langkah 1: Terapkan fungsi Lambda Anda

```
Resources:
MyLambdaFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: index.handler
    Runtime: nodejs12.x
    CodeUri: s3://bucket/code.zip

    AutoPublishAlias: live
```

Langkah 2: Lakukan penyebaran bertahap

```
Resources:
MyLambdaFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: index.handler
    Runtime: nodejs12.x
    CodeUri: s3://bucket/code.zip

    AutoPublishAlias: live

DeploymentPreference:
  Type: Canary10Percent10Minutes
  Alarms:
```

```
# A list of alarms that you want to monitor
- !Ref AliasErrorMetricGreaterThanZeroAlarm
- !Ref LatestVersionErrorMetricGreaterThanZeroAlarm
Hooks:
# Validation Lambda functions that are run before and after traffic shifting
PreTraffic: !Ref PreTrafficLambdaFunction
PostTraffic: !Ref PostTrafficLambdaFunction
```

Pelajari selengkapnya

Untuk contoh langsung mengonfigurasi penerapan bertahap, lihat [Modul 5 - Penerapan Canary](#) di Lokakarya Lengkap. AWS SAM

Telemetri di AWS SAMCLI

Pada AWS, kami mengembangkan dan meluncurkan layanan berdasarkan apa yang kami pelajari dari interaksi dengan pelanggan. Kami menggunakan umpan balik pelanggan untuk mengulangi produk kami. Telemetri adalah informasi tambahan yang membantu kami untuk lebih memahami kebutuhan pelanggan kami, mendiagnosis masalah, dan memberikan fitur yang meningkatkan pengalaman pelanggan.

Antarmuka baris AWS SAM perintah (CLI) mengumpulkan telemetri, seperti metrik penggunaan generik, informasi sistem dan lingkungan, dan kesalahan. Untuk detail tentang jenis telemetri yang dikumpulkan, lihat [Jenis informasi yang dikumpulkan](#).

Kami AWS SAM CLI tidak mengumpulkan informasi pribadi, seperti nama pengguna atau alamat email. DynamoDB lokal juga tidak mengekstraksi informasi tingkat proyek yang sensitif.

Pelanggan mengontrol apakah telemetri dihidupkan, dan mereka dapat mengubah pengaturan mereka kapan saja. Jika telemetri tetap aktif, data telemetri AWS SAM CLI dikirim di latar belakang tanpa memerlukan interaksi pelanggan tambahan.

Matikan telemetri untuk sesi

Di sistem operasi macOS dan Linux, Anda dapat menonaktifkan telemetri untuk satu sesi. Untuk menonaktifkan telemetri untuk sesi Anda saat ini, jalankan perintah berikut untuk mengatur variabel lingkungan SAM_CLI_TELEMETRY ke `false`. Ulangi perintah tersebut untuk setiap sesi atau terminal baru.

```
export SAM_CLI_TELEMETRY=0
```

Menonaktifkan telemetri untuk profil Anda di semua sesi

Jalankan perintah berikut untuk mematikan telemetri untuk semua sesi saat Anda menjalankan AWS SAM CLI pada sistem operasi Anda.

Untuk menonaktifkan telemetri di Linux

1. Jalankan:.

```
echo "export SAM_CLI_TELEMETRY=0" >> ~/.profile
```

2. Jalankan:.

```
source ~/.profile
```

Untuk menonaktifkan telemetri di macOS

1. Jalankan:.

```
echo "export SAM_CLI_TELEMETRY=0" >> ~/.profile
```

2. Jalankan:.

```
source ~/.profile
```

Untuk menonaktifkan telemetri di Windows

1. Jalankan:.

```
setx SAM_CLI_TELEMETRY 0
```

2. Jalankan:.

```
refreshenv
```

Jenis informasi yang dikumpulkan

- Informasi penggunaan — Perintah umum dan subperintah yang dijalankan pelanggan.

- Kesalahan dan informasi diagnostik — Status dan durasi perintah yang dijalankan pelanggan, termasuk kode keluar, nama pengecualian internal, dan kegagalan saat menghubungkan ke Docker.
- Informasi sistem dan lingkungan — Versi Python, sistem operasi (Windows, Linux, atau macOS), lingkungan di mana AWS SAM CLI berjalan (misalnya, toolkit AWS IDEAWS CodeBuild, atau terminal), dan nilai hash atribut penggunaan.

Pelajari selengkapnya

Data telemetri yang AWS SAM CLI dikumpulkan mematuhi kebijakan privasi data. AWS Untuk informasi selengkapnya, lihat hal berikut:

- [AWSKetentuan Layanan](#)
- [FAQ Privasi Data](#)

Catatan penting

Bagian ini berisi catatan penting dan pengumuman untuk AWS Serverless Application Model (AWS SAM).

Topik

- [Catatan penting untuk 2023](#)
- [Catatan penting untuk tahun 2020](#)

Catatan penting untuk 2023

Oktober 2023

AWS SAMCLImenghentikan dukungan untuk Python 3.7

Diterbitkan: 2023-10-20

Python 3.7menerima end-of-life status pada bulan Juni 2023. Ini AWS SAM CLI akan menghentikan dukungan untuk Python 3.7 pada 24 Oktober 2023. Untuk informasi lebih lanjut, lihat [pengumuman](#) di [aws-sam-cli](#) GitHubrepositori.

Perubahan ini berdampak pada pengguna berikut:

- Jika Anda menggunakan Python 3.7 dan menginstal AWS SAM CLI melalui pip.
- Jika Anda menggunakan `aws-sam-cli` sebagai perpustakaan dan membangun aplikasi Anda dengan Python 3.7.

Jika Anda menginstal dan mengelola AWS SAM CLI melalui metode lain, Anda tidak terpengaruh.

Untuk pengguna yang terkena dampak, kami sarankan Anda meningkatkan lingkungan pengembangan Anda ke Python 3.8 atau yang lebih baru.

Perubahan ini tidak memengaruhi dukungan untuk lingkungan Python 3.7 AWS Lambda runtime. Untuk informasi selengkapnya, lihat [kebijakan penghentian waktu proses](#) di Panduan Pengembang AWS Lambda

Catatan penting untuk tahun 2020

Juni 2020

Menginstal AWS SAM CLI pada 32-bit Windows

Support untuk AWS SAM CLI Windows 32-bit akan segera usang. Jika Anda beroperasi pada sistem 32 bit, kami rekomendasikan Anda meningkatkan ke sistem 64 bit dan ikuti petunjuk yang ditemukan di [Instalasi AWS SAM CLI](#).

Jika Anda tidak dapat meningkatkan ke sistem 64-bit, Anda dapat menggunakan [Legacy Docker Toolbox](#) dengan AWS SAM CLI sistem 32-bit. Namun, ini akan menyebabkan Anda menghadapi batasan tertentu dengan AWS SAM CLI. Misalnya, Anda tidak dapat menjalankan kontainer Docker 64 bit pada sistem 32 bit. Jadi, jika fungsi Lambda Anda tergantung pada 64 bit kontainer asli yang dikompilasi, Anda tidak akan dapat mengujinya secara lokal pada sistem 32 bit.

Untuk menginstal AWS SAM CLI pada sistem 32-bit, jalankan perintah berikut:

```
pip install aws-sam-cli
```

Important

Meskipun `pip install aws-sam-cli` perintah ini juga berfungsi pada Windows 64-bit, kami sarankan Anda menggunakan [MSI 64-bit](#) untuk menginstal AWS SAM CLI pada sistem 64-bit.

Riwayat dokumen untuk AWS SAM

Tabel berikut menjelaskan perubahan penting dalam setiap perilsan Panduan Developer AWS Serverless Application Model. Untuk notifikasi tentang pembaruan dokumentasi ini, Anda dapat berlangganan umpan RSS.

- Pembaruan dokumentasi terbaru: 6 Desember 2023

Perubahan	Deskripsi	Tanggal
Tambahkan penginstal paket AWS SAMCLI arm64 untuk Linux	Untuk petunjuk, lihat Menginstal AWS SAMCLI .	6 Desember 2023
Tambahkan opsi --watch-exclude untuk perintah sam sync AWS SAMCLI	Kecualikan file dan folder dari memulai sinkronisasi. Untuk mempelajari selengkapnya, lihat Menentukan file dan folder yang tidak akan memulai sinkronisasi .	6 Desember 2023
Tambahkan -- build-in-source opsi untuk perintah AWS SAMCLI sam sync	Bangun proyek Anda di folder sumber Anda untuk mempercepat proses pembuatan. Untuk mempelajari lebih lanjut, lihat Mempercepat waktu pembuatan dengan membuat project di folder sumber .	6 Desember 2023
Tambahkan -- build-in-source opsi untuk perintah AWS SAMCLI sam build	Bangun proyek Anda di folder sumber Anda untuk mempercepat proses pembuatan. Untuk mempelajari lebih lanjut, lihat Mempercepat waktu pembuatan dengan	6 Desember 2023

membuat project di folder sumber.		
Tambahkan dukungan sumber daya baru untuk perintah pemanggilan AWS SAMCLI jarak jauh	Gunakan sam remote invoke dengan aplikasi Kinesis Data Streams, antrian Amazon SQS, dan mesin status Step Functions. Untuk mempelajari lebih lanjut, lihat Menggunakan sam remote invoke .	15 November 2023
Tambahkan perintah uji-acara AWS SAMCLI jarak jauh baru untuk acara pengujian yang dapat dibagikan	Gunakan AWS SAM CLI untuk mengakses dan mengelola peristiwa pengujian yang dapat dibagikan dari registri EventBridge skema untuk menguji fungsi Lambda Anda di. AWS Cloud Untuk mempelajari lebih lanjut, lihat Menggunakan sam remote test-event .	3 Oktober 2023
AWS SAMCLIdukungan untuk Terraform sekarang tersedia secara umum	Untuk mempelajari lebih lanjut tentang AWS SAM CLI dukunganTerraform, lihat AWS SAMCLITerraformdukungan .	5 September 2023
Tambahkan AWS SAMCLI dukungan untuk Terraform Cloud	AWS SAMCLISekarang mendukung pengujian lokal untukTerraform Cloud. Untuk mempelajari lebih lanjut, lihat Mengatur untuk Terraform Cloud .	5 September 2023

Tambahkan dukungan format YAML file untuk file AWS SAMCLI konfigurasi	AWS SAMCLISekarang mendukung format file [.yaml]. Mengkonfigurasi halaman file AWS SAMCLI dan AWS SAMCLI konfigurasi telah diperbarui.	Juli 18, 2023
Tambahkan dukungan AWS SAMCLIsam local start-api perintah untuk Terraform	Untuk apa AWS SAMCLI dukunganTerraform? bagian telah diperbarui untuk menyertakan dukungan AWS SAM CLI sam local start-api perintah untukTerraform.	6 Juli 2023
Tambahkan perintah pemanggilan AWS SAMCLI jarak jauh baru	Untuk mulai menggunakan ansam remote invoke, lihat Menggunakan sam remote invoke .	22 Juni 2023
Tambahkan jenis AWS AppSyncGraphQL API sumber daya tanpa server	Buat AWS::Serverless::GraphQLApi bagian baru yang menjelaskan cara mendefinisikan GraphQL API sumber daya denganAWS SAM.	22 Juni 2023
Tambahkan AWS SAMCLI dukungan untuk Ruby 3.2	Perbarui halaman sam init untuk menyertakan gambar dasar dan nilai runtime baru. Perbarui halaman repositori Gambar dengan 3.2 Ruby Amazon ECR URI.	6 Juni 2023

Tambahkan langkah-langkah opsional untuk verifikasi integritas penginstal AWS SAMCLI paket	Perbarui Menginstal AWS SAMCLI halaman untuk mencerminkan langkah opsional. Buat Verifikasi integritas halaman AWS SAMCLI penginstal untuk mendokumentasikan langkah-langkah.	31 Mei 2023
Tambahkan opsi sinkronisasi sam untuk melewati sinkronisasi infrastruktur	Sesuaikan apakah AWS CloudFormation penerapan diperlukan setiap kali sam sync dijalankan. Untuk mempelajari lebih lanjut, lihat Lewati AWS CloudFormation penerapan awal .	Maret 23, 2023
Tambahkan dukungan untuk jenis sumber acara DocumentDB	Spesifikasi AWS SAM template sekarang mendukung jenis sumber DocumentDB peristiwa untuk <code>AWS::Serverless::Function</code> sumber daya. Untuk mempelajari lebih lanjut, lihat DocumentDB .	10 Maret 2023
Bangun fungsi Rust Lambda dengan Cargo Lambda	Gunakan AWS SAM CLI untuk membangun fungsi Lambda Rust Anda menggunakan <code>Cargo Lambda</code> . Untuk mempelajari lebih lanjut, lihat Membangun fungsi Lambda Karat dengan <code>Cargo Lambda</code> .	23 Februari 2023

[Membangun sumber daya fungsi di luar AWS SAM](#)

Menambahkan panduan tentang melewati fungsi saat menggunakan sam build perintah. Untuk mempelajari lebih lanjut, lihat [Membangun fungsi di luar AWS SAM](#).

14 Februari 2023

[Sintaks konektor tertanam baru](#)

Gunakan sintaks konektor tertanam baru untuk menentukan `AWS::Serverless::Connector` sumber daya Anda. Untuk mempelajari selengkapnya, lihat [Mengelola izin sumber daya dengan AWS SAM konektor](#).

Februari 8, 2023

[Ditambahkan perintah daftar sam baru untuk AWS SAMCLI](#)

Gunakan sam list untuk melihat informasi penting tentang sumber daya dalam aplikasi tanpa server Anda. Untuk mempelajari lebih lanjut, lihat [daftar sam](#).

2 Februari 2023

[Ditambahkan Format dan OutExtension membangun properti untuk esbuild](#)

Membangun fungsi Lambda Node.js dengan esbuild sekarang mendukung `Format` dan `OutExtension` membangun properti. Untuk mempelajari selengkapnya, lihat [Membangun fungsi Lambda Node.js dengan esbuild](#).

2 Februari 2023

Menambahkan opsi manajemen runtime ke spesifikasi AWS SAM template	Konfigurasi opsi manajemen runtime untuk fungsi Lambda Anda. Untuk mempelajari selengkapnya, lihat RuntimeManagementConfig .	Januari 24, 2023
Properti target ditambahkan EventSource untuk AWS::Serverless::StateMachine sumber daya.	<code>AWS::Serverless::StateMachine</code> jenis sumber daya mendukung Target properti untuk EventBridgeRule dan sumber Schedule acara.	13 Januari 2023
Konfigurasi penskalaan poller SQS untuk fungsi Lambda	Konfigurasi penskalaan poller SQS dengan properti untuk <code>ScalingConfig</code> <code>AWS::Serverless::Function</code> Untuk mempelajari selengkapnya, lihat ScalingConfig .	Januari 12, 2023
Validasi AWS SAM aplikasi dengan cfn-lint	Anda dapat menggunakan <code>cfn-lint</code> untuk memvalidasi template Anda AWS SAM melalui <code>AWS SAM CLI</code> Untuk mempelajari lebih lanjut, lihat Memvalidasi dengan cfn-lint .	11 Januari 2023
Pantau aplikasi tanpa server Anda dengan CloudWatch Application Insights	Konfigurasi Wawasan CloudWatch Aplikasi Amazon untuk memantau AWS SAM aplikasi Anda. Untuk mempelajari lebih lanjut, lihat Memantau aplikasi tanpa server Anda dengan Wawasan CloudWatch Aplikasi .	19 Desember 2022

Menambahkan penginstal AWS SAMCLI paket untuk macOS	Instal AWS SAM CLI menggunakan penginstal paket macOS baru. Untuk mempelajari lebih lanjut, lihat Menginstal AWS SAMCLI .	6 Desember 2022
Ditambahkan dukungan untuk Lambda SnapStart	Konfigurasi fungsi Lambda Anda SnapStart untuk membuat snapshot, yang merupakan status cache dari fungsi yang diinisialisasi. Untuk mempelajari selengkapnya, lihat AWS::Serverless::Function .	28 November 2022
Ditambahkan AWS SAMCLI dukungan untuk nodejs18.x	AWS SAMCLI sekarang mendukung runtime nodejs18.x. Untuk mempelajari lebih lanjut, lihat sam init .	17 November 2022
Menambahkan panduan tentang mengonfigurasi akses dan izin	AWS SAM menyediakan dua opsi yang menyederhanakan pengelolaan akses dan izin untuk aplikasi tanpa server Anda. Untuk mempelajari lebih lanjut, lihat Mengelola akses dan izin sumber daya .	17 November 2022

[Menambahkan dukungan untuk membangun fungsi.NET 7 Lambda dengan kompilasi AOT Asli](#)

Bangun dan kemas fungsi-fungsi Lambda .NET 7 Anda AWS SAM, gunakan kompilasi Native Ahead-of-Time (AOT) untuk meningkatkan waktu mulai dingin Lambda. Untuk mempelajari lebih lanjut, lihat [Membangun fungsi.NET 7 Lambda dengan kompilasi AOT Asli](#).

15 November 2022

[Menambahkan AWS SAM CLI Terraform dukungan untuk debugging dan pengujian lokal](#)

Gunakan bagian AWS SAM CLI dalam Terraform proyek Anda untuk melakukan debugging lokal dan pengujian fungsi dan lapisan Lambda Anda. Untuk mempelajari lebih lanjut, lihat [AWS SAM dukungan CLI Terraform](#).

November 14, 2022

[Ditambahkan AWS SAM dukungan untuk EventBridge Scheduler](#)

Spesifikasi template AWS Serverless Application Model (AWS SAM) menyediakan sintaks sederhana dan singkat yang dapat Anda gunakan untuk menjadwalkan acara dengan EventBridge Scheduler untuk dan. AWS Lambda AWS Step Functions Untuk informasi selengkapnya, lihat [Menjadwalkan acara dengan EventBridge Penjadwal](#).

10 November 2022

Menyederhanakan instruksi AWS SAMCLI instalasi	AWS SAMCLIPrasyarat dan langkah-langkah opsional telah dipindahkan ke halaman terpisah. Langkah-langkah instalasi untuk sistem operasi yang didukung dapat ditemukan di Instalasi AWS SAMCLI .	4 November 2022
Menambahkan perbaikan untuk memungkinkan jalur panjang untuk pengguna Windows 10	Repositori template AWS SAM CLI aplikasi berisi beberapa jalur file panjang yang dapat menyebabkan kesalahan saat berjalan sam init karena keterbatasan Windows 10MAX_PATH. Untuk informasi selengkapnya, lihat Menginstal AWS SAMCLI	4 November 2022
Memperbarui proses penyebaran bertahap untuk penerapan pertama kali	Secara bertahap menerapkan fungsi Lambda AWS CodeDeploy dengan membutuhkan dua langkah. Untuk mempelajari lebih lanjut, lihat Menerapkan fungsi Lambda secara bertahap untuk pertama kalinya .	13 Oktober 2022
Dukungan penyaringan acara Lambda tambahan untuk lebih banyak jenis acara	FilterCriteria properti ditambahkan ke MSK , MQ , dan jenis sumber SelfManagedKafka acara.	13 Oktober 2022

[Menambahkan dukungan OpenID Connect \(OIDC\) untuk pipeline AWS SAM](#)

AWS SAM mendukung otentikasi pengguna OpenID Connect (OIDC) untuk platform Bitbucket, GitHub Actions, dan integrasi GitLab berkelanjutan dan pengiriman berkelanjutan (CI/CD). Untuk mempelajari lebih lanjut, lihat [Menggunakan Akun Pengguna OIDC dengan AWS SAM pipeline](#).

13 Oktober 2022

[Catatan tentang JwtConfiguration properti](#)

Menambahkan catatan tentang mendefinisikan issuer dan audience properti di bawah JwtConfiguration untuk [OAuth2Authorizer](#).

Oktober 7, 2022

[Properti baru untuk Fungsi dan StateMachine EventSource](#)

Enabled dan State properti ditambahkan ke sumber CloudWatchEvent acara untuk [AWS::Serverless::Function](#). State properti ditambahkan ke sumber Schedule acara untuk [AWS::Serverless::Function](#) dan [AWS::Serverless::StateMachine](#).

6 Oktober 2022

AWS SAMkonektor sekarang umumnya tersedia	Konektor adalah jenis sumber daya AWS SAM abstrak, diidentifikasi sebagai <code>AWS::Serverless::Connector</code> , yang menyediakan metode penyediaan izin yang sederhana dan aman antara sumber daya aplikasi tanpa server Anda. Untuk mempelajari selengkapnya, lihat Mengelola izin sumber daya dengan AWS Serverless Application Model konektor .	6 Oktober 2022
Menambahkan opsi sinkronisasi sam baru ke AWS SAMCLI	<code>--dependency-layer</code> dan <code>--use-container</code> opsi ditambahkan ke sam sync .	September 20, 2022
Menambahkan opsi penerapan sam baru ke AWS SAMCLI	<code>--on-failure</code> opsi ditambahkan ke sam deploy .	9 September 2022
dukungan esbuild sekarang tersedia secara umum	Untuk membangun dan mengemas fungsi Lambda Node.js, Anda dapat menggunakan bundler AWS SAM CLI with JavaScript esbuild .	September 1, 2022
AWS SAMCLITelemetri yang diperbarui	Deskripsi informasi sistem dan lingkungan yang dikumpulkan telah diperbarui untuk menyertakan nilai hash dari atribut penggunaan.	September 1, 2022

[Menambahkan dukungan variabel lingkungan lokal ke AWS SAMCLI](#)

Gunakan variabel lingkungan dengan AWS SAM CLI saat [menjalankan fungsi Lambda](#) secara lokal dan saat menjalankan [API Gateway](#) secara lokal.

September 1, 2022

[Support untuk arsitektur set instruksi Lambda](#)

Gunakan AWS SAM CLI untuk membangun fungsi Lambda dan lapisan Lambda untuk x86_64 atau arm64 arsitektur set instruksi. Untuk informasi selengkapnya, lihat properti [Arsitektur](#) dari tipe `AWS::Serverless::Function` sumber daya dan [CompatibleArchitectures](#) properti tipe `AWS::Serverless::LayerVersion` sumber daya.

1 Oktober 2021

[Menghasilkan contoh konfigurasi pipa](#)

Gunakan AWS SAM CLI untuk menghasilkan contoh pipeline untuk beberapa sistem CI/CD, menggunakan perintah baru [sam pipeline bootstrap](#) dan [sam pipeline init](#). Untuk informasi selengkapnya, lihat [Membuat contoh alur CI/CD](#).

21 Juli 2021

AWS SAM CLI AWS CDK integrasi (pratinjau, fase 2)	Dengan fase 2 dari rilis pratinjau publik, Anda sekarang dapat menggunakan paket AWS SAM CLI untuk dan menyebarkan AWS CDK aplikasi. Anda juga dapat mengunduh AWS CDK aplikasi sampel secara langsung menggunakan file AWS SAM CLI. Untuk informasi selengkapnya, lihat AWS Cloud Development Kit (AWS CDK) (Pratinjau) .	13 Juli 2021
Support untuk RabbitMQ sebagai sumber acara untuk fungsi	Menambahkan dukungan untuk RabbitMQ sebagai sumber peristiwa untuk fungsi nirserver. Untuk informasi selengkapnya, lihat properti SourceAccessConfigurations sumber peristiwa MQ dari tipe sumber daya AWS::Serverless::Function .	7 Juli 2021
Menerapkan aplikasi tanpa server menggunakan Amazon ECR build container image	Gunakan image container build Amazon ECR untuk menyebarkan aplikasi tanpa server dengan sistem CI/CD umum seperti, JenkinsAWS CodePipeline, CI/CD, dan Actions. GitLab GitHub Untuk informasi selengkapnya, lihat Men-deploy aplikasi nirserver .	24 Juni 2021

Debugging AWS SAM aplikasi dengan Toolkit AWS	Kit alat AWS sekarang mendukung debugging bertahap dengan lebih banyak kombinasi lingkungan pengembangan terpadu (IDE) dan waktu aktif. Untuk informasi selengkapnya, lihat Menggunakan Kit Alat AWS .	20 Mei 2021
AWS SAM CLI AWS CDK integrasi (pratinjau)	Anda sekarang dapat menggunakan AWS SAM CLI untuk menguji dan membangun AWS CDK aplikasi secara lokal. Ini adalah perilis publik pratinjau. Untuk informasi selengkapnya, lihat AWS Cloud Development Kit (AWS CDK) (Pratinjau) .	29 April 2021
Repositori gambar kontainer default diubah menjadi Amazon ECR Public	Repositori gambar kontainer default berubah dari DockerHub Amazon ECR Public . Untuk informasi selengkapnya, lihat Repositori citra .	6 April 2021
Membangun malam AWS SAM CLI	Anda sekarang dapat menginstal versi pra-rilis AWS SAM CLI, yang dibangun setiap malam. Untuk informasi selengkapnya, lihat bagian Nightly build dari subtopik OS pilihan Anda di bawah Menginstal. AWS SAM CLI	25 Maret 2021

[Membangun dukungan variabel lingkungan kontainer](#)

Sekarang Anda dapat meneruskan variabel lingkungan untuk membangun kontainer. Untuk informasi selengkapnya, lihat opsi `--container-env-var` and `--container-env-var-file` dalam [sam build](#).

4 Maret 2021

[Proses instalasi Linux baru](#)

Anda sekarang dapat menginstal AWS SAM CLI menggunakan installer Linux asli. Untuk informasi selengkapnya, lihat [Menginstal AWS SAMCLI di Linux](#).

10 Februari 2021

[Support untuk antrian dead-letter untuk EventBridge](#)

Menambahkan dukungan untuk antrian surat mati untuk dan sumber Schedule acara untuk EventBridge fungsi tanpa server dan mesin status. Untuk informasi selengkapnya, lihat properti `DeadLetterConfig` dari sumber peristiwa `EventBridgeRule` dan `Schedule`, untuk kedua tipe sumber daya [AWS::Serverless::Function](#) dan [AWS::Serverless::StateMachine](#).

29 Januari 2021

[Support untuk pos pemeriksaan kustom](#)

Menambahkan dukungan untuk titik pemeriksaan kustom untuk sumber peristiwa DynamoDB dan Kinesis untuk fungsi nirserver. Untuk informasi selengkapnya, lihat properti `FunctionResponseTypes` dari tipe data [Kinesis](#) dan [DynamoDB](#) tipe sumber daya [AWS::Serverless::Function](#) .

29 Januari 2021

[Support untuk jendela yang jatuh](#)

Menambahkan dukungan untuk windows yang jatuh untuk sumber peristiwa DynamoDB dan Kinesis untuk fungsi nirserver. Untuk informasi selengkapnya, lihat properti `TrailingWindowInSeconds` dari tipe data [Kinesis](#) dan [DynamoDB](#) tipe sumber daya [AWS::Serverless::Function](#) .

17 Desember 2020

[Support untuk wadah hangat](#)

Menambahkan dukungan untuk wadah hangat saat menguji secara lokal menggunakan AWS SAM CLI perintah [sam local start-api](#) dan [sam local start-lambda](#) . Untuk informasi selengkapnya, lihat opsi `--warm-containers` untuk perintah tersebut.

16 Desember 2020

[Support untuk gambar kontainer Lambda](#)

Menambahkan dukungan untuk citra kontainer Lambda. Untuk informasi selengkapnya, lihat [Membangun aplikasi](#).

1 Desember 2020

[Support untuk penandatangan kode](#)

Menambahkan dukungan untuk penandatanganan kode dan deployment tepercaya dari kode aplikasi nirserver. Untuk informasi selengkapnya, lihat [Mengonfigurasi penandatangan kode untuk aplikasi AWS SAM](#).

23 November 2020

[Support untuk build paralel dan cache](#)

Peningkatan performa aplikasi nirserver dibangun dengan menambahkan dua opsi ke perintah [sam build](#): `--parallel`, yang membangun fungsi dan lapisan secara paralel, bukan secara berurutan, dan `--cached`, yang menggunakan pembangunan artefak bangunan dari bangunan sebelumnya ketika tidak ada perubahan yang dibuat yang memerlukan pembangunan ulang.

10 November 2020

[Support untuk Amazon MQ, dan otentikasi TLS timbal balik](#)

Menambahkan dukungan untuk Amazon MQ sebagai sumber peristiwa untuk fungsi nirserver. Untuk informasi selengkapnya, lihat tipe data [EventSource](#) dan [MQ](#) tipe sumber daya [AWS::Serverless::Function](#) . Juga menambahkan dukungan untuk autentikasi Keamanan Lapisan Pengangkutan (TLS) bersama untuk API dari API Gateway dan API HTTP. Untuk informasi selengkapnya, lihat tipe data [DomainConfiguration](#) dari tipe sumber daya [AWS::Serverless::Api](#) , atau tipe data [HttpApiDomainConfiguration](#) dari tipe sumber daya [AWS::Serverless::HttpApi](#) .

5 November 2020

[Support untuk Lambda Authorizer untuk HTTP API](#)

Menambahkan dukungan untuk otorisasi Lambda untuk tipe sumber daya [AWS::Serverless::HttpApi](#) . Untuk informasi selengkapnya, lihat [Contoh otorisasi Lambda \(AWS::Serverless::HttpApi\)](#) .

27 Oktober 2020

[Support untuk beberapa file konfigurasi dan lingkungan](#)

Menambahkan dukungan untuk beberapa file konfigurasi dan lingkungan untuk menyimpan nilai parameter default untuk AWS SAM CLI perintah. Untuk informasi selengkapnya, lihat [file AWS SAMCLI konfigurasi](#).

24 September 2020

[Support untuk X-Ray dengan Step Functions, dan referensi saat mengontrol akses ke API](#)

Menambahkan dukungan untuk X-Ray sebagai sumber peristiwa untuk mesin status nirserver. Untuk informasi selengkapnya, lihat properti [Tracing](#) tipe sumber daya [AWS::Serverless::StateMachine](#). Juga menambahkan dukungan untuk referensi saat mengontrol akses ke API. Untuk informasi selengkapnya, lihat tipe data [ResourcePolicyStatement](#).

17 September 2020

[Dukungan untuk Amazon MSK](#)

Menambahkan dukungan untuk Amazon MSK sebagai sumber peristiwa untuk fungsi nirserver. Hal ini memungkinkan catatan dalam topik Amazon MSK untuk memicu fungsi Lambda Anda. Untuk informasi selengkapnya, lihat tipe data [EventSource](#) dan [MSK](#) dari tipe sumber daya [AWS::Serverless::Function](#).

13 Agustus 2020

Dukungan untuk Amazon EFS	Menambahkan dukungan untuk pemasangan sistem file Amazon EFS ke direktori lokal. Hal ini memungkinkan kode fungsi Lambda Anda untuk mengakses dan mengubah sumber daya bersama. Untuk informasi selengkapnya, lihat properti FileSystemConfigs dari tipe sumber daya AWS::Serverless::Function .	16 Juni 2020
Mengatur aplikasi tanpa server	Menambahkan dukungan untuk mengorkestrasi aplikasi dengan membuat mesin status Step Functions menggunakan AWS SAM. Untuk informasi selengkapnya, lihat Mengorkestrasikan sumber daya AWS dengan AWS Step Functions dan tipe sumber daya AWS::Serverless::StateMachine .	27 Mei 2020
Membangun runtime kustom	Menambahkan kemampuan untuk membangun waktu aktif kustom. Untuk informasi selengkapnya, lihat Membangun waktu aktif kustom .	21 Mei 2020

Membangun lapisan	Menambahkan kemampuan untuk membangun sumber daya LayerVersion individu. Untuk informasi selengkapnya, lihat Membangun lapisan .	19 Mei 2020
AWS CloudFormation Sumber daya yang dihasilkan	Memberikan detail tentang sumber daya AWS CloudFormation yang dibuat AWS SAM dan cara mereferensikannya. Untuk informasi selengkapnya, lihat Sumber daya AWS CloudFormation yang dibuat .	8 April 2020
Menyiapkan AWS kredensial	Menambahkan instruksi untuk menyiapkan kredensial AWS saat Anda belum mengaturnya untuk digunakan dengan alat AWS lain, seperti salah satu AWS SDK atau AWS CLI. Untuk informasi selengkapnya, lihat Menyiapkan kredensial AWS .	17 Januari 2020
AWS SAM spesifikasi dan AWS SAM CLI pembaruan	Memigrasikan AWS SAM spesifikasi dari GitHub. Untuk informasi selengkapnya, lihat Spesifikasi AWS SAM . Juga memperbarui alur kerja deployment dengan perubahan pada perintah sam deploy .	25 November 2019

[Opsi baru untuk mengontrol akses ke API Gateway API dan pembaruan templat kebijakan](#)

Menambahkan opsi baru untuk mengontrol akses ke API dari API Gateway: izin IAM, kunci API, dan kebijakan sumber daya. Untuk informasi selengkapnya, lihat [Mengontrol akses ke API dari API Gateway](#). Juga memperbarui dua templat kebijakan: RekognitionFacesPolicy dan ElasticsearchHttpPostPolicy. Untuk informasi selengkapnya, lihat [Templat kebijakan AWS SAM](#).

29 Agustus 2019

[Memulai pembaruan](#)

Memperbarui babak memulai dengan instruksi instalasi yang ditingkatkan untuk tutorial Hello World AWS SAM CLI dan Hello. Untuk informasi selengkapnya, lihat [Memulai dengan AWS SAM](#).

25 Juli 2019

[Mengontrol akses ke API Gateway API](#)

Menambahkan dukungan untuk mengontrol akses ke API dari API Gateway. Untuk informasi selengkapnya, lihat [Mengontrol akses ke API dari API Gateway](#).

21 Maret 2019

Ditambahkan <code>sam publish</code> ke AWS SAM CLI	sam publish Perintah baru dalam AWS SAM CLI menyederhanakan proses untuk menerbitkan aplikasi tanpa server di file. AWS Serverless Application Repository Untuk informasi selengkapnya, lihat Menerbitkan aplikasi tanpa server menggunakan . AWS SAM CLI	21 Desember 2018
Dukungan aplikasi dan lapisan bersarang	Menambahkan dukungan untuk aplikasi dan lapisan bersarang. Untuk informasi selengkapnya, lihat Menggunakan aplikasi bersarang dan Bekerja dengan lapisan .	29 November 2018
Ditambahkan <code>sam build</code> ke AWS SAM CLI	sam build Perintah baru dalam AWS SAM CLI menyederhanakan proses untuk mengkompilasi aplikasi tanpa server dengan dependensi sehingga Anda dapat menguji dan menyebarkan aplikasi ini secara lokal. Untuk informasi selengkapnya, lihat Membangun aplikasi .	19 November 2018
Menambahkan opsi instalasi baru untuk AWS SAM CLI	Menambahkan opsi instalasi Linuxbrew (Linux), MSI (Windows), dan Homebrew (macOS) untuk AWS SAM CLI Untuk informasi selengkapnya, lihat Menginstal AWS SAM CLI .	7 November 2018

[Panduan baru](#)

Ini adalah perilsan pertama
dari Panduan Developer AWS
Serverless Application Model.

17 Oktober 2018

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.