개발자 가이드

AWS Serverless Application Model

AWS Serverless Application Model: 개발자 가이드

Table of Contents

AWS SAM란 무엇인가요? 1
- 사용 시기 AWS SAM 1
- 주요 기능 2
- 관련 정보 3
- 작동 방식 3
  - AWS SAM 템플릿 사양이란 무엇입니까? 3
  - AWS SAM 프로젝트와 AWS SAM 템플릿이란 무엇입니까? 4
  - 란 무엇입니까 AWS SAMCLI? 10
  - 자세히 알아보기 19
  - 다음 단계 19
- 서버리스 개념 19
  - 서버리스 개념 20
- 코드형 인프라 21
  - AWS 서버리스 애플리케이션을 위한 IaC 도구 21
  - 서버리스 애플리케이션을 위한 IaC 도구 비교 22
  - 자세히 알아보기 23

시작하기 24
- 사전 조건 24
  - 1단계: AWS 계정 가입 25
  - 2단계: IAM 사용자 계정 생성 25
  - 3단계: 액세스 키 ID 및 비밀 액세스 키 생성 26
  - 4단계: 설치 AWS CLI 28
  - 5단계: AWS CLI 를 사용하여 AWS 자격 증명 구성 28
  - 6단계: (선택 사항) VS Code용 AWS 도구 키트 설치 29
  - 다음 단계 30
- AWS SAM CLI 설치 30
  - AWS SAMCLI의 설치 31
  - 설치 오류 문제 해결 41
  - 다음 단계 42
  - 선택 사항: AWS SAM CLI 설치관리자 프로그램 확인 43
- Hello World 자습서 55
  - 사전 조건 56
  - 1단계: Hello World 샘플 애플리케이션 초기화 57
  - 2단계: 귀하의 애플리케이션 빌드 61

3단계: AWS 클라우드에 대한 애플리케이션 배포 63
4단계: 애플리케이션의 실행 67
5단계: AWS 클라우드에서 함수와 상호작용하십시오. 69
6단계: 애플리케이션을 수정하고 AWS 클라우드에 동기화합니다. 69
7단계: (선택 사항) 로컬에서 애플리케이션 테스트 73
8단계: AWS 클라우드에서 애플리케이션 삭제 75
문제 해결 76
자세히 알아보기 76
Lambda에서 변환하기 76
자세히 알아보기 77
AWS SAM 사용 방법 78
AWS SAM CLI 79
AWS SAMCLI 명령을 문서화하는 방법 79
AWS SAM CLI 구성 81
구성 파일(`samconfig` 파일)을 생성하는 방법 81
프로젝트 설정 구성 82
자격 증명 및 기본 설정 구성 86
핵심 명령 86
AWS SAM CLI을(를) 사용한 로컬 테스트 88
로컬에서 AWS SAM 애플리케이션 테스트 88
로컬에서 CloudFormation 템플릿 테스트 88
로컬에서 AWS CDK 애플리케이션 테스트 88
Terraform 애플리케이션 로컬 테스트 89
AWS SAM 템플릿 90
템플릿 구조 91
YAML 91
템플릿 섹션 92
다음 단계 94
Globals 94
리소스 및 속성 99
AWS::Serverless::Api 100
AWS::Serverless::Application 157
AWS::Serverless::CapacityProvider 162
AWS::Serverless::Connector 170
AWS::Serverless::Function 185
AWS::Serverless::GraphQLApi 335

AWS::Serverless::HttpApi 370
AWS::Serverless::LayerVersion 401
AWS::Serverless::SimpleTable 407
AWS::Serverless::StateMachine 412
생성된 리소스 454
생성된 CloudFormation 리소스 참조 454
생성된 CloudFormation 리소스 시나리오 455
AWS::Serverless::Api 458
AWS::Serverless::Application 459
AWS::Serverless::Connector 460
AWS::Serverless::Function 461
AWS::Serverless::GraphQLApi 467
AWS::Serverless::HttpApi 468
AWS::Serverless::LayerVersion 470
AWS::Serverless::SimpleTable 470
AWS::Serverless::StateMachine 471
지원되는 리소스 속성 472
예외 473
API 게이트웨이 익스텐션 474
내장 함수 475
애플리케이션의 예 477
DynamoDB 이벤트 처리 477
시작하기 전 준비 사항 477
1단계: 애플리케이션 초기화 477
2단계: 애플리케이션 로컬 테스트 478
3단계: 애플리케이션 패키징 478
4단계: 애플리케이션 배포 479
다음 단계 480
Amazon S3 이벤트 처리 480
시작하기 전 준비 사항 480
1단계: 애플리케이션 초기화 481
2단계: 애플리케이션 패키징 481
3단계: 애플리케이션 배포 482
2단계: 애플리케이션 로컬 테스트 483
다음 단계 483
애플리케이션 개발 484

- 애플리케이션 생성 ... 484
  - 새 서버리스 애플리케이션 초기화 ... 485
  - sam init에 대한 옵션 ... 491
  - 문제 해결 ... 491
  - 예제 ... 491
  - 자세히 알아보기 ... 492
  - 다음 단계 ... 492
- 인프라 정의 ... 492
  - 애플리케이션 리소스 정의 ... 493
  - 액세스 설정 ... 494
  - API 액세스 제어 ... 572
  - 계층으로 효율성 향상 ... 585
  - 코드 재사용 ... 588
  - 시간 기반 이벤트 관리 ... 591
  - 애플리케이션 오케스트레이션 ... 594
  - 코드 서명 설정 ... 596
  - AWS SAM 템플릿 파일 검증 ... 599
- 애플리케이션 빌드 ... 599
  - sam build 소개 ... 600
  - 기본 빌드 ... 614
  - 빌드 사용자 지정 ... 622

애플리케이션 테스트 ... 647
- sam local 소개 ... 647
  - sam local 명령 사용 ... 648
  - sam local generate-event 소개 ... 648
  - sam local invoke 소개 ... 654
  - sam local start-api 소개 ... 660
  - sam local start-lambda 소개 ... 666
- 로컬로 함수 호출 ... 669
  - 환경 변수 파일 ... 670
  - 계층 ... 671
  - 자세히 알아보기 ... 671
- 로컬로 API Gateway 실행 ... 671
  - 환경 변수 파일 ... 673
  - 계층 ... 674
- sam remote test-event를 사용하여 테스트 ... 675

sam remote test-event를 사용하도록 AWS SAMCLI를 설정합니다. 675
sam remote test-event 명령 사용 676
공유 가능한 테스트 이벤트 사용 679
공유 가능한 테스트 이벤트 관리 679
sam remote invoke를 사용하여 테스트 680
sam remote invoke 명령 사용 681
sam remote invoke 명령 옵션 사용 686
프로젝트 구성 파일을 구성합니다. 691
예시 691
관련 링크 707
통합 테스트 자동화 707
샘플 페이로드 생성 709
내구성 함수 테스트 및 디버깅 709
로컬 테스트 워크플로 709
일반적인 테스트 시나리오 710
문제 해결 711
디버깅 팁 711
자세히 알아보기 712
애플리케이션 디버깅 713
로컬로 함수 디버깅 713
AWS 도구 키트 사용 714
디버그 모드에서 AWS SAM을 로컬에서 실행 716
여러 런타임 인수 전달 717
cfn-lint로 검증하기 718
예시 718
자세히 알아보기 718
애플리케이션 및 리소스 배포 719
sam deploy 소개 719
사전 조건 720
sam deploy를 사용하여 애플리케이션 배포하기 720
모범 사례 730
sam deploy 옵션 730
문제 해결 731
예시 731
자세히 알아보기 739
배포 옵션 739

AWS SAM CLI를 사용하여 수동으로 배포하는 방법 740
CI/CD 시스템 및 파이프라인을 사용하여 배포 740
점진적 배포 740
AWS SAM CLI를 이용한 배포 문제 해결 741
자세히 알아보기 671
CI/CD 시스템 및 파이프라인을 사용하여 배포 741
파이프라인이란 무엇입니까? 742
AWS SAM이 로컬 파일을 업로드하는 방법 743
스타터 파이프라인 생성 751
스타터 파이프라인 사용자 지정 756
배포 자동화 758
OIDC 인증 사용 762
sam sync 소개 765
로컬 변경 사항을 자동으로 감지하고 AWS 클라우드에 동기화 766
AWS 클라우드에 동기화할 로컬 변경 사항 사용자 지정 767
테스트 및 검증을 위해 클라우드에서 애플리케이션 준비하기 767
sam sync 명령의 옵션 767
문제 해결 770
예시 770
자세히 알아보기 777
애플리케이션 모니터링 778
Application Insights 778
AWS SAM를 사용하여 CloudWatch Application Insights 구성하기 778
다음 단계 782
로그 작업 782
CloudFormation 스택으로 로그 가져오기 782
Lambda 함수 이름으로 로그 가져오기 782
테일링 로그 783
특정 시간 범위의 로그 보기 783
로그 필터링 783
오류 강조 표시 783
JSON 스타일리시 인쇄 783
AWS SAM 참조 784
AWS SAM 사양 및 AWS SAM 템플릿 784
AWS SAMCLI 명령 참조 784
AWS SAM 정책 템플릿 785

주제 785
AWS SAM CLI 명령 785
sam build 786
sam delete 792
sam deploy 794
sam init 799
sam list 802
sam local callback 810
sam local execution 816
sam local generate-event 822
sam local invoke 823
sam local start-api 829
sam local start-lambda 834
sam logs 838
sam package 842
sam pipeline bootstrap 845
sam pipeline init 850
sam publish 851
sam remote callback 853
sam remote execution 859
sam remote invoke 865
sam remote test-event 870
sam sync 877
sam traces 884
sam validate 885
AWS SAM CLI 관리 887
AWS SAMCLI구성 파일 887
AWS SAM CLI 버전 관리 894
AWS 보안 인증 설정 903
AWS SAM원격 측정CLI 905
문제 해결 907
커넥터 참조 912
지원되는 커넥터 리소스 유형 913
커넥터에서 생성한 IAM 정책 923
Docker 설치 946
Docker 설치 946

다음 단계 ... 949
Finch 설치하기 ... 949
Finch 설치하기 ... 950
AWS SAM CLI로 Finch 구성하기 ... 952
설치 확인하기 ... 953
Finch 중지하기 ... 954
문제 해결 ... 955
다음 단계 ... 955
이미지 리포지토리 ... 956
이미지 리포지토리 URI ... 956
예제 ... 958
점진적 배포 ... 958
Lambda 함수를 처음으로 점진적으로 배포 ... 961
자세히 알아보기 ... 962
중요 정보 ... 963
2023 ... 963
Terraform 지원 ... 964
시작하기 ... 964
사전 조건 ... 964
Terraform과 함께 AWS SAMCLI 명령 사용 ... 965
Terraform 프로젝트 설정 ... 965
Terraform Cloud에 대해 설정 ... 971
Terraform과 함께 AWS SAM CLI 사용 ... 972
sam local invoke를 사용한 로컬 테스트 ... 973
sam local start-api를 사용한 로컬 테스트 ... 973
sam local start-lambda를 사용한 로컬 테스트 ... 975
Terraform 제한 사항 ... 975
Serverless.tf와 함께 AWS SAMCLI 사용 ... 975
Terraform 참조 ... 976
AWS SAM 지원되는 기능 참조 ... 976
Terraform 특정 참조 ... 976
sam 메타데이터 ... 977
AWS SAM CLI Terraform 지원 ... 979
란 무엇입니까 AWS SAMCLI? ... 980
Terraform와 함께 AWS SAM CLI를 사용하려면 어떻게 해야 합니까? ... 981
다음 단계 ... 981

다른 사용자가 사용할 수 있도록 게시 982
사전 조건 982
애플리케이션 게시() 983
1단계: AWS SAM 템플릿에 `Metadata` 섹션 추가 983
2단계: 애플리케이션 패키징 984
3단계: 애플리케이션 게시 985
4단계: 애플리케이션 공유(선택 사항) 985
기존 애플리케이션의 새 버전 게시 985
추가 주제 986
메타데이터 섹션 속성 986
속성 986
사용 사례 988
예제 989
문서 기록 991
..... mxi

AWS Serverless Application Model (AWS SAM)란 무엇입니까?

AWS Serverless Application Model (AWS SAM)는 코드형 인프라(IaC)를 사용하여 서버리스 애플리케이션을 빌드하기 위한 오픈 소스 프레임워크입니다. 개발자는 AWS SAM의 간편 구문을 사용하여 배포 중에 인프라로 변환되는 [CloudFormation](https://aws.amazon.com/cloudformation/) 리소스와 특수 서버리스 리소스를 선언합니다. 작업 시 다음과 상호 작용 AWS SAM합니다.

1. AWS SAM CLI - 서버리스 애플리케이션을 개발, 로컬 테스트 및 배포하는 데 도움이 되는 명령줄 도구입니다.
2. AWS SAM 템플릿 - 서버리스 리소스를 정의 CloudFormation 하기 위한 간소화된 구문을 제공하는의 확장입니다.

sam init 명령을 사용하면 일반적으로 AWS SAM 템플릿, 애플리케이션 코드 및 기타 구성 파일이 포함된 프로젝트라고 하는 AWS SAM 프로젝트 디렉터리가 생성됩니다.

사용 시기 AWS SAM

AWS SAM 는 완전한 기능을 갖춘 간소화된 서버리스 개발을 원하는 시나리오에 이상적인 IaC 솔루션입니다 CloudFormation. 예를 들어 SAM을 다음에 사용할 수 있습니다.

- 서버리스 애플리케이션: SAM을 사용하여 최소한의 코드로 AWS Lambda 함수, Lambda 내구성 함수, Amazon API Gateway APIs, Amazon DynamoDB 테이블 및 기타 서버리스 리소스를 빠르게 정의할 수 있습니다.
- CloudFormation 기능 향상: SAM을 기존 CloudFormation 템플릿과 결합하여 기존 인프라에 서버리스 구성 요소를 추가할 수 있습니다. SAM 리소스는 동일한 템플릿의 표준 CloudFormation 리소스와 함께 작동합니다.
- 로컬 개발 및 테스트: SAM CLI을(를) 사용하여 AWS에 배포하기 전에 개발 머신에서 Lambda 함수를 로컬로 테스트하고, API Gateway 엔드포인트를 시뮬레이션하며, 서버리스 애플리케이션을 디버그할 수 있습니다.
- 서버리스용 CI/CD: 스테이징 및 프로덕션 환경에 필요한 CloudFormation 인프라를 자동으로 생성하는 SAM 템플릿을 사용하여 배포 파이프라인을 구축할 수 있습니다.
- 콘솔 생성 리소스에서 마이그레이션: SAM 템플릿을 사용하여 AWS 관리 콘솔에서 생성된 Lambda 함수 및 API Gateway 리소스를 코드형 인프라로 변환할 수 있습니다.

AWS SAM 다른 IaC 도구와 비교

- 템플릿 호환성을 유지하면서 서버리스 리소스 정의를 단순화하려면 CloudFormation 대신 SAM을 사용하세요.
- 프로그래밍 방식 대신 인프라를 설명하는 선언적 접근 방식을 선호하는 AWS CDK 경우 대신 SAM 을 사용합니다.
- SAM CLI의 로컬 테스트 기능을 사용하여 CDK 애플리케이션을 개선 AWS CDK 하여 SAM을와 결합 합니다.

주요 기능

AWS SAM 는 다음을 수행하여 개발자 경험을 개선하는 다양한 이점을 제공합니다.

더 적은 코드를 사용하여 애플리케이션 인프라 코드 빠르게 정의하기

AWS SAM 템플릿을 작성하여 서버리스 애플리케이션 인프라 코드를 정의합니다. 템플릿을에 직접 배포 CloudFormation 하여 리소스를 프로비저닝합니다.

전체 개발 수명 주기에 걸쳐 서버리스 애플리케이션을 관리합니다.

개발 수명 주기의 작성, 구축, 배포, 테스트 및 모니터링 단계에 걸쳐 서버리스 애플리케이션을 관리하려면 AWS SAMCLI를 사용합니다. 자세한 내용은 AWS SAM CLI 단원을 참조하십시오.

AWS SAM 커넥터를 사용하여 리소스 간에 권한을 빠르게 프로비저닝

AWS SAM 템플릿의 AWS SAM 커넥터를 사용하여 AWS 리소스 간에 권한을 정의합니다.는 코드를 의도를 용이하게 하는 데 필요한 IAM 권한으로 AWS SAM 변환합니다. 자세한 내용은 AWS SAM 커넥터를 사용한 리소스 권한 관리 단원을 참조하십시오.

개발 과정에서 로컬 변경 사항을 클라우드에 지속적으로 동기화

sam sync 명령을 AWS SAMCLI 사용하여 로컬 변경 사항을 클라우드에 자동으로 동기화하여 개발 및 클라우드 테스트 워크플로의 속도를 높입니다. 자세한 내용은 sam sync를 사용하는 AWS 클라우드 동기화 소개 단원을 참조하십시오.

Terraform 서버리스 애플리케이션 관리

AWS SAMCLI를 사용하여 Lambda 함수 및 계층의 로컬 디버깅 및 테스트를 수행할 수 있습니다. 자세한 내용은 AWS SAM CLI Terraform 지원 단원을 참조하십시오.

관련 정보

- AWS SAM 작동 방식에 대한 자세한 내용은 단원을 참조하십시오AWS SAM 작동 방식.
- 사용을 시작하려면 단원을 AWS SAM참조하십시오AWS SAM 시작.
- AWS SAM 를 사용하여 서버리스 애플리케이션을 생성하는 방법에 대한 개요는 섹션을 참조하세요AWS SAM 사용 방법.

AWS SAM 작동 방식

AWS SAM 를 사용하여 서버리스 애플리케이션을 생성할 때 다음 구성 요소와 상호 작용합니다.

1. AWS SAM 템플릿 - AWS 리소스를 정의하는 중요한 파일입니다. 이 템플릿에는 AWS SAM 템플릿 사양, 즉 서버리스 애플리케이션의 함수, 이벤트, API, 구성 및 권한을 정의하는 데 사용하는 간소화된 간편 구문을 제공하는 오픈 소스 프레임워크가 포함되어 있습니다. 이 파일은 sam init 명령을 실행할 때 생성되는 애플리케이션 폴더인 AWS SAM 프로젝트에 있습니다.
2. AWS SAM CLI - AWS SAM 프로젝트 및 지원되는 타사 통합과 함께 사용하여 서버리스 애플리케이션을 구축하고 실행할 수 있는 명령줄 도구입니다. AWS SAMCLI는 AWS SAM 프로젝트에서 명령을 실행하고 결국 서버리스 애플리케이션으로 전환하는 데 사용하는 도구입니다.

서버리스 애플리케이션을 정의하는 리소스, 이벤트 소스 매핑 및 기타 속성을 표현하려면 AWS SAM 프로젝트의 AWS SAM 템플릿 및 기타 파일에서 리소스를 정의하고 애플리케이션을 개발합니다. AWS SAMCLI를 사용하여 AWS SAM 프로젝트에서 명령을 실행합니다. 이렇게 하면 서버리스 애플리케이션을 초기화, 빌드, 테스트 및 배포할 수 있습니다.

> 서버리스 개념이 생소하신가요?
>
> AWS Serverless Application Model의 서버리스 개념 섹션을 검토하는 것이 좋습니다.

AWS SAM 템플릿 사양이란 무엇입니까?

AWS SAM 템플릿 사양은 서버리스 애플리케이션 인프라 코드를 정의하고 관리하는 데 사용할 수 있는 오픈 소스 프레임워크입니다. AWS SAM 템플릿 사양은 다음과 같습니다.

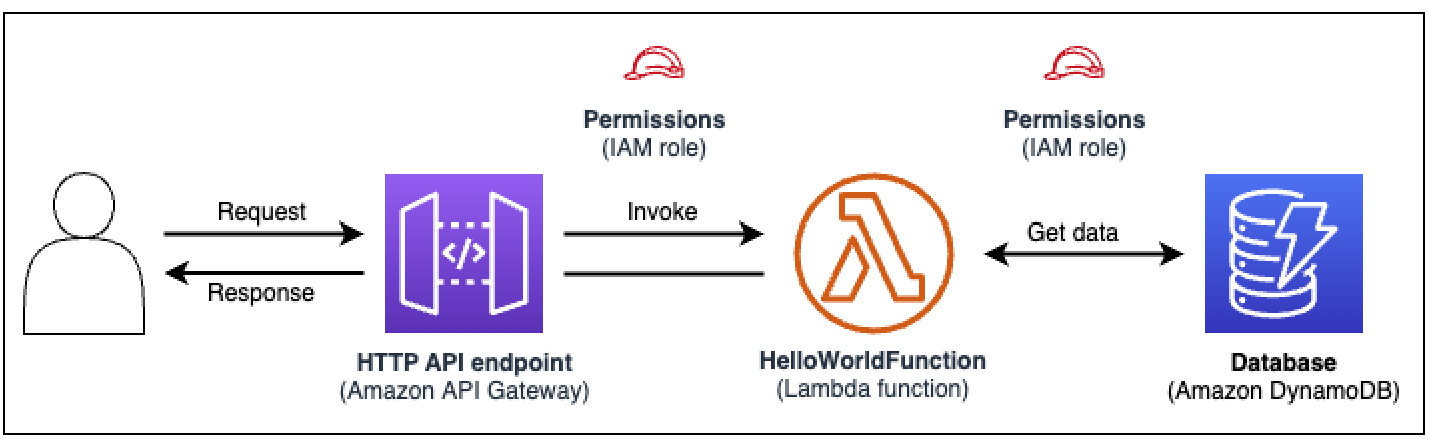
- 기본 제공 AWS CloudFormation CloudFormation - AWS SAM 템플릿에서 직접 구문을 사용하여 리소스 및 속성 구성에 대한 광범위한 지원을 활용합니다. 이미에 익숙하다면 애플리케이션 인프라 코드를 관리하기 위해 새 서비스를 배울 필요가 CloudFormation없습니다.
- 의 확장 CloudFormation-는 특히 서버리스 개발 속도 향상에 초점을 맞춘 고유한 구문을 AWS SAM 제공합니다. 동일한 템플릿 내에서 CloudFormation 및 AWS SAM 구문을 모두 사용할 수 있습니다.
- 추상적이고 간단한 구문 - AWS SAM 구문을 사용하면 코드 줄 수를 줄이고 오류 발생 가능성을 줄이면서 인프라를 빠르게 정의할 수 있습니다. 구문은 서버리스 애플리케이션 인프라를 정의하는데 있어 복잡함을 해소할 수 있도록 특별히 엄선되었습니다.
- 변환 - 템플릿을 인프라를 프로비저닝하는 데 필요한 코드로 변환하는 복잡한 작업을 AWS SAM 수행합니다 CloudFormation.

AWS SAM 프로젝트와 AWS SAM 템플릿이란 무엇입니까?

AWS SAM 프로젝트에는 AWS SAM 템플릿 사양이 포함된 AWS SAM 템플릿이 포함됩니다. 이 사양은 서버리스 애플리케이션 인프라를 정의하는 데 사용하는 오픈 소스 프레임워크이며 AWS, 몇 가지 추가 구성 요소를 사용하여 작업을 더 쉽게 수행할 수 있습니다. 이러한 의미에서 AWS SAM 템플릿은 CloudFormation 템플릿의 확장입니다.

다음은 기본 서버리스 애플리케이션의 예입니다. 이 애플리케이션은 HTTP 요청을 통해 데이터베이스의 모든 항목을 가져오는 요청을 처리합니다. 이것은 다음과 같은 요소로 구성됩니다.

1. 요청을 처리하는 로직이 포함된 함수.
2. 클라이언트(요청자)와 애플리케이션 간의 통신 역할을 하는 HTTP API.
3. 항목을 저장하는 데이터베이스.
4. 애플리케이션을 안전하게 실행할 수 있는 권한.



이 애플리케이션의 인프라 코드는 다음 AWS SAM 템플릿에서 정의할 수 있습니다.

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

23줄의 코드에는 다음과 같은 인프라가 정의되어 있습니다.

- AWS Lambda 서비스를 사용하는 함수입니다.
- Amazon API Gateway 서비스를 사용하는 HTTP API.
- Amazon DynamoDB 서비스를 사용하는 데이터베이스.
- 이러한 서비스가 서로 상호 작용하는 데 필요한 AWS Identity and Access Management (IAM) 권한입니다.

이 인프라를 프로비저닝하기 위해 템플릿이 CloudFormation로 배포됩니다. 배포 중에는 23줄의 코드를 이러한 리소스를 생성하는 데 필요한 CloudFormation 구문으로 AWS SAM 변환합니다 AWS. 변환된 CloudFormation 템플릿에는 200줄 이상의 코드가 포함되어 있습니다.

변환된 CloudFormation 템플릿

```
{
```

```
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
    "getAllItemsFunction": {
      "Type": "AWS::Lambda::Function",
      "Metadata": {
        "SamResourceId": "getAllItemsFunction"
      },
      "Properties": {
        "Code": {
          "S3Bucket": "amzn-s3-demo-source-bucket-1a4x26zbcdkqr",
          "S3Key": "what-is-app/a6f856abf1b2c4f7488c09b367540b5b"
        },
        "Handler": "src/get-all-items.getAllItemsHandler",
        "Role": {
          "Fn::GetAtt": [
            "getAllItemsFunctionRole",
            "Arn"
          ]
        },
        "Runtime": "nodejs12.x",
        "Tags": [
          {
            "Key": "lambda:createdBy",
            "Value": "SAM"
          }
        ]
      }
    },
    "getAllItemsFunctionRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "AssumeRolePolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Action": [
                "sts:AssumeRole"
              ],
              "Effect": "Allow",
              "Principal": {
                "Service": [
                  "lambda.amazonaws.com"
                ]
              }
```

```
          }
        ]
      },
      "ManagedPolicyArns": [
        "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
      ],
      "Tags": [
        {
          "Key": "lambda:createdBy",
          "Value": "SAM"
        }
      ]
    }
  },
  "getAllItemsFunctionApiPermission": {
    "Type": "AWS::Lambda::Permission",
    "Properties": {
      "Action": "lambda:InvokeFunction",
      "FunctionName": {
        "Ref": "getAllItemsFunction"
      },
      "Principal": "apigateway.amazonaws.com",
      "SourceArn": {
        "Fn::Sub": [
          "arn:${AWS::Partition}:execute-api:${AWS::Region}:${AWS::AccountId}:
${__ApiId__}/${__Stage__}/GET/",
          {
            "__ApiId__": {
              "Ref": "ServerlessHttpApi"
            },
            "__Stage__": "*"
          }
        ]
      }
    }
  },
  "ServerlessHttpApi": {
    "Type": "AWS::ApiGatewayV2::Api",
    "Properties": {
      "Body": {
        "info": {
          "version": "1.0",
          "title": {
            "Ref": "AWS::StackName"
```

```
            }
          },
          "paths": {
            "/": {
              "get": {
                "x-amazon-apigateway-integration": {
                  "httpMethod": "POST",
                  "type": "aws_proxy",
                  "uri": {
                    "Fn::Sub": "arn:${AWS::Partition}:apigateway:
${AWS::Region}:lambda:path/2015-03-31/functions/${getAllItemsFunction.Arn}/invocations"
                  },
                  "payloadFormatVersion": "2.0"
                },
                "responses": {}
              }
            }
          },
          "openapi": "3.0.1",
          "tags": [
            {
              "name": "httpapi:createdBy",
              "x-amazon-apigateway-tag-value": "SAM"
            }
          ]
        }
      }
    },
    "ServerlessHttpApiApiGatewayDefaultStage": {
      "Type": "AWS::ApiGatewayV2::Stage",
      "Properties": {
        "ApiId": {
          "Ref": "ServerlessHttpApi"
        },
        "StageName": "$default",
        "Tags": {
          "httpapi:createdBy": "SAM"
        },
        "AutoDeploy": true
      }
    },
    "SampleTable": {
      "Type": "AWS::DynamoDB::Table",
      "Metadata": {
```

```
        "SamResourceId": "SampleTable"
      },
      "Properties": {
        "AttributeDefinitions": [
          {
            "AttributeName": "id",
            "AttributeType": "S"
          }
        ],
        "KeySchema": [
          {
            "AttributeName": "id",
            "KeyType": "HASH"
          }
        ],
        "BillingMode": "PAY_PER_REQUEST"
      }
    },
    "getAllItemsFunctionMyConnPolicy": {
      "Type": "AWS::IAM::ManagedPolicy",
      "Metadata": {
        "aws:sam:connectors": {
          "getAllItemsFunctionMyConn": {
            "Source": {
              "Type": "AWS::Serverless::Function"
            },
            "Destination": {
              "Type": "AWS::Serverless::SimpleTable"
            }
          }
        }
      },
      "Properties": {
        "PolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Effect": "Allow",
              "Action": [
                "dynamodb:GetItem",
                "dynamodb:Query",
                "dynamodb:Scan",
                "dynamodb:BatchGetItem",
                "dynamodb:ConditionCheckItem",
```

```
                "dynamodb:PartiQLSelect"
              ],
              "Resource": [
                {
                  "Fn::GetAtt": [
                    "SampleTable",
                    "Arn"
                  ]
                },
                {
                  "Fn::Sub": [
                    "${DestinationArn}/index/*",
                    {
                      "DestinationArn": {
                        "Fn::GetAtt": [
                          "SampleTable",
                          "Arn"
                        ]
                      }
                    }
                  ]
                }
              ]
            }
          ]
        },
        "Roles": [
          {
            "Ref": "getAllItemsFunctionRole"
          }
        ]
      }
    }
  }
}
```

를 사용하면 23줄의 인프라 코드를 AWS SAM정의합니다.는 애플리케이션을 프로비저닝하는 데 필요한 200줄 이상의 CloudFormation 코드로 코드를 AWS SAM 변환합니다.

란 무엇입니까 AWS SAMCLI?

AWS SAMCLI는 AWS SAM 템플릿 및 지원되는 타사 통합과 함께 사용하여 서버리스 애플리케이션을 빌드하고 실행할 수 있는 명령줄 도구입니다. AWS SAMCLI를 사용하여 다음을 수행할 수 있습니다.

- 새 애플리케이션 프로젝트를 빠르게 초기화합니다.
- 배포할 애플리케이션을 빌드합니다.
- 로컬 디버깅 및 테스트를 수행합니다.
- 애플리케이션 배포
- CI/CD 배포 파이프라인을 구성합니다.
- 클라우드에서 애플리케이션을 모니터링하고 문제를 해결합니다.
- 개발 과정에서 로컬 변경 사항을 클라우드에 동기화합니다.
- 이 뿐만이 아닙니다!

AWS SAMCLI는 AWS SAM 및 CloudFormation 템플릿과 함께 사용할 때 가장 잘 활용됩니다. Terraform과 같은 타사 제품에서도 사용할 수 있습니다.

새 프로젝트 초기화

스타터 템플릿 중에서 선택하거나 사용자 지정 템플릿 위치를 선택하여 새 프로젝트를 시작할 수 있습니다.

여기서는 sam init 명령을 사용하여 새 애플리케이션 프로젝트를 초기화합니다. 먼저 시작할 Hello World 예제 프로젝트를 선택합니다. 그러면 AWS SAMCLI가 스타터 템플릿을 다운로드하고 프로젝트 폴더 디렉터리 구조가 생성합니다.

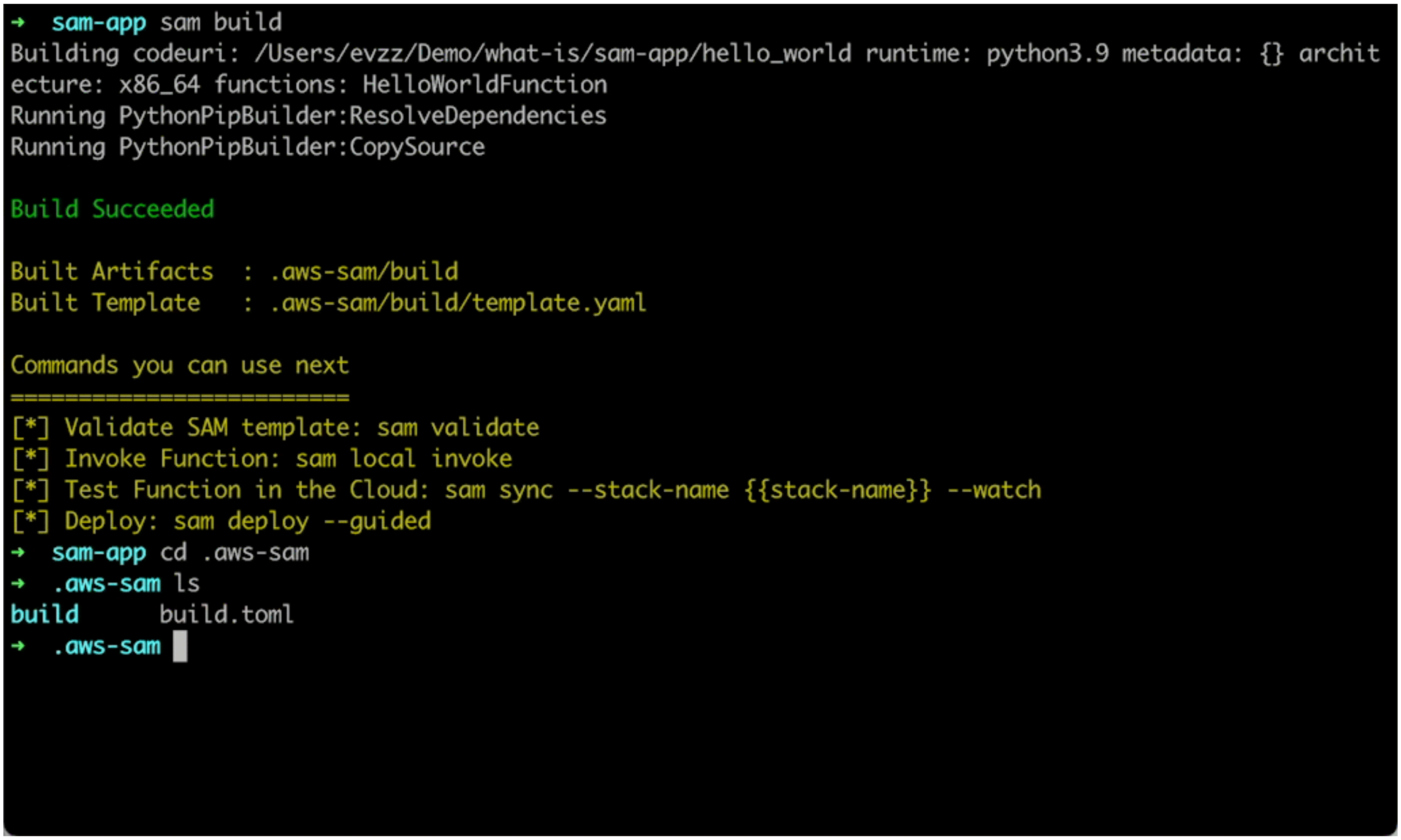
```
→  what-is sam init

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]:
```

자세한 내용은 에서 애플리케이션 생성 AWS SAM 섹션을 참조하세요.

배포를 위한 애플리케이션 빌드

함수 종속 항목을 패키징하고 프로젝트 코드 및 폴더 구조를 구성하여 배포를 준비합니다.

여기서는 sam build 명령을 사용하여 애플리케이션을 배포할 준비를 합니다. AWS SAMCLI는 `.aws-sam` 디렉터리를 만들고 배포를 위해 여기에 애플리케이션 종속성 및 파일을 구성합니다.



자세한 내용은 [애플리케이션 빌드](#) 섹션을 참조하세요.

로컬 디버깅 및 테스트 수행

로컬 컴퓨터에서 이벤트를 시뮬레이션하고, API를 테스트하고, 함수를 간접 호출하는 등 다양한 작업을 수행하여 애플리케이션을 디버깅 및 테스트할 수 있습니다.

여기서는 sam local invoke 명령을 사용하여 로컬에서 `HelloWorldFunction`를 간접 호출합니다. 이를 위해 AWS SAMCLI는 로컬 컨테이너를 만들고, 함수를 빌드하고, 간접 호출하고, 결과를 출력합니다. Docker와 같은 애플리케이션을 사용하여 시스템에서 컨테이너를 실행할 수 있습니다.

```
➜  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\"}"}
```

자세한 내용은 애플리케이션 테스트 및 애플리케이션 디버깅 섹션을 참조하세요.

애플리케이션 배포

애플리케이션의 배포 설정을 구성하고 AWS 클라우드에 배포하여 리소스를 프로비저닝합니다.

여기서는 sam deploy --guided 명령을 사용하여 대화형 흐름을 통해 애플리케이션을 배포합니다. 는 애플리케이션의 배포 설정을 구성하는 과정을 AWS SAMCLI 안내하고, 템플릿을 로 변환하고 CloudFormation,에 배포하여 CloudFormation 리소스를 생성합니다.

```
→ 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
```

자세한 내용은 애플리케이션 및 리소스 배포 섹션을 참조하세요.

CI/CD 배포 파이프라인 구성

지원되는 CI/CD 시스템을 사용하여 안전한 지속적 통합 및 제공(CI/CD) 파이프라인을 생성합니다.

여기서는 sam pipeline init --bootstrap 명령을 사용하여 애플리케이션을 위한 CI/CD 배포 파이프라인을 구성합니다. 는 옵션을 AWS SAMCLI 안내하고 CI/CD 시스템과 함께 사용할 AWS 리소스 및 구성 파일을 생성합니다.

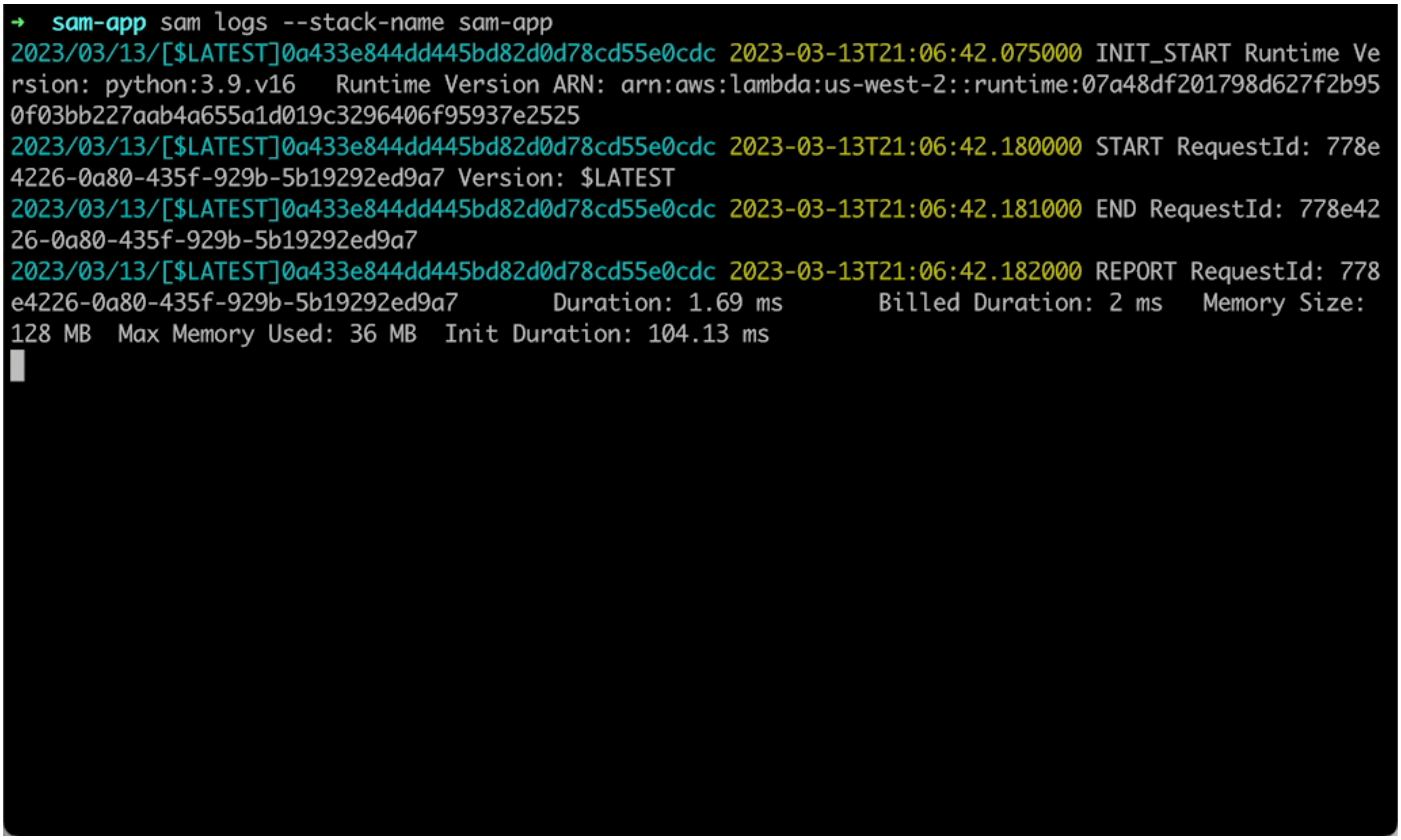
```
[3] Reference application build resources
Enter the pipeline execution role ARN if you have previously created one, or we will create one for y
ou []:
Enter the CloudFormation execution role ARN if you have previously created one, or we will create one
 for you []:
Please enter the artifact bucket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

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

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

자세한 내용은 CI/CD 시스템 및 파이프라인을 사용하여 배포 섹션을 참조하세요.

클라우드에서 애플리케이션 모니터링 및 문제 해결

배포된 리소스에 대한 중요한 정보를 보고, 로그를 수집하고, AWS X-Ray와 같은 기본 제공 모니터링 도구를 활용합니다.

여기서는 sam list 명령을 사용하여 배포된 리소스를 확인합니다. API 엔드포인트를 가져와서 간접 호출하면 함수가 트리거됩니다. 그런 다음 sam logs를 사용하여 함수의 로그를 확인합니다.



자세한 내용은 애플리케이션 모니터링 섹션을 참조하세요.

개발 과정에서 로컬 변경 사항을 클라우드에 동기화

로컬 시스템에서 개발할 때 변경 내용을 클라우드에 자동으로 동기화합니다. 변경 사항을 빠르게 확인하고 클라우드에서 테스트 및 검증을 수행할 수 있습니다.

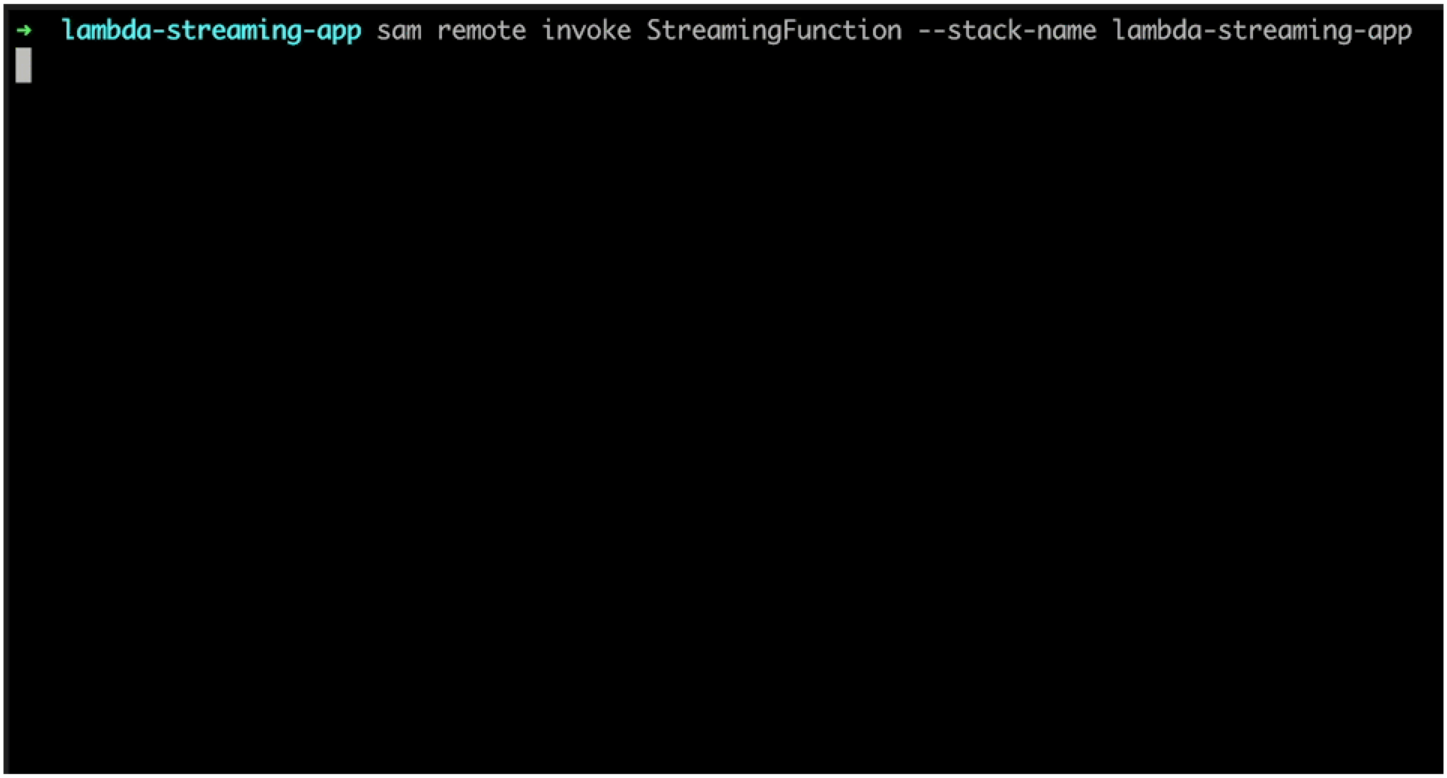
여기서는 sam sync --watch 명령을 사용하여 AWS SAMCLI가 로컬 변경 사항을 감시하도록 합니다. `HelloWorldFunction` 코드를 수정하면 AWS SAMCLI가 변경 사항을 자동으로 감지하고 업데이트를 클라우드에 배포합니다.

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

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

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

Stack update succeeded. Sync infra completed.

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

클라우드에서 지원되는 리소스 테스트

이벤트를 간접 호출하고 클라우드의 지원되는 리소스에 전달합니다.

여기서는 sam remote invoke 명령을 사용하여 클라우드에 배포된 Lambda 함수를 테스트합니다. Lambda 함수를 간접 호출하고 해당 로그와 응답을 받습니다. 응답을 스트리밍하도록 구성된 Lambda 함수를 사용하면 AWS SAMCLI가 실시간으로 응답을 다시 스트리밍합니다.



자세히 알아보기

계속 알아보려면 다음 리소스를 AWS SAM참조하세요.

- 전체 AWS SAM 워크숍 -가 AWS SAM 제공하는 많은 주요 기능을 가르치도록 설계된 워크숍입니다.
- SAM을 사용한 세션 -를 사용하는 서버 AWS 리스 개발자 지지 팀이 만든 비디오 시리즈입니다 AWS SAM.
- 서버리스 랜드 - AWS 서버리스에 대한 최신 정보, 블로그, 비디오, 코드 및 학습 리소스를 통합하는 사이트입니다.

다음 단계

를 처음 사용하는 경우 단원을 AWS SAM참조하십시오AWS SAM 시작.

AWS Serverless Application Model의 서버리스 개념

AWS Serverless Application Model(AWS SAM)을 사용하기 전에 기본적인 서버리스 개념에 대해 알아보세요.

서버리스 개념

이벤트 중심 아키텍처

서버리스 애플리케이션은 컴퓨팅을 위한 AWS Lambda, 데이터베이스 관리를 위한 Amazon DynamoDB와 같이 각각 특화된 역할을 수행하는 개별 AWS 서비스로 구성됩니다. 이러한 서비스는 이벤트 기반 아키텍처를 통해 서로 느슨하게 통합됩니다. 이벤트 기반 아키텍처에 대해 자세히 알아보려면 이벤트 기반 아키텍처란 무엇인가요?를 참조하세요.

코드형 인프라(IaC)

코드형 인프라(IaC)는 개발자가 코드를 다루는 것과 동일한 방식으로 인프라를 취급하는 방식으로, 인프라 프로비저닝에도 동일하게 애플리케이션 코드 개발의 엄격함을 적용합니다. 템플릿 파일에서 인프라를 정의하고, AWS에 배포하면 AWS가 리소스를 대신 생성합니다. IaC를 사용하면 AWS가 프로비저닝하기를 바라는 대상을 코드로 정의할 수 있습니다. AWS에 사용할 수 있는 IaC 도구 비교는코드형 인프라(IaC)을(를) 참조하세요.

서버리스 기술

AWS 서버리스 기술을 사용하면 서버를 직접 관리할 필요 없이 애플리케이션을 구축하고 실행할 수 있습니다. AWS가 수행하는 모든 서버 관리는 자동 확장 및 내장된 고가용성과 같은 많은 이점을 제공하여 아이디어를 신속하게 프로덕션에 적용할 수 있도록 지원합니다. 서버리스 기술을 사용하면 서버 관리 및 운영에 대해 걱정할 필요 없이 제품의 핵심에만 집중할 수 있습니다. 서버리스에 대한 자세한 내용은 다음 주제를 참조하세요.

- 의 서버리스AWS
- 서버리스 개발자 가이드 - AWS 클라우드에서의 서버리스 개발에 대한 개념적 개요를 제공합니다.

핵심 AWS 서버리스 서비스에 대한 기본 소개는 서버리스 랜드의 서버리스 101: 서버리스 서비스에 대한 이해 섹션을 참조하세요.

서버리스 애플리케이션

AWS SAM을 사용하면 AWS SAM 프로젝트와 템플릿으로 구성된 애플리케이션에서 관련 리소스를 관리할 수 있습니다. 애플리케이션의 모든 리소스는 AWS SAM 템플릿에서 정의되거나 참조됩니다. AWS SAM이 템플릿을 처리할 때 CloudFormation 리소스가 생성됩니다. CloudFormation에서 리소스는 스택이라는 단일 단위로 관리되며 스택의 모든 리소스는 스택의 CloudFormation 템플릿에 의해 정의됩니다.

코드형 인프라(IaC)

코드형 인프라(IaC)를 사용하면 서버리스 애플리케이션을 포함한 AWS 리소스의 배포 및 관리를 자동화할 수 있습니다. IaC를 사용하면 코드를 사용하여 인프라를 정의할 수 있으므로 배포를 버전 관리, 공유 및 복제하기가 더 쉽습니다. 이 접근 방식은 다음과 같은 이점을 제공합니다.

- 개발 주기 단축
- 구성 관리 간소화
- 배포의 안정성 및 일관성 향상

AWS 서버리스 애플리케이션을 위한 IaC 도구

AWS은(는) 클라우드 리소스를 빌드, 배포 및 관리하는 데 도움이 되는 여러 IaC 도구를 제공합니다. 이 섹션에서는 AWS SAM이(가) 이 생태계 내에서 어떻게 적용되며 다른 AWS IaC 도구와 어떻게 작동하는지 설명합니다.

AWS CloudFormation

CloudFormation을(를) 사용하면 YAML 또는 JSON 템플릿을 사용하여 전체 AWS 인프라를 모델링하고 프로비저닝할 수 있습니다. CloudFormation은(는) 리소스 생성, 업데이트 및 삭제를 자동으로 처리합니다. AWS SAM 애플리케이션을 배포할 때, CloudFormation은(는) 변환된 템플릿을 처리하여 리소스를 생성하고 관리합니다.

AWS Serverless Application Model (AWS SAM)

AWS SAM은(는) 서버리스 리소스 정의를 위한 간소화된 구문을 통해 서버리스 애플리케이션 구축을 돕습니다. AWS SAM 템플릿을 사용하여 간결한 YAML 구문으로 Lambda 함수, API, 데이터베이스 및 이벤트 소스를 프로비저닝할 수 있습니다. AWS SAM은(는) 배포 중에 이러한 템플릿을 CloudFormation 템플릿으로 변환합니다.

> **Note**
>
> AWS SAM이(가) 서버리스 애플리케이션에 특화되어 있지만, AWS SAM 템플릿에서 모든 CloudFormation 리소스 유형을 사용할 수 있습니다. 이를 통해 필요할 때 비서버리스 리소스를 유연하게 포함할 수 있습니다.

AWS Cloud Development Kit(AWS CDK)

AWS CDK을(를) 사용하면 TypeScript, Python, Java, C#/.Net, 또는 Go와 같은 친숙한 프로그래밍 언어를 사용하여 서버리스 인프라를 정의할 수 있습니다. 프로그래밍 구문(예: 루프 및 조건)을 사용하여 인프라를 정의할 수 있으며, AWS CDK은(는) 배포를 위해 CloudFormation 템플릿을 생성합니다. AWS SAM CLI을(를) 사용하여 AWS CDK로 생성된 애플리케이션을 로컬로 테스트하고 디버깅할 수 있습니다. 자세한 내용은 로컬에서 CDK 애플리케이션 테스트를 참조하세요.

서버리스 애플리케이션을 위한 IaC 도구 비교

서버리스 애플리케이션을 위한 IaC 도구를 선택할 때는 팀의 선호도, 프로젝트 요구 사항 및 기존 워크플로를 고려하세요. 다음 표는 서버리스 개발을 위한 AWS IaC 도구의 주요 특징을 비교합니다.

도구	프라이머리 사용	최적의 용도	AWS SAM와(과) 함께 작동	선택 시기
CloudFormation	복잡한 AWS 인프라 관리	AWS 리소스에 대한 상세 제어가 필요한 애플리케이션	AWS SAM 배포 중 템플릿이 CloudFormation 템플릿으로 변환됨	비서버리스 리소스에 대한 세밀한 제어의 경우
AWS SAM	서버리스 애플리케이션 개발	Lambda를 사용하여 서버리스 애플리케이션을 구축하는 팀	네이티브 기능	Lambda 함수, API Gateway API 및 기타 서버리스 리소스로 구성된 서버리스 아키텍처에 주로 초점을 맞출 때
AWS CDK	프로그래밍 언어를 사용한 인프라 정의	입력한 언어와 코드 우선 접근 방식을 선호하는 팀	AWS SAM 템플릿 생성 및 AWS SAM CLI 테스트에 사용	프로그래밍 방식의 인프라 정의 또는 복잡한 리소스 구성 로직이 필요할 때

> **Note**
>
> 이 가이드는 AWS 네이티브 IaC 도구에 중점을 두지만, Terraform은 서버리스 애플리케이션을 정의하는 데 사용할 수 있는 또 다른 인기 있는 IaC 솔루션입니다. AWS SAM CLI은(는) Terraform에 정의된 Lambda 함수의 로컬 테스트를 지원합니다. 자세한 내용은 AWS SAM CLI Terraform 지원 섹션을 참조하세요.

자세히 알아보기

- AWS의 DevOps 관행에 대해 자세히 알아보려면 AWS에 대한 DevOps 소개를 참조하세요.
- 다양한 IaC 도구와 함께 Lambda를 사용하는 방법에 대한 자세한 내용은 코드형 인프라(IaC)와 함께 Lambda 사용을 참조하세요.

AWS SAM 시작

이 섹션의 주제를 검토하고 완료하여 AWS SAM을 시작합니다. AWS SAM 사전 조건에서는 AWS 계정 설정, IAM 사용자 생성, 키 액세스 생성, AWS SAM CLI 설치 및 구성에 대한 자세한 지침을 제공합니다. 필수 조건을 완료한 후, 개발 접근 방식을 선택할 수 있습니다.

- AWS SAM CLI 설치을(를) Linux, Windows 및 macOS 운영 체제에서 수행할 수 있습니다.
- 통합 디버깅 및 테스트 기능을 갖춘 시각적 개발 환경을 위해 VS Code Toolkit을 사용하여 AWS SAM 시작하기를 사용하세요.

설치가 완료되면 선택적으로 AWS SAM Hello World 자습서를 살펴볼 수 있습니다. 이 자습서에서는 AWS SAM을 사용하여 기본 서버리스 애플리케이션을 생성하는 프로세스를 안내합니다. 자습서를 완료한 후에는 AWS Serverless Application Model(AWS SAM) 사용 방법에 자세히 설명된 개념을 검토할 준비가 됩니다.

주제

- AWS SAM 사전 조건
- AWS SAM CLI 설치
- 자습서: AWS SAM에서 Hello World 애플리케이션 배포
- Lambda 함수를 AWS SAM 애플리케이션으로 변환하기

AWS SAM 사전 조건

AWS Serverless Application Model 명령줄 인터페이스()를 설치하고 사용하기 전에 다음 사전 조건을 완료합니다AWS SAMCLI.

를 사용하려면 다음이 AWS SAMCLI필요합니다.

- AWS 계정, AWS Identity and Access Management (IAM) 자격 증명 및 IAM 액세스 키 페어.
- 자격 AWS 증명을 구성하는 AWS Command Line Interface (AWS CLI)입니다.

주제

- 1단계: AWS 계정 가입
- 2단계: IAM 사용자 계정 생성

- 3단계: 액세스 키 ID 및 비밀 액세스 키 생성
- 4단계: 설치 AWS CLI
- 5단계: AWS CLI 를 사용하여 AWS 자격 증명 구성
- 6단계: (선택 사항) VS Code용 AWS 도구 키트 설치
- 다음 단계

1단계: AWS 계정 가입

이 없는 경우 다음 단계를 AWS 계정완료하여 생성합니다.

에 가입하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 엽니다.
2. 온라인 지시 사항을 따릅니다.

   등록 절차 중 전화 또는 텍스트 메시지를 받고 전화 키패드로 확인 코드를 입력하는 과정이 있습니다.

   에 가입하면 AWS 계정AWS 계정 루트 사용자이 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스에 액세스할 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

2단계: IAM 사용자 계정 생성

다음 옵션 중 하나를 선택하여 관리 사용자를 생성합니다.

관리자를 관리하는 방법 한 가지 선택	목적	By	다른 방법
IAM Identity	단기 보안 인증 정보를 사용하여 AWS에 액세스합니다.	AWS IAM Identity Center 사용 설명서의 시작하기 지침을 따릅니다.	AWS Command Line Interface 사용 설명서에서 사용하도록 AWS CLI 를 구성 AWS IAM

관리자를 관리하는 방법 한 가지 선택	목적	By	다른 방법
Center 에서 (권장)	이는 보안 모범 사례와 일치합니다. 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.		Identity Center하여 프로그래밍 방식 액세스를 구성합니다.
IAM에서 (권장되지 않음)	장기 보안 인증 정보를 사용하여 AWS에 액세스합니다.	IAM 사용 설명서의 비상 액세스를 위한 IAM 사용자 생성에 나와 있는 지침을 따르세요.	IAM 사용 설명서에 나온 IAM 사용자의 액세스 키 관리 단계를 수행하여 프로그래밍 방식의 액세스를 구성합니다.

3단계: 액세스 키 ID 및 비밀 액세스 키 생성

CLI 액세스를 위해서는 액세스 키 ID 및 비밀 액세스 키가 필요합니다. 가능하다면 장기 액세스 키 대신 임시 보안 인증 정보를 사용하세요. 임시 보안 인증도 액세스 키 ID와 비밀 액세스 키로 구성되지만 보안 인증이 만료되는 시간을 나타내는 보안 토큰이 포함되어 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 리소스에서 임시 자격 증명 사용을 참조하세요.

사용자는 AWS 외부에서와 상호 작용하려는 경우 프로그래밍 방식으로 액세스해야 합니다 AWS Management 콘솔. 프로그래밍 방식 액세스를 부여하는 방법은에 액세스하는 사용자 유형에 따라 다릅니다 AWS.

사용자에게 프로그래밍 방식의 액세스 권한을 부여하려면 다음 옵션 중 하나를 선택합니다.

프로그래밍 방식 액세스가 필요한 사용자	목적	방법
IAM	(권장) 콘솔 자격 증명을 임시 자격 증명으로 사용하여 AWS CLI, AWS SDKs 또는 AWS APIs.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다.

프로그래밍 방식 액세스가 필요한 사용자	목적	방법
		• 자세한 AWS CLI내용은 AWS Command Line Interface 사용 설명서의 Login for AWS local development를 참조하세요. • AWS SDKs 경우 SDK 및 도구 참조 안내서의 AWS 로컬 개발을 위한 로그인을 참조하세요. AWS SDKs
작업 인력 ID (IAM Identity Center에서 관리되는 사용자)	임시 자격 증명을 사용하여 AWS CLI, AWS SDKs 또는 AWS APIs.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • 자세한 AWS CLI내용은 AWS Command Line Interface 사용 설명서의 AWS CLI 를 사용하도록 구성을 AWS IAM Identity Center 참조하세요. • AWS SDKs, 도구 및 AWS APIs의 경우 SDK 및 도구 참조 안내서의 IAM Identity Center 인증을 참조하세요. AWS SDKs
IAM	임시 자격 증명을 사용하여 AWS CLI, AWS SDKs 또는 AWS APIs.	IAM 사용 설명서의 AWS 리소스에서 임시 자격 증명 사용의 지침을 따릅니다.

프로그래밍 방식 액세스가 필요한 사용자	목적	방법
IAM	(권장되지 않음) 장기 자격 증명을 사용하여 AWS CLI, AWS SDKs 또는 AWS APIs.	사용하고자 하는 인터페이스에 대한 지침을 따릅니다. • 자세한 AWS CLI내용은 AWS Command Line Interface 사용 설명서의 IAM 사용자 자격 증명을 사용하여 인증을 참조하세요. • AWS SDKs 및 도구의 경우 SDK 및 도구 참조 안내서의 장기 자격 증명을 사용하여 인증을 참조하세요. AWS SDKs • AWS APIs 경우 IAM 사용 설명서의 IAM 사용자의 액세스 키 관리를 참조하세요.

4단계: 설치 AWS CLI

AWS CLI 는 명령줄 셸에서 명령을 AWS 서비스 사용하여와 상호 작용할 수 있는 오픈 소스 도구입니다. 에는 자격 증명 구성과 같은 활동을 AWS CLI 위해가 AWS SAMCLI 필요합니다. 에 대한 자세한 내용은 AWS Command Line Interface 사용 설명서의 란 무엇입니까 AWS Command Line Interface?를 AWS CLI참조하세요.

를 설치하려면 AWS Command Line Interface 사용 설명서의 의 최신 버전 설치 또는 업데이트를 AWS CLI AWS CLI참조하세요.

5단계: AWS CLI 를 사용하여 AWS 자격 증명 구성

IAM Identity Center로 자격 증명을 구성하려면

- IAM Identity Center로 자격 증명을 구성하려면 Configure your profile with the AWS configure sso 마법사를 참조하세요.

를 사용하여 자격 증명을 구성하려면 AWS CLI

1. 명령줄에서 `aws configure`명령을 실행합니다.
2. 다음을 구성합니다. 각 링크를 선택하여 자세히 알아보십시오.

    a. 액세스 키 ID
    b. 보안 액세스 키
    c. AWS 리전
    d. 출력 형식

    다음 예제는 샘플 값을 보여줍니다.

    ```
    $ 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
    ```

는이 정보를 `credentials` 및 `config` 파일의 프로파일(설정 모음)`default`에 AWS CLI 저장합니다. 이 파일은 `.aws` 디렉터리에 위치합니다. 기본적으로이 프로필의 정보는 사용할 프로필을 명시적으로 지정하지 않는 AWS CLI 명령을 실행할 때 사용됩니다. `credentials` 파일 사용에 대한 자세한 내용은 AWS Command Line Interface 사용자 가이드의 구성 및 보안 인증 파일 설정을 참조하세요.

기존 구성 및 보안 인증 파일 사용과 같은 보안 인증 구성에 대한 자세한 내용은 AWS Command Line Interface 사용자 가이드의 빠른 설정을 참조하세요.

6단계: (선택 사항) VS Code용 AWS 도구 키트 설치

통합 개발 환경을 선호하는 개발자를 위해 AWS Toolkit for VS Code는 시각적 디버깅, CodeLens 통합, 간소화된 배포 워크플로 등 서버리스 개발을 위한 강력한 기능을 제공합니다.

VS Code 개발을 위한 사전 조건

- 시스템에 Visual Studio Code(버전 1.73.0 이상)가 설치되어 있어야 합니다
- VS Code용 YAML 언어 지원 확장

AWS Toolkit for VS Code를 설치하려면

1. Visual Studio Code를 엽니다
2. 확장 보기(Ctrl+Shift+X 또는 Cmd+Shift+X)를 엽니다
3. "AWS Toolkit" 검색
4. Amazon Web Services에서 "AWS Toolkit" 확장 프로그램 설치
5. Red Hat의 "YAML" 확장(SAM 템플릿 CodeLens 기능에 필요)을 설치합니다

AWS SAM에서 VS Code 사용의 이점

- 시각적 디버깅: 로컬에서 Lambda 함수에 중단점을 설정하고 단계별로 실행합니다
- CodeLens 통합: SAM 템플릿에서 직접 함수를 빌드, 배포 및 간접 호출합니다
- 통합 터미널: 편집기에서 나가지 않고 AWS SAM AWS CLI 명령에 액세스
- 템플릿 유효성 검사: SAM 템플릿에 대한 실시간 유효성 검사 및 IntelliSense

VS Code에서 AWS 자격 증명을 구성하는 방법에 대한 자세한 내용은 AWS Toolkit for VS Code 사용 설명서의 자격 증명 설정을 참조하세요.

다음 단계

- 이제를 설치하고 사용을 AWS SAMCLI 시작할 준비가 되었습니다 AWS SAM. 를 설치하려면 단원을 AWS SAMCLI참조하십시오AWS SAM CLI 설치.
- 서버리스 개발을 위해 Visual Studio 코드를 설정하려면 VS Code용 AWS 도구 키트 설정을 참조하세요.

AWS SAM CLI 설치

4단계: 설치 AWS CLI에서 지침에 따라 지원되는 운영 체제에 AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI)의 최신 릴리스를 설치하세요.

자동 빌드를 업그레이드 제거 또는 관리하는 방법을 포함하여 현재 설치된 버전의 AWS SAMCLI에 대한 자세한 내용은을 참조하세요AWS SAM CLI 버전 관리.

> **AWS SAM CLI을 처음 설치하시나요?**
>
> 진행하기 전에 이전 섹션의 모든 사전 요구 사항을 완료하십시오. 여기에는 다음이 포함됩니다.
>
> 1. AWS 계정 가입
> 2. AWS에 대한 보안 액세스 설정.
> 3. 액세스 키 ID 및 보안 액세스 키 생성
> 4. AWS CLI의 설치
> 5. AWS 보안 인증의 구성

주제

- AWS SAMCLI의 설치
- 설치 오류 문제 해결
- 다음 단계
- 선택 사항: AWS SAM CLI 설치관리자 프로그램의 무결성 확인

AWS SAMCLI의 설치

> **Note**
>
> 2023년 9월부터 AWS은 AWS SAM CLI `aws/tap/aws-sam-cli`을 위한 AWS 관리형 Homebrew 설치관리자 프로그램을 더 이상 유지 관리하지 않습니다. AWS SAM CLI을 설치 및 관리하는데 Homebrew을 사용하는 경우 다음 옵션을 참조하세요.
>
> - Homebrew을 계속 사용하려면 커뮤니티 관리 설치관리자 프로그램을 사용할 수 있습니다. 자세한 내용은 AWS SAM로 CLIHomebrew 관리을 참조하십시오
> - 이 페이지에 설명된 자사 설치 방법 중 하나를 사용하는 것이 좋습니다. 이러한 방법 중 하나를 사용하기 전에 Homebrew에서 전환 을 참조하세요.
> - 자세한 내용은 릴리스 버전: 1.121.0을 참조하세요.

AWS SAM CLI을 설치하려면 귀하의 운영 체제에 대한 지침을 따르세요.

Linux

x86_64 - command line installer

1. 선택한 디렉터리에 AWS SAM CLI .zip file을 복사합니다.
2. (선택 사항) 설치 전에 설치관리자 프로그램의 무결성을 확인할 수 있습니다. 지침은 선택 사항: AWS SAM CLI 설치관리자 프로그램의 무결성 확인 섹션을 참조하세요.
3. 선택한 디렉터리에 설치 파일의 압축을 풉니다. 다음은 `sam-installation` 하위 디렉토리의 사용을 보여주는 예입니다.

> **Note**
>
> 운영 체제에 기본 제공 unzip 명령이 없는 경우 이와 동등한 명령을 사용하세요.

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

4. AWS SAM CLI를 실행하여 실행 `install` 파일을 설치합니다. 이 실행 파일은 이전 단계에서 사용된 디렉터리에 위치합니다. 다음은 `sam-installation` 하위 디렉토리의 사용을 보여주는 예입니다.

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

5. 설치를 확인합니다.

```
$ sam --version
```

설치에 성공했는지 확인하려면 출력에서 다음 괄호 안의 텍스트가 사용 가능한 최신 버전으로 대체되었는지 확인합니다.

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

arm64 - command line installer

1. 선택한 디렉터리에 AWS SAM CLI .zip file을 복사합니다.
2. (선택 사항) 설치 전에 설치관리자 프로그램의 무결성을 확인할 수 있습니다. 지침은 선택 사항: AWS SAM CLI 설치관리자 프로그램의 무결성 확인 섹션을 참조하세요.

3. 선택한 디렉터리에 설치 파일의 압축을 풉니다. 다음은 `sam-installation` 하위 디렉토리의 사용을 보여주는 예입니다.

   > **Note**
   >
   > 운영 체제에 기본 제공 unzip 명령이 없는 경우 이와 동등한 명령을 사용하세요.

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

4. AWS SAM CLI를 실행하여 실행 `install` 파일을 설치합니다. 이 실행 파일은 이전 단계에서 사용된 디렉터리에 위치합니다. 다음은 `sam-installation` 하위 디렉토리의 사용을 보여주는 예입니다.

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

5. 설치를 확인합니다.

   ```
   $ sam --version
   ```

   설치에 성공했는지 확인하려면 다음과 같은 출력에서 괄호 안의 텍스트가 최신 SAM CLI 버전으로 대체되었는지 확인합니다.

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

macOS

설치 단계

패키지 설치관리자 프로그램을 사용하여 AWS SAM CLI를 설치합니다. 또한 패키지 설치관리자 프로그램에서는 GUI와 명령줄의 두 가지 설치 방법 중에서 선택할 수 있습니다. 모든 사용자를 위해 설치하거나 현재 사용자만을 위해 설치할 수 있습니다. 모든 사용자를 대상으로 설치하려면 수퍼유저 인증이 필요합니다.

> **Note**
>
> AWS SAM CLI은(는) MacOS 13.x보다 오래된 MacOS 버전을 지원하지 않습니다. 지원되는 버전의 MacOS로 업그레이드하거나 Homebrew을(를) 사용하여 AWS SAM CLI를 설치하세요.

GUI - All users

패키지 설치관리자 프로그램을 다운로드하여 AWS SAM CLI를 설치하려면

> **Note**
>
> 이전에 Homebrew 혹은 pip를 통해 AWS SAM CLI를 설치한 경우, 이를 먼저 제거해야 합니다. 지침은 AWS SAM CLI 제거 섹션을 참조하세요.

1. 선택한 디렉터리에 macOS `pkg`를 다운로드합니다.
   - Intel 프로세서를 실행하는 Mac의 경우 x86_64 선택 – aws-sam-cli-macos-x86_64.pkg
   - Apple Silicon을 실행하는 Mac의 경우 arm64 선택 – aws-sam-cli-macos-arm64.pkg

   > **Note**
   >
   > 설치 전에 설치관리자 프로그램의 무결성을 확인하는 옵션이 있습니다. 지침은 선택 사항: AWS SAM CLI 설치관리자 프로그램의 무결성 확인 섹션을 참조하세요.

2. 다운로드한 파일을 실행하고 화면에 나타난 지침에 따라 소개, Read Me 및 라이선스 단계를 계속 진행합니다.
3. 대상 선택에서 이 컴퓨터의 모든 사용자를 위한 설치를 선택합니다.
4. 설치 유형에서 AWS SAM CLI을 설치할 위치를 선택하고 설치를 누릅니다. 권장되는 기본 위치는 `/usr/local/aws-sam-cli`입니다.

   > **Note**
   >
   > sam 명령으로 AWS SAMCLI를 간접 호출하기 위해 설치관리자 프로그램은 `/usr/local/bin/sam`와 `/usr/local/aws-sam-cli/sam` 또는 귀하가 선택한 설치 폴더 중 하나 간에 심링크를 자동으로 생성합니다.

5. AWS SAM CLI가 설치하면 설치 성공 메시지가 표시됩니다. 닫기를 누릅니다.

성공적으로 설치되었는지 확인하려면

- 다음을 실행하여 AWS SAM CLI가 제대로 설치되었고 심링크가 구성되어 있는지 확인합니다.

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

GUI - Current user

AWS SAM CLI를 다운로드하여 설치하려면

> **Note**
>
> 이전에 Homebrew 혹은 pip를 통해 AWS SAM CLI를 설치한 경우, 이를 먼저 제거해야 합니다. 지침은 AWS SAM CLI 제거 섹션을 참조하세요.

1. 선택한 디렉터리에 macOS pkg를 다운로드합니다.

   - Intel 프로세서를 실행하는 Mac의 경우 x86_64 선택 – aws-sam-cli-macos-x86_64.pkg
   - Apple Silicon을 실행하는 Mac의 경우 arm64 선택 – aws-sam-cli-macos-arm64.pkg

   > **Note**
   >
   > 설치 전에 설치관리자 프로그램의 무결성을 확인하는 옵션이 있습니다. 지침은 선택 사항: AWS SAM CLI 설치관리자 프로그램의 무결성 확인 섹션을 참조하세요.

2. 다운로드한 파일을 실행하고 화면에 나타난 지침에 따라 소개, Read Me 및 라이선스 단계를 계속 진행합니다.
3. 대상 선택에서 나만을 위한 설치를 선택합니다. 이 옵션이 표시되지 않으면 다음 단계를 진행합니다.
4. 설치 유형에서 다음을 수행합니다.

   1. AWS SAM CLI를 설치할 위치를 선택합니다. 기본 위치는 `/usr/local/aws-sam-cli`입니다. 쓰기 권한이 있는 위치를 선택합니다. 설치 위치를 변경하려면 로컬을 선택하고 위치를 선택합니다. 완료되면 계속을 누르십시오.
   2. 이전 단계에서 나를 위한 설치를 선택하는 옵션이 표시되지 않은 경우 설치 위치 변경 > 내게만 설치를 선택하고 계속을 누르십시오.
   3. 설치를 누릅니다.
5. AWS SAM CLI가 설치하면 설치 성공 메시지가 표시됩니다. 닫기를 누릅니다.

심링크 생성

- sam 명령으로 AWS SAMCLI를 간접 호출하려면 AWS SAM CLI프로그램과 귀하의 $PATH 사이에 심링크를 수동으로 만들어야 합니다. 다음 명령을 수정하고 실행하여 심링크를 생성합니다.

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

  - *sudo* – 사용자에게 $PATH에 대한 쓰기 권한이 있는 경우에는 sudo은 필요하지 않습니다. 그렇지 않으면 sudo이 필요합니다.
  - *path-to* - AWS SAM CLI 프로그램을 설치한 위치의 경로. 예를 들어 `/Users/myUser/Desktop`입니다.
  - *path-to-symlink-directory* – 귀하의 `$PATH` 환경 변수입니다. 기본 위치는 `/usr/local/bin`입니다.

성공적으로 설치되었는지 확인하려면

- 다음을 실행하여 AWS SAM CLI가 제대로 설치되었고 심링크가 구성되어 있는지 확인합니다.

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

Command line - All users

AWS SAM CLI를 다운로드하여 설치하려면

> **Note**
>
> 이전에 Homebrew 혹은 pip를 통해 AWS SAM CLI를 설치한 경우, 이를 먼저 제거해야 합니다. 지침은 AWS SAM CLI 제거 섹션을 참조하세요.

1. 선택한 디렉터리에 macOS pkg를 다운로드합니다.
   - Intel 프로세서를 실행하는 Mac의 경우 x86_64 선택 – aws-sam-cli-macos-x86_64.pkg
   - Apple Silicon을 실행하는 Mac의 경우 arm64 선택 – aws-sam-cli-macos-arm64.pkg

   > **Note**
   >
   > 설치 전에 설치관리자 프로그램의 무결성을 확인하는 옵션이 있습니다. 지침은 선택 사항: AWS SAM CLI 설치관리자 프로그램의 무결성 확인 섹션을 참조하세요.

2. 설치 스크립트를 변경하고 실행합니다.

   ```
   $ 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**
   >
   > `/usr/local/bin/sam` `/usr/local/aws-sam-cli/sam` sam명령으로 AWS SAMCLI를 간접 호출하려면, 설치관리자 프로그램은 와 간에 심링크를 자동으로 생성합니다.

성공적으로 설치되었는지 확인하려면

- 다음을 실행하여 AWS SAM CLI가 제대로 설치되었고 심링크가 구성되어 있는지 확인합니다.

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

Command line - Current user

AWS SAM CLI를 다운로드하여 설치하려면

> **Note**
>
> 이전에 Homebrew 혹은 pip를 통해 AWS SAM CLI를 설치한 경우, 이를 먼저 제거해야 합니다. 지침은 AWS SAM CLI 제거 섹션을 참조하세요.

1. 선택한 디렉터리에 macOS pkg를 다운로드합니다.

   - Intel 프로세서를 실행하는 Mac의 경우 x86_64 선택 – aws-sam-cli-macos-x86_64.pkg
   - Apple Silicon을 실행하는 Mac의 경우 arm64 선택 – aws-sam-cli-macos-arm64.pkg

   > **Note**
   >
   > 설치 전에 설치관리자 프로그램의 무결성을 확인하는 옵션이 있습니다. 지침은 선택 사항: AWS SAM CLI 설치관리자 프로그램의 무결성 확인 섹션을 참조하세요.

2. 귀하에게 쓰기 권한이 있는 설치 디렉터리를 결정하십시오. 그런 다음 템플릿을 사용하여 xml 파일을 만들고 설치 디렉토리를 반영하도록 수정하십시오. 디렉토리는 이미 존재해야 합니다.

   예를 들어, *path-to-my-directory*를 /Users/myUser/Desktop로 바꾸면, aws-sam-cli 프로그램 폴더가 거기에 설치됩니다.

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

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

3. xml파일을 저장하고 다음을 실행하여 파일이 유효한지 확인합니다.

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

   출력 결과는 AWS SAM CLI 프로그램에 적용할 기본 설정이 표시되어야 합니다.

4. AWS SAM CLI을 설치하려면 다음 명령을 실행합니다.

```
$ 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.
```

심링크 생성

- sam 명령으로 AWS SAMCLI를 간접 호출하려면 AWS SAM CLI프로그램과 귀하의 $PATH 사이에 심링크를 수동으로 만들어야 합니다. 다음 명령을 수정하고 실행하여 심링크를 생성합니다.

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

  - *sudo* – 사용자에게 $PATH에 대한 쓰기 권한이 있는 경우에는 sudo은 필요하지 않습니다. 그렇지 않으면 sudo이 필요합니다.

- *path-to* - AWS SAM CLI 프로그램을 설치한 위치의 경로. 예를 들어 `/Users/myUser/Desktop`입니다.
- *path-to-symlink-directory* – 귀하의 `$PATH` 환경 변수입니다. 기본 위치는 `/usr/local/bin`입니다.

성공적으로 설치되었는지 확인하려면

- 다음을 실행하여 AWS SAM CLI가 제대로 설치되었고 심링크가 구성되어 있는지 확인합니다.

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

Windows

Windows 설치관리자 프로그램 (MSI) 파일은 Windows 운영 체제의 패키지 설치 프로그램 파일입니다.

MSI 파일을 사용하여AWS SAM CLI를 설치하려면 다음 단계를 따르십시오.

1. AWS SAM CLI 64비트를 다운로드하십시오.
2. (선택 사항) 설치 전에 설치관리자 프로그램의 무결성을 확인할 수 있습니다. 지침은 선택 사항: AWS SAM CLI 설치관리자 프로그램의 무결성 확인 섹션을 참조하세요.
3. 설치를 확인합니다.

   설치를 완료한 후 새 명령 프롬프트 또는 PowerShell 프롬프트를 열어 확인합니다. `sam`을 명령줄에서 간접 호출할 수 있어야 합니다.

   ```
   sam --version
   ```

   AWS SAM CLI를 성공적으로 설치하면 다음과 같은 출력 결과가 표시됩니다.

   ```
   SAM CLI, <latest version>
   ```
4. 긴 경로를 활성화합니다(Windows 10 이상만 해당).

> **Important**
>
> AWS SAM CLI는 Windows 최대 경로 제한을 초과하는 파일 경로와 상호 작용할 수 있습니다. 이로 인해 `sam init`를 실행할 때 Windows 10 MAX_PATH 제한을 초과하여 오류가 발생할 수 있습니다. 이 문제를 해결하기 위해서는 새로운 긴 경로 동작을 구성해야 합니다.

긴 경로를 활성화하려면 Microsoft Windows 앱 개발 설명서의 Windows 10, 버전 1607 이상에서 긴 경로 사용 활성화를 참조하세요.

5. Git을 설치합니다.

   `sam init` 명령을 사용하여 샘플 애플리케이션을 다운로드하려면 Git도 설치해야 합니다. 자세한 내용은 Git 설치를 참조하세요

설치 오류 문제 해결

Linux

도커 오류: “도커 데몬에 연결할 수 없습니다. docker 데몬이 이 호스트에서 실행되고 있습니까?”

경우에 따라서는 `ec2-user`가 Docker 대몬에 액세스할 수 있는 권한을 제공하기 위해 인스턴스를 재부팅해야 할 수도 있습니다. 이 오류가 발생하면 인스턴스를 재부팅합니다.

셸 오류: ‘명령을 찾을 수 없음’

이 오류가 발생하면 셸이 경로에서 AWS SAM CLI 실행 파일을 찾을 수 없는 것입니다. AWS SAM CLI 실행 파일을 설치한 디렉터리의 위치를 확인한 다음 디렉터리가 경로에 있는지 확인합니다.

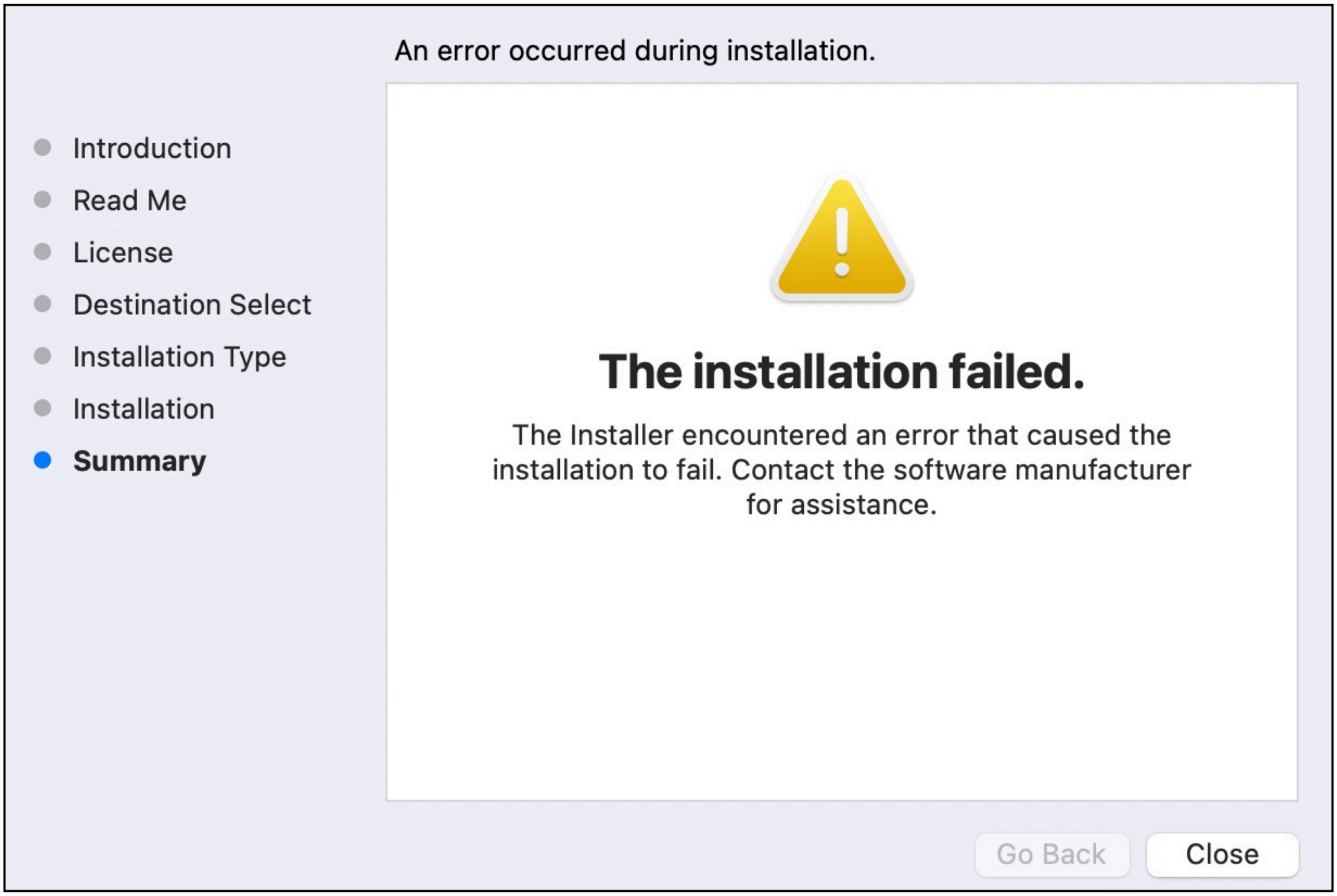
AWS SAM CLI error: “/lib64/libc.so.6: version ‘GLIBC_2.14’ not found (required by /usr/local/aws-sam-cli/dist/libz.so.1)”

이 오류가 표시되면 지원되지 않는 Linux 버전을 사용하고 있고 기본 제공 glibc 버전이 오래된 것입니다. 다음 중 하나를 사용하세요.

- Linux 호스트를 CentOS, Fedora, Ubuntu 또는 Amazon Linux 2 최신 배포판의 64비트 버전으로 업그레이드합니다.
- AWS SAM CLI 설치에 대해서는 이 지침을 따릅니다.

macOS

설치 실패



사용자용 AWS SAM CLI를 설치하고 쓰기 권한이 없는 설치 디렉터리를 선택한 경우 이 오류가 발생할 수 있습니다. 다음 중 하나를 사용하세요.

1. 쓰기 권한이 있는 다른 설치 디렉터리를 선택합니다.
2. 설치 프로그램을 삭제합니다. 그런 다음 다시 다운로드하고 실행합니다.

다음 단계

AWS SAM CLI에 대한 자세한 내용과 자체 서버리스 애플리케이션 구축을 시작하려면 다음을 참조하세요.

- 자습서: AWS SAM에서 Hello World 애플리케이션 배포— 기본 서버리스 애플리케이션을 다운로드, 빌드 및 배포하기 위한 단계별 지침

- 전체 AWS SAM 워크숍 – AWS SAM가 제공하는 여러 주요 기능을 설명하기 위해 마련된 워크숍입니다.
- AWS SAM예제 애플리케이션 및 패턴 – 추가로 실험해 볼 수 있는 커뮤니티 작성자의 샘플 애플리케이션 및 패턴

선택 사항: AWS SAM CLI 설치관리자 프로그램의 무결성 확인

패키지 설치관리자 프로그램을 사용하여 AWS Serverless Application Model 명령줄 인터페이스(AWS SAMCLI)를 설치하는 경우 설치 전에 무결성을 확인할 수 있습니다. (이 단계는 선택 사항이며, 권장 사항은 아닙니다.)

사용할 수 있는 두 가지 확인 옵션은 다음과 같습니다.

- 패키지 설치관리자 프로그램 서명을 확인합니다.
- 패키지 설치관리자 프로그램 해시 값을 확인합니다.

플랫폼에서 사용할 수 있는 경우 서명 파일 옵션을 확인하는 것이 좋습니다. 이 옵션은 키 값이 여기에 게시되고 GitHub 리포지토리와 별도로 관리되므로 추가 보안 레이어를 제공합니다.

주제

- 설치관리자 서명 파일을 확인합니다.
- 해시 값을 확인합니다.

설치관리자 서명 파일을 확인합니다.

Linux

arm64 - 명령줄 설치관리자

AWS SAM은 GnuPG를 사용하여AWS SAM CLI .zip 설치 프로그램에 서명합니다. 확인은 다음 단계들을 수행합니다.

1. 기본 공개 키를 사용하여 서명자 공개 키를 확인합니다.
2. 서명자 공개 키를 사용하여 AWS SAM CLI 패키지 설치관리자 프로그램을 확인합니다.

서명자 공개 키의 무결성 확인

1. 기본 공개 키를 복사하여 귀하의 로컬 기기에 `.txt` 파일로 저장합니다. 예:*primary-public-key.txt*


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

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


2. 주요 퍼블릭 키를 귀하의 키링으로 가져옵니다.

```
$ 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. 서명자 공개 키를 복사하여 로컬 시스템에 `.txt` 파일로 저장합니다. 예:*signer-public-key.txt*

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

mQINBGgrxIgBEADGCTudveeeVbWpZDGX9Ni57mBRMVSJwQJ6F/PC34jw0DozxTtd
H+ZPsXLvLwerN/DVXbK8E1qNZ5RGptak8j7MPz+MC3n4txibEJpB61vpjJJM+9cC
7whaMLDT/SbykHYXdrnHqa8KsUJl7rPLJcaRN722NSxvYVMIOA9ffVXV7cfEyZi5
MbYF2Gc9LNbKaknImIva7EKeeh2/wI6YCqC5yytyfWU5dL6oHXsgTnFL9mhziMxv
WhyzawyJG6EJZsJ3WLlbIKApN6XZSXyCxOvlBrebYZjD5v0nA+TJaQ7is8atjtOI
DGe0AViw7kO8ChTpjA7YG/Uu7n/Fy7qLF/3Nz0b6cBNjemjBazQ3A3KNCpi5hqFM
Uo1WpoVLr5CXQnc0B3fBUnTIoxi0Sk5MKjH9AbYxfgqEX0ZJB9hAlc6LIEy0Yru6
MMBrIHE86IMl1NfE/DeLnCdPG23+1PttwyOt3+9z5QwmPe3VPpEfCySPcdxHKZSP
rLile8qDznEvlPDvQ0qkBxdMtVa2yct5VJkdqy6UrN2xa0dpspHjRUjHh/EY/xMt
fwMUjOKohaZ/1pjotCcksAsZWUxCNcFvLYxuxeytVk4F09Es1hj4ihhLUI+43/ic
3DHSEiext7Q8/UccNArkhSCT7UOvvL7QTuP+pjYTyiC8Vx6g/Y5Ht5+qywARAQAB
tDBBV1MgU0FNIENMSSBUZWFtIDxhd3Mtc2FtLWNsaS1zaWduZXJAYW1hem9uLmNv
bT6JAj8EEwEJACkFAmgrxIgCGy8FCQPCZwAHCwkIBwMCAQYVCAIJCgsEFgIDAQIe
AQIXgAAKCRBAlKuxvt/atJo6EAC/5C8uJs76W5f5V5XNAMzwBFiZuYpop3DRReCo
P68ZZylokAC9ShRZnIOujpDJtlNS7T/G00BzmcpspkYYE531ALaXcHWmb9XV0Ajg
J8iboAVBLY0C7mhL/cbJ3v9QlpXXjyTuhexkJCV8rdHVX/0H8WqTZplEaRuZ7p8q
PMxddg4ClwstYuH3O/dmNdlGqfb4Fqy8MnV1yGSXRs5Jf+sDlN2UO4mbpyk/mr1c
f/jFxmx86IkCWJVvdXWCVTe2AFy3NHCdLtdnEvFhokCOQd9wibUWX0j9vq4cVRZT
qamnpAQaOlH3lXOwrjqo8b1AIPoRWSfMtCYvh6kA8MAJv4cAznzXILSLtOE0mzaU
qp5qoy37wNIjeztX6c/q4wss05qTlJhnNu4s3nh5VHultooaYpmDxp+ala5TWeuM
KZDI4KdAGF4z0Raif+N53ndOYIiXkY0goUbsPCnVrCwoK9PjjyoJncq7c14wNl5O
IQUZEjyYAQDGZqs5XSfY4zW2cCXatrfozKF7R1kSU14DfJwPUyksoNAQEQezfXyq
kr0gfIWK1r2nMdqS7WgSx/ypS5kdyrHuPZdaYfEVtuezpoT2lQQxOSZqqlp5hI4R
nqmPte53WXJhbC0tgTIJWn+Uy/d5Q/aSIfD6o8gNLS1BDs1j1ku0XKu1sFCHUcZG
aerdsIkCHAQQAQkABgUCaCvFeAAKCRBC/V96c62IWt3/D/9gOLzWtz62lqJRCsri
wcA/yz88ayKb/GUv3FCT5Nd9JZt8y1tW+AE3SPTdcpfZmt5UN2sRzljO61mpKJzp
eBvYQ9og/34ZrRQqeg8bz02u34LKYl1gD0xY0bWtB7TGIxIZZYqZECoPR0Dp6ZzB
```

```
abzkRSsJkEk0vbZzJhfWFYs98qfp/G0suFSBE7908Am33DB2jQ/Sollh1VmNE6Sv
EOgR6+2yEkS2D0+msJMa/V82v9gBTPnxSlNV1d8Dduvt9rbM3LoxiNXUgx/s52yY
U6H3bwUcQ3UY6uRe1UWo5QnMFcDwfg43+q5rmjB4xQyX/BaQyF5K0hZyG+42/pH1
EMwl8qN617FTxo3hvQUi/cBahlhQ8EVYsGnHDVxLCisbq5iZvp7+XtmMy1Q417gT
EQRo8feJh31elGWlccVR2pZgIm1PQ69dzzseHnnKkGhifik0bDGo5/IH2EgI1KFn
SG399RMU/qRzOPLVP3i+zSJmhMqG8cnZaUwE5V4P21vQSclhhd2Hv/C4SVKNqA2i
+oZbHj2vAkuzTTL075AoANebEjPGqwsKZi5mWUE5Pa931JeiXxWZlEB7rkgQ1PAB
fsDBhYLt4MxCWAhifLMA6uQ4BhXu2RuXOqNfSbqa8jVF6DB6cD8eAHGpPKfJ0l30
LtZnq+n4SfeNbZjD2FQWZR4CrA==
=lHfs
-----END PGP PUBLIC KEY BLOCK-----
```


4. 서명자 퍼블릭 키를 귀하의 키링으로 가져옵니다.

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

gpg: key 4094ABB1BEDFDAB4: public key "AWS SAM CLI Team <aws-sam-cli-
signer@amazon.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
gpg: no ultimately trusted keys found
```

   출력 결과의 키 값을 기록해 두십시오. 예:*4094ABB1BEDFDAB4*

5. 키 값을 사용하여 서명자 공개 키 핑거프린트를 획득하고 확인합니다.

```
$ gpg --fingerprint 4094ABB1BEDFDAB4

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

   지문은 다음과 일치해야 합니다.

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

   지문 스트링이 일치하지 않으면 AWS SAM CLI 설치관리자를 사용하지 마십시오. aws-sam-cli GitHub 리포지토리에 이슈를 생성하여 AWS SAM팀으로 에스컬레이션하십시오.

6. 서명자 공개 키의 서명을 확인하십시오.

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

```
pub   rsa4096 2025-05-19 [SCEA] [expires: 2027-05-19]
      EF463E86CA31933BB688CC1A4094ABB1BEDFDAB4
uid           [ unknown] AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>
sig!3        4094ABB1BEDFDAB4 2025-05-19  [self-signature]
sig!         42FD5F7A73AD885A 2025-05-19  AWS SAM CLI Primary <aws-sam-cli-
primary@amazon.com>
```

`1 signature not checked due to a missing key`이 확인되면 이전 단계를 반복하여 기본 및 서명자 공개 키를 키링으로 가져오세요.

기본 공개 키와 서명자 공개 키 모두의 키 값이 표시되어 있어야 합니다.

서명자 공개 키의 무결성을 확인했으므로 이제 서명자 공개 키를 사용하여 AWS SAM CLI패키지 설치 관리자 프로그램을 확인할 수 있습니다.

AWS SAMCLI 패키지 설치관리자 프로그램의 무결성을 확인하려면

1. AWS SAMCLI패키지 서명 파일 가져오기 - 다음 명령을 사용하여 AWS SAM CLI 패키지 설치관리자 프로그램의 서명 파일을 다운로드합니다.

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

2. 서명 파일을 확인 - 다운로드된 `.sig` 및 `.zip` 파일 모두를 gpg 명령의 파라미터로 전달합니다. 다음은 예제입니다.

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

다음과 같이 출력됩니다

```
gpg: Signature made Mon 19 May 2025 01:21:57 AM UTC using RSA key ID
 4094ABB1BEDFDAB4
gpg: Good signature from "AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: EF46 3E86 CA31 933B B688  CC1A 4094 ABB1 BEDF DAB4
```

- `WARNING: This key is not certified with a trusted signature!` 메시지는 무시해도 됩니다. 이것은 귀하의 개인 PGP 키(보유하신 경우)와 AWS SAM CLI PGP 키 사이에 신뢰 체인이 없기 때문에 발생한 것입니다. 자세한 내용은 [Web of trust](#)를 참조하세요.
- 출력 결과에 `BAD signature` 문구가 포함된 경우 귀하가 절차를 올바르게 수행했는지 확인합니다. 이 응답을 계속 받으면 aws-sam-cli GitHub 리포지토리에 [이슈를 제기하여](#) AWS SAM팀에 올리고 다운로드한 파일은 사용하지 마십시오.

`Good signature from "AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>"` 메시지는 서명이 확인되었으며 귀하는 설치를 진행할 수 있다는 의미입니다.

x86_64 - 명령줄 설치관리자

AWS SAM은 [GnuPG](#)를 사용하여AWS SAM CLI .zip 설치 프로그램에 서명합니다. 확인은 다음 단계들을 수행합니다.

1. 기본 공개 키를 사용하여 서명자 공개 키를 확인합니다.
2. 서명자 공개 키를 사용하여 AWS SAM CLI 패키지 설치관리자 프로그램을 확인합니다.

서명자 공개 키의 무결성 확인

1. 기본 공개 키를 복사하여 귀하의 로컬 기기에 `.txt` 파일로 저장합니다. 예:*`primary-public-key.txt`*

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

mQINBGRuSzMBEADsqiwOy78w7F4+sshaMFRIwRGNRm94p5Qey2KMZBxekFtoryVD
D9jEOnvupx4tvhfBHz5EcUHCEOdl4MTqdBy6vVAshozgxVb9RE8JpECn5lw7XC69
4Y7Gy1TKKQMEWtDXElkGxIFdUWvWjSnPlzfnoXwQYGeE93CUS3h5dImP22Yk1Ct6
eGGhlcbg1X4L8EpFMj7GvcsU8f7ziVI/PyC1Xwy39Q8/I67ip5eU5ddxO/xHqrbL
YC7+8pJPbRMej2twT2LrcpWWYAbprMtRoa6WfE0/thoo3xhHpIMHdPfAA86ZNGIN
kRLjGUg7jnPTRW4Oin3pCc8nT4Tfc1QERkHm641gTC/jUvpmQsM6h/FUVP2i5iE/
JHpJcMuL2Mg6zDo3x+3gTCf+Wqz3rZzxB+wQT3yryZs6efcQy7nROiRxYBxCSXX0
2cNYzsYLb/bYaW8yqWIHD5IqKhw269gp2E5Khs60zgS3CorMb5/xHgXjUCVgcu8a
a8ncdf9fjl3WS5p0ohetPbO2ZjWv+MaqrZOmUIgKbA4RpWZ/fU97P5BW9ylwmIDB
sWy0cMxg8MlvSdLytPieogaM0qMg3u5qXRGBr6Wmevkty0qgnmpGGc5zPiUbtOE8
CnFFqyxBpj5IOnG0KZGVihvn+iRxrv6GO7WWO92+Dc6m94U0EEiBR7QiOwARAQAB
tDRBV1MgU0FNIENMSSBQcmltYXJ5IDxhd3Mtc2FtLWNsaS1wcmltYXJ5QGFtYXpv
```

```
bi5jb20+iQI/BBMBCQApBQJkbkszAhsvBQkHhM4ABwsJCAcDAgEGFQgCCQoLBBYC
AwECHgECF4AACgkQQv1fenOtiFqTuhAAzi5+ju5UVOWqHKevOJSO08T4QB8HcqAE
SVO3mY6/j29knkcL8ubZP/DbpV7QpHPI2PB5qSXsiDTP3IYPbeY78zHSDjljaIK3
njJLMScFeGPyfPpwMsuY4nzrRIgAtXShPA8N/k4ZJcafnpNqKj7QnPxiC1KaIQWm
pOtvb8msUF3/s0UTa5Ys/lNRhVC0eGg32ogXGdojZA2kHZWdm9udLo4CDrDcrQT7
NtDcJASapXSQL63XfAS3snEc4e1941YxcjfYZ33rel8K9juyDZfi1slWR/L3AviI
QFIaqSHzyOtP1oinUkoVwL8ThevKD3Ag9CZflZLzNCV7yqlF8RlhEZ4zcE/3s9El
WzCFsozb5HfE1AZonmrDh3SyOEIBMcS6vG5dWnvJrAuSYv2rX38++K5Pr/MIAfOX
DOI1rtA+XDsHNv9lSwSy0lt+iClawZANO9IXCiN1rOYcVQlwzDFwCNWDgkwdOqS0
gOA2f8NF9lE5nBbeEuYquoOl1Vy8+ICbgOFs9LoWZlnVh7/RyY6ssowiU9vGUnHI
L8f9jqRspIz/Fm3JD86ntZxLVGkeZUz62FqErdohYfkFIVcv7GONTEyrz5HLlnpv
FJ0MR0HjrMrZrnOVZnwBKhpbLocTsH+3t5It4ReYEX0f1DIOL/KRwPvjMvBVkXY5
hblRVDQoOWc=
=d9oG
-----END PGP PUBLIC KEY BLOCK-----
```


2. 주요 퍼블릭 키를 귀하의 키링으로 가져옵니다.

```
$ 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. 서명자 공개 키를 복사하여 로컬 시스템에 `.txt` 파일로 저장합니다. 예:*signer-public-key.txt*


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

mQINBGgrxIgBEADGCTudveeeVbWpZDGX9Ni57mBRMVSJwQJ6F/PC34jw0DozxTtd
H+ZPsXLvLwerN/DVXbK8E1qNZ5RGptak8j7MPz+MC3n4txibEJpB61vpjJJM+9cC
7whaMLDT/SbykHYXdrnHqa8KsUJl7rPLJcaRN722NSxvYVMIOA9ffVXV7cfEyZi5
MbYF2Gc9LNbKaknImIva7EKeeh2/wI6YCqC5yytyfWU5dL6oHXsgTnFL9mhziMxv
WhyzawyJG6EJZsJ3WLlbIKApN6XZSXyCxOvlBrebYZjD5v0nA+TJaQ7is8atjtOI
```

```
DGe0AViw7kO8ChTpjA7YG/Uu7n/Fy7qLF/3Nz0b6cBNjemjBazQ3A3KNCpi5hqFM
Uo1WpoVLr5CXQnc0B3fBUnTIoxi0Sk5MKjH9AbYxfgqEX0ZJB9hAlc6LIEy0Yru6
MMBrIHE86IMl1NfE/DeLnCdPG23+1PttwyOt3+9z5QwmPe3VPpEfCySPcdxHKZSP
rLile8qDznEvlPDvQ0qkBxdMtVa2yct5VJkdqy6UrN2xa0dpspHjRUjHh/EY/xMt
fwMUjOKohaZ/1pjotCcksAsZWUxCNcFvLYxuxeytVk4F09Es1hj4ihhLUI+43/ic
3DHSEiext7Q8/UccNArkhSCT7UOvvL7QTuP+pjYTyiC8Vx6g/Y5Ht5+qywARAQAB
tDBBV1MgU0FNIENMSSBUZWFtIDxhd3Mtc2FtLWNsaS1zaWduZXJAYW1hem9uLmNv
bT6JAj8EEwEJACkFAmgrxIgCGy8FCQPCZwAHCwkIBwMCAQYVCAIJCgsEFgIDAQIe
AQIXgAAKCRBAlKuxvt/atJo6EAC/5C8uJs76W5f5V5XNAMzwBFiZuYpop3DRReCo
P68ZZylokAC9ShRZnIOujpDJtlNS7T/G00BzmcpspkYYE531ALaXcHWmb9XV0Ajg
J8iboAVBLY0C7mhL/cbJ3v9QlpXXjyTuhexkJCV8rdHVX/0H8WqTZplEaRuZ7p8q
PMxddg4ClwstYuH3O/dmNdlGqfb4Fqy8MnV1yGSXRs5Jf+sDlN2UO4mbpyk/mr1c
f/jFxmx86IkCWJVvdXWCVTe2AFy3NHCdLtdnEvFhokCOQd9wibUWX0j9vq4cVRZT
qamnpAQaOlH3lXOwrjqo8b1AIPoRWSfMtCYvh6kA8MAJv4cAznzXILSLtOE0mzaU
qp5qoy37wNIjeztX6c/q4wss05qTlJhnNu4s3nh5VHultooaYpmDxp+ala5TWeuM
KZDI4KdAGF4z0Raif+N53ndOYIiXkY0goUbsPCnVrCwoK9PjjyoJncq7c14wNl5O
IQUZEjyYAQDGZqs5XSfY4zW2cCXatrfozKF7R1kSU14DfJwPUyksoNAQEQezfXyq
kr0gfIWK1r2nMdqS7WgSx/ypS5kdyrHuPZdaYfEVtuezpoT2lQQxOSZqqlp5hI4R
nqmPte53WXJhbC0tgTIJWn+Uy/d5Q/aSIfD6o8gNLS1BDs1j1ku0XKu1sFCHUcZG
aerdsIkCHAQQAQkABgUCaCvFeAAKCRBC/V96c62IWt3/D/9gOLzWtz62lqJRCsri
wcA/yz88ayKb/GUv3FCT5Nd9JZt8y1tW+AE3SPTdcpfZmt5UN2sRzljO61mpKJzp
eBvYQ9og/34ZrRQqeg8bz02u34LKYl1gD0xY0bWtB7TGIxIZZYqZECoPR0Dp6ZzB
abzkRSsJkEk0vbZzJhfWFYs98qfp/G0suFSBE79O8Am33DB2jQ/Sollh1VmNE6Sv
EOgR6+2yEkS2D0+msJMa/V82v9gBTPnxSlNV1d8Dduvt9rbM3LoxiNXUgx/s52yY
U6H3bwUcQ3UY6uRe1UWo5QnMFcDwfg43+q5rmjB4xQyX/BaQyF5K0hZyG+42/pH1
EMwl8qN617FTxo3hvQUi/cBahlhQ8EVYsGnHDVxLCisbq5iZvp7+XtmMy1Q417gT
EQRo8feJh31elGWlccVR2pZgIm1PQ69dzzseHnnKkGhifik0bDGo5/IH2EgI1KFn
SG399RMU/qRzOPLVP3i+zSJmhMqG8cnZaUwE5V4P21vQSclhhd2Hv/C4SVKNqA2i
+oZbHj2vAkuzTTL075AoANebEjPGqwsKZi5mWUE5Pa931JeiXxWZlEB7rkgQ1PAB
fsDBhYLt4MxCWAhifLMA6uQ4BhXu2RuXOqNfSbqa8jVF6DB6cD8eAHGpPKfJOl30
LtZnq+n4SfeNbZjD2FQWZR4CrA==
=lHfs
-----END PGP PUBLIC KEY BLOCK-----
```

4. 서명자 퍼블릭 키를 귀하의 키링으로 가져옵니다.

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

gpg: key 4094ABB1BEDFDAB4: public key "AWS SAM CLI Team <aws-sam-cli-
signer@amazon.com>" imported
gpg: Total number processed: 1
gpg:               imported: 1  (RSA: 1)
gpg: no ultimately trusted keys found
```

출력 결과의 키 값을 기록해 두십시오. 예:*4094ABB1BEDFDAB4*

5. 키 값을 사용하여 서명자 공개 키 핑거프린트를 획득하고 확인합니다.

```
$ gpg --fingerprint 4094ABB1BEDFDAB4

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

지문은 다음과 일치해야 합니다.

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

지문 스트링이 일치하지 않으면 AWS SAM CLI 설치관리자를 사용하지 마십시오. aws-sam-cli GitHub 리포지토리에 [이슈를 생성하여](#) AWS SAM팀으로 에스컬레이션하십시오.

6. 서명자 공개 키의 서명을 확인하십시오.

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

pub   rsa4096 2025-05-19 [SCEA] [expires: 2027-05-19]
      EF463E86CA31933BB688CC1A4094ABB1BEDFDAB4
uid           [ unknown] AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>
sig!3        4094ABB1BEDFDAB4 2025-05-19  [self-signature]
sig!         42FD5F7A73AD885A 2025-05-19  AWS SAM CLI Primary <aws-sam-cli-
primary@amazon.com>
```

`1 signature not checked due to a missing key`이 확인되면 이전 단계를 반복하여 기본 및 서명자 공개 키를 키링으로 가져오세요.

기본 공개 키와 서명자 공개 키 모두의 키 값이 표시되어 있어야 합니다.

서명자 공개 키의 무결성을 확인했으므로 이제 서명자 공개 키를 사용하여 AWS SAM CLI패키지 설치관리자 프로그램을 확인할 수 있습니다.

AWS SAMCLI 패키지 설치관리자 프로그램의 무결성을 확인하려면

1. AWS SAMCLI패키지 서명 파일 가져오기 - 다음 명령을 사용하여 AWS SAM CLI 패키지 설치관리자 프로그램의 서명 파일을 다운로드합니다.

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

2. 서명 파일을 확인 - 다운로드된 `.sig` 및 `.zip` 파일 모두를 gpg 명령의 파라미터로 전달합니다. 다음은 예제입니다.

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

다음과 같이 출력됩니다

```
gpg: Signature made Mon 19 May 2025 01:21:57 AM UTC using RSA key ID
 4094ABB1BEDFDAB4
gpg: Good signature from "AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: EF46 3E86 CA31 933B B688  CC1A 4094 ABB1 BEDF DAB4
```

- `WARNING: This key is not certified with a trusted signature!` 메시지는 무시해도 됩니다. 이것은 귀하의 개인 PGP 키(보유하신 경우)와 AWS SAM CLI PGP 키 사이에 신뢰 체인이 없기 때문에 발생한 것입니다. 자세한 내용은 Web of trust를 참조하세요.
- 출력 결과에 `BAD signature` 문구가 포함된 경우 귀하가 절차를 올바르게 수행했는지 확인합니다. 이 응답을 계속 받으면 aws-sam-cli GitHub 리포지토리에 이슈를 제기하여 AWS SAM팀에 올리고 다운로드한 파일은 사용하지 마십시오.

`Good signature from "AWS SAM CLI Team <aws-sam-cli-signer@amazon.com>"` 메시지는 서명이 확인되었으며 귀하는 설치를 진행할 수 있다는 의미입니다.

macOS

GUI 및 명령줄 설치관리자 프로그램

`pkgutil` 도구를 사용하거나 수동으로 AWS SAM CLI 패키지 설치관리자 서명 파일의 무결성을 확인할 수 있습니다.

pkgutil을 사용하여 확인하려면

1. 귀하의 로컬 기기에서 다운로드한 설치관리자 경로를 제공하여 다음 명령을 실행하십시오.

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

다음은 예제입니다.

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

2. 출력 결과에서 Developer ID Installer: AMZN Mobile LLC를 위해 SHA256 fingerprint을 찾습니다. 다음은 예제입니다.

```
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 fingerprint은 다음 값과 일치해야 합니다.

```
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
```

지문 스트링이 일치하지 않으면 AWS SAM CLI 설치관리자를 사용하지 마십시오. aws-sam-cli GitHub 리포지토리에 이슈를 생성하여 AWS SAM팀으로 에스컬레이션하십시오. 지문 스트링이 일치하면 패키지 설치관리자 프로그램을 사용하여 계속 진행할 수 있습니다.

패키지 설치관리자 프로그램을 수동으로 확인하려면

- Apple 지원 웹사이트에서 수동으로 다운로드한 Apple 소프트웨어 업데이트의 정품 여부를 확인하는 방법을 참조하세요.

Windows

AWS SAMCLI설치관리자 프로그램은 Windows 운영 체제용 MSI 파일로 패키지됩니다.

설치관리자 프로그램 무결성 확인

1. 설치관리자 프로그램의을 마우스 오른쪽 버튼으로 클릭하고 속성 창을 엽니다.
2. 디지털 서명(Digital Signatures) 탭을 선택합니다.
3. 서명 목록에서 Amazon Services Web Services, Inc.를 선택한 후 세부 정보를 선택합니다.
4. 일반 탭이 선택되어 있지 않으면 이 탭을 선택한 후 인증서 보기(View Certificate)를 선택합니다.
5. 세부 정보 탭을 선택한 다음, 선택되어 있지 않은 경우 표시 드롭다운 목록에서 모두(All)를 선택합니다.
6. 지문(Thumbprint) 필드가 보일 때까지 아래로 스크롤한 후 지문(Thumbprint)을 선택합니다. 그러면 아래 창에 전체 지문 값이 표시됩니다.
7. 지문 값을 다음 값과 일치시킵니다. 값이 일치하면 설치를 진행합니다. 그렇지 않은 경우 aws-sam-cli GitHub 리포지토리에서 이슈를 제기하여 AWS SAM팀에 에스컬레이션합니다.

```
cd62479397f09d72a04c7399a254b0a91da53d6c
```

해시 값을 확인합니다.

Linux

x86_64 - 명령줄 설치관리자

다음 명령을 사용하여 해시 값을 생성하여 다운로드한 설치관리자 파일의 무결성 및 신뢰성을 확인합니다.

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

출력 결과는 다음과 같아야 합니다.

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

64자의 SHA-256 해시 값을 GitHub에 관한 AWS SAMCLI 릴리스 노트에 있는 원하는 AWS SAM CLI 버전의 값과 비교합니다.

macOS

GUI 및 명령줄 설치관리자 프로그램

다음 명령을 사용하여 해시 값을 생성하여 다운로드한 설치관리자의 무결성 및 신뢰성을 확인합니다.

```
$ 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
```

귀하의 64자의 SHA-256 해시 값을 AWS SAMCLI릴리스 노트 GitHub리포지토리의 해당 값과 비교합니다.

자습서: AWS SAM에서 Hello World 애플리케이션 배포

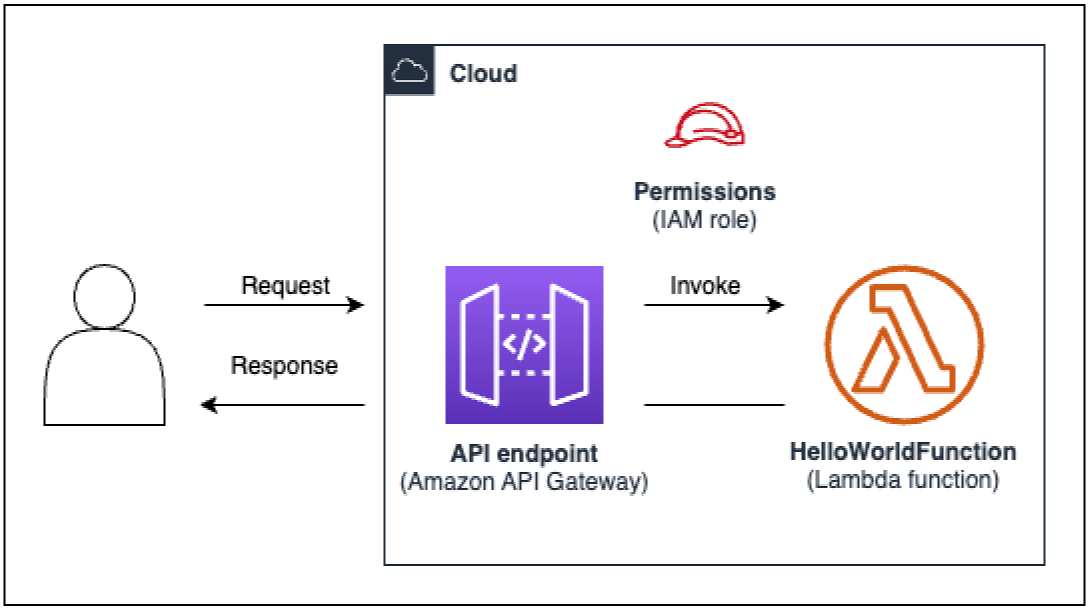
이 사용 지침서에서는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAMCLI)를 사용하여 다음을 완료합니다.

- 샘플 Hello World 애플리케이션을 초기화, 빌드 및 배포합니다.
- 로컬에서 변경하고 AWS CloudFormation에 동기화합니다.
- 개발 호스트에서 로컬 테스트 수행
- AWS 클라우드에서 샘플 애플리케이션을 삭제합니다.

샘플 Hello World 애플리케이션은 기본적인 API 백엔드를 구현합니다. 다음 리소스로 구성됩니다.

- Amazon API Gateway – 함수를 간접 호출하는 데 사용할 API 엔드포인트입니다.
- AWS Lambda – HTTP API GET 요청을 처리하고 `hello world` 메시지를 반환하는 함수입니다.
- AWS Identity and Access Management(IAM) 역할 – 서비스가 안전하게 상호 작용할 수 있는 권한을 제공합니다.

다음 다이어그램은 이 애플리케이션의 구성 요소를 보여줍니다.



주제

- 사전 조건
- 1단계: Hello World 샘플 애플리케이션 초기화
- 2단계: 귀하의 애플리케이션 빌드
- 3단계: AWS 클라우드에 대한 애플리케이션 배포
- 4단계: 애플리케이션의 실행
- 5단계: AWS 클라우드에서 함수와 상호작용하십시오.
- 6단계: 애플리케이션을 수정하고 AWS 클라우드에 동기화합니다.
- 7단계: (선택 사항) 로컬에서 애플리케이션 테스트
- 8단계: AWS 클라우드에서 애플리케이션 삭제
- 문제 해결
- 자세히 알아보기

사전 조건

다음 단계들을 귀하가 완료했는지 확인하십시오.

- AWS SAM 사전 조건
- AWS SAM CLI 설치

> **Note**
>
> 서버리스 개발을 위해 Visual Studio Code 사용을 선호한다면, VS Code 인터페이스를 사용한 단계별 지침은 VS Code용 AWS Toolkit 사용자 가이드의 서버리스 애플리케이션 생성을 참조하세요.

1단계: Hello World 샘플 애플리케이션 초기화

이 단계에서는 AWS SAM CLI를 사용하여 귀하의 로컬 기기에 샘플 Hello World 애플리케이션 프로젝트를 생성합니다.

샘플 Hello World 애플리케이션을 초기화하려면

1. 명령줄 프롬프트에 선택한 시작 디렉터리에서 다음 명령을 실행합니다.

```
$ sam init
```

> **Note**
>
> 이 명령은 서버리스 애플리케이션을 초기화하고 프로젝트 디렉터리를 생성합니다. 이 디렉터리에는 `template.yaml` 파일을 포함하여 여러 파일과 폴더가 있으며, 이는 귀하의 AWS SAM 템플릿입니다.
> 설치된 Python 버전은 `template.yaml`에 지정된 `Runtime` 속성과 일치해야 합니다. 프로덕션 워크로드의 경우, AWS 환경에서 완전히 지원되는 Python 3.12 또는 이전 버전을 사용할 것을 권장합니다. Python 버전이 일치하지 않으면 빌드 오류가 발생합니다.
> 버전 호환성 문제를 해결하려면:
>
> - 초기화하는 동안 호환되는 런타임을 지정하세요: `sam init --runtime python3.9`
> - 초기화 후 `template.yaml`의 `Runtime` 속성을 수정하세요

2. AWS SAM CLI가 새 애플리케이션을 초기화하는 과정을 안내합니다. 다음을 구성합니다.

   1. AWS퀵 스타트 템플릿을 선택하여 시작 템플릿을 선택합니다.
   2. Hello World 예제 템플릿을 선택하고 다운로드하십시오.
   3. Python런타임 및 `zip` 패키지 유형을 사용합니다.

4. 이 사용 지침서에서는 AWS X-Ray 추적을 옵트아웃합니다. 자세한 내용은 AWS X-Ray 개발자 가이드에서 AWS X-Ray란 무엇입니까?를 참조하세요.
5. 이 사용 지침서에서는 Amazon CloudWatch Application Insights를 사용한 모니터링을 옵트아웃합니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon CloudWatch Application Insights를 참조하세요.
6. 이 자습서에서는 Lambda 함수에서 JSON 형식의 구조화된 로깅 설정을 옵트아웃합니다.
7. 애플리케이션 이름을 sam-app으로 지정합니다.

AWS SAMCLI대화형 흐름을 사용하려면:

- 대괄호([ ])는 기본값을 나타냅니다. 기본값을 선택하려면 답을 비워둡니다.
- 예에 **y**를 입력하고 아니요에 **n**를 입력합니다.

다음은 `sam init` 대화형 흐름에 대한 예제입니다.

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

Choose an AWS Quick Start application template
    1 - Hello World Example
    2 - Data processing
    3 - Hello World Example with Powertools for AWS Lambda
    4 - Multi-step workflow
    5 - Scheduled task
    6 - Standalone function
    7 - Serverless API
    8 - Infrastructure event management
    9 - Lambda Response Streaming
   10 - GraphQLApi Hello World Example
   11 - Full Stack
   12 - Lambda EFS example
   13 - Serverless Connector Hello World Example
   14 - Multi-step workflow with Connectors
   15 - DynamoDB Example
   16 - Machine Learning
```

```
Template: 1

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

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

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

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

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

3. AWS SAM CLI는 시작 템플릿을 다운로드하고 귀하의 로컬 기기에 애플리케이션 프로젝트 디렉터리 구조를 만듭니다. 다음은 AWS SAM CLI 출력의 예시입니다.

```
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. 명령줄에서 새로 만든 `sam-app` 디렉터리로 이동합니다. 다음은 AWS SAM CLI에서 만든 내용의 예입니다.

   Note

   `tree` 명령이 자동으로 설치되지 않은 경우 다음 명령을 실행하세요: `brew install tree`

```
$ cd sam-app

$ tree

### README.md
### __init__.py
### events
#   ### event.json
### hello_world
#   ### __init__.py
#   ### app.py
#   ### requirements.txt
### samconfig.toml
### template.yaml
### tests
    ### __init__.py
    ### integration
    #   ### __init__.py
    #   ### test_api_gateway.py
    ### requirements.txt
    ### unit
        ### __init__.py
        ### test_handler.py

6 directories, 14 files
```

   강조해야 할 몇 가지 중요한 파일:

   - `hello_world/app.py` – 귀하의 Lambda 함수 코드가 포함되어 있습니다.
   - `hello_world/requirements.txt` – Lambda 함수에 필요한 모든 Python 종속물을 포함합니다.

- `samconfig.toml` – AWS SAM CLI에 의해 사용되는 기본 파라미터를 저장하는 애플리케이션의 구성 파일.
- `template.yaml` – 귀하의 애플리케이션 인프라 코드가 들어 있는 AWS SAM 템플릿입니다.

이제 로컬 컴퓨터에 완전히 작성된 서버리스 애플리케이션이 생겼습니다!

2단계: 귀하의 애플리케이션 빌드

이 단계에서는 AWS SAM CLI를 사용하여 애플리케이션을 빌드하고 배포를 준비합니다. 빌드하면, AWS SAM CLI가 `.aws-sam` 디렉토리를 만들고 여기에 함수 종속성, 프로젝트 코드 및 프로젝트 파일을 구성합니다.

귀하의 애플리케이션을 빌드하려면

- 명령줄에서 `sam-app` 프로젝트 디렉터리로부터 다음을 실행합니다.

```
$ sam build
```

> **Note**
>
> 로컬 기기에 Python이 설치되어 있지 않은 경우, sam build --use-container 명령어를 대신 사용합니다. AWS SAMCLI는 함수의 런타임과 종속물을 포함하는 Docker 컨테이너를 생성합니다. 이 명령은 로컬 기기에 Docker을 필요로 합니다. Docker의 설치는 Docker 설치을 참조하십시오

다음은 AWS SAM CLI 출력의 예시입니다.

```
$ 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
```

다음은 AWS SAM CLI가 생성한 `.aws-sam` 디렉토리의 축약된 예입니다.

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

강조해야 할 몇 가지 중요한 파일:

- `build/HelloWorldFunction` – Lambda 함수 코드 및 종속 항목이 포함되어 있습니다. AWS SAM CLI는 애플리케이션의 각 함수에 대한 디렉토리를 생성합니다.
- `build/template.yaml` – 배포 시 CloudFormation이 참조하는 AWS SAM 템플릿 사본이 들어 있습니다.
- `build.toml` – 애플리케이션을 빌드하고 배포할 때 AWS SAM CLI이 참조하는 기본 파라미터 값을 저장하는 구성 파일입니다.

이제 AWS 클라우드에 귀하의 애플리케이션을 배포할 준비가 되었습니다.

3단계: AWS 클라우드에 대한 애플리케이션 배포

> **Note**
>
> 이 단계에는 AWS 보안 인증 구성이 필요합니다. 자세한 내용은 AWS SAM 사전 조건에서 5단계: AWS CLI 를 사용하여 AWS 자격 증명 구성을 참조하세요

이 단계에서는 AWS SAM CLI를 사용하여 애플리케이션을 AWS 클라우드에 배포합니다. AWS SAM CLI는 다음과 같은 작업을 수행합니다.

- 배포를 위한 애플리케이션 설정을 구성하는 과정을 안내합니다.
- Amazon Simple Storage Service(S3)에 귀하의 애플리케이션 파일을 업로드합니다.
- AWS SAM템플릿을 CloudFormation 템플릿으로 변환하십시오. 그런 다음 템플릿을 CloudFormation 서비스에 업로드하여 AWS 리소스를 프로비저닝합니다.

애플리케이션을 배포하려면

1. 명령줄에서 `sam-app` 프로젝트 디렉터리로부터 다음을 실행합니다.

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

2. AWS SAMCLI 대화형 흐름에 따라 애플리케이션 설정을 구성하십시오. 다음을 구성합니다.

   1. CloudFormation스택 이름 - 스택이란 하나의 단위로 관리할 수 있는 AWS 리소스의 모음입니다. 자세히 알아보려면 AWS CloudFormation 사용자 가이드의 스택 작업을 참조하세요.
   2. CloudFormation스택을 배포할 AWS 리전 자세한 내용은 AWS CloudFormation 사용자 가이드의 CloudFormation엔드포인트를 참조하세요.
   3. 이 사용 지침서에서는 배포 전 변경 사항 확인을 옵트아웃합니다.
   4. IAM 역할 생성 허용 - 이를 통해 AWS SAM가 API Gateway 리소스와 Lambda 함수 리소스가 상호 작용하는 데 필요한 IAM 역할을 생성할 수 있습니다.
   5. 이 사용 지침서에서는 롤백 비활성화를 옵트아웃합니다.
   6. 정의된 권한 부여 없이 HelloWorldFunction 허용 – API Gateway 엔드포인트가 승인 없이 공개적으로 액세스할 수 있도록 구성되어 있기 때문에 이 메시지가 표시됩니다. 이것이 Hello World 애플리케이션을 위한 의도된 구성이므로 AWS SAM CLI가 계속 진행되도록 합니다. 권한 부여 전송에 대한 자세한 내용은을 참조하세요AWS SAM 템플릿으로 API 액세스 제어

7. 구성 파일에 인수 저장 - 그러면 배포 우선 설정에 따라 애플리케이션의 `samconfig.toml` 파일이 업데이트됩니다.
8. 기본 구성 파일 이름을 선택합니다.
9. 기본 구성 환경을 선택합니다.

다음은 `sam deploy --guided` 대화형 플로우의 출력의 예제입니다.

```
$ 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 SAM CLI는 다음을 수행하여 애플리케이션을 배포합니다.

   - AWS SAM CLI는 Amazon S3 버킷을 생성하고 귀하의 `.aws-sam` 디렉터리를 업로드합니다.
   - AWS SAM CLI는 귀하의 AWS SAM템플릿을 CloudFormation로 변환하여 CloudFormation 서비스에 업로드합니다.
   - CloudFormation은 귀하의 리소스를 프로비저닝합니다.

배포 중에는 AWS SAM CLI는 귀하의 진행 상황을 표시합니다. 다음은 출력의 예제입니다.

```
Looking for resources needed for deployment:

    Managed S3 bucket: aws-sam-cli-managed-default-samclisam-s3-demo-
bucket-1a4x26zbcdkqr
    A different default S3 bucket can be set in samconfig.toml

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

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

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

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

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

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

```
Waiting for changeset to be created..

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


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


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

CloudFormation events from stack operations (refresh every 0.5 seconds)
-------------------------------------------------------------------------------------------------
ResourceStatus                       ResourceType
 LogicalResourceId                    ResourceStatusReason
-------------------------------------------------------------------------------------------------
UPDATE_IN_PROGRESS                   AWS::Lambda::Function
 HelloWorldFunction                   -
UPDATE_COMPLETE                      AWS::Lambda::Function
 HelloWorldFunction                   -
UPDATE_COMPLETE_CLEANUP_IN_PROGRE    AWS::CloudFormation::Stack          sam-app
                         -
SS
DELETE_IN_PROGRESS                   AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt    -
                                                                         ack
DELETE_COMPLETE                      AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt    -
                                                                         ack
```

```
UPDATE_COMPLETE                     AWS::CloudFormation::Stack          sam-app
                         -
-------------------------------------------------------------------------------------

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

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

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



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

이제 귀하의 애플리케이션이 배포되어 AWS 클라우드에서 실행 중입니다!

4단계: 애플리케이션의 실행

이 단계에서는 API 엔드포인트에 GET 요청을 보내고 Lambda 함수 출력을 확인합니다.

API 엔드포인트 값을 가져오려면

1. 이전 단계에서 AWS SAM CLI에 의해 표시된 정보에서 `Outputs` 섹션을 찾습니다. 이 섹션에서 `HelloWorldApi` 리소스를 찾아 HTTP 엔드포인트 값을 찾으십시오. 다음은 출력의 예제입니다.

```
-------------------------------------------------------------------------------------
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. 다른 방식으로는 sam list endpoints --output json 명령을 사용하여 이 정보를 가져올 수 있습니다. 다음은 출력의 예제입니다.

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

함수를 간접 호출하려면

- 브라우저 또는 명령줄을 사용하여 API 엔드포인트에 GET 요청을 보냅니다. 다음은 curl 명령을 사용한 예입니다.

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

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

5단계: AWS 클라우드에서 함수와 상호작용하십시오.

이 단계에서는 AWS SAM CLI를 사용하여 AWS 클라우드에서 귀하의 Lambda 함수를 간접 호출합니다.

Lambda 함수를 클라우드에서 간접 호출하려면

1. 이전 단계의 함수의 `LogicalResourceId`을 기록해 두십시오. 그것은 `HelloWorldFunction`이어야 합니다.
2. 명령줄에서 `sam-app` 프로젝트 디렉터리로부터 다음을 실행합니다.

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

3. AWS SAM CLI는 클라우드에서 귀하의 함수를 간접 호출하고 응답을 반환합니다. 다음은 출력의 예제입니다.

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

6단계: 애플리케이션을 수정하고 AWS 클라우드에 동기화합니다.

이 단계에서는 AWS SAM CLI sam sync --watch 명령을 사용하여 로컬 변경 내용을 AWS 클라우드에 동기화합니다.

sam sync를 사용하려면

1. 명령줄에서 `sam-app` 프로젝트 디렉터리로부터 다음을 실행합니다.

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

2. AWS SAM CLI는 프롬프트를 통해 귀하가 개발 스택을 동기화하고 있는지 확인을 구합니다. 이 sam sync --watch 명령은 로컬 변경 내용을 실시간으로 AWS 클라우드에 자동 배포하므로 개발 환경에서만 사용하는 것이 좋습니다.

   AWS SAM CLI는 로컬 변경 모니터링을 시작하기 전에 초기 배포를 수행합니다. 다음은 출력의 예제입니다.

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

Confirm that you are synchronizing a development stack.

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

Build Succeeded

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


    Deploying with following values
    ===============================
    Stack name                   : sam-app
    Region                       : us-west-2
    Disable rollback             : False
    Deployment s3 bucket         : aws-sam-cli-managed-default-samclisam-s3-demo-
bucket-1a4x26zbcdkqr
    Capabilities                 : ["CAPABILITY_NAMED_IAM",
 "CAPABILITY_AUTO_EXPAND"]
```

```
    Parameter overrides          : {}
    Signing Profiles             : null

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



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

CloudFormation events from stack operations (refresh every 0.5 seconds)
-------------------------------------------------------------------------------------------------
ResourceStatus                    ResourceType
 LogicalResourceId                 ResourceStatusReason
-------------------------------------------------------------------------------------------------
UPDATE_IN_PROGRESS                AWS::CloudFormation::Stack          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.
```

다음으로 귀하는 Lambda 함수 코드를 수정합니다. AWS SAM CLI는 이 변경 사항을 자동으로 감지하고 애플리케이션을 AWS 클라우드에 동기화합니다.

애플리케이션을 수정하고 동기화하려면

1. 귀하가 선택한 IDE에서 `sam-app/hello_world/app.py` 파일을 엽니다.
2. `message`을 변경하고 파일을 저장합니다. 다음은 예제입니다.

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

3. AWS SAM CLI는 귀하의 변경 사항을 감지하고 애플리케이션을 AWS 클라우드에 동기화합니다. 다음은 출력의 예제입니다.

```
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. 변경 사항을 확인하려면 API 엔드포인트에 GET 요청을 다시 보내십시오.

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

7단계: (선택 사항) 로컬에서 애플리케이션 테스트

> **Note**
>
> 이 단계는 선택 사항입니다.

> **Important**
>
> 이 단계는 로컬 기기에 Docker을 필요로 합니다. 로컬 테스트를 위해 AWS SAM CLI을(를) 사용하려면 Docker이(가) 설치 및 구성되어 있어야 합니다. 자세한 내용은 Docker 설치 섹션을 참조하십시오.

이 단계에서는 AWS SAM CLI sam local 명령을 사용하여 애플리케이션을 로컬에서 테스트합니다. 이를 위해 AWS SAM CLI는 Docker를 사용하여 로컬 환경을 만듭니다. 이 로컬 환경은 Lambda 함수의 클라우드 기반 실행 환경을 에뮬레이션합니다.

귀하는 다음을 수행합니다.

1. Lambda 함수를 위한 로컬 환경을 생성하고 이를 간접 호출합니다.
2. 로컬에서 HTTP API 엔드포인트를 호스팅하고 Lambda 함수를 간접 호출하는 데 사용합니다.

Lambda 함수를 로컬에서 간접 호출하려면

1. 명령줄에서 sam-app 프로젝트 디렉터리로부터 다음을 실행합니다.

```
$ sam local invoke
```

2. AWS SAM CLI는 로컬 Docker 컨테이너를 생성하고 귀하의 함수를 간접 호출합니다. 다음은 출력의 예제입니다.

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

API를 로컬에서 호스팅하려면

1. 명령줄에서 sam-app 프로젝트 디렉터리로부터 다음을 실행합니다.

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

2. AWS SAMCLI는 Lambda 함수를 위한 로컬 Docker 컨테이너를 생성하고 API 엔드포인트를 시뮬레이션하기 위한 로컬 HTTP 서버를 생성합니다. 다음은 출력의 예제입니다.

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

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

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

3. 귀하의 브라우저 또는 명령줄을 사용하여 로컬 API 엔드포인트에 GET 요청을 보냅니다. 다음은 curl 명령을 사용한 예입니다.

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

8단계: AWS 클라우드에서 애플리케이션 삭제

이 단계에서는 AWS SAM CLI sam delete 명령을 사용하여 AWS 클라우드에서 애플리케이션을 삭제합니다.

AWS 클라우드에서 귀하의 애플리케이션을 삭제하려면

1. 명령줄에서 `sam-app` 프로젝트 디렉터리로부터 다음을 실행합니다.

```
$ sam delete
```

2. AWS SAM CLI는 확인을 요청할 것입니다. 그러면 애플리케이션의 Amazon S3 버킷과 CloudFormation 스택이 삭제됩니다. 다음은 출력의 예제입니다.

```
$ 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
```

문제 해결

AWS SAM CLI 문제를 해결하려면 AWS SAMCLI 문제 해결 섹션을 참조하세요.

자세히 알아보기

AWS SAM에 대해 계속 알아보고자 한다면 다음 리소스를 참조하세요.

- 전체 AWS SAM 워크숍 – AWS SAM가 제공하는 여러 주요 기능을 설명하기 위해 마련된 워크숍입니다.
- SAM과의 세션 – 당사의 AWS 서버리스 개발자 지원 팀이 제작한 AWS SAM의 사용에 관한 동영상 시리즈입니다.
- 서버리스 랜드 – AWS 서버리스에 대한 최신 정보, 블로그, 동영상, 코드 및 학습 리소스를 한데 모아 놓은 사이트입니다.

Lambda 함수를 AWS SAM 애플리케이션으로 변환하기

Lambda 콘솔에 VS Code as Infrastructure as Code(IaC)에서 관리하려는 함수가 있는 경우 이를 VS Code로 전송한 다음 AWS SAM 템플릿으로 변환할 수 있습니다. 함수를 AWS SAM 템플릿으로 변환하면 인프라의 버전 관리를 제어하고, 배포를 자동화하고, 함수를 원격으로 디버깅하고, 개발 수명 주기 전반에 걸쳐 일관된 환경을 유지할 수 있습니다. 로컬 및 원격 디버깅 기능을 사용하면 코드를 단계별로 살펴보고, 변수를 검사하고, 로컬 및 AWS 클라우드 모두에서 중단점을 설정하여 문제를 보다 효과적으로 해결할 수도 있습니다.

콘솔에서 함수를 이동하고 로컬 개발을 설정하는 방법에 대한 지침은 AWS Lambda 개발자 안내서의 VS Code를 사용하여 로컬에서 Lambda 함수 개발을 참조하세요. 다음 지침에 따라 Lambda 함수를 콘솔에서 VS Code로 이동한 다음 AWS SAM 템플릿으로 변환합니다. 이 작업을 수행한 후 다음을 수행할 수 있습니다.

- AWS 클라우드에서 함수를 원격으로 디버깅합니다. 자세한 내용은 AWS Lambda 개발자 안내서의 VS Code를 사용하여 Lambda 함수를 원격으로 디버깅을 참조하세요.

- 연결된 리소스 및 권한을 코드형 인프라로 관리합니다. 자세한 내용은 AWS Lambda 개발자 안내서의 코드형 인프라(IaC)와 함께 Lambda 사용을 참조하세요.
- AWS SAMCLI을(를) 사용하여 애플리케이션을 배포하고 업데이트합니다. 자세한 내용은 AWS SAM CLI을 참조하세요.
- Infrastructure Composer를 사용하여 서버리스 아키텍처를 시각적으로 설계합니다. 자세한 내용은 Infrastructure Composer 개발자 안내서를 참조하세요.

자세히 알아보기

AWS SAM에 대해 계속 알아보고자 한다면 다음 리소스를 참조하세요.

- 전체 AWS SAM 워크숍 – AWS SAM가 제공하는 여러 주요 기능을 설명하기 위해 마련된 워크숍입니다.
- SAM과의 세션 – 당사의 AWS 서버리스 개발자 지원 팀이 제작한 AWS SAM의 사용에 관한 동영상 시리즈입니다.
- 서버리스 랜드 – AWS 서버리스에 대한 최신 정보, 블로그, 동영상, 코드 및 학습 리소스를 한데 모아 놓은 사이트입니다.

AWS Serverless Application Model(AWS SAM) 사용 방법

애플리케이션을 개발하는 데 사용하는 기본 도구는 AWS SAM CLI, AWS SAM 템플릿 및 AWS SAM 프로젝트(애플리케이션 프로젝트 디렉터리)입니다. 이러한 도구를 사용하여 다음을 수행할 수 있습니다.

1. 애플리케이션 개발. 여기에는 애플리케이션 초기화, 리소스 정의, 애플리케이션 빌드가 포함됩니다.
2. 애플리케이션 테스트.
3. 애플리케이션 디버깅.
4. 애플리케이션 및 리소스 배포.
5. 애플리케이션 모니터링.

sam init 명령을 실행하고 후속 워크플로를 완료하면 AWS SAM이 AWS SAM 프로젝트를 생성합니다. AWS SAM 프로젝트에 코드를 추가하여 서버리스 애플리케이션을 정의합니다. AWS SAM 프로젝트는 일련의 파일 및 폴더로 구성되지만 그 중에 가장 중요한 파일은 `template.yaml`이라는 AWS SAM 템플릿입니다. 이 템플릿에서 코드를 작성하여 서버리스 애플리케이션을 정의하는 리소스, 이벤트 소스 매핑 및 기타 속성을 표현합니다.

AWS SAM CLI에는 AWS SAM 프로젝트에서 사용하는 명령의 리포지토리가 포함되어 있습니다. 보다 구체적으로, AWS SAM CLI는 AWS SAM 프로젝트의 빌드, 변환, 배포, 디버깅, 패키징, 초기화 및 동기화에 사용하는 것입니다. 즉, AWS SAM 프로젝트를 서버리스 애플리케이션으로 전환하는 데 사용하는 것입니다.

이러한 도구에 대한 자세한 내용은 다음 장을 참조하세요.

AWS SAM CLI - AWS SAM CLI에 대해 배우고 서버리스 애플리케이션을 구축하고 실행하는 데 그것을 사용하는 방법을 알아봅니다.

AWS SAM 프로젝트 및 AWS SAM 템플릿 - AWS SAM 프로젝트 구조와 AWS SAM 템플릿을 사용하여 서버리스 애플리케이션을 정의하는 방법을 알아봅니다.

AWS SAM CLI

AWS Serverless Application Model 명령줄 인터페이스(AWS SAMCLI)는 서버리스 애플리케이션의 로컬 개발 및 테스트를 위한 명령줄 도구입니다. AWS SAMCLI은(는) 클라우드에 배포하기 전에 서버리스 애플리케이션을 로컬에서 빌드, 변환, 배포, 디버그, 패키지, 초기화 및 동기화할 수 있도록 지원합니다.

AWS SAMCLI은(는) 다양한 프레임워크 및 코드형 인프라(IaC) 도구를 사용하여 정의된 서버리스 애플리케이션과 함께 작동하며, 지원 수준은 다음과 같이 다릅니다.

- AWS SAM 템플릿 - 로컬 테스트, 디버깅, 패키징 및 배포 기능을 포함하는 전체 기능 세트를 통해 네이티브 지원을 제공합니다.
- AWS CDK 애플리케이션 - ccdk synth 명령어를 사용하여 AWS CDK 애플리케이션을 CloudFormation 템플릿으로 합성한 후 Lambda 함수의 로컬 테스트를 지원합니다.
- CloudFormation 템플릿 - AWS SAM이(가) CloudFormation을(를) 확장하므로 직접적인 호환성을 제공하여, 표준 CloudFormation 템플릿에 정의된 서버리스 리소스를 지원합니다.
- Terraform 애플리케이션 - Lambda 함수의 빌드 및 로컬 테스트에 대한 제한된 지원을 제공합니다. Terraform으로 정의된 Lambda 함수를 나타내는 AWS SAM 템플릿 아티팩트 생성을 필요로 합니다.

가장 포괄적인 기능 지원과 간소화된 개발자 경험을 위해서는 네이티브 AWS SAM 템플릿을 사용하는 것이 좋습니다.

주제

- AWS SAMCLI 명령을 문서화하는 방법
- AWS SAM CLI 구성
- AWS SAM CLI 핵심 명령
- AWS SAM CLI을(를) 사용한 로컬 테스트

AWS SAMCLI 명령을 문서화하는 방법

AWS SAM CLI 명령은 다음 형식을 사용하여 문서화됩니다.

- 프롬프트 - Linux 프롬프트는 기본적으로 문서화되며 ($)로 표시됩니다. Windows와 관련된 명령의 경우 (>)가 프롬프트로 사용됩니다. 명령을 입력할 때 프롬프트를 포함시키지 마세요.

- 디렉터리 - 특정 디렉터리에서 명령을 실행해야 하는 경우 프롬프트 기호 앞에 디렉터리 이름이 표시됩니다.
- 사용자 입력 - 명령줄에 입력하는 명령 텍스트는 **`user input`**으로 형식이 지정됩니다.
- 교체 가능한 텍스트 - 파일 이름 및 파라미터와 같은 변수 텍스트는 *`## ### ###`* 형식으로 지정됩니다. 특정 키보드 입력이 필요한 여러 줄 명령에서는 키보드 명령도 대체 가능한 텍스트로 표시될 수 있습니다. 예를 들면 *ENTER*를 입력합니다.
- 출력 - 명령에 대한 응답으로 반환되는 출력은 `computer output`와 같은 형식입니다.

다음은 `sam deploy` 명령의 출력 예입니다.

```
$ 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`은 명령줄에 입력하는 명령입니다.
2. **`sam deploy --guided --template`**은 있는 그대로 제공되어야 합니다.
3. *`template.yaml`*은 특정 파일 이름으로 바꿀 수 있습니다.
4. 출력은 `Configuring SAM deploy`에서 시작됩니다.

5. 출력에서 *ENTER*와 *y*는 사용자가 제공한 대체 가능한 값을 나타냅니다.

AWS SAM CLI 구성

AWS SAM의 이점 중 하나는 반복 작업을 제거하여 개발자의 시간을 최적화한다는 것입니다. AWS SAM CLI에는 이 용도로 사용하는 `samconfig`라는 구성 파일이 포함되어 있습니다. 기본적으로 AWS SAM CLI에는 구성이 필요하지 않지만 구성 파일을 업데이트하면 AWS SAM이 구성 파일에서 사용자 지정 파라미터를 참조하게 하고 명령을 실행할 때 사용하는 파라미터를 최소화할 수 있습니다. 다음 표의 예제에서는 명령을 최적화하는 방법을 보여줍니다.

원본	**`samconfig`** 로 최적화
sam build --cached --parallel --use-containers	sam build
sam local invoke --env-vars locals.json	sam local invoke
sam local start-api --env-vars locals.json --warm-containers EAGER	sam local start-api

AWS SAM CLI는 개발자가 서버리스 애플리케이션을 생성, 개발 및 배포하는 데 도움이 되는 일련의 명령을 제공합니다. 이러한 각 명령은 애플리케이션 및 개발자의 기본 설정에 따라 선택적 플래그를 사용하여 구성할 수 있습니다. 자세한 내용은 GitHub에서 AWS SAM CLI 콘텐츠를 참조하세요.

이 섹션의 주제에서는 AWS SAMCLI구성 파일을 생성하고 해당 기본 설정을 사용자 지정하여 서버리스 애플리케이션의 개발 시간을 최적화하는 방법을 보여줍니다.

주제

- 구성 파일(samconfig 파일)을 생성하는 방법
- 프로젝트 설정 구성
- 자격 증명 및 기본 설정 구성

구성 파일(**`samconfig`** 파일)을 생성하는 방법

AWS SAM CLI 구성 파일(파일 이름 `samconfig`)은 일반적으로 TOML 구조를 사용하지만 YAML 형식일 수도 있는 텍스트 파일입니다. AWS 빠른 시작 템플릿을 사용할 경우 이 파일은 sam init 명령을 실

행하면 생성됩니다. 애플리케이션을 배포할 때 sam deploy -\-guided 명령을 사용하여 이 파일을 업데이트할 수 있습니다.

기본값을 사용하여 배포를 완료하면 `samconfig` 파일에 `default`라는 프로파일이 포함됩니다. deploy 명령을 다시 실행할 경우 AWS SAM은 이 프로파일의 저장된 구성 설정을 적용합니다.

`samconfig` 파일의 이점은 AWS SAM이 deploy 명령 외에도 사용 가능한 다른 명령에 대한 구성 설정을 저장한다는 것입니다. 새 배포에서 생성된 이러한 값 외에도 AWS SAM CLI를 사용하는 개발자 워크플로의 다른 측면을 간소화할 수 있는 여러 속성을 `samconfig` 파일에 설정할 수 있습니다.

프로젝트 설정 구성

AWS SAM CLI와 함께 사용할 구성 파일에서 AWS SAM CLI 명령 파라미터 값과 같은 프로젝트별 설정을 지정할 수 있습니다. 이 구성에 대한 자세한 정보는 AWS SAMCLI구성 파일 섹션을 참조하세요.

구성 파일 사용

구성 파일은 환경, 명령 및 파라미터 값으로 구성됩니다. 자세한 내용은 구성 파일 기본 사항 섹션을 참조하세요.

새 환경을 구성하려면

1. 구성 파일에 새 환경을 지정합니다.

   다음은 새 `prod` 환경을 지정하는 예제입니다.

   TOML

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

   YAML

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

2. 구성 파일의 파라미터 섹션에서 파라미터 값을 키-값 쌍으로 지정합니다.

   다음은 `prod` 환경에 대한 애플리케이션 스택 이름을 지정하는 예제입니다.

TOML

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

YAML

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

3. `--config-env` 옵션을 사용하여 사용할 환경을 지정합니다.

   다음은 예제입니다.

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

파라미터 값을 구성하려면

1. 파라미터 값을 구성하려는 AWS SAM CLI 명령을 지정합니다. 모든 AWS SAM CLI 명령의 파라미터 값을 구성하려면 `global` 식별자를 사용합니다.

   다음은 `default` 환경 `sam deploy` 명령에 대한 파라미터 값을 지정하는 예제입니다.

   TOML

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

   YAML

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

   다음은 `default` 환경의 모든 AWS SAM CLI 명령에 대한 파라미터 값을 지정하는 예입니다.

TOML

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

YAML

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

2. AWS SAM CLI 대화형 흐름을 통해 파라미터 값을 지정하고 구성 파일을 수정할 수도 있습니다.

   다음은 `sam deploy --guided` 대화형 흐름에 대한 예제입니다.

```
$ 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
```

자세한 내용은 구성 파일 생성 및 수정 섹션을 참조하세요.

예시

기본 TOML 예제

다음은 samconfig.toml 구성 파일의 예입니다.

```
...
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
```

기본 YAML 예제

다음은 samconfig.yaml 구성 파일의 예입니다.

```
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
```

자격 증명 및 기본 설정 구성

AWS Command Line Interface(AWS CLI)을 사용하여 AWS 자격 증명, 기본 리전 이름, 기본 출력 형식과 같은 기본 설정을 구성합니다. 구성한 후에는 이러한 설정을 AWS SAM CLI와 함께 사용할 수 있습니다. 자세한 내용은 AWS Command Line Interface 사용 설명서의 다음 세션을 참조하세요.

- 구성 기본 사항
- 구성 및 자격 증명 파일 설정
- 에 사용되는 명명된 프로파일AWS CLI
- IAM Identity Center 지원 명명된 프로파일 사용

빠른 설정 지침은 5단계: AWS CLI 를 사용하여 AWS 자격 증명 구성 섹션을 참조하세요.

AWS SAM CLI 핵심 명령

AWS SAM CLI에는 서버리스 애플리케이션의 생성, 빌드, 테스트, 배포 및 동기화를 수행하는 데 사용하는 몇 가지 기본 명령이 있습니다. 아래 표에는 이러한 명령이 나와 있으며 각 명령에 대한 자세한 정보로 연결되는 링크를 제공합니다.

전체 AWS SAM CLI 명령 목록은 AWS SAM CLI 명령 참조 섹션을 참조하세요.

Command	하는 일	관련 주제
sam build	로컬 테스트 또는 AWS 클라우드에 대한 배포와 같은 개발자 워크플로의 후속 단계를 위해 응용 프로그램을 준비합니다.	• AWS SAM의 빌드 소개 • sam build
sam deploy	CloudFormation을 사용하여 애플리케이션을 AWS 클라우드에 배포합니다.	• AWS SAM 배포 소개 • sam deploy
sam init	새 서버리스 애플리케이션을 초기화하고 생성하는 옵션을 제공합니다.	• 에서 애플리케이션 생성 AWS SAM • sam init
sam local	서버리스 애플리케이션을 로컬에서 테스트하는 하위 명령을 제공합니다.	• sam local 명령을 사용한 테스트 소개 • sam local generate-event • sam local invoke • sam local start-api • sam local start-lambda
sam remote invoke	AWS 클라우드에서 지원되는 AWS 리소스와 상호 작용하는 방법을 제공합니다.	• sam remote invoke를 사용하여 클라우드에서 테스트 소개 • sam remote invoke
sam remote test-event	AWS Lambda 함수의 공유 가능한 테스트 이벤트에 액세스하고 관리하는 방법을 제공합니다.	• sam remote test-event를 사용한 클라우드 테스트 소개 • sam remote test-event
sam sync	로컬 애플리케이션 변경 사항을 AWS 클라우드에 빠르게 동기화하는 옵션을 제공합니다.	• sam sync를 사용하는 AWS 클라우드 동기화 소개 • sam sync

AWS SAM CLI을(를) 사용한 로컬 테스트

AWS SAMCLI은(는) 다양한 코드형 인프라(IaC) 도구에서 서버리스 애플리케이션을 로컬로 테스트할 수 있도록 지원합니다. 이 가이드는 다양한 IaC 프레임워크와 함께 AWS SAM CLI 을(를) 사용하여 로컬 테스트를 수행하는 방법을 설명합니다.

AWS SAM CLI을(를) 로컬 테스트에 사용할 때의 주요 이점은 다음과 같습니다.

- 빠른 개발 - AWS에 배포하지 않고도 코드 변경 사항을 테스트할 수 있습니다
- 비용 효율성 - AWS 비용을 발생시키지 않고 개발 및 테스트를 수행합니다
- 오프라인 기능 - 인터넷 연결 없이 애플리케이션에서 작업
- 간소화된 디버깅 - 디버거를 사용하여 로컬에서 Lambda 함수 코드를 단계별로 실행할 수 있습니다
- 현실적인 테스트 - AWS 서비스의 로컬 에뮬레이션을 사용하여 애플리케이션을 테스트할 수 있습니다

로컬에서 AWS SAM 애플리케이션 테스트

AWS SAM 템플릿을 사용하여 정의된 애플리케이션 테스트에 대한 자세한 내용은 이 가이드의 테스트 및 서버리스 애플리케이션 디버깅을 참조하세요.

로컬에서 CloudFormation 템플릿 테스트

AWS SAMCLI의 로컬 테스트 기능을 CloudFormation와(과) 함께 사용하려면 CloudFormation 템플릿에 AWS SAM 변환을 추가해야 합니다. 자세한 내용은 AWS SAM Template Anatomy를 참조하세요.

- CloudFormation 템플릿에 AWS SAM 변환 추가
- 템플릿 호환성 이해
- 서버리스 리소스 구문 탐색

로컬에서 AWS CDK 애플리케이션 테스트

cdk synth 명령어를 사용하여 AWS CDK 애플리케이션을 CloudFormation 템플릿으로 합성한 후, AWS SAM CLI을(를) 사용하여 AWS CDK 애플리케이션을 로컬로 테스트할 수 있습니다.

자세한 내용은 AWS CDK 개발자 가이드의 AWS SAM CLI을(를) 사용한 AWS CDK 애플리케이션 로컬 테스트 및 빌드를 참조하세요. 여기에는 다음이 포함됩니다.

- [로컬 테스트 시작하기]
- [AWS SAM을(를) 사용한 AWS CDK 애플리케이션 로컬 테스트]

Terraform 애플리케이션 로컬 테스트

AWS SAM CLI은(는) Terraform 프로젝트와 Terraform Cloud를 지원합니다. 이를 사용하여 다음 항목의 로컬 디버깅 및 테스트를 수행할 수 있습니다. Lambda 함수 및 레이어, 그리고 Amazon API Gateway HTTP 및 REST API.

환경을 설정하고 사용 가능한 모든 기능에 대해 알아보려면 이 가이드의 Terraform 지원을 참조하세요. 여기에는 다음이 포함됩니다.

- AWS SAM CLI의 Terraform 지원 시작하기
- AWS SAM CLI을(를) Terraform과 함께 사용하여 로컬 디버깅 및 테스트

AWS SAM 템플릿

sam init 명령을 실행하고 후속 워크플로를 완료하면 AWS SAM에서 애플리케이션 프로젝트 디렉터리를 생성하며 이것이 AWS SAM 프로젝트입니다. AWS SAM 프로젝트에 코드를 추가하여 서버리스 애플리케이션을 정의합니다. AWS SAM 프로젝트는 일련의 파일 및 폴더로 구성되지만 주로 작업하는 파일은 `template.yaml`이라는 AWS SAM 템플릿입니다. 이 템플릿에서 코드를 작성하여 서버리스 애플리케이션을 정의하는 리소스, 이벤트 소스 매핑 및 기타 속성을 표현합니다.

> **Note**
>
> AWS SAM 템플릿의 주요 요소는 AWS SAM 템플릿 사양입니다. 이 사양은 CloudFormation과 비교할 때 훨씬 더 적은 수의 코드 줄을 사용하여 서버리스 애플리케이션의 리소스, 이벤트 소스 매핑, 권한, API 및 기타 속성을 정의할 수 있는 약식 구문을 제공합니다.

이 섹션에서는 AWS SAM 템플릿에서 섹션을 사용하여 리소스 유형, 리소스 속성, 데이터 유형, 리소스 특성, 내장 함수 및 API Gateway 확장을 정의하는 방법에 대해 자세히 설명합니다.

AWS SAM 템플릿은 CloudFormation 템플릿의 확장으로, CloudFormation과 비교하여 코드 줄의 수가 훨씬 적은 약식 구문을 사용하는 고유한 구문 유형을 지원합니다. 이 구문 유형을 사용하면 서버리스 애플리케이션을 빌드할 때 개발 속도가 빨라집니다. 자세한 정보는 [AWS SAM 리소스 및 속성](#) 섹션을 참조하세요. CloudFormation 템플릿에 대한 전체 참조는 AWS CloudFormation 사용 설명서의 [CloudFormation 템플릿 참조](#)를 참조하세요.

개발할 때는 애플리케이션을 더 잘 구성하고 관리하려면 애플리케이션 코드를 별도의 파일로 분할하는 것이 유용할 때가 많습니다. 기본적인 예가 AWS SAM 템플릿에 코드를 작성하는 대신 AWS Lambda 함수 코드마다 별도의 파일을 사용하는 것입니다. 이 작업을 수행하려면 Lambda 함수 코드를 프로젝트의 하위 디렉터리에 구성하고 AWS Serverless Application Model(AWS SAM) 템플릿 내에서 해당 로컬 경로를 참조하면 됩니다.

주제

- AWS SAM 템플릿 구조
- AWS SAM 리소스 및 속성
- AWS SAM용으로 생성된 CloudFormation 리소스
- AWS SAM에서 지원되는 리소스 속성
- AWS SAM의 API Gateway 확장

- AWS SAM의 내장 함수

AWS SAM 템플릿 구조

AWS SAM 템플릿 파일은 CloudFormation 템플릿 파일의 형식을 유사하게 따르는데, 이는 AWS CloudFormation사용자 가이드의 템플릿 구조에 설명되어 있습니다. AWS SAM 템플릿 파일과 CloudFormation 템플릿 파일 간의 주요 차이점은 다음과 같습니다.

- 변환 선언. AWS SAM 템플릿 파일에는 `Transform: AWS::Serverless-2016-10-31` 선언이 필요합니다. 이 선언은 CloudFormation 템플릿 파일을 AWS SAM 템플릿 파일로 식별합니다. 변환에 대한 자세한 내용은 AWS CloudFormation 사용자 가이드의 변환을 참조하세요.
- 글로벌 섹션. `Globals` 섹션은 AWS SAM에 고유합니다. 이것은 귀하의 모든 서버리스 함수 및 API에 공통적인 속성을 정의합니다. `AWS::Serverless::Function`, `AWS::Serverless::Api`, `AWS::Serverless::SimpleTable` 리소스들 모두 `Globals` 섹션에 정의된 속성을 계승합니다. 이 섹션에 대한 자세한 내용을 알아보려면 AWS SAM 템플릿의 Globals 섹션 섹션을 참조하세요.
- 리소스 섹션 AWS SAM 템플릿에서 `Resources` 섹션은 CloudFormation 리소스와 AWS SAM 리소스의 조합을 포함할 수 있습니다. CloudFormation리소스에 관한 자세한 내용은 AWS CloudFormation 사용자 가이드의 AWS리소스 및 속성 유형 참조를 참조하세요. AWS SAM 리소스에 대한 자세한 내용은 AWS SAM 리소스 및 속성 섹션을 참조하세요.

AWS SAM 템플릿 파일의 다른 모든 섹션은 같은 이름의 CloudFormation 템플릿 파일 섹션에 해당합니다.

YAML

다음 예제에서는 YAML 형식의 템플릿 조각을 보여줍니다.

```
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
```

템플릿 섹션

AWS SAM 템플릿에는 여러 주요 섹션이 포함되어 있습니다. 오직 `Transform` 및 `Resources` 섹션만 필요합니다.

템플릿 섹션들은 어떤 순서로든 포함시킬 수 있습니다. 그러나 언어 확장을 사용하는 경우 다음 예제와 같이 서버리스 변환 이전(즉, `AWS::Serverless-2016-10-31` 이전)에 `AWS::LanguageExtensions`를 추가해야 합니다.

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

템플릿을 작성할 때 다음 목록의 논리적 순서를 사용하는 것이 유용할 수 있습니다. 이는 한 섹션의 값이 이전 섹션의 값을 참조할 수 있기 때문입니다.

[변환(필수 사항)]

AWS SAM 템플릿의 경우 귀하는 이 섹션을 `AWS::Serverless-2016-10-31`의 값으로 포함시켜야 합니다.

추가 변환은 선택 사항입니다. 변환에 대한 자세한 내용은 AWS CloudFormation 사용자 가이드의 [변환]을 참조하세요.

[글로벌(선택 사항)]

모든 서버리스 함수, API 및 단순 테이블에 공통되는 속성. `AWS::Serverless::Function`, `AWS::Serverless::Api`, `AWS::Serverless::SimpleTable` 리소스들 모두 `Globals` 섹션에 정의된 속성을 계승합니다.

이 섹션은 AWS SAM에 고유합니다. CloudFormation 템플릿에는 해당 섹션이 없습니다.

[설명(선택 사항)]

템플릿을 설명하는 텍스트 문자열입니다.

이 섹션은 CloudFormation 템플릿의 `Description` 섹션들에 직접 관련됩니다.

[Metadata(선택 사항)]

템플릿에 대한 추가 정보를 제공하는 객체입니다.

이 섹션은 CloudFormation 템플릿의 `Metadata` 섹션들에 직접 관련됩니다.

[파라미터(선택 사항)]

(스택을 생성하거나 업데이트할 때) 실행 시간에 템플릿에 전달하는 값입니다. 템플릿의 `Resources` 및 `Outputs` 섹션에서 파라미터를 참조할 수 있습니다. `Parameters` 섹션에 선언된 객체는 sam deploy --guided 명령으로 하여금 사용자에게 추가 프롬프트를 표시하게 합니다.

`sam deploy`명령의 `--parameter-overrides` 파라미터를 사용하여 전달된 값과 구성 파일의 항목이 AWS SAM 템플릿 파일의 항목보다 우선합니다. `sam deploy` 명령에 대한 자세한 내용은 AWS SAMCLI 명령 참조 내 [sam deploy] 섹션을 참조하세요. 구성 파일에 대한 자세한 내용은 [AWS SAMCLI구성 파일] 섹션을 참조하세요.

[Mappings(선택 사항)]

조건부 파라미터 값을 지정하는 데 사용할 수 있는 키와 관련 값의 매핑으로, 조회 테이블과 비슷합니다. `Resources` 및 `Outputs` 섹션의 [`Fn::FindInMap`] 내장 함수를 사용하여 키를 상응하는 값에 매칭할 수 있습니다.

이 섹션은 CloudFormation 템플릿의 `Mappings` 섹션들에 직접 관련됩니다.

[조건(선택 사항)]

스택 생성 또는 업데이트 시 특정 리소스 속성에 값이 할당되는지 또는 특정 리소스가 생성되는지 여부를 제어하는 조건입니다. 예를 들어, 스택이 프로덕션용인지 테스트 환경용인지에 따라 달라지는 리소스를 조건부로 생성할 수 있습니다.

이 섹션은 CloudFormation 템플릿의 `Conditions` 섹션들에 직접 관련됩니다.

[리소스(필수 사항)]

Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스 또는 Amazon Simple Storage Service(S3) 버킷 같은 스택 리소스들 및 이들의 속성. 템플릿의 `Resources` 및 `Outputs` 섹션에서 리소스를 참조할 수 있습니다.

이 섹션은 CloudFormation 템플릿의 `Resources` 섹션과 비슷합니다. AWS SAM 템플릿에서 이 섹션은 CloudFormation 리소스 외에도 AWS SAM 리소스를 포함할 수 있습니다.

[Outputs(선택 사항)]

귀하가 귀하의 스택의 속성을 볼 때마다 표시되는 값을 설명합니다. 예를 들어, 귀하는 S3 버킷 이름에 대해 출력을 선언하고 나서 `aws cloudformation describe-stacks` AWS Command Line Interface(AWS CLI) 명령을 직접 호출하여 해당 이름을 볼 수 있습니다.

이 섹션은 CloudFormation 템플릿의 `Outputs` 섹션들에 직접 관련됩니다.

다음 단계

AWS SAM 템플릿 파일이 포함된 샘플 서버리스 애플리케이션을 다운로드 및 배포하려면 AWS SAM 시작를 참조하고 자습서: AWS SAM에서 Hello World 애플리케이션 배포의 지침을 따르십시오.

AWS SAM 템플릿의 Globals 섹션

AWS SAM 템플릿에서 선언하는 리소스에 공통 구성이 있는 경우가 있습니다. 예를 들어 동일한 `AWS::Serverless::Function`, `Runtime`, `Memory`, `VPCConfig` 및 `Environment`의 구성을 갖는 여러 `Cors` 리소스가 있는 애플리케이션이 있을 수 있습니다. 모든 리소스에서 이 정보를 복제하는 대신 `Globals` 섹션에서 정보를 한 번 선언하고 리소스가 해당 정보를 승계하도록 할 수 있습니다.

`Globals` 섹션은 다음 AWS SAM 리소스 유형을 지원합니다.

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

예시

```
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
```

이 예제에서는 `HelloWorldFunction` 및 `ThumbnailFunction` 둘 다 `Runtime`을 위하여 "nodejs12.x", `Timeout`를 위하여 "180"초, `Handler`를 위하여 "index.handler"를 각각 사용합니다. `HelloWorldFunction`은 승계된 TABLE_NAME 외에도 메시지 환경 변수를 추가합니다. `ThumbnailFunction`은 모든 `Globals` 속성을 승계하고 API 이벤트 소스를 추가합니다.

지원되는 리소스 및 속성

AWS SAM 는 다음 리소스 및 속성을 지원합니다.

```
Globals:
  Api:
    AccessLogSetting:
    Auth:
    BinaryMediaTypes:
    CacheClusterEnabled:
```

```
    CacheClusterSize:
    CanarySetting:
    Cors:
    DefinitionUri:
    Domain:
    EndpointConfiguration:
    GatewayResponses:
    MethodSettings:
    MinimumCompressionSize:
    Name:
    OpenApiVersion:
    PropagateTags:
    TracingEnabled:
    Variables:

  Function:
    Architectures:
    AssumeRolePolicyDocument:
    AutoPublishAlias:
    CodeSigningConfigArn:
    CodeUri:
    DeadLetterQueue:
    DeploymentPreference:
    Description:
    Environment:
    EphemeralStorage:
    EventInvokeConfig:
    FileSystemConfigs:
    FunctionUrlConfig:
    Handler:
    KmsKeyArn:
    Layers:
    LoggingConfig:
    MemorySize:
    PermissionsBoundary:
    PropagateTags:
    ProvisionedConcurrencyConfig:
    RecursiveLoop:
    ReservedConcurrentExecutions:
    RolePath:
    Runtime:
    RuntimeManagementConfig:
    SnapStart:
    SourceKMSKeyArn:
```

```
    Tags:
    TenancyConfig:
    Timeout:
    Tracing:
    VpcConfig:

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

  SimpleTable:
    SSESpecification:

  StateMachine:
    PropagateTags:
```

> **Note**
>
> 이전 목록에 없는 리소스 및 속성은 일체 지원되지 않습니다. 지원되지 않는 몇 가지 이유는 다음과 같습니다. 1) 이들이 잠재적 보안 문제를 야기하거나 2) 이들이 템플릿을 이해하기 어렵게 만듭니다.

묵시적 API

AWS SAM 는 `Events` 섹션에서 APIs를 선언할 때 암시적 API를 생성합니다. `Globals`을 사용하여 묵시적 API의 모든 속성을 재설정할 수 있습니다.

재설정 가능한 속성

리소스는 `Globals` 섹션에서 선언한 속성을 재설정할 수 있습니다. 예를 들어 환경 변수 맵에 새 변수를 추가하거나 전역적으로 선언된 변수를 재설정할 수 있습니다. 하지만 리소스는 `Globals` 섹션에 지정된 속성을 제거할 수 없습니다.

보다 일반적으로 `Globals` 섹션은 모든 리소스가 공유하는 속성을 선언합니다. 일부 리소스는 전역적으로 선언된 속성에 새 값을 제공할 수 있지만 제거할 수는 없습니다. 일부 리소스는 속성을 사용하지만 다른 리소스는 사용하지 않는 경우 `Globals` 섹션에서 속성을 선언해서는 안 됩니다.

다음 단원에서는 데이터 유형별로 재설정이 작동하는 방식을 설명합니다.

기본 데이터 유형 대체됨

기본 데이터 유형에는 문자열, 숫자, 부울 등이 포함됩니다.

`Resources` 섹션에 지정된 값이 `Globals` 섹션의 값을 대체합니다.

예시

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

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

`Runtime`에 대한 `MyFunction`은 `python3.9`으로 설정됩니다.

맵 병합됨

맵은 사전 또는 키-값 페어의 모음이라고도 합니다.

`Resources` 섹션의 맵 항목은 글로벌 맵 항목과 병합됩니다. 중복된 항목이 있는 경우 `Resource` 섹션 항목이 `Globals` 섹션 항목에 우선합니다.

예시

```
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
```

MyFunction의 환경 변수는 다음과 같이 설정됩니다.

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

목록 추가 기능

리스트는 배열이라고도 합니다.

Globals 섹션의 목록 항목은 Resources 섹션의 목록 앞에 추가됩니다.

예시

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

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

SecurityGroupIds의 MyFunction을 위한 VpcConfig 양식은 다음과 같이 설정됩니다.

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

AWS SAM 리소스 및 속성

이 섹션에서는 특정 리소스 및 속성 유형을 설명합니다 AWS SAM. AWS SAM 간편 구문을 사용하여 이러한 리소스 및 속성을 정의합니다.는 CloudFormation 리소스 및 속성 유형 AWS SAM 도 지원합

니다. 모든 AWS 리소스 및 속성 유형 CloudFormation 과 AWS SAM 지원에 대한 참조 정보는 AWS CloudFormation 사용 설명서의 AWS 리소스 및 속성 유형 참조를 참조하세요.

주제

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

AWS::Serverless::Api

HTTPS 엔드포인트를 통해 간접 호출할 수 있는 Amazon API Gateway 리소스 및 메서드 컬렉션을 생성합니다.

AWS::Serverless::Api 리소스를 AWS서버리스 애플리케이션 정의 템플릿에 명시적으로 추가할 필요는 없습니다. 이 유형의 리소스는 AWS::Serverless::Function 리소스를 참조하지 않는 템플릿에 정의된 AWS::Serverless::Api 리소스에 정의된 Api 이벤트를 결합하여 묵시적으로 생성됩니다.

기본 Amazon API Gateway 리소스를 구성할 수 있는 더 많은 기능을 제공하는 OpenAPI를 사용하여 API를 정의하고 문서화하려면 AWS::Serverless::Api리소스를 사용해야 합니다.

CloudFormation 후크 또는 IAM 정책을 사용하여 API Gateway 리소스에 액세스를 제어할 권한 부여자가 연결되어 있는지 확인하는 것이 좋습니다.

CloudFormation 후크 사용에 대한 자세한 내용은 CLI 사용자 가이드의 CloudFormation후크 등록 및 apigw-enforce-authorizer GitHub 리포지토리를 참조하세요.

IAM 정책 사용에 대한 자세한 내용은 API 게이트웨이 개발자 가이드의 API 경로에 권한 부여 필요를 참조하세요.

> **Note**
>
> AWS CloudFormation에 배포하면 AWS SAM는 귀하의 AWS SAM 리소스를 CloudFormation 리소스로 변환됩니다. 자세한 내용은 AWS SAM용으로 생성된 CloudFormation 리소스 섹션을 참조하십시오.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

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
  Policy: JSON
```

```
  StageName: String
  Tags: Map
  TracingEnabled: Boolean
  Variables: Map
```

속성

`AccessLogSetting`

특정 단계를 위한 액세스 로그 세팅 구성에 대한 설정입니다.

유형: AccessLogSetting

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::ApiGateway::Stage` 리소스의 `AccessLogSetting` 속성으로 직접 전달됩니다.

`AlwaysDeploy`

API에 대한 변경 사항이 감지되지 않은 경우에도 항상 API를 배포합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`ApiKeySourceType`

사용량 계획에 따라 측정 요청을 위한 API 키의 원본입니다. 유효한 값은 `HEADER` 및 `AUTHORIZER`입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::ApiGateway::RestApi` 리소스의 `ApiKeySourceType` 속성으로 직접 전달됩니다.

`Auth`

권한 부여를 구성하여 API Gateway API에 대한 액세스를 제어합니다.

AWS SAM을 사용하는 액세스 구성에 대한 자세한 내용은 AWS SAM 템플릿으로 API 액세스 제어 섹션을 참조하세요. 글로벌 권한 부여자를 재정의하는 방법을 보여주는 예시는 the section called "Amazon API Gateway REST API에 대한 글로벌 권한 부여자 재정의"에서 확인하세요.

유형: APIAuth

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`BinaryMediaTypes`

API가 반환할 수 있는 MIME 유형 목록입니다. 이를 사용하여 API에 대한 바이너리 지원을 활성화할 수 있습니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `BinaryMediaTypes` 리소스의 `AWS::ApiGateway::RestApi` 속성과 유사합니다. BinaryMediaTypes 목록은 CloudFormation 리소스와 OpenAPI 문서 모두에 추가됩니다.

`CacheClusterEnabled`

해당 단계에 대해 캐싱이 활성화되는지를 나타냅니다. 응답을 캐시하려면 `CachingEnabled`에 따라 `true`을 `MethodSettings`으로 설정해야 합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::ApiGateway::Stage` 리소스의 `CacheClusterEnabled` 속성으로 직접 전달됩니다.

`CacheClusterSize`

해당 단계의 캐시 클러스터 크기입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::ApiGateway::Stage` 리소스의 `CacheClusterSize` 속성으로 직접 전달됩니다.

`CanarySetting`

canary 설정을 정규 배포 단계로 구성합니다.

유형: CanarySetting

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::ApiGateway::Stage` 리소스의 `CanarySetting` 속성으로 직접 전달됩니다.

`Cors`

모든 API 게이트웨이 API에 대한 CORS(Cross-Origin Resource Sharing)를 관리합니다. 허용할 도메인을 문자열로 지정하거나 추가 Cors 구성을 사용하여 사전을 지정합니다.

> **Note**
>
> OpenAPI 정의를 수정하려면 CORS는 AWS SAM이 필요합니다. `DefinitionBody`에서 인라인 OpenAPI 정의를 생성하여 CORS를 켭니다.

자세한 내용은 API 게이트웨이 개발자 가이드의 API 게이트웨이 REST API 리소스 활성화를 참조하세요.

형식: 문자열 | CorsConfiguration

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`DefinitionBody`

API를 설명하는 OpenAPI 사양입니다. `DefinitionUri`이나 `DefinitionBody` 어느 것도 지정되지 않은 경우 SAM은 템플릿 구성을 기반으로 `DefinitionBody`을 생성합니다.

API를 정의하는 로컬 OpenAPI 파일을 참조하려면 `AWS::Include` 변환을 사용하십시오. 자세한 내용은 AWS SAM이 로컬 파일을 업로드하는 방법을 참조하십시오.

유형: JSON

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 Body 리소스의 AWS::ApiGateway::RestApi 속성과 유사합니다. 특정 속성이 제공되는 경우, 콘텐츠는 CloudFormation으로 전달되기 전에 DefinitionBody에 삽입되거나 수정될 수 있습니다. 속성에는 Auth, BinaryMediaTypes, Cors, GatewayResponses, Models 및 해당 EventSource에 대한 AWS::Serverless::Function 유형의 Api가 포함됩니다.

DefinitionUri

Amazon S3 Uri, 로컬 파일 경로 또는 API를 정의하는 OpenAPI 문서의 위치 객체입니다. 이 속성이 참조하는 Amazon S3 객체는 유효한 OpenAPI 파일이어야 합니다. DefinitionUri이나 DefinitionBody 어느 것도 지정되지 않은 경우 SAM은 템플릿 구성을 기반으로 DefinitionBody을 생성합니다.

로컬 파일 경로를 제공하는 경우 템플릿은 sam deploy 또는 sam package명령이 포함된 워크플로를 거쳐야 정의가 제대로 변환됩니다.

DefinitionUri이 참조하는 외부 OpenAPI 파일에서는 내장 함수가 지원되지 않습니다. 대신 DefinitionBody변환 포함과 함께 속성을 사용하여 OpenAPI 정의를 템플릿으로 가져오십시오.

형식: 문자열 | APIdeEfinition

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 BodyS3Location 리소스의 AWS::ApiGateway::RestApi 속성과 유사합니다. 중첩된 Amazon S3 속성은 다르게 지정됩니다.

Description

Api 리소스에 대한 설명입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS::ApiGateway::RestApi 리소스의 Description 속성으로 직접 전달됩니다.

DisableExecuteApiEndpoint

클라이언트가 기본 execute-api 엔드포인트를 사용하여 API를 간접 호출할 수 있는지를 지정합니다. 기본적으로 클라이언트는 기본 https://{api_id}.execute-api.

`{region}.amazonaws.com`(으)로 API를 간접 호출할 수 있습니다. 클라이언트가 사용자 지정 도메인 이름을 사용하여 API를 간접 호출하도록 요구하려면 `True`을(를) 지정합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `DisableExecuteApiEndpoint` 리소스의 `AWS::ApiGateway::RestApi` 속성과 유사합니다. 이것은 `x-amazon-apigateway-endpoint-configuration` 확장의 `disableExecuteApiEndpoint` 속성으로 직접 전달되며, 이것은 `Body` 리소스의 `AWS::ApiGateway::RestApi` 속성에 추가됩니다.

`Domain`

이 API 게이트웨이 API의 사용자 지정 도메인을 구성합니다.

유형: DomainConfiguration

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`EndpointConfiguration`

REST API의 엔드포인트 유형.

유형: EndpointConfiguration

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `EndpointConfiguration` 리소스의 `AWS::ApiGateway::RestApi` 속성과 유사합니다. 중첩된 구성 속성들의 이름은 각각 다르게 지정됩니다.

`FailOnWarnings`

경고가 발생할 때 API 생성을 롤백할 것인지(`true`) 하지 않을 것인지(`false`) 지정합니다. 기본값은 `false`입니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::ApiGateway::RestApi` 리소스의 [`FailOnWarnings`] 속성으로 직접 전달됩니다.

`GatewayResponses`

API에 대한 게이트웨이 응답을 구성합니다. 게이트웨이 응답은 API 게이트웨이에서 직접 또는 Lambda 권한 부여자를 통해 반환한 응답입니다. 자세한 내용은 [게이트웨이 응답을 위한 Api Gateway OpenAPI 확장] 설명서를 참조하세요.

유형: 맵

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`MergeDefinitions`

AWS SAM은 API 이벤트 소스에서 OpenAPI 사양을 생성합니다. `true`가 이것을 AWS SAM 리소스에 정의된 인라인 OpenAPI사양에 병합하도록 `AWS::Serverless::Api`을 지정하십시오. 병합하지 않도록 `false`을 지정하십시오.

`MergeDefinitions`은 `DefinitionBody` 속성이 `AWS::Serverless::Api`을 정의할 것을 요구합니다. `MergeDefinitions`는 `DefinitionUri`에 관하여 `AWS::Serverless::Api` 속성과 호환되지 않습니다.

기본 값: `false`

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`MethodSettings`

로깅, 지표, Cache TTL, 제한을 포함하여 API 단계의 모든 설정을 구성합니다.

유형: [MethodSetting]의 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::ApiGateway::Stage` 리소스의 [`MethodSettings`] 속성으로 직접 전달됩니다.

`MinimumCompressionSize`

클라이언트의 Accept-Encoding 헤더를 기반으로 응답 본문을 압축할 수 있습니다. 응답 본문 크기가 구성된 임계값보다 크거나 같으면 압축이 트리거됩니다. 최대 본문 크기 임계값은 10MB(10,485,760바이트)입니다. - 지원되는 압축 유형은 gzip, deflate 및 identity입니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::ApiGateway::RestApi` 리소스의 `MinimumCompressionSize` 속성으로 직접 전달됩니다.

`Mode`

이 속성은 OpenAPI를 사용하여 REST API를 정의하는 경우에만 적용됩니다. `Mode`는 API 게이트웨이에서 리소스 업데이트를 처리하는 방법을 결정합니다. 자세한 내용은 리소스의 AWS::ApiGateway::RestApi모드 속성을 참조하세요.

유효한 값: `overwrite` 또는 `merge`

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::ApiGateway::RestApi` 리소스의 `Mode` 속성으로 직접 전달됩니다.

`Models`

API 메서드에서 사용할 스키마입니다. 이러한 스키마는 JSON 또는 YAML을 사용하여 설명할 수 있습니다. 예제 모델은 이 페이지 하단의 예제 섹션을 참조하세요.

유형: 맵

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Name`

API 게이트웨이 RestApi 리소스의 이름

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::ApiGateway::RestApi` 리소스의 [Name](#) 속성으로 직접 전달됩니다.

`OpenApiVersion`

사용할 OpenApi의 버전입니다. Swagger 사양의 경우 `2.0`를 사용할 수도 있고, `3.0.1`와 같은 OpenApi 3.0 버전 중 하나를 사용할 수도 있습니다. OpenAPI에 대한 자세한 내용은 [OpenAPI 사양](#)을 참조하세요.

> **Note**
>
> AWS SAM은 `Stage` 기본적으로 호출되는 스테이지를 생성합니다. 이 속성을 임의의 유효한 값으로 설정하면 스테이지 `Stage`가 생성되지 않습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`PropagateTags`

`Tags`속성의 태그를 [AWS::Serverless::Api](#) 생성된 리소스로 전달할지 여부를 지정합니다. 귀하의 생성된 리소스에 태그를 전파하도록 `True`을 지정합니다.

유형: 부울

필수 항목 여부: 아니요

기본값: `False`

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Policy`

API에 대한 권한이 포함된 정책 문서입니다. 정책의 ARN을 설정하려면 `!Join` 내장 함수를 구분 기호 `""`, 값 `"execute-api:/"` 및 `"*"`와 함께 사용합니다.

유형: JSON

필수 항목 여부: 아니요

CloudFormation 호환성: 이 속성은 `AWS::ApiGateway::RestApi` 리소스의 정책 속성에 직접 전달됩니다.

`StageName`

API 게이트웨이에서 간접 호출된 URI(Uniform Resource Identifier)의 첫 번째 경로 세그먼트로 사용하는 단계의 이름입니다.

스테이지 리소스를 참조하려면 `<api-logical-id>.Stage`를 사용하십시오. AWS::Serverless::Api 리소스를 지정할 때 생성된 리소스를 참조하는 방법에 대한 자세한 내용은 CloudFormation가 지정된 경우 생성되는 AWS::Serverless::Api 리소스 섹션을 참조하세요. 생성된 CloudFormation 리소스에 대한 일반적인 정보는 AWS SAM용으로 생성된 CloudFormation 리소스 섹션을 참조하세요.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `StageName` 리소스의 `AWS::ApiGateway::Stage` 속성과 유사합니다. SAM에서는 필수이지만 API 게이트웨이에서는 필요하지 않습니다

추가 참고 사항: 묵시적 API의 단계 이름은 “Prod”입니다.

`Tags`

이 API 게이트웨이 단계에 추가할 태그를 지정하는 맵(문자열 간)입니다. 태그의 유효한 키와 값에 대한 자세한 내용은 사용자 가이드의 AWS CloudFormation리소스 태그를 참조하세요.

유형: 맵

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `Tags` 리소스의 `AWS::ApiGateway::Stage` 속성과 유사합니다. SAM의 태그 속성은 키:값의 쌍으로 구성되고, CloudFormation에서는 태그 객체 목록으로 구성됩니다.

`TracingEnabled`

해당 단계에 대해 X-Ray를 사용한 활성 추적을 활성화하는지를 지시합니다. X-Ray에 대한 자세한 내용은 API 게이트웨이 개발자 가이드의 X-Ray를 사용하여 REST API에 대한 사용자 요청 추적을 참조하세요.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::ApiGateway::Stage` 리소스의 `TracingEnabled` 속성으로 직접 전달됩니다.

`Variables`

단계 변수를 정의하는 맵(문자열 대 문자열)으로서, 여기서 변수 이름이 키가 되고 변수 값이 값이 됩니다. 변수 이름에는 영숫자 문자만 사용할 수 있습니다. 값은 정규식 `[A-Za-z0-9._~:/?#&=,-]+`과 일치해야 합니다.

유형: 맵

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::ApiGateway::Stage` 리소스의 `Variables` 속성으로 직접 전달됩니다.

반환 값

Ref

`Ref` 내장 함수에 이 리소스의 논리적 ID가 제공되면 그것은 기저의 API 게이트웨이 API의 ID를 반환합니다.

`Ref` 함수의 사용에 대한 자세한 내용은 AWS CloudFormation 사용자 가이드의 `Ref` 섹션을 참조하세요.

Fn::GetAtt

`Fn::GetAtt`은 이 유형의 지정된 속성에 대한 값을 반환합니다. 다음은 사용 가능한 속성과 반환되는 샘플 값.

`Fn::GetAtt`의 사용에 대한 자세한 내용은 AWS CloudFormation 사용자 가이드의 `Fn::GetAtt` 섹션을 참조하세요.

`RootResourceId`

`RestApi` 리소스의 루트 리소스 ID입니다(예: `a0bc123d4e`).

예시

SimpleApi예시

API 엔드포인트가 있는 Lambda 함수를 포함하는 Hello World AWS SAM 템플릿 파일입니다. 이 파일은 작동하는 서버리스 애플리케이션을 위한 전체 AWS SAM 템플릿 파일입니다.

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}
```

ApiCors예시

Lambda 통합 및 CORS 구성과 함께 외부 Swagger 파일에 API가 정의된 AWS SAM 템플릿 스니펫. 이는 AWS SAM 정의를 보여주는 AWS::Serverless::Api 템플릿 파일의 일부에 불과합니다.

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
```

ApiCognitoAuth예시

Amazon Cognito를 사용하여 API를 상대로 하는 요청을 승인하는 API가 포함된 AWS SAM 템플릿 스니펫. 이는 AWS SAM 정의를 보여주는 AWS::Serverless::Api 템플릿 파일의 일부에 불과합니다.

YAML

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

ApiModels예시

모델 스키마가 포함된 API가 포함된 AWS SAM 템플릿 스니펫. 이는 AWS SAM 템플릿 파일의 일부에 불과하며, 두 개의 모델 스키마가 포함된 AWS::Serverless::Api 정의를 보여줍니다.

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
```

캐싱 예제

API 엔드포인트가 있는 Lambda 함수를 포함하는 Hello World AWS SAM 템플릿 파일입니다. API에는 하나의 리소스와 메서드에 대한 캐싱이 활성화되어 있습니다. 캐싱에 대한 자세한 내용은 API 게이트웨이 개발자 가이드의 응답성 향상을 위한 API 캐싱 활성화 를 참조하세요.

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}
```

프라이빗 API를 사용한 사용자 지정 도메인 예시

VPC 엔드포인트가 프라이빗 도메인에 매핑된 API 엔드포인트를 가진 Lambda 함수를 포함하는 Hello World AWS SAM 템플릿 파일입니다. 이 템플릿은 VPC 엔드포인트와 프라이빗 도메인 사이에 도메인 액세스 연결을 생성합니다. 자세한 내용은 API 게이트웨이의 프라이빗 API의 사용자 지정 도메인 이름을 참조하세요.

YAML

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

Parameters:
    DomainName:
      Type: String
      Default: mydomain.example.com
    CertificateArn:
```

```
      Type: String
    HostedZoneId:
      Type: String
    VpcEndpointId:
      Type: String
    VpcEndpointDomainName:
      Type: String
    VpcEndpointHostedZoneId:
      Type: String

Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Properties:
      InlineCode: |
        def handler(event, context):
            return {'body': 'Hello World!', 'statusCode': 200}
      Handler: index.handler
      Runtime: python3.13
      Events:
        Fetch:
          Type: Api
          Properties:
            RestApiId:
              Ref: MyApi
            Method: Get
            Path: /get
  MyApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      EndpointConfiguration:
        Type: PRIVATE
        VPCEndpointIds:
        - !Ref VpcEndpointId
      Policy:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal: '*'
          Action: execute-api:Invoke
          Resource: execute-api:/*
        - Effect: Deny
          Principal: '*'
```

```
        Action: execute-api:Invoke
        Resource: execute-api:/*
        Condition:
          StringNotEquals:
            aws:SourceVpce: !Ref VpcEndpointId
    Domain:
      DomainName: !Ref DomainName
      CertificateArn: !Ref CertificateArn
      EndpointConfiguration: PRIVATE
      BasePath:
      - /
      Route53:
        HostedZoneId: !Ref HostedZoneId
        VpcEndpointDomainName: !Ref VpcEndpointDomainName
        VpcEndpointHostedZoneId: !Ref VpcEndpointHostedZoneId
      AccessAssociation:
        VpcEndpointId: !Ref VpcEndpointId
      Policy:
        Version: '2012-10-17       '
        Statement:
        - Effect: Allow
          Principal: '*'
          Action: execute-api:Invoke
          Resource: execute-api:/*
        - Effect: Deny
          Principal: '*'
          Action: execute-api:Invoke
          Resource: execute-api:/*
          Condition:
            StringNotEquals:
              aws:SourceVpce: !Ref VpcEndpointId
```

ApiAuth

권한 부여를 구성하여 API Gateway API에 대한 액세스를 제어합니다.

AWS SAM을 사용하여 액세스를 구성하는 방법에 대한 자세한 내용 및 예는 [AWS SAM 템플릿으로 API 액세스 제어](#) 섹션을 참조하세요.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

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

> **Note**
>
> `Authorizers` 속성은 `AWS_IAM`을(를) 포함하지만, `AWS_IAM`에 필요한 추가 구성은 없습니다. 문제 해결 예는 [AWS IAM](#)을(를) 참조하세요.

속성

`AddApiKeyRequiredToCorsPreflight`

`ApiKeyRequired` 및 `Cors` 속성이 설정된 경우 `AddApiKeyRequiredToCorsPreflight`를 설정하면 API 키가 `Options` 속성에 추가됩니다.

유형: 부울

필수 항목 여부: 아니요

기본값: `True`

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`AddDefaultAuthorizerToCorsPreflight`

`DefaultAuthorizer` 및 `Cors` 속성이 설정된 경우 `AddDefaultAuthorizerToCorsPreflight`를 설정하면 기본 권한 부여자가 OpenAPI 섹션의 `Options` 속성에 추가됩니다.

유형: 부울

필수 항목 여부: 아니요

기본값: True

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`ApiKeyRequired`

true로 설정하면 모든 API 이벤트에 API 키가 필요합니다. 자세한 내용을 알아보려면 API Gateway 개발자 안내서의 Amazon API Gateway에서 API 사용량 계획 생성 및 사용을 참조하세요.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Authorizers`

API Gateway API에 대한 액세스를 제어하는 데 사용되는 권한 부여자.

자세한 내용은 AWS SAM 템플릿으로 API 액세스 제어을 참조하십시오

유형: CognitoAuthorizer | LambdaTokenAuthorizer | LambdaRequestAuthorizer | AWS_IAM

필수 항목 여부: 아니요

기본값: 없음

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

추가 참고 사항: SAM은 Api의 OpenApi 정의에 권한 부여자를 추가합니다.

`DefaultAuthorizer`

기본적으로 API 호출을 승인하는 데 사용되는 API Gateway API의 기본 권한 부여자를 지정합니다.

> **Note**
>
> 이 API와 연결된 함수의 Api EventSource가 IAM 권한을 사용하도록 구성된 경우, 이 속성을 `AWS_IAM`로 설정해야 합니다. 그렇지 않으면 오류가 발생합니다.

유형: 문자열

필수 항목 여부: 아니요

기본값: 없음

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`InvokeRole`

모든 리소스 및 메서드의 통합 보안 인증 정보를 이 값으로 설정합니다.

`CALLER_CREDENTIALS`이 호출자 보안 인증 정보를 사용하여 엔드포인트를 간접 호출하는 `arn:aws:iam:::<user>/`에 매핑됩니다.

유효한 값: `CALLER_CREDENTIALS`, `NONE`, `IAMRoleArn`

유형: 문자열

필수 항목 여부: 아니요

기본값: `CALLER_CREDENTIALS`

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`ResourcePolicy`

API의 모든 메서드와 경로에 대한 리소스 정책을 구성합니다.

유형: 리소스 정책 설명

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

추가 참고 사항: 이 설정은 ApiFunctionAuth를 사용하여 개별 `AWS::Serverless::Function`에서도 정의할 수도 있습니다. 이는 `EndpointConfiguration: PRIVATE`를 포함하는 API에 필요합니다.

`UsagePlan`

이 API와 연결된 사용량 계획을 구성합니다. 사용량 계획에 대한 자세한 내용은 API Gateway 개발자 안내서의 API 키를 이용한 사용량 계획의 생성 및 사용 섹션을 참조하세요.

이 AWS SAM 속성은 이 속성이 설정되면 3개의 AWS::ApiGateway::UsagePlan, AWS::ApiGateway::UsagePlanKey, AWS::ApiGateway::ApiKey 추가 CloudFormation 리소스를 생성합니다. 이 시나리오에 대한 자세한 내용은 UsagePlan 속성 지정됨 섹션을 참조하세요. 생성된 CloudFormation 리소스에 대한 일반적인 정보는 AWS SAM용으로 생성된 CloudFormation 리소스 섹션을 참조하세요.

유형: ApiUsagePlan

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

CognitoAuth

Cognito 인증 예제

YAML

```
Auth:
  Authorizers:
    MyCognitoAuth:
     UserPoolArn:
       Fn::GetAtt:
         - MyUserPool
         - Arn
     AuthType: "COGNITO_USER_POOLS"
  DefaultAuthorizer: MyCognitoAuth
  InvokeRole: CALLER_CREDENTIALS
  AddDefaultAuthorizerToCorsPreflight: false
  ApiKeyRequired: false
  ResourcePolicy:
    CustomStatements: [{
      "Effect": "Allow",
      "Principal": "*",
      "Action": "execute-api:Invoke",
      "Resource": "execute-api:/Prod/GET/pets",
      "Condition": {
          "IpAddress": {
              "aws:SourceIp": "1.2.3.4"
```

```
            }
          }
      }]
      IpRangeDenylist:
        - "10.20.30.40"
```

AWS IAM

AWS IAM 예제

YAML

```
Auth:
  Authorizers: AWS_IAM
```

ApiUsagePlan

API Gateway API의 사용량 계획을 구성합니다. 사용량 계획에 대한 자세한 내용은 API Gateway 개발자 안내서의 API 키를 이용한 사용량 계획의 생성 및 사용을 참조하세요.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

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

속성

`CreateUsagePlan`

이 사용량 계획의 구성 방법을 결정합니다. 유효한 값은 `PER_API`, `SHARED`, `NONE`입니다.

`PER_API`는 이 API와 관련된 AWS::ApiGateway::UsagePlan, AWS::ApiGateway::ApiKey 및 AWS::ApiGateway::UsagePlanKey 리소스를 생성합니다. 이러한 리소스는 *<api-logical-*

*id>*UsagePlan, *<api-logical-id>*ApiKey, *<api-logical-id>*UsagePlanKey의 논리적 ID를 각각 가지고 있습니다.

SHARED는 동일한 AWS::ApiGateway::UsagePlan 템플릿에서 CreateUsagePlan: SHARED도 가지는 모든 API에서 공유되는 AWS SAM, AWS::ApiGateway::ApiKey 및 AWS::ApiGateway::UsagePlanKey 리소스를 생성합니다. 이러한 리소스는 ServerlessUsagePlan, ServerlessApiKey, ServerlessUsagePlanKey의 논리적 ID를 각각 가지고 있습니다. 이 옵션을 사용하는 경우 정의 충돌과 불확실한 상태를 방지하기 위해 하나의 API 리소스에만 이 사용량 계획의 추가 구성을 추가하는 것이 좋습니다.

NONE는 이 API를 사용한 사용량 계획 생성 또는 연결을 비활성화합니다. 이는 AWS SAM 템플릿의 Globals 섹션에서 SHARED 또는 PER_API가 지정된 경우에만 필요합니다.

유효한 값: PER_API, SHARED, NONE

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

Description

사용량 단계에 대한 설명입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성: 이 속성은 AWS::ApiGateway::UsagePlan 리소스의 Description 속성으로 직접 전달됩니다.

Quota

사용자가 지정된 간격 내에서 실행할 수 있는 요청 수를 구성합니다.

유형: QuotaSettings

필수 항목 여부: 아니요

CloudFormation 호환성: 이 속성은 AWS::ApiGateway::UsagePlan 리소스의 Quota 속성으로 직접 전달됩니다.

`Tags`

사용 계획과 연결할 임의 태그(키-값 페어)의 배열입니다.

이 속성은 CloudFormation 태그 유형을 사용합니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation 호환성: 이 속성은 `AWS::ApiGateway::UsagePlan` 리소스의 `Tags` 속성으로 직접 전달됩니다.

`Throttle`

전체 요청 빈도(초당 평균 요청 수) 및 버스트 용량을 구성합니다.

유형: ThrottleSettings

필수 항목 여부: 아니요

CloudFormation 호환성: 이 속성은 `AWS::ApiGateway::UsagePlan` 리소스의 `Throttle` 속성으로 직접 전달됩니다.

`UsagePlanName`

사용량 계획의 이름입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성: 이 속성은 `AWS::ApiGateway::UsagePlan` 리소스의 `UsagePlanName` 속성으로 직접 전달됩니다.

예시

UsagePlan

다음은 사용량 계획의 예시입니다.

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

Amazon Cognito 사용자 풀 권한 부여자를 정의합니다.

자세한 정보와 지침은 [AWS SAM 템플릿으로 API 액세스 제어](#) 섹션을 참조하세요.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

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

속성

`AuthorizationScopes`

이 권한 부여자의 권한 부여 범위 목록.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Identity`

이 속성은 권한 부여자에 대한 수신 요청에서 `IdentitySource`를 지정하는 데 사용할 수 있습니다.

유형: CognitoAuthorizationIdentity

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`UserPoolArn`

사용자 풀을 참조하거나 이 cognito 권한 부여자를 추가할 사용자 풀을 지정할 수 있습니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

CognitoAuth

Cognito 인증 예제

YAML

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

```
        ValidationExpression: myauthvalidationexpression
```

CognitoAuthorizationIdentity

이 속성은 권한 부여자에 대한 수신 요청에서 IdentitySource를 지정하는 데 사용할 수 있습니다. IdentitySource에 대한 자세한 내용은 APIGateway Authorizer OpenAPI 확장 프로그램을 참조하세요.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

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

속성

`Header`

OpenAPI 정의에서 권한 부여를 위한 헤더 이름을 지정합니다.

유형: 문자열

필수 여부: 아니요

기본값: 권한 부여

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`ReauthorizeEvery`

API Gateway가 권한 부여자 결과를 캐싱하는 기간을 지정하는 TTL(Time to Live) 기간(초)입니다. 0보다 큰 값을 지정한 경우 API Gateway에서 권한 부여자 응답을 캐싱합니다. 기본적으로 API Gateway는 이 속성을 300으로 설정합니다. 최대 값은 3600 또는 1시간입니다.

유형: 정수

필수 항목 여부: 아니요

기본값: 300

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`ValidationExpression`

수신되는 자격 증명에 대한 확인 표현식입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

CognitoAuthIdentity

YAML

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

LambdaRequestAuthorizer

Lambda 함수를 사용하여 API에 대한 액세스를 제어하도록 Lambda 함수를 구성합니다.

자세한 정보와 지침은 [AWS SAM 템플릿으로 API 액세스 제어](#) 섹션을 참조하세요.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

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

속성

`DisableFunctionDefaultPermissions`

`AWS::Serverless::Api` 리소스와 권한 부여자 Lambda 함수 간에 권한을 프로비저닝하기 위해 AWS SAM이 `AWS::Lambda::Permissions` 리소스를 자동으로 생성하지 않도록 `true`를 지정합니다.

기본 값: `false`

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`FunctionArn`

API에 대한 권한 부여를 제공하는 Lambda 함수의 함수 ARN을 지정합니다.

> **Note**
>
> `AWS::Serverless::Api`에 대해 `FunctionArn`가 지정되면 AWS SAM은 자동으로 `AWS::Lambda::Permissions` 리소스를 생성합니다. `AWS::Lambda::Permissions` 리소스는 API와 권한 부여자 Lambda 함수 간에 권한을 프로비저닝합니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`FunctionInvokeRole`

Lambda 권한 부여자의 OpenAPI 정의에 권한 부여자 보안 인증 정보를 추가합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`FunctionPayloadType`

이 속성은 API의 Lambda 권한 부여자 유형을 정의하는 데 사용할 수 있습니다.

유효한 값: `TOKEN` 또는 `REQUEST`

유형: 문자열

필수 항목 여부: 아니요

기본값: `TOKEN`

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Identity`

이 속성은 권한 부여자에 대한 수신 요청에서 `IdentitySource`를 지정하는 데 사용할 수 있습니다. 이 속성은 `FunctionPayloadType` 속성이 `REQUEST`로 설정된 경우에만 필요합니다.

유형: [LambdaRequestAuthorizationIdentity]

필수 항목 여부: 조건부

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

LambdaRequestAuth

YAML

```
Authorizers:
  MyLambdaRequestAuth:
    FunctionPayloadType: REQUEST
    FunctionArn:
      Fn::GetAtt:
        - MyAuthFunction
        - Arn
```

```
    FunctionInvokeRole:
      Fn::GetAtt:
        - LambdaAuthInvokeRole
        - Arn
    Identity:
      Headers:
        - Authorization1
```

LambdaRequestAuthorizationIdentity

이 속성은 권한 부여자에 대한 수신 요청에서 IdentitySource를 지정하는 데 사용할 수 있습니다. IdentitySource에 대한 자세한 내용은 APIGateway Authorizer OpenAPI 확장 프로그램을 참조하세요.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

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

속성

`Context`

지정된 컨텍스트 문자열을 `context.contextString` 형식의 매핑 표현식으로 변환합니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Headers`

헤더를 `method.request.header.name` 형식의 쉼표로 구분된 매핑 표현식 문자열로 변환합니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`QueryStrings`

지정된 쿼리 문자열을 `method.request.querystring.queryString` 형식의 쉼표로 구분된 매핑 표현식 문자열로 변환합니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`ReauthorizeEvery`

API Gateway가 권한 부여자 결과를 캐싱하는 기간을 지정하는 TTL(Time to Live) 기간(초)입니다. 0보다 큰 값을 지정한 경우 API Gateway에서 권한 부여자 응답을 캐싱합니다. 기본적으로 API Gateway는 이 속성을 300으로 설정합니다. 최대 값은 3600 또는 1시간입니다.

유형: 정수

필수 항목 여부: 아니요

기본값: 300

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`StageVariables`

지정된 단계 변수를 `stageVariables.stageVariable` 형식의 쉼표로 구분된 매핑 표현식 문자열로 변환합니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

LambdaRequestIdentity

YAML

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

LambdaTokenAuthorizer

Lambda 함수를 사용하여 API에 대한 액세스를 제어하도록 Lambda 함수를 구성합니다.

자세한 정보와 지침은 AWS SAM 템플릿으로 API 액세스 제어 섹션을 참조하세요.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

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

속성

`DisableFunctionDefaultPermissions`

AWS::Serverless::Api 리소스와 권한 부여자 Lambda 함수 간에 권한을 프로비저닝하기 위해 AWS SAM이 AWS::Lambda::Permissions 리소스를 자동으로 생성하지 않도록 true를 지정합니다.

기본 값: `false`

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`FunctionArn`

API에 대한 권한 부여를 제공하는 Lambda 함수의 함수 ARN을 지정합니다.

> **Note**
>
> `AWS::Serverless::Api`에 대해 `FunctionArn`가 지정되면 AWS SAM은 자동으로 `AWS::Lambda::Permissions` 리소스를 생성합니다. `AWS::Lambda::Permissions` 리소스는 API와 권한 부여자 Lambda 함수 간에 권한을 프로비저닝합니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`FunctionInvokeRole`

Lambda 권한 부여자의 OpenAPI 정의에 권한 부여자 보안 인증 정보를 추가합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`FunctionPayloadType`

이 속성은 API의 Lambda 권한 부여자 유형을 정의하는 데 사용할 수 있습니다.

유효한 값: `TOKEN` 또는 `REQUEST`

유형: 문자열

필수 항목 여부: 아니요

기본값: TOKEN

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Identity`

이 속성은 권한 부여자에 대한 수신 요청에서 `IdentitySource`를 지정하는 데 사용할 수 있습니다. 이 속성은 `FunctionPayloadType` 속성이 `REQUEST`로 설정된 경우에만 필요합니다.

유형: [LambdaTokenAuthorizationIdentity]

필수 항목 여부: 조건부

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

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

이 속성은 권한 부여자에 대한 수신 요청에서 IdentitySource를 지정하는 데 사용할 수 있습니다. IdentitySource에 대한 자세한 내용은 APIGateway Authorizer OpenAPI 확장 프로그램을 참조하세요.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

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

속성

`Header`

OpenAPI 정의에서 권한 부여를 위한 헤더 이름을 지정합니다.

유형: 문자열

필수 여부: 아니요

기본값: 권한 부여

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`ReauthorizeEvery`

API Gateway가 권한 부여자 결과를 캐싱하는 기간을 지정하는 TTL(Time to Live) 기간(초)입니다. 0보다 큰 값을 지정한 경우 API Gateway에서 권한 부여자 응답을 캐싱합니다. 기본적으로 API Gateway는 이 속성을 300으로 설정합니다. 최대 값은 3600 또는 1시간입니다.

유형: 정수

필수 항목 여부: 아니요

기본값: 300

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`ValidationExpression`

수신되는 자격 증명에 대한 확인 표현식입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

LambdaTokenIdentity

YAML

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

ResourcePolicyStatement

API의 모든 메서드와 경로에 대한 리소스 정책을 구성합니다. 리소스 정책에 대한 자세한 내용은 API Gateway 개발자 안내서의 [API Gateway 리소스 정책을 사용하는 액세스 제어]를 참조하세요.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  AwsAccountBlacklist: List
  AwsAccountWhitelist: List
```

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

속성

`AwsAccountBlacklist`

차단할 AWS 계정.

유형: 문자열 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`AwsAccountWhitelist`

허용할 AWS 계정. 이 속성의 예제 사용 예제를 알아보려면 이 페이지 하단의 예제을 참조하세요

유형: 문자열 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`CustomStatements`

이 API에 적용할 사용자 지정 리소스 정책 설명 목록. 이 속성의 예제 사용 예제를 알아보려면 이 페이지 하단의 예제을 참조하세요

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`IntrinsicVpcBlacklist`

차단할 Virtual Private Cloud(VPC))목록입니다. 여기서 각 VPC는 동적 참조 또는 Ref 내장 함수와 같은 참조로 지정됩니다. 이 속성의 예제 사용 예제를 알아보려면 이 페이지 하단의 예제을 참조하세요

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`IntrinsicVpcWhitelist`

허용할 VPC 목록. 여기서 각 VPC는 동적 참조 또는 Ref 내장 함수와 같은 참조로 지정됩니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`IntrinsicVpceBlacklist`

차단할 VPC 엔드포인트 목록. 여기서 각 VPC 엔드포인트는 동적 참조 또는 Ref 내장 함수와 같은 참조로 지정됩니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`IntrinsicVpceWhitelist`

허용할 VPC 엔드포인트 목록. 여기서 각 VPC 엔드포인트는 동적 참조 또는 Ref 내장 함수와 같은 참조로 지정됩니다. 이 속성의 예제 사용 예제를 알아보려면 이 페이지 하단의 예제을 참조하세요

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`IpRangeBlacklist`

차단할 IP 주소 또는 주소 범위. 이 속성의 예제 사용 예제를 알아보려면 이 페이지 하단의 예제을 참조하세요

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`IpRangeWhitelist`

허용할 IP 주소 또는 주소 범위.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`SourceVpcBlacklist`

차단할 소스 VPC 또는 VPC 엔드포인트. 소스 VPC 이름은 `"vpc-"`로 시작하고 소스 VPC 엔드포인트 이름은 `"vpce-"`로 시작해야 합니다. 이 속성의 예제 사용 예제를 알아보려면 이 페이지 하단의 예제을 참조하세요

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`SourceVpcWhitelist`

허용할 소스 VPC 또는 VPC 엔드포인트. 소스 VPC 이름은 `"vpc-"`로 시작하고 소스 VPC 엔드포인트 이름은 `"vpce-"`로 시작해야 합니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

리소스 정책 예시

다음 예시에서는 두 개의 IP 주소와 한 개의 소스 VPC를 차단하고 한 AWS계정을 허용합니다.

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

API를 정의하는 OpenAPI 문서.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

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

속성

`Bucket`

OpenAPI 파일이 저장되는 Amazon S3 버킷의 이름입니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `Bucket` `AWS::ApiGateway::RestApi` 데이터 유형의 `S3Location` 속성에 직접 전달됩니다.

`Key`

OpenAPI 파일의 Amazon S3 키.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `Key` `AWS::ApiGateway::RestApi` 데이터 유형의 `S3Location` 속성에 직접 전달됩니다.

`Version`

버전이 지정된 객체의 경우 OpenAPI 파일의 버전.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [Version](#) `AWS::ApiGateway::RestApi` 데이터 유형의 `S3Location` 속성에 직접 전달됩니다.

예시

Uri 정의 예제

API 정의 예

YAML

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

CorsConfiguration

API Gateway API의 교차 오리진 리소스 공유(CORS)를 관리합니다. 허용할 도메인을 문자열로 지정하거나 추가 Cors 구성을 사용하여 사전을 지정합니다.

> **Note**
>
> OpenAPI 정의를 수정하려면 CORS는 AWS SAM이 필요합니다. `DefinitionBody`에서 인라인 OpenAPI 정의를 생성하여 CORS를 켭니다. `CorsConfiguration`이 OpenAPI 정의에서 설정되고 속성 수준에서도 설정된 경우 AWS SAM이 이를 병합합니다. 속성 수준이 OpenAPI 정의보다 우선합니다.

자세한 내용은 [API 게이트웨이 개발자 가이드](#)의 API 게이트웨이 REST API 리소스 활성화를 참조하세요.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  AllowCredentials: Boolean
```

```
  AllowHeaders: String
  AllowMethods: String
  AllowOrigin: String
  MaxAge: String
```

속성

`AllowCredentials`

요청에 보안 인증 정보를 포함할 수 있는지 여부를 나타내는 부울.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`AllowHeaders`

허용할 헤더 문자열.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`AllowMethods`

허용할 HTTP 메서드가 포함된 문자열입.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`AllowOrigin`

허용할 오리진 문자열. 이것은 문자열 형식의 쉼표로 구분된 목록일 수 있습니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`MaxAge`

CORS Preflight 요청을 캐시하는 데 걸리는 시간(초)이 포함된 문자열.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

CorsConfiguration

CORS 구성의 예입니다. 이것은 CORS가 구성되어 있는 AWS::Serverless::Api 정의 및 AWS::Serverless::Function을 보여주는 AWS SAM 템플릿 파일의 일부에 불과합니다. Lambda 프록시 통합 또는 HTTP 프록시 통합을 사용하는 경우 백엔드는 `Access-Control-Allow-Origin`, `Access-Control-Allow-Methods` 및 `Access-Control-Allow-Headers` 헤더를 반환해야 합니다.

YAML

```
Resources:
  ApiGatewayApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      Cors:
        AllowMethods: "'POST, GET'"
        AllowHeaders: "'X-Forwarded-For'"
        AllowOrigin: "'https://example.com'"
        MaxAge: "'600'"
        AllowCredentials: true
  ApiFunction: # Adds a GET method at the root resource via an Api event
    Type: AWS::Serverless::Function
    Properties:
```

```
      Events:
        ApiEvent:
          Type: Api
          Properties:
            Path: /
            Method: get
            RestApiId:
              Ref: ApiGatewayApi
      Runtime: python3.10
      Handler: index.handler
      InlineCode: |
        import json
        def handler(event, context):
          return {
          'statusCode': 200,
          'headers': {
            'Access-Control-Allow-Headers': 'Content-Type',
            'Access-Control-Allow-Origin': 'https://example.com',
            'Access-Control-Allow-Methods': 'POST, GET'
            },
          'body': json.dumps('Hello from Lambda!')
          }
```

DomainConfiguration

API의 사용자 지정 도메인을 구성합니다.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  AccessAssociation: DomainAccessAssociation
  BasePath: List
  CertificateArn: String
  DomainName: String
  EndpointConfiguration: String
  MutualTlsAuthentication: MutualTlsAuthentication
  NormalizeBasePath: Boolean
  OwnershipVerificationCertificateArn: String
  Policy: Json
```

```
  Route53: Route53Configuration
  SecurityPolicy: String
```

속성

AccessAssociation

AWS::ApiGateway::DomainNameAccessAssociation 리소스를 생성하는 데 필요한 구성입니다.

AWS SAM은 이 속성이 설정되면 AWS::ApiGateway::DomainNameAccessAssociation 리소스를 생성합니다. CloudFormation 리소스 태그 지정에 대한 자세한 내용은 AWS SAM용으로 생성된 CloudFormation 리소스 섹션을 참조하세요.

유형: DomainAccessAssociation

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

BasePath

Amazon API Gateway 도메인 이름을 사용하여 구성할 basepath 목록입니다.

유형: 목록

필수 항목 여부: 아니요

기본값: /

CloudFormation 호환성: 이 속성은 `AWS::ApiGateway::BasePathMapping` 리소스의 `BasePath` 속성과 유사합니다. AWS SAM은 이 속성에 지정된 `BasePath`당 하나씩 여러 `AWS::ApiGateway::BasePathMapping` 리소스를 만듭니다.

CertificateArn

이 도메인 이름 엔드포인트의 AWS 관리형 인증서의 Amazon 리소스 이름(ARN)입니다. 유일하게 지원되는 소스는 AWS Certificate Manager입니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation 호환성: 이 속성은 AWS::ApiGateway::DomainName 리소스의 CertificateArn 속성과 유사합니다. EndpointConfiguration이 REGIONAL(기본값)로 설정된 경우 CertificateArn이 AWS::ApiGateway::DomainName의 RegionalCertificateArn에 매핑됩니다. EndpointConfiguration이 EDGE로 설정된 경우 CertificateArn이 AWS::ApiGateway::DomainName의 CertificateArn에 매핑됩니다. EndpointConfiguration이(가) PRIVATE(으)로 설정된 경우, 이 속성은 AWS::ApiGateway::DomainNameV2 리소스로 전달됩니다.

추가 참고 사항: EDGE 엔드포인트의 경우 us-east-1 AWS 리전에 인증서를 생성해야 합니다.

DomainName

Amazon API Gateway의 API에 대한 사용자 지정 도메인 이름입니다. 대문자는 지원되지 않습니다.

AWS SAM은 이 속성이 설정되면 AWS::ApiGateway::DomainName 리소스를 생성합니다. 이 시나리오에 대한 자세한 내용은 DomainName 속성이 지정되었습니다. 섹션을 참조하세요. CloudFormation 리소스 태그 지정에 대한 자세한 내용은 AWS SAM용으로 생성된 CloudFormation 리소스 섹션을 참조하세요.

유형: 문자열

필수 항목 여부: 예

CloudFormation 호환성: 이 속성은 EndpointConfiguration이 PRIVATE(으)로 설정된 경우 AWS::ApiGateway::DomainName 리소스의 DomainName 속성으로 직접 전달되거나, AWS::ApiGateway::DomainNameV2(으)로 전달됩니다.

EndpointConfiguration

사용자 지정 도메인에 매핑할 API Gateway 엔드포인트의 유형을 정의합니다. 이 속성의 값에 따라 CertificateArn 속성이 CloudFormation에 매핑되는 방식이 결정됩니다.

유효한 값: EDGE, REGIONAL 또는 PRIVATE

유형: 문자열

필수 항목 여부: 아니요

기본값: REGIONAL

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`MutualTlsAuthentication`

사용자 지정 도메인 이름에 대한 상호 전송 계층 보안(TLS) 인증 구성입니다.

유형: MutualTlsAuthentication

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::ApiGateway::DomainName` 리소스의 `MutualTlsAuthentication` 속성으로 직접 전달됩니다.

`NormalizeBasePath`

`BasePath` 속성으로 정의된 basepath에 영숫자가 아닌 문자를 사용할 수 있는지 여부를 나타냅니다. `True`로 설정하면 basepath에서 영숫자가 아닌 문자가 제거됩니다.

`BasePath` 속성과 함께 `NormalizeBasePath`을 사용합니다.

유형: 부울

필수 항목 여부: 아니요

기본값: True

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`OwnershipVerificationCertificateArn`

사용자 지정 도메인의 소유권을 확인하기 위해 ACM에서 발급한 공인 인증서의 ARN입니다. 상호 TLS를 구성하고 `CertificateArn`에 대해 ACM 가져오기 또는 사설 CA 인증서 ARN을 지정하는 경우에만 필요합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::ApiGateway::DomainName` 리소스의 `OwnershipVerificationCertificateArn` 속성으로 직접 전달됩니다.

`Policy`

API Gateway 도메인 이름에 연결할 IAM 정책입니다. `EndpointConfiguration`이(가) `PRIVATE`(으)로 설정된 경우에만 적용됩니다.

유형: Json

필수 항목 여부: 아니요

CloudFormation호환성: `EndpointConfiguration`이(가) `PRIVATE`(으)로 설정된 경우, 이 속성은 `AWS::ApiGateway::DomainNameV2` 리소스의 `Policy` 속성으로 직접 전달됩니다. 유효한 정책 문서의 예시는 AWS::ApiGateway::DomainNameV2를 참조하세요.

`Route53`

Amazon Route 53 구성을 정의합니다.

유형: Route53Configuration

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`SecurityPolicy`

이 도메인 이름에 대한 TLS 버전과 암호 그룹의 결합입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: `EndpointConfiguration`이(가) `PRIVATE`(으)로 설정된 경우, 이 속성은 `AWS::ApiGateway::DomainName` 리소스의 `SecurityPolicy` 속성으로 직접 전달되거나, `AWS::ApiGateway::DomainNameV2`(으)로 전달됩니다. `PRIVATE` 엔드포인트의 경우, TLS_1_2 만 지원됩니다.

예시

DomainName

DomainName 예제

YAML

```
Domain:
```

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

DomainAccessAssociation

액세스 연결 소스와 프라이빗 커스텀 도메인 이름 간의 도메인 액세스 연결을 구성합니다. 지원되는 유일한 액세스 연결 소스는 VPC 엔드포인트 ID입니다. 자세한 내용은 API 게이트웨이의 프라이빗 API의 사용자 지정 도메인 이름을 참조하세요.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
VpcEndpointId: String
```

속성

`VpcEndpointId`

API Gateway VPC 서비스와 연결된 VPC 인터페이스 엔드포인트의 엔드포인트 ID입니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `AWS::ApiGateway::DomainNameAccessAssociation` 리소스의 `AccessAssociationSource` 속성으로 직접 전달됩니다.

Route53Configuration

API에 대한 Route53 레코드 세트를 구성합니다.

구문

AWS Serverless Application Model (AWS SAM) 템플릿에서이 개체를 선언하려면 다음 구문을 사용합니다.

YAML

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

속성

`DistributionDomainName`

API 사용자 지정 도메인 이름의 사용자 지정 배포를 구성합니다.

유형: 문자열

필수 항목 여부: 아니요

기본값: API Gateway 배포를 사용합니다.

CloudFormation 호환성:이 속성은 `AWS::Route53::RecordSetGroup AliasTarget` 리소스의 DNSName 속성으로 직접 전달됩니다.

추가 참고 사항: CloudFront 배포의 도메인 이름입니다.

`EvaluateTargetHealth`

EvaluateTargetHealth가 true인 경우 별칭 레코드는 ELB 로드 밸런서 또는 호스팅 영역의 다른 레코드와 같이 참조된 AWS 리소스의 상태를 상속합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Route53::RecordSetGroup AliasTarget` 리소스의 [`EvaluateTargetHealth`] 속성으로 직접 전달됩니다.

추가 참고 사항: 별칭 대상이 CloudFront 배포인 경우, EvaluateTargetHealth를 true로 설정할 수 없습니다.

`HostedZoneId`

기록을 생성하려는 호스팅 영역의 ID입니다.

`HostedZoneName` 또는 `HostedZoneId` 중 하나를 지정하며 둘 다 지정하지 않습니다. 동일한 도메인 이름을 가진 호스팅 영역이 여러 개 있는 경우 `HostedZoneId`를 사용하여 호스팅 영역을 지정해야 합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Route53::RecordSetGroup RecordSet` 리소스의 [`HostedZoneId`] 속성으로 직접 전달됩니다.

`HostedZoneName`

기록을 생성하려는 호스팅 영역의 이름입니다.

`HostedZoneName` 또는 `HostedZoneId` 중 하나를 지정하며 둘 다 지정하지 않습니다. 동일한 도메인 이름을 가진 호스팅 영역이 여러 개 있는 경우 `HostedZoneId`를 사용하여 호스팅 영역을 지정해야 합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Route53::RecordSetGroup RecordSet` 리소스의 [`HostedZoneName`] 속성으로 직접 전달됩니다.

`IpV6`

이 속성이 설정되면는 `AWS::Route53::RecordSet` 리소스를 AWS SAM 생성하고 제공된 HostedZone에 AAAA 대해 [유형을] 로 설정합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은에 고유 AWS SAM 하며 CloudFormation 동등한 속성이 없습니다.

`Region`

지연 시간 기반 리소스 기록 세트만 해당: 이 리소스 기록 세트가 참조하는 리소스를 생성한 Amazon EC2 리전입니다. 리소스는 일반적으로 EC2 인스턴스 또는 ELB 로드 밸런서와 같은 AWS 리소스이며 레코드 유형에 따라 IP 주소 또는 DNS 도메인 이름으로 참조됩니다.

Amazon Route 53에서 지연 리소스 기록 세트에 대해 생성된 도메인 이름 및 유형에 대한 DNS 쿼리를 수신한 경우 Route 53은 최종 사용자와 연결된 Amazon EC2 리전 간에 지연 시간이 가장 짧은 지연 리소스 기록 세트를 선택합니다. 그 다음에 Route 53은 선택된 리소스 기록 세트와 연결된 값을 반환합니다.

다음 사항에 유의하세요.

- 지연 리소스 기록 세트별로 `ResourceRecord`를 1개씩만 지정할 수 있습니다.
- 각 Amazon EC2 리전에 지연 시간 리소스 기록 세트를 1개씩만 생성할 수 있습니다.
- 모든 Amazon EC2 리전에 지연 시간 리소스 기록 세트를 생성할 필요는 없습니다. Route 53은 지연 시간 리소스 기록 세트를 생성할 리전 중에서 지연 시간이 가장 좋은 리전을 선택합니다.
- `Name` 및 `Type` 요소 값이 지연 시간 리소스 기록 세트와 같은 비-지연 시간 리소스 기록 세트를 생성할 수 없습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Route53::RecordSetGroup` `RecordSet` 데이터 유형의 [`Region`] 속성으로 직접 전달됩니다.

`SetIdentifier`

단순하지 않은 라우팅 정책이 있는 리소스 기록 세트: 이름과 유형의 조합이 동일한 여러 리소스 기록 세트(이름이 acme.example.com이고 유형이 A인 여러 가중치 기반 리소스 기록 세트) 사이에서 차별화되는 식별자입니다. 이름과 유형이 동일한 리소스 기록 세트 그룹에서 각 리소스 기록 세트의 `SetIdentifier` 값은 고유해야 합니다.

라우팅 정책에 대한 자세한 내용을 알아보려면 Amazon Route 53 개발자 안내서의 [라우팅 정책 선택]을 참조하세요.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Route53::RecordSetGroup` `RecordSet` 데이터 유형의 [SetIdentifier](#) 속성으로 직접 전달됩니다.

`VpcEndpointDomainName`

API Gateway VPC 서비스와 연결된 VPC 인터페이스 엔드포인트의 DNS 이름입니다. 이 속성은 프라이빗 도메인에만 필요합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Route53::RecordSet` `AliasTarget` 필드의 [DNSName](#) 속성으로 직접 전달됩니다.

`VpcEndpointHostedZoneId`

API Gateway VPC 서비스와 연결된 VPC 인터페이스 엔드포인트의 호스팅된 영역 ID입니다. 이 속성은 프라이빗 도메인에만 필요합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Route53::RecordSet` `AliasTarget` 필드의 [HostedZoneId](#) 속성으로 직접 전달됩니다.

예제

기본 예제

이 예시에서는 API에 대한 사용자 지정 도메인과 Route 53 기록 세트를 구성합니다.

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

REST API의 엔드포인트 유형.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

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

속성

`Type`

REST API의 엔드포인트 유형.

유효한 값: `EDGE` 또는 `REGIONAL` 또는 `PRIVATE`

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [`Types`] `AWS::ApiGateway::RestApi` 데이터 유형의 `EndpointConfiguration` 속성에 직접 전달됩니다.

`VPCEndpointIds`

Route53 별칭을 생성할 REST API의 VPC 엔드포인트 ID 목록입니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [`VpcEndpointIds`] `AWS::ApiGateway::RestApi` 데이터 유형의 `EndpointConfiguration` 속성에 직접 전달됩니다.

예시

EndpointConfiguration

엔드포인트 구성 예제

YAML

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

AWS::Serverless::Application

AWS Serverless Application Repository로부터, 또는 Amazon S3 버킷으로부터 서버리스 애플리케이션을 중첩 애플리케이션으로 내장합니다. 중첩된 애플리케이션은 중첩된 리소스로 배포되며, 중첩된 AWS::CloudFormation::Stack리소스에는 다른 리소스를 비롯한 다른 AWS::Serverless::Application리소스가 포함될 수 있습니다.

> **Note**
>
> AWS CloudFormation에 배포하면 AWS SAM는 귀하의 AWS SAM 리소스를 CloudFormation 리소스로 변환됩니다. 자세한 내용은 AWS SAM용으로 생성된 CloudFormation 리소스 섹션을 참조하십시오.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

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

```
  Tags: Map
  TimeoutInMinutes: Integer
```

속성

Location

중첩된 애플리케이션의 템플릿 URL, 파일 경로 또는 위치 개체.

템플릿 URL을 제공하는 경우, 템플릿 URL은 클라우드포메이션 템플릿URL 설명서에 지정된 형식을 따라야 하며 유효한 클라우드포메이션또는 SAM 템플릿을 포함해야 합니다. ApplicationLocationObject는 AWS Serverless Application Repository에 게시된 애플리케이션을 지정하는 데 사용할 수 있습니다.

로컬 파일 경로를 제공하는 경우, 애플리케이션이 제대로 변환되려면 템플릿이 `sam deploy` 또는 `sam package` 명령을 포함하는 워크플로를 거쳐야 합니다.

유형: 문자열 | ApplicationLocationObject

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `AWS::CloudFormation::Stack` 리소스의 `TemplateURL` 속성과 유사합니다. CloudFormation 버전에서는 AWS Serverless Application Repository에서 애플리케이션을 인출하는 데 ApplicationLocationObject가 필요하지 않습니다.

NotificationARNs

스택 이벤트에 대한 알림을 전송하는 기존의 Amazon SNS 주제 목록.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::CloudFormation::Stack` 리소스의 `NotificationARNs` 속성으로 직접 전달됩니다.

Parameters

애플리케이션 파라미터 값

유형: 맵

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::CloudFormation::Stack` 리소스의 `Parameters` 속성으로 직접 전달됩니다.

`Tags`

이 응용 프로그램에 추가할 태그를 지정하는 맵(문자열 간)입니다. 키와 값에는 영숫자 문자만 사용할 수 있습니다. 키는 길이가 1~127자(유니코드 문자)이며 “aws:”로 시작할 수 없습니다. 값은 길이가 1~255자인 유니코드 문자입니다.

유형: 맵

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::CloudFormation::Stack` 리소스의 `Tags` 속성과 유사합니다. SAM의 태그 속성은 키:값의 쌍으로 구성되고, CloudFormation에서는 태그 객체 목록으로 구성됩니다. 스택이 생성되면 SAM은 이 애플리케이션에 `lambda:createdBy:SAM` 태그를 자동으로 추가합니다. 또한 이 애플리케이션이 AWS Serverless Application Repository의 애플리케이션인 경우 SAM은 두 개의 추가 `serverlessrepo:applicationId:`*`ApplicationId`* 및 `serverlessrepo:semanticVersion:`*`SemanticVersion`*태그도 자동으로 추가합니다.

`TimeoutInMinutes`

중첩 스택이 `CREATE_COMPLETE` 상태에 도달할 때까지 CloudFormation이 대기하는 시간(분)입니다. 기본값은 제한 시간 없음입니다. CloudFormation이 중첩 스택이 `CREATE_COMPLETE` 상태에 도달했음을 감지하면 상위 스택에서 해당 중첩 스택 리소스를 `CREATE_COMPLETE`로 표시하고 상위 스택 생성을 재개합니다. 중첩 스택이 `CREATE_COMPLETE` 상태에 도달하기 전에 제한 시간이 만료되면 CloudFormation이 해당 중첩 스택을 실패로 표시하고 중첩 스택과 상위 스택을 둘 다 롤백합니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::CloudFormation::Stack` 리소스의 `TimeoutInMinutes` 속성으로 직접 전달됩니다.

반환 값

Ref

`Ref` 내장 함수에 이 리소스의 논리적 ID를 제공하면 기저의 `AWS::CloudFormation::Stack` 리소스의 리소스 이름을 반환합니다.

Ref 함수의 사용에 대한 자세한 내용은 AWS CloudFormation 사용자 가이드의 Ref 섹션을 참조하세요.

Fn::GetAtt

Fn::GetAtt은 이 유형의 지정된 속성에 대한 값을 반환합니다. 다음은 사용 가능한 속성과 반환되는 샘플 값.

Fn::GetAtt의 사용에 대한 자세한 내용은 AWS CloudFormation 사용자 가이드의 Fn::GetAtt 섹션을 참조하세요.

Outputs.ApplicationOutputName

*ApplicationOutputName*라는 이름의 스택 출력 결과의 값.

예시

SAR 애플리케이션

서버리스 애플리케이션 리포지토리의 템플릿을 사용하는 애플리케이션

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
```

정상-애플리케이션

S3 URL을 통한 애플리케이션

YAML

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

ApplicationLocationObject

AWS Serverless Application Repository에 게시된 애플리케이션.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

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

속성

`ApplicationId`

애플리케이션의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`SemanticVersion`

애플리케이션의 의미 체계 버전입니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

애플리케이션

예제 애플리케이션 위치 객체

YAML

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

AWS::Serverless::CapacityProvider

고객 소유 Amazon Elastic Compute Cloud 인스턴스에서 Lambda 관리형 인스턴스를 실행할 수 있는 AWS Lambda 함수의 용량 공급자를 생성합니다. 이 리소스는 EC2 요금 모델을 활용하여 대규모 Lambda 워크로드에 대한 비용 최적화를 제공하는 Lambda 관리형 인스턴스 기능의 일부입니다.

용량 공급자는 EC2 인스턴스의 수명 주기를 관리하고 Lambda 함수가 서버리스 프로그래밍 모델을 유지하면서 고객 소유 컴퓨팅 리소스에서 실행하는 데 필요한 인프라를 제공합니다.

> **Note**
>
> 에 배포하면가 AWS SAM 리소스를 CloudFormation 리소스로 AWS CloudFormation AWS SAM 변환합니다. 자세한 내용은 AWS SAM용으로 생성된 CloudFormation 리소스 단원을 참조하십시오.

구문

AWS Serverless Application Model (AWS SAM) 템플릿에서이 개체를 선언하려면 다음 구문을 사용합니다.

YAML

```
Type: AWS::Serverless::CapacityProvider
Properties:
  CapacityProviderName: String
  VpcConfig: VpcConfig
  OperatorRole: String
  Tags: Map
  PropagateTags: Boolean
  InstanceRequirements: InstanceRequirements
  ScalingConfig: ScalingConfig
  KMSKeyArn: String
```

속성

`CapacityProviderName`

용량 공급자의 이름입니다. 이 이름은 AWS 계정 및 리전 내에서 고유해야 합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::CapacityProvider` 리소스의 `CapacityProviderName` 속성과 유사합니다.

`VpcConfig`

용량 공급자의 VPC 구성입니다. Amazon EC2 인스턴스가 시작될 VPC 서브넷 및 보안 그룹을 지정합니다.

유형: [VpcConfig]

필수 항목 여부: 예

CloudFormation 호환성:이 속성은 `AWS::Lambda::CapacityProvider` 리소스의 [`VpcConfig`] 속성으로 직접 전달됩니다.

`OperatorRole`

고객 계정에서 Amazon EC2 인스턴스 및 관련 리소스를 생성하고 관리할 수 있는 권한이 있는 Lambda에 대한 연산자 역할의 ARN입니다. 제공되지 않은 경우는 필요한 권한이 있는 역할을 AWS SAM 자동으로 생성합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::CapacityProvider` 리소스의 [`PermissionsConfig`] 속성으로 직접 전달됩니다.

`Tags`

용량 공급자 및 관련 리소스에 적용할 키-값 페어의 맵입니다.

유형: 맵

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 AWS::Lambda::CapacityProvider 리소스의 Tags 속성으로 직접 전달됩니다. 의 Tags 속성은 키-값 페어로 AWS SAM 구성됩니다(이 CloudFormation 속성에서는 태그 객체 목록으로 구성됨). 또한는이 Lambda 함수와이 함수에 대해 생성된 기본 역할에 lambda:createdBy:SAM 태그를 AWS SAM 자동으로 추가합니다.

PropagateTags

태그 속성에서 AWS::Serverless::CapacityProvider 생성된 리소스로 태그를 전달할지 여부를 나타냅니다. 생성된 리소스에 태그를 전파True하려면이 값을 로 설정합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은에 고유 AWS SAM 하며 CloudFormation 동등한 속성이 없습니다.

InstanceRequirements

용량 공급자가 사용할 수 있는 컴퓨팅 인스턴스 유형에 대한 사양입니다. 여기에는 아키텍처 요구 사항 및 allowed 또는 excluded 인스턴스 유형이 포함됩니다.

유형: InstanceRequirements

필수 항목 여부: 아니요

CloudFormation compatibility:maps이 속성을에 AWS SAM 매핑합니다InstanceRequirements.

ScalingConfig

용량 공급자의 조정 구성입니다. 용량 공급자가 수요에 따라 Amazon EC2 인스턴스를 확장하는 방법을 정의합니다.

Type: ScalingConfig

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 AWS::Lambda::CapacityProvider 리소스의 CapacityProviderScalingConfig 속성으로 직접 전달됩니다.

KMSKeyArn

용량 공급자의 저장 및 전송 중 데이터를 암호화하는 데 사용되는 AWS KMS 키의 ARN입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 `KmsKeyArn` 속성으로 직접 전달됩니다.

반환 값

Ref

이 리소스의 논리적 ID가 `Ref` 내장 함수에 제공되면 용량 공급자의 이름을 반환합니다.

`Ref` 함수의 사용에 대한 자세한 내용은 AWS CloudFormation 사용자 가이드의 `Ref` 섹션을 참조하세요.

Fn::GetAtt

`Fn::GetAtt`은 이 유형의 지정된 속성에 대한 값을 반환합니다. 다음은 사용 가능한 속성과 반환되는 샘플 값.

`Fn::GetAtt`의 사용에 대한 자세한 내용은 AWS CloudFormation 사용자 가이드의 `Fn::GetAtt` 섹션을 참조하세요.

`Arn`

용량 공급자의 ARN입니다.

예제

기본 용량 공급자

다음 예제에서는 VPC 구성을 사용하여 기본 용량 공급자를 생성합니다.

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

```
    Tags:
      Environment: Production
      Team: ServerlessTeam
```

확장 기능이 있는 고급 용량 공급자

다음 예시에서는 사용자 지정 인스턴스 요구 사항 및 조정 구성을 사용하여 용량 공급자를 생성합니다.

```
AdvancedCapacityProvider:
  Type: AWS::Serverless::CapacityProvider
  Properties:
    CapacityProviderName: advanced-capacity-provider
    VpcConfig:
      SubnetIds:
        - subnet-12345678
        - subnet-87654321
      SecurityGroupIds:
        - sg-12345678
    OperatorRole: arn:aws:iam::123456789012:role/MyCapacityProviderRole
    PropagateTags: true
    InstanceRequirements:
      Architectures:
        - x86_64
      ExcludedTypes:
        - t2.micro
    ScalingConfig:
      MaxInstanceCount: 10
      ManualScalingPolicies:
        AverageCPUUtilization: 70.0
    KMSKeyArn: arn:aws:kms:us-
east-1:123456789012:key/12345678-1234-1234-1234-123456789012
    Tags:
      Environment: Production
      CostCenter: Engineering
```

VpcConfig

EC2 인스턴스가 시작될 서브넷 및 보안 그룹을 포함하여 용량 공급자에 대한 VPC 설정을 구성합니다.

구문

AWS Serverless Application Model (AWS SAM) 템플릿에서이 개체를 선언하려면 다음 구문을 사용합니다.

YAML

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

속성

SubnetIds

EC2 인스턴스가 시작될 서브넷 IDs 목록입니다. 하나 이상의 서브넷을 지정해야 합니다.

유형: 목록

필수 항목 여부: 예

CloudFormation 호환성:이 속성은 `AWS::Lambda::CapacityProvider` 리소스의 `SubnetIds` 속성으로 직접 전달됩니다.

SecurityGroupIds

EC2 인스턴스와 연결할 보안 그룹 IDs의 목록입니다. 지정하지 않으면 VPC의 기본 보안 그룹이 사용됩니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::CapacityProvider` 리소스의 `SecurityGroupIds` 속성으로 직접 전달됩니다.

예제

VPC 구성

다음 예제에서는 여러 서브넷과 보안 그룹이 있는 VPC 구성을 보여줍니다.

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

InstanceRequirements

아키텍처 및 인스턴스 유형 제약 조건을 포함하여 용량 공급자가 시작할 EC2 인스턴스에 대한 요구 사항을 지정합니다.

구문

AWS Serverless Application Model (AWS SAM) 템플릿에서이 개체를 선언하려면 다음 구문을 사용합니다.

YAML

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

> **Note**
>
> 용량 공급자에 대한 인스턴스 요구 사항을 정의할 ExcludedTypes 때 AllowedTypes 또는 중 하나를 지정할 수 있지만 둘 다 지정할 수는 없습니다.

속성

`Architectures`

용량 공급자 인스턴스의 명령 세트 아키텍처입니다.

유효한 값: x86_64 또는 arm64

유형: 목록

필수 항목 여부: 아니요

기본값: x86_64

CloudFormation 호환성:이 속성은 AWS::Lambda::CapacityProvider 리소스의 Architectures 속성으로 직접 전달됩니다.

`AllowedTypes`

용량 공급자 인스턴스에 허용되는 EC2 인스턴스 유형의 목록입니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::CapacityProvider` 리소스의 `AllowedInstanceTypes` 속성으로 직접 전달됩니다.

`ExcludedTypes`

용량 공급자에서 제외할 EC2 인스턴스 유형의 목록입니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::CapacityProvider` 리소스의 `ExcludedInstanceTypes` 속성으로 직접 전달됩니다.

예제

인스턴스 요구 사항 구성

다음 예제에서는 특정 아키텍처 및 인스턴스 유형 제약 조건이 있는 인스턴스 요구 사항을 보여줍니다.

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

ScalingConfig

용량 공급자가 최대 인스턴스 제한 및 조정 정책을 포함하여 수요에 따라 EC2 인스턴스를 조정하는 방법을 구성합니다.

구문

AWS Serverless Application Model (AWS SAM) 템플릿에서이 개체를 선언하려면 다음 구문을 사용합니다.

YAML

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

속성

`MaxVCpuCount`

용량 공급자가 모든 컴퓨팅 인스턴스에 프로비저닝할 수 있는 최대 vCPUs.

유형: 정수

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::CapacityProvider` 리소스의 [`MaxVCpuCount`] 속성으로 직접 전달됩니다.

`AverageCPUUtilization`

조정 결정을 위한 목표 평균 CPU 사용률(0~100)입니다. 평균 CPU 사용률이이 임계값을 초과하면 용량 공급자는 Amazon EC2 인스턴스를 확장합니다.

유형: Double

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::CapacityProvider` 리소스의 [`ScalingPolicies`] 속성으로 직접 전달됩니다.

예제

조정 구성

다음 예제는 최대 VCpu 수와 평균 CPU 사용률을 가진 조정 구성을 보여줍니다.

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

AWS::Serverless::Connector

두 리소스 간의 권한을 구성합니다. 커넥터에 대한 소개는 AWS SAM 커넥터를 사용한 리소스 권한 관리 섹션을 참조하세요.

생성된 AWS CloudFormation 리소스에 대한 자세한 내용은 CloudFormation을 지정한 경우 생성되는 AWS::Serverless::Connector 리소스 섹션을 참조하세요.

커넥터에 대한 피드백을 제공하려면 서버리스 애플리케이션 모델AWSGitHub 리포지토리에 새 문제를 제출하십시오.

> **Note**
>
> AWS CloudFormation에 배포하면 AWS SAM는 귀하의 AWS SAM 리소스를 CloudFormation 리소스로 변환됩니다. 자세한 내용은 AWS SAM용으로 생성된 CloudFormation 리소스 섹션을 참조하십시오.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 구문을 사용합니다.

> **Note**
>
> 대부분의 사용 사례에서는 내장 커넥터 구문을 사용하는 것이 좋습니다. 소스 리소스에 내장되어 있으면 시간이 지나도 읽고 유지 관리하기가 더 쉬워집니다. 중첩된 스택의 리소스 또는 공유 리소스와 같이 동일한 AWS SAM 템플릿 내에 있지 않은 소스 리소스를 참조해야 하는 경우 이러한 `AWS::Serverless::Connector` 구문을 사용합니다.

내장 커넥터

```
<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
```

속성

`Destination`

대상 리소스.

유형: 리소스 참조 | 리소스 참조 목록

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Permissions`

원본 리소스가 대상 리소스에서 수행할 수 있는 권한 유형입니다.

`Read`은 리소스에서 데이터를 읽을 수 있는 AWS Identity and Access Management(IAM) 작업을 포함하고 있습니다.

`Write`은 리소스에 데이터를 시작하고 쓸 수 있는 IAM 작업을 포함합니다.

유효한 값: `Read` 또는 `Write`

유형: 목록

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Source`

소스 리소스. `AWS::Serverless::Connector` 명령문을 사용할 때 필요합니다.

유형: 리SourceReference

필수 여부: 조건부

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`SourceReference`

소스 리소스.

> **Note**
>
> 소스 리소스의 추가 속성을 정의할 때 내장된 커넥터 명령문과 함께 사용합니다.

유형: SourceReference

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

내장 커넥터

다음 예제에서는 내장 커넥터를 사용하여 AWS Lambda 함수와 Amazon DynamoDB 테이블 간의 `Write` 데이터 연결을 정의합니다.

```
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
    ...
```

다음 예제에서는 내장 커넥터를 사용하여 `Read`을 정의하고 권한을 `Write`합니다.

```
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
    ...
```

다음 예제에서는 내장된 커넥터를 사용하여 Id이 아닌 속성으로 소스 리소스를 정의합니다.

```
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

다음 예제에서는 AWS::Serverless::Connector 리소스를 사용하여 Amazon DynamoDB 테이블에서 AWS Lambda함수를 읽고 쓰도록 합니다.

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

```
    Source:
      Id: MyFunction
    Destination:
      Id: MyTable
    Permissions:
      - Read
      - Write
```

다음 예제에서는 AWS::Serverless::Connector 리소스를 사용하여 Lambda 함수가 Amazon SNS 주제에 쓰도록 하고 두 리소스 모두 동일한 템플릿에서 이를 실행합니다.

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

다음 예제에서는 AWS::Serverless::Connector 리소스를 사용하여 Amazon SNS 주제가 Lambda 함수에 쓰도록 하고, 그 다음 Amazon DynamoDB 테이블에 쓰도록 하며, 모든 리소스가 동일한 템플릿에서 이를 실행합니다.

```
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
```

다음은 위 예제에서 변환된 AWS CloudFormation 템플릿입니다.

```
"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

AWS::Serverless::Connector 리소스 유형이 사용하는 리소스에 대한 참조.

> **Note**
>
> 동일한 템플릿에 있는 리소스의 경우 `Id`를 제공하십시오. 동일한 템플릿에 있지 않은 리소스의 경우 다른 속성을 조합하여 사용합니다. 자세한 내용은을 참조하십시오AWS SAM 커넥터 참조

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

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

속성

`Arn`

리소스의 ARN.

유형: 문자열

필수 항목 여부: 조건부

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Id`

동일한 템플릿에 있는 리소스의 논리적 ID.

> **Note**
>
> `Id`이 지정된 경우, 커넥터가 AWS Identity and Access Management(IAM) 정책을 생성하면 해당 정책과 관련된 IAM 역할이 리소스`Id`에서 유추됩니다. `Id`이 지정되지 않은 경우, 커넥터가 생성된 IAM 정책을 IAM 역할에 연결할 수 있도록 리소스의 `RoleName`를 제공하십시오.

유형: 문자열

필수 항목 여부: 조건부

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Name`

리소스의 이름.

유형: 문자열

필수 항목 여부: 조건부

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Qualifier`

범위를 좁히는 리소스의 한정자입니다. `Qualifier`는 리소스 제약 조건 ARN의 끝에 있는 * 값을 대체합니다. 예시는 API Gateway가 Lambda 함수를 호출에서 확인하십시오.

> **Note**
> 한정자 정의는 리소스 유형별로 다릅니다. 지원되는 소스 및 대상 리소스 유형 목록은 AWS SAM 커넥터 참조 섹션을 참조하세요.

유형: 문자열

필수 항목 여부: 조건부

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`QueueUrl`

Amazon SQS 대기열의 URL. 이 속성은 Amazon SQS 리소스에만 적용됩니다.

유형: 문자열

필수 항목 여부: 조건부

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`ResourceId`

리소스의 ID. 에를 들면, API 게이트웨이 API ID.

유형: 문자열

필수 항목 여부: 조건부

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`RoleName`

리소스와 관련된 역할 이름.

> **Note**
>
> Id이 지정된 경우, 커넥터가 IAM 정책을 생성하면 해당 정책에 연결된 IAM 역할이 리소스 Id에서 유추됩니다. Id이 지정되지 않은 경우, 커넥터가 생성된 IAM 정책을 IAM 역할에 연결할 수 있도록 리소스의 RoleName를 제공하십시오.

유형: 문자열

필수 항목 여부: 조건부

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

Type

리소스의 CloudFormation 유형. 자세한 내용은 AWS리소스 및 속성 유형 참조를 참조하세요.

유형: 문자열

필수 항목 여부: 조건부

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

API Gateway가 Lambda 함수를 호출

다음 예제에서는 AWS::Serverless::Connector 리소스를 사용하여 Amazon API Gateway에서 AWS Lambda 함수를 호출할 수 있도록 허용합니다.

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

AWS::Serverless::Connector 리소스 유형이 사용하는 소스 리소스에 대한 참조입니다.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
Qualifier: String
```

속성

`Qualifier`

범위를 좁히는 리소스의 한정자입니다. `Qualifier`는 리소스 제약 조건 ARN의 끝에 있는 `*` 값을 대체합니다.

> **Note**
>
> 한정자 정의는 리소스 유형별로 다릅니다. 지원되는 소스 및 대상 리소스 유형 목록은 [AWS SAM 커넥터 참조](#) 섹션을 참조하세요.

유형: 문자열

필수 항목 여부: 조건부

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

다음 예제에서는 내장된 커넥터를 사용하여 **`Id`**이 아닌 속성으로 소스 리소스를 정의합니다.

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

AWS::Serverless::Function

AWS Lambda 함수, AWS Identity and Access Management (IAM) 실행 역할 및 함수를 트리거하는 이벤트 소스 매핑을 생성합니다.

AWS::Serverless::Function 리소스는 `Metadata` 리소스 속성도 지원하므로 애플리케이션에 필요한 사용자 지정 런타임을 빌드 AWS SAM 하도록에 지시할 수 있습니다. 사용자 지정 런타임을 구축하는 방법에 대한 자세한 정보는 AWS SAM에서 사용자 지정 런타임을 사용하여 Lambda 함수 빌드 섹션을 참조하세요.

> **Note**
>
> 에 배포하면가 AWS SAM 리소스를 CloudFormation 리소스로 AWS CloudFormation AWS SAM 변환합니다. 자세한 내용은 AWS SAM용으로 생성된 CloudFormation 리소스 단원을 참조하십시오.

구문

AWS Serverless Application Model (AWS SAM) 템플릿에서이 개체를 선언하려면 다음 구문을 사용합니다.

YAML

```
Type: AWS::Serverless::Function
Properties:
  Architectures: List
  AssumeRolePolicyDocument: JSON
  AutoPublishAlias: String
  AutoPublishAliasAllProperties: Boolean
  AutoPublishCodeSha256: String
  CapacityProviderConfig: CapacityProviderConfig
  CodeSigningConfigArn: String
  CodeUri: String | FunctionCode
  DeadLetterQueue: Map | DeadLetterQueue
  DeploymentPreference: DeploymentPreference
  Description: String
  DurableConfig: DurableConfig
  Environment: Environment
  EphemeralStorage: EphemeralStorage
  EventInvokeConfig: EventInvokeConfiguration
```

```
  Events: EventSource
  FileSystemConfigs: List
  FunctionName: String
  FunctionScalingConfig: FunctionScalingConfig
  FunctionUrlConfig: FunctionUrlConfig
  Handler: String
  ImageConfig: ImageConfig
  ImageUri: String
  InlineCode: String
  KmsKeyArn: String
  Layers: List
  LoggingConfig: LoggingConfig
  MemorySize: Integer
  PackageType: String
  PermissionsBoundary: String
  Policies: String | List | Map
  PublishToLatestPublished: Boolean
  PropagateTags: Boolean
  ProvisionedConcurrencyConfig: ProvisionedConcurrencyConfig
  RecursiveLoop: String
  ReservedConcurrentExecutions: Integer
  Role: String
  RolePath: String
  Runtime: String
  RuntimeManagementConfig: RuntimeManagementConfig
  SnapStart: SnapStart
  SourceKMSKeyArn: String
  Tags: Map
  TenancyConfig: TenancyConfig
  Timeout: Integer
  Tracing: String
  VersionDescription: String
  VersionDeletionPolicy: String
  VpcConfig: VpcConfig
```

속성

`Architectures`

함수의 명령 세트 아키텍처입니다.

이 속성에 대한 자세한 내용은 AWS Lambda 개발자 안내서의 Lambda 명령 세트 아키텍처를 참조하십시오.

유효한 값: `x86_64` 혹은 `arm64` 중 하나

유형: 목록

필수 항목 여부: 아니요

기본값: `x86_64`

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 [`Architectures`] 속성으로 직접 전달됩니다.

`AssumeRolePolicyDocument`

이 함수에 대해 생성된 기본값에 대한 `Role`AssumeRolePolicyDocument를 추가합니다. 이 속성이 지정되지 않은 경우이 함수에 대한 기본 수임 역할을 AWS SAM 추가합니다.

유형: JSON

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::IAM::Role` 리소스의 [`AssumeRolePolicyDocument`] 속성과 유사합니다.는이 속성을이 함수에 대해 생성된 IAM 역할에 AWS SAM 추가합니다. 역할의 Amazon 리소스 이름(ARN)이 이 함수에 대해 제공된 경우 이 속성은 아무 작업도 수행하지 않습니다.

`AutoPublishAlias`

Lambda 별칭의 이름. Lambda의 별칭에 대한 자세한 내용은 AWS Lambda 개발자 안내서의 [Lambda 함수 별칭]을 참조하세요. 이 속성을 사용하는 예제는 [AWS SAM에서 서버리스 애플리케이션의 점진적 배포] 섹션을 참조하세요.

AWS SAM 는이 속성이 설정되면 [AWS::Lambda::Version] 및 [AWS::Lambda::Alias] 리소스를 생성합니다. 이 시나리오에 대한 자세한 내용은 [AutoPublishalias 속성 지정됨] 섹션을 참조하세요. 생성된 CloudFormation 리소스에 대한 일반적인 내용은 섹션을 참조하세요[AWS SAM용으로 생성된 CloudFormation 리소스].

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은에 고유 AWS SAM 하며 CloudFormation 동등한 속성이 없습니다.

`AutoPublishAliasAllProperties`

새 [AWS::Lambda::Version](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-lambda-version.html)이 생성되는 시기를 지정합니다. `true`의 경우, Lambda 함수의 속성이 변경되면 새 Lambda 버전이 생성됩니다. `false`의 경우, 다음 속성 중 하나가 변경된 경우에만 새 Lambda 버전이 생성됩니다.

- `Environment`, `MemorySize`, 또는 `SnapStart`.
- `Code` 속성의 업데이트를 초래하는 일체의 변경(예:`CodeDict`, `ImageUri`, 혹은 `InlineCode`).

이 속성은 `AutoPublishAlias`이 정의될 것을 요구합니다.

만약 `AutoPublishCodeSha256`도 지정되면, 그것의 행위는 `AutoPublishAliasAllProperties: true`에 우선합니다.

유형: 부울

필수 항목 여부: 아니요

기본값: `false`

CloudFormation 호환성:이 속성은에 고유 AWS SAM 하며 CloudFormation 동등한 속성이 없습니다.

`AutoPublishCodeSha256`

이 문자열을 사용할 경우 `CodeUri` 값과 함께 새 Lambda 버전을 게시해야 하는지 여부를 결정하는 데 사용됩니다. Amazon S3 위치에 저장된 배포 패키지가 업데이트된 Lambda 함수 코드가 포함된 새 배포 패키지로 교체되지만, `CodeUri` 속성은 변경되지 않은 상태로 유지되는 경우(새 배포 패키지가 새 Amazon S3 위치에 업로드되고 `CodeUri`가 새 위치로 변경되는 경우와 반대) 발생할 수 있는 배포 문제를 해결하기 위해 이 속성을 주로 사용합니다.

이 문제는 다음과 같은 특성을 가진 AWS SAM 템플릿으로 표시됩니다.

- `DeploymentPreference` 객체가 점진적 배포를 위해 구성됩니다([AWS SAM에서 서버리스 애플리케이션의 점진적 배포](#)에 설명되어 있음).
- `AutoPublishAlias` 속성이 설정되어 있으며 배포 간에 변경되지 않습니다.
- `CodeUri` 속성이 설정되어 있으며 배포 간에 변경되지 않습니다.

이 시나리오에서 `AutoPublishCodeSha256`을 업데이트하면 새 Lambda 버전이 성공적으로 생성됩니다. 하지만 Amazon S3에 배포된 새 함수 코드는 인식되지 않습니다. 새 함수 코드를 인식하려면 Amazon S3 버킷에서 버전 관리를 사용하는 것이 좋습니다. Lambda 함수의 `Version` 속성을 지정하고 항상 최신 배포 패키지를 사용하도록 버킷을 구성합니다.

이 시나리오에서 점진적 배포를 성공적으로 트리거하려면 `AutoPublishCodeSha256`에 대한 고유한 값을 제공해야 합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은에 고유 AWS SAM 하며 CloudFormation 동등한 속성이 없습니다.

`CapacityProviderConfig`

함수의 게시된 버전을 연결할 용량 공급자를 구성합니다. 이렇게 하면 Lambda 관리형 인스턴스에서 관리하는 고객 소유 EC2 인스턴스에서 함수를 실행할 수 있습니다.

유형: CapacityProviderConfig

필수 항목 여부: 아니요

CloudFormation 호환성: SAM은 `AWS::Lambda::Function` 리소스의 속성에 전달된 `CapacityProviderConfig` 속성을 평면화하고 중첩된 구조를 재구성합니다.

`CodeSigningConfigArn`

이 함수에 대해 코드 서명을 활성화하는 데 사용되는 AWS::Lambda::CodeSigningConfig리소스의 ARN. 코드 서명에 대한 자세한 내용은 AWS SAM 애플리케이션에 대한 코드 서명 설정 섹션을 참조하세요.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 `CodeSigningConfigArn` 속성으로 직접 전달됩니다.

`CodeUri`

함수의 코드입니다. 허용 가능한 값은 다음을 포함합니다.

- 함수의 Amazon S3 URI. 예를 들어 `s3://bucket-123456789/sam-app/1234567890abcdefg`입니다.
- 함수의 로컬 경로. 예를 들어 `hello_world/`입니다.
- FunctionCode 객체입니다.

> **Note**
>
> 함수의 Amazon S3 URI 또는 FunctionCode 객체를 제공하는 경우 유효한 Lambda 배포 패키지를 참조해야 합니다.
> 로컬 파일 경로를 제공하는 경우 배포 시 AWS SAMCLI를 사용하여 로컬 파일을 업로드하십시오. 자세한 내용은 배포 시 AWS SAM이 로컬 파일을 업로드하는 방법를 참조하세요.
> `CodeUri` 속성에서 내장 함수를 사용하는 경우 AWS SAM 는 값을 올바르게 구문 분석할 수 없습니다. 대신 AWS::LanguageExtensions transform을 사용하는 것이 좋습니다.

유형: [문자열 | FunctionCode]

필수 항목 여부: 조건부. `PackageType`이 `Zip`로 설정된 경우, `CodeUri` 또는 중 `InlineCode` 하나가 필요합니다.

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 `Code` 속성과 유사합니다. 중첩된 Amazon S3 속성은 다르게 지정됩니다.

`DeadLetterQueue`

Lambda가 처리할 수 없는 이벤트를 전송하는 Amazon Simple Notification Service(Amazon SNS) 주제 또는 Amazon Simple Queue Service(SQS) 대기열을 구성합니다. DLQ(Dead Letter Queue) 기능에 대한 자세한 내용은 AWS Lambda 개발자 안내서의 DLQ(Dead Letter Queue)를 참조하세요.

> **Note**
>
> 귀하의 Lambda 함수의 이벤트 소스가 Amazon SQS 대기열인 경우, Lambda 함수가 아닌 소스 대기열에 대하여 DLQ(Dead Letter Queue) 대기열을 구성합니다. 함수에 구성하는 배달하지 못한 편지 대기열은 이벤트 소스 대기열이 아닌 함수의 비동기식 간접 호출 대기열에 사용됩니다.

유형: 맵 | DLQ

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 `DeadLetterConfig` 속성과 유사합니다. CloudFormation 에서 유형은에서 파생되는 반면`TargetArn`, 에서는 AWS SAM와 함께 유형을 전달해야 합니다`TargetArn`.

`DeploymentPreference`

점진적 Lambda 배포를 가능하게 하는 설정.

`DeploymentPreference` 객체가 지정되면 AWS::CodeDeploy::Application 호출된 `ServerlessDeploymentApplication` (스택당 하나), AWS::CodeDeploy::DeploymentGroup 호출된 및 호출AWS::IAM::Role된 *<function-logical-id>*`DeploymentGroup`를 AWS SAM 생성합니다`CodeDeployServiceRole`.

유형: DeploymentPreference

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은에 고유 AWS SAM 하며 CloudFormation 동등한 속성이 없습니다.

추가 참조: 이 속성에 대한 자세한 내용은 AWS SAM에서 서버리스 애플리케이션의 점진적 배포 섹션을 참조하세요.

`Description`

함수에 대한 설명입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 `Description` 속성으로 직접 전달됩니다.

`DurableConfig`

내구성 함수에 대한 구성입니다. 자동 체크포인트 및 재생 기능을 사용하여 상태 저장 실행을 활성화합니다.

유형: DurableConfig

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은에 고유 AWS SAM 하며 CloudFormation 동등한 속성이 없습니다.

`Environment`

런타임 환경에 대한 구성.

유형: Environment

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 `Environment` 속성으로 직접 전달됩니다.

`EphemeralStorage`

`/tmp`의 Lambda 함수에서 사용할 수 있는 디스크 공간 (MB)을 지정하는 객체.

실행 역할에 대한 자세한 내용은 AWS Lambda 개발자 가이드의 Lambda 실행 역할을 참조하세요.

유형: EphemeralStorage

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 `EphemeralStorage` 속성으로 직접 전달됩니다.

`EventInvokeConfig`

Lambda 함수의 이벤트 간접 호출 구성을 설명하는 객체입니다.

유형: EventInvokeConfiguration

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은에 고유 AWS SAM 하며 CloudFormation 동등한 속성이 없습니다.

`Events`

이 함수를 트리거하는 이벤트를 지정합니다. 이벤트는 유형 및 각 유형에 따라 달라지는 속성 집합으로 구성됩니다.

유형: EventSource

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은에 고유 AWS SAM 하며 CloudFormation 동등한 속성이 없습니다.

`FileSystemConfigs`

Amazon Elastic File System(Amazon EFS) 파일 시스템에 대한 연결 설정을 지정하는 FileSystemConfig 객체 목록입니다.

귀하의 템플릿이 AWS::EFS::MountTarget 리소스를 포함하는 경우, 함수 전에 마운트 대상이 생성되거나 업데이트되도록 `DependsOn` 리소스 속성도 지정해야 합니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 `FileSystemConfigs` 속성으로 직접 전달됩니다.

`FunctionName`

함수의 이름. 이름을 지정하지 않으면 고유한 이름 하나가 귀하를 위해 생성됩니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 `FunctionName` 속성으로 직접 전달됩니다.

`FunctionScalingConfig`

용량 공급자에서 실행되는 Lambda 함수의 조정 동작을 구성합니다. 최소 및 최대 실행 환경 수를 정의합니다.

유형: FunctionScalingConfig

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 `FunctionScalingConfig` 속성으로 직접 전달됩니다.

`FunctionUrlConfig`

함수 URL을 설명하는 객체입니다. 함수 URL은 귀하의 함수를 간접 호출하는 데 사용할 수 있는 HTTP(S) 엔드포인트입니다.

자세한 내용은 AWS Lambda 개발자 안내서의 Function URLs를 참조하세요.

유형: FunctionUrlConfig

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은에 고유 AWS SAM 하며 CloudFormation 동등한 속성이 없습니다.

`Handler`

코드 내에서 실행을 시작하기 위해 직접 호출되는 함수입니다. 이 속성은 `PackageType` 속성이 `Zip`로 설정된 경우에만 필요합니다.

유형: 문자열

필수 항목 여부: 조건부

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 `Handler` 속성으로 직접 전달됩니다.

`ImageConfig`

Lambda 컨테이너 이미지 설정을 구성하는 데 사용되는 객체입니다. 자세한 내용은 AWS Lambda 개발자 가이드에서 Lambda로 컨테이너 이미지 사용하기를 참조하세요.

유형: ImageConfig

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 `ImageConfig` 속성으로 직접 전달됩니다.

`ImageUri`

Lambda 함수의 컨테이너 이미지에 대한 Amazon Elastic Container Registry(Amazon ECR) 리포지토리의 URI입니다. 이 속성은 `PackageType` 속성이 `Image`로 설정된 경우에만 적용되며, 그렇지 않으면 무시됩니다. 자세한 내용은 AWS Lambda 개발자 가이드에서 Lambda로 컨테이너 이미지 사용하기를 참조하세요.

> **Note**
>
> `PackageType` 속성이 로 설정된 경우 `ImageImageUri`가 필요하거나 AWS SAM 템플릿 파일에 필요한 `Metadata` 항목을 사용하여 애플리케이션을 빌드해야 합니다. 자세한 내용은 AWS SAM을 사용한 기본 빌드 단원을 참조하십시오.

필요한 `Metadata` 항목을 사용하여 애플리케이션을 빌드하는 것이 `ImageUri`에 우선하므로 둘 다 지정하면 `ImageUri`은 무시됩니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` `Code` 데이터 유형의 [`ImageUri`] 속성으로 직접 전달됩니다.

`InlineCode`

템플릿에 직접 작성된 Lambda 함수 코드입니다. 이 속성은 `PackageType` 속성이 `Zip`로 설정된 경우에만 적용되며, 그렇지 않으면 무시됩니다.

> **Note**
>
> `PackageType` 속성이 `Zip`(기본값)으로 설정된 경우 `CodeUri` 또는 `InlineCode` 중 하나가 요구됩니다.

유형: 문자열

필수 항목 여부: 조건부

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` `Code` 데이터 유형의 [`ZipFile`] 속성으로 직접 전달됩니다.

`KmsKeyArn`

Lambda가 함수의 환경 변수를 암호화하고 해독하는 데 사용하는 AWS Key Management Service (AWS KMS) 키의 ARN입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 [`KmsKeyArn`] 속성으로 직접 전달됩니다.

`Layers`

이 함수가 사용해야 하는 `LayerVersion` ARN 목록. 여기에 지정된 순서는 Lambda 함수를 실행할 때 들여오는 순서입니다. 버전은 버전이 포함된 전체 ARN이거나 LayerVersion 리소스에 대한 참조입니다. 예를 들어, `LayerVersion`에 대한 참조는 `!Ref MyLayer`일 것이고, 버전이 포함된

전체 ARN은 `arn:aws:lambda:`*`region`*`:`*`account-id`*`:layer:`*`layer-name`*`:`*`version`*일 것입니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 `Layers` 속성으로 직접 전달됩니다.

`LoggingConfig`

함수의 Amazon CloudWatch Logs 구성입니다.

Type: LoggingConfig

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 `LoggingConfig` 속성으로 직접 전달됩니다.

`MemorySize`

함수의 간접 호출당 할당된 메모리 크기(MB)입니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 `MemorySize` 속성으로 직접 전달됩니다.

`PackageType`

Lambda 함수의 배포 패키지 유형. 자세한 내용은 AWS Lambda 개발자 안내서의 Lambda 배포 패키지를 참조하세요.

참고:

1. 이 속성을 `Zip`(기본값)으로 설정하면 `CodeUri` 또는 `InlineCode`이 적용되며 `ImageUri`는 무시됩니다.

2. 이 속성을 `Image`로 설정하면 `ImageUri` 속성만 적용되고 `CodeUri` 및 `InlineCode`는 둘 다 무시됩니다. 함수의 컨테이너 이미지를 저장하는 데 필요한 Amazon ECR 리포지토리는에서 자동으로 생성할 수 있습니다 AWS SAMCLI. 자세한 내용은을 참조하십시오sam deploy.

유효한 값: `Zip` 또는 `Image`

유형: 문자열

필수 항목 여부: 아니요

기본값: `Zip`

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 `PackageType` 속성으로 직접 전달됩니다.

`PermissionsBoundary`

이 함수의 실행 역할에 사용할 권한 경계의 ARN입니다. 이 속성은 역할이 자동으로 생성된 경우에만 작동합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::IAM::Role` 리소스의 `PermissionsBoundary` 속성으로 직접 전달됩니다.

`Policies`

이 함수의 권한 정책. 정책은 함수의 기본 AWS Identity and Access Management (IAM) 실행 역할에 추가됩니다.

이 속성은 단일 값 또는 값 목록을 허용합니다. 허용되는 값은 다음과 같습니다.

- AWS SAM 정책 템플릿.
- AWS 관리형 정책 또는 고객 관리형 정책의 ARN.
- 다음 목록에 있는 AWS 관리형 정책의 이름입니다.
- 맵 형식의 YAML 인라인 IAM 정책입니다.

> **Note**
>
> `Role` 속성을 설정하면 이 속성은 무시됩니다.

유형: 문자열 | 목록 | 맵

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 AWS::IAM::Role 리소스의 Policies 속성과 유사합니다.

PublishToLatestPublished

함수가 업데이트될 때 최신 함수 버전을 게시할지 여부를 지정합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 AWS::Lambda::Function 리소스의 PublishToLatestPublished 속성으로 직접 전달됩니다.

PropagateTags

Tags속성의 태그를 AWS::Serverless::Function 생성된 리소스로 전달할지 여부를 지정합니다. 귀하의 생성된 리소스에 태그를 전파하도록 True을 지정합니다.

유형: 부울

필수 항목 여부: 아니요

기본값: False

CloudFormation 호환성:이 속성은에 고유 AWS SAM 하며 CloudFormation 동등한 속성이 없습니다.

ProvisionedConcurrencyConfig

함수의 별칭에 대한 프로비저닝된 동시성 구성.

> **Note**
>
> AutoPublishAlias가 설정된 경우에만 ProvisionedConcurrencyConfig을 지정할 수 있습니다. 이렇게 하지 않으면 오류가 발생합니다.

유형: ProvisionedConcurrencyConfig

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 AWS::Lambda::Alias 리소스의 ProvisionedConcurrencyConfig 속성으로 직접 전달됩니다.

`RecursiveLoop`

함수의 재귀 루프 감지 구성 상태입니다.

이 값이 `Allow`로 설정되어 있으면 Lambda가 재귀 루프의 일부로 간접 호출되는 함수를 감지해도 아무런 조치를 취하지 않습니다.

이 값이 `Terminate`로 설정되어 있으면 Lambda가 재귀 루프의 일부로 간접 호출되는 함수를 감지한 경우 함수가 간접 호출되는 것을 중지하고 사용자에게 알립니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 [`RecursiveLoop`] 속성으로 직접 전달됩니다.

`ReservedConcurrentExecutions`

함수에 대해 예비하고 싶은 최대 동시 실행 수를 지정합니다.

이 속성에 대한 자세한 내용은 AWS Lambda 개발자 안내서[의 Lambda함수] 규모 조정을 참조하세요.

유형: 정수

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 [`ReservedConcurrentExecutions`] 속성으로 직접 전달됩니다.

`Role`

이 함수의 실행 역할로 사용할 IAM 역할의 ARN.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 [`Role`] 속성과 유사합니다. 이는 에는 필요하지 CloudFormation 만 에는 필요하지 않습니다 AWS SAM. 역할을 지정하지 않으면 *<function-logical-id>*`Role`의 논리 ID로 귀하에게 하나가 생성됩니다.

`RolePath`

함수의 IAM 실행 역할의 경로입니다.

역할이 자동으로 생성될 때 이 속성을 사용하십시오. `Role` 속성에 역할이 지정된 경우에는 사용하지 마십시오.

유형: 문자열

필수 항목 여부: 조건부

CloudFormation 호환성:이 속성은 `AWS::IAM::Role` 리소스의 `Path` 속성으로 직접 전달됩니다.

`Runtime`

함수 런타임의 식별자입니다. 이 속성은 `PackageType` 속성이 `Zip`로 설정된 경우에만 필요합니다.

> **Note**
>
> 이 속성의 `provided` 식별자를 지정하는 경우 `Metadata` 리소스 속성을 사용하여이 함수에 필요한 사용자 지정 런타임을 빌드 AWS SAM 하도록에 지시할 수 있습니다. 사용자 지정 런타임을 구축하는 방법에 대한 자세한 정보는 AWS SAM에서 사용자 지정 런타임을 사용하여 Lambda 함수 빌드 섹션을 참조하세요.

유형: 문자열

필수 항목 여부: 조건부

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 `Runtime` 속성으로 직접 전달됩니다.

`RuntimeManagementConfig`

런타임 환경 업데이트, 롤백 동작, 특정 런타임 버전 선택 등 Lambda 함수의 런타임 관리 옵션을 구성합니다. 자세한 내용은 AWS Lambda 개발자 안내서의 Lambda 런타임 업데이트를 참조하세요.

유형: RuntimeManagementConfig

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 `RuntimeManagementConfig` 속성으로 직접 전달됩니다.

`SnapStart`

모든 새 Lambda 함수 버전의 스냅샷을 생성합니다. 스냅샷은 모든 종속성을 포함하여 초기화된 함수의 캐시된 상태입니다. 함수는 한 번만 초기화되고 캐시된 상태는 향후 모든 간접 호출에 재사용되므로 함수를 초기화해야 하는 횟수를 줄여 애플리케이션 성능을 개선합니다. 자세한 내용은 AWS Lambda 개발자 안내서의 Lambda SnapStart를 통한 시작 성능 개선을 참조하세요.

유형: SnapStart

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 `SnapStart` 속성으로 직접 전달됩니다.

`SourceKmsKeyArn`

고객의 ZIP 함수 코드를 암호화하는 데 사용되는 KMS 키 ARN을 나타냅니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` `Code` 데이터 유형의 `SourceKmsKeyArn` 속성으로 직접 전달됩니다.

`Tags`

이 함수에 추가된 태그를 지정하는 맵(문자열에 문자열)입니다. 태그의 유효한 키와 값에 관한 자세한 내용은 AWS Lambda 개발자 안내서의 태그 키 및 값 요구 사항을 참조하세요.

스택이 생성되면는이 Lambda 함수와이 함수에 대해 생성된 기본 역할에 `lambda:createdBy:SAM` 태그를 AWS SAM 자동으로 추가합니다.

유형: 맵

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 `Tags` 속성과 유사합니다. 의 `Tags` 속성은 키-값 페어로 AWS SAM 구성됩니다(이 CloudFormation 속성은

Tag 객체 목록으로 구성됨). 또한는이 Lambda 함수와이 함수에 대해 생성된 기본 역할에 `lambda:createdBy:SAM` 태그를 AWS SAM 자동으로 추가합니다.

`TenancyConfig`

Lambda 테넌트 격리 모드에 대한 구성입니다. 실행 환경이 서로 다른 테넌트 IDs 간에 공유되지 않도록 하여 다중 테넌트 애플리케이션을 위한 컴퓨팅 수준 격리를 제공합니다.

유형: [TenancyConfig](#)

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 [`TenancyConfig`](#) 속성으로 직접 전달됩니다.

`Timeout`

중지되기 전까지 함수를 실행할 수 있는 최대 시간(초).

유형: 정수

필수 항목 여부: 아니요

기본값: 3

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 [`Timeout`](#) 속성으로 직접 전달됩니다.

`Tracing`

함수의 X-Ray 추적 모드를 지정하는 문자열.

- `Active` – 함수에 대한 X-Ray 추적을 활성화합니다.
- `Disabled` – 함수에 대한 X-Ray를 비활성화합니다.
- `PassThrough` – 함수에 대한 X-Ray 추적을 활성화합니다. 샘플링 결정은 다운스트림 서비스에 위임됩니다.

`Active` 또는 `PassThrough`로 지정되고 `Role` 속성이 설정되지 않은 경우, AWS SAM 는 귀하를 위해 생성한 Lambda 실행 역할에 `arn:aws:iam::aws:policy/AWSXrayWriteOnlyAccess` 정책을 추가합니다.

X-Ray에 대한 자세한 내용은 AWS Lambda 개발자 안내서의와 [AWS Lambda 함께 사용을 AWS X-Ray](#) 참조하세요.

유효한 값: [Active|Disabled|PassThrough]

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 AWS::Lambda::Function 리소스의 TracingConfig 속성과 유사합니다.

VersionDescription

새 Lambda 버전 리소스에 추가되는 Description 필드를 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 AWS::Lambda::Version 리소스의 Description 속성으로 직접 전달됩니다.

VersionDeletionPolicy

가 AutoPublishAlias 설정될 때 생성되는 Lambda 버전 리소스에 대한 삭제 정책을 지정합니다. 이렇게 하면 스택이 삭제될 때 버전 리소스가 보존 또는 삭제되는지 여부를 제어합니다.

유효한 값: Delete, Retain 또는 Snapshot

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은에 고유 AWS SAM 하며 CloudFormation 동등한 속성이 없습니다. 생성된 AWS::Lambda::Version 리소스에 DeletionPolicy 속성을 설정합니다.

VpcConfig

이 함수가 Virtual Private Cloud(VPC) 내의 프라이빗 리소스에 액세스할 수 있도록 하는 구성입니다.

유형: VpcConfig

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 `VpcConfig` 속성으로 직접 전달됩니다.

반환 값

Ref

`Ref` 내장 함수에 이 리소스의 논리적 ID를 입력하면 기저의 Lambda 함수의 리소스 이름이 반환됩니다.

`Ref` 함수의 사용에 대한 자세한 내용은 AWS CloudFormation 사용자 가이드의 `Ref` 섹션을 참조하세요.

Fn::GetAtt

`Fn::GetAtt`은 이 유형의 지정된 속성에 대한 값을 반환합니다. 다음은 사용 가능한 속성과 반환되는 샘플 값.

`Fn::GetAtt`의 사용에 대한 자세한 내용은 AWS CloudFormation 사용자 가이드의 `Fn::GetAtt` 섹션을 참조하세요.

`Arn`

기저의 Lambda 함수의 ARN.

예제

단순 함수

다음은 Amazon S3 버킷에 있는 패키지 유형 `Zip`(기본값) 및 함수 코드의 AWS::Serverless::Function 리소스에 대한 기본 예제입니다.

YAML

```
Type: AWS::Serverless::Function
Properties:
  Handler: index.handler
  Runtime: python3.9
  CodeUri: s3://bucket-name/key-name
```

함수 속성 예제

다음은 `InlineCode`, `Layers`, `Tracing`, `Policies`, `Amazon EFS` 및 `Api` 이벤트 소스를 사용하는 패키지 유형 `Zip`(기본값) AWS::Serverless::Function의 예입니다.

YAML

```
Type: AWS::Serverless::Function
DependsOn: MyMountTarget        # This is needed if an AWS::EFS::MountTarget resource
 is declared for EFS
Properties:
  Handler: index.handler
  Runtime: python3.9
  InlineCode: |
    def handler(event, context):
      print("Hello, world!")
  ReservedConcurrentExecutions: 30
  Layers:
    - Ref: MyLayer
  Tracing: Active
  Timeout: 120
  FileSystemConfigs:
    - Arn: !Ref MyEfsFileSystem
      LocalMountPath: /mnt/EFS
  Policies:
    - AWSLambdaExecute
    - Version: '2012-10-17       '
      Statement:
        - Effect: Allow
          Action:
            - s3:GetObject
            - s3:GetObjectACL
          Resource: 'arn:aws:s3:::sam-s3-demo-bucket/*'
  Events:
    ApiEvent:
      Type: Api
      Properties:
        Path: /path
        Method: get
```

ImageConfig 예제

다음은 패키지 유형`Image`의 Lambda 함수에 대한 `ImageConfig`의 예제입니다.

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 예제

현재 동작에 따라 런타임 환경을 업데이트하도록 구성된 Lambda 함수:

```
TestFunction
  Type: AWS::Serverless::Function
  Properties:
    ...
    Runtime: python3.9
    RuntimeManagementConfig:
      UpdateRuntimeOn: Auto
```

함수가 업데이트될 때 런타임 환경을 업데이트하도록 구성된 Lambda 함수:

```
TestFunction
  Type: AWS::Serverless::Function
  Properties:
    ...
    Runtime: python3.9
    RuntimeManagementConfig:
      UpdateRuntimeOn: FunctionUpdate
```

런타임 환경을 수동으로 업데이트하도록 구성된 Lambda 함수:

```
TestFunction
  Type: AWS::Serverless::Function
  Properties:
    ...
```

```
    Runtime: python3.9
    RuntimeManagementConfig:
      RuntimeVersionArn: arn:aws:lambda:us-
east-1::runtime:4c459dd0104ee29ec65dcad056c0b3ddbe20d6db76b265ade7eda9a066859b1e
      UpdateRuntimeOn: Manual
```

SnapStart 예제

향후 버전에서 SnapStart가 켜져 있는 Lambda 함수의 예:

```
TestFunc
  Type: AWS::Serverless::Function
  Properties:
    ...
    SnapStart:
      ApplyOn: PublishedVersions
```

TenancyConfig 예제

테넌트 격리 모드가 켜져 있는 Lambda 함수의 예:

```
TestFunction
  Type: AWS::Serverless::Function
  Properties:
    ...
    TenancyConfig:
      TenantIsolationMode: PER_TENANT
```

DeadLetterQueue

AWS Lambda(Lambda)에서 이벤트를 처리할 수 없을 때 이벤트를 전송하는 SQS 대기열 또는 SNS 주제를 지정합니다. DLQ(Dead Letter Queue) 기능에 대한 자세한 내용은 AWS Lambda 개발자 안내서의 DLQ(Dead Letter Queue)를 참조하세요.

SAM은 Lambda 함수 실행 역할에 적절한 권한을 자동으로 추가하여 Lambda 서비스에 리소스에 대한 액세스 권한을 부여합니다. SQS 대기열에는 SQS:sendMessage가 추가되고 SNS 주제에는 SNS:Publish가 추가됩니다.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
TargetArn: String
Type: String
```

속성

TargetArn

Amazon SQS 대기열 또는 Amazon SNS 주제의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `TargetArn` `AWS::Lambda::Function` 데이터 유형의 `DeadLetterConfig` 속성에 직접 전달됩니다.

Type

DLQ(Dead Letter Queue) 유형.

유효한 값: SNS, SQS

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

DeadLetterQueue

SNS 주제에 대한 DLQ 예제

YAML

```
DeadLetterQueue:
  Type: SNS
```

```
TargetArn: arn:aws:sns:us-east-2:123456789012:my-topic
```

DeploymentPreference

점진적 Lambda 배포를 지원하는 구성을 지정합니다. 점진적 Lambda 배포 구성에 대한 자세한 내용은 AWS SAM에서 서버리스 애플리케이션의 점진적 배포을 참조하세요.

> **Note**
>
> `AutoPublishAlias` 객체를 사용하려면 AWS::Serverless::Function에서 `DeploymentPreference`을 지정해야 합니다. 그렇지 않으면 오류가 발생합니다.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
Alarms: List
Enabled: Boolean
Hooks: Hooks
PassthroughCondition: Boolean
Role: String
TriggerConfigurations: List
Type: String
```

속성

`Alarms`

배포로 인해 발생한 오류로 인해 트리거하고자 하는 CloudWatch 경보의 목록입니다.

이 속성은 `Fn::If` 내장 함수를 받아들입니다. `Fn::If`를 사용하는 예제 템플릿은 이 항목 하단의 예제 섹션을 참조하세요.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Enabled`

이 배포 환경 설정이 활성화되어 있는지 여부입니다.

유형: 부울

필수 항목 여부: 아니요

기본값: True

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Hooks`

트래픽 이동 전후에 실행되는 Lambda 함수를 검증합니다.

유형: Hooks

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`PassthroughCondition`

True인 경우 이 배포 선호가 활성화되어 있으면, 생성된 CodeDeploy 리소스로 함수의 조건이 전달됩니다. 일반적으로 이 값을 True로 설정해야 합니다. 그렇지 않으면 함수의 조건이 False로 확인되더라도 CodeDeploy 리소스가 생성됩니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Role`

CodeDeploy가 트래픽 이동에 사용할 IAM 역할 ARN입니다. 이것이 제공되면 IAM 역할이 생성되지 않습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`TriggerConfigurations`

배포 그룹에 연결하고자 하는 트리거 구성의 목록입니다. 라이프사이클 이벤트에 관한 SNS 주제를 알리는 데 사용됩니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::CodeDeploy::DeploymentGroup` 리소스의 `TriggerConfigurations` 속성으로 직접 전달됩니다.

`Type`

현재 배포 유형에는 Linear와 Canary라는 두 가지 카테고리가 있습니다. 사용할 수 있는 배포 유형에 대한 자세한 내용은 AWS SAM에서 서버리스 애플리케이션의 점진적 배포섹션을 참조하세요.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

사전 및 사후 트래픽 후크가 포함된 DeploymentPreference.

트래픽 전후 훅이 포함된 배포 환경 설정 예입니다.

YAML

```
DeploymentPreference:
  Enabled: true
  Type: Canary10Percent10Minutes
  Alarms:
```

```
    - !Ref: AliasErrorMetricGreaterThanZeroAlarm
    - !Ref: LatestVersionErrorMetricGreaterThanZeroAlarm
  Hooks:
    PreTraffic:
      !Ref: PreTrafficLambdaFunction
    PostTraffic:
      !Ref: PostTrafficLambdaFunction
```

Fn: :If 내장 함수를 사용한 DeploymentPreference

알람 구성에 `Fn::If`를 사용하는 배포 환경 설정의 예입니다. 이 예제에서는 `Alarm1`가 `MyCondition`인 경우 `true`이 구성되고, `Alarm2`이 `Alarm5`이면 `MyCondition` 및 `false`가 구성될 것입니다.

YAML

```
DeploymentPreference:
  Enabled: true
  Type: Canary10Percent10Minutes
  Alarms:
    Fn::If:
      - MyCondition
      - - Alarm1
      - - Alarm2
        - Alarm5
```

Hooks

트래픽 이동 전후에 실행되는 Lambda 함수를 검증합니다.

> **Note**
>
> 이 속성에서 참조되는 Lambda 함수는 결과 `CodeDeployLambdaAliasUpdate` 리소스의 AWS::Lambda::Alias 객체를 구성합니다. 자세한 내용은 사용자 가이드의 AWS CloudFormationCodeDeploy LambdaAliasUpdate 정책을 참조하세요.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  PostTraffic: String
  PreTraffic: String
```

속성

PostTraffic

트래픽 이동 후에 실행되는 Lambda 함수입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

PreTraffic

트래픽이 이동하기 전에 실행되는 Lambda 함수입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

후크

예제 후크 함수

YAML

```
Hooks:
  PreTraffic:
    Ref: PreTrafficLambdaFunction
  PostTraffic:
```

```
    Ref: PostTrafficLambdaFunction
```

DurableConfig

AWS Lambda 함수에 대한 내구성 있는 실행 설정을 구성합니다. 내구성 있는 함수는 최대 1년 동안 실행되고 진행 상황을 자동으로 체크포인트하여 장기 실행 워크플로와 내결함성 애플리케이션을 지원합니다. 내구성 함수에 대한 자세한 내용은 AWS Lambda 개발자 안내서의 [Lambda 내구성 함수](#)를 참조하세요.

구문

AWS Serverless Application Model (AWS SAM) 템플릿에서이 개체를 선언하려면 다음 구문을 사용합니다.

YAML

```
  ExecutionTimeout: Integer
  RetentionPeriodInDays: Integer
```

속성

`ExecutionTimeout`

Lambda가 영구 함수를 중지하기 전에 실행하도록 허용하는 시간(초)입니다. 최대값은 366일 또는 31,622,400초입니다.

유형: 정수

필수 항목 여부: 예

최소: 1

최대: 31622400

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` `DurableConfig` 데이터 유형의 `ExecutionTimeout` 속성으로 직접 전달됩니다.

`RetentionPeriodInDays`

내구성이 뛰어난 실행이 종료된 후 Lambda가 기록을 유지하는 일수로, 1~90일입니다. 기본값은 14일입니다.

유형: 정수

필수 항목 여부: 아니요

기본값: 14

최소: 1

최대: 90

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` `DurableConfig` 데이터 유형의 [`RetentionPeriodInDays`]() 속성으로 직접 전달됩니다.

예제

DurableConfig

실행 제한 시간이 1시간이고 보존 기간이 7일인 함수에 대한 내구성 구성 예제입니다.

YAML

```
DurableConfig:
  ExecutionTimeout: 3600
  RetentionPeriodInDays: 7
```

EventInvokeConfiguration

[비동기식]() Lambda 별칭 또는 버전 간접 호출을 위한 구성 옵션입니다.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  DestinationConfig: EventInvokeDestinationConfiguration
  MaximumEventAgeInSeconds: Integer
  MaximumRetryAttempts: Integer
```

속성

`DestinationConfig`

Lambda가 이벤트를 처리한 후 이벤트의 대상을 지정하는 구성 객체입니다.

유형: [EventInvokeDestinationConfiguration](#)

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [`DestinationConfig`](#) 리소스의 `AWS::Lambda::EventInvokeConfig` 속성과 유사합니다. SAM에는 CloudFormation에 없는 추가 매개변수인 “유형”이 필요합니다.

`MaximumEventAgeInSeconds`

Lambda가 처리를 위해 함수에 보내는 요청의 최대 사용 기간입니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventInvokeConfig` 리소스의 [`MaximumEventAgeInSeconds`](#) 속성으로 직접 전달됩니다.

`MaximumRetryAttempts`

함수가 오류를 반환할 때 재시도하는 최대 횟수입니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [`MaximumRetryAttempts`](#) 리소스의 `AWS::Lambda::EventInvokeConfig` 속성으로 직접 전달됩니다.

예시

MaximumEventAgeInSeconds

MaximumEventAgeInSeconds 예제

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

Lambda가 이벤트를 처리한 후 이벤트의 대상을 지정하는 구성 객체입니다.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  OnFailure: OnFailure
  OnSuccess: OnSuccess
```

속성

`OnFailure`

처리에 실패한 이벤트의 대상입니다.

유형: OnFailure

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `OnFailure` 리소스의 `AWS::Lambda::EventInvokeConfig` 속성과 유사합니다. 추가 SAM 전용 속성인 `Type`이 요구됩니다.

`OnSuccess`

성공적으로 처리된 이벤트의 대상입니다.

유형: OnSuccess

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 OnSuccess 리소스의 `AWS::Lambda::EventInvokeConfig` 속성과 유사합니다. 추가 SAM 전용 속성인 `Type`이 요구됩니다.

예시

OnSuccess

OnSuccess 예제

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SQS
      Destination: arn:aws:sqs:us-west-2:012345678901:my-queue
    OnFailure:
      Type: Lambda
      Destination: !GetAtt DestinationLambda.Arn
```

OnFailure

처리에 실패한 이벤트의 대상입니다.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  Destination: String
  Type: String
```

속성

`Destination`

대상 리소스의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 조건부

CloudFormation호환성: 이 속성은 `OnFailure` 리소스의 `AWS::Lambda::EventInvokeConfig` 속성과 유사합니다. SAM은 이 속성에서 참조되는 리소스에 액세스하는 데 필요한 권한을 이 함수와 관련된 자동 생성 IAM 역할에 추가합니다.

추가 참고 사항: 유형이 Lambda/EventBridge인 경우 대상이 필요합니다.

`Type`

대상에서 참조되는 리소스의 유형입니다. 지원되는 유형은 `SQS`, `SNS`, `S3`, `Lambda` 및 `EventBridge`입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

추가 참고 사항: 유형이 SQS/SNS이고 `Destination` 속성이 비어 있는 경우 SAM이 SQS/SNS 리소스를 자동 생성합니다. 리소스를 참조하려면 SQS의 경우 `<function-logical-id>.DestinationQueue`를, SNS의 경우 `<function-logical-id>.DestinationTopic`를 사용합니다. 유형이 Lambda/EventBridge인 경우 `Destination`이 필수입니다.

예시

SQS 및 Lambda 대상을 사용한 EventInvoke 구성 예제

이 예제에서는 SQS OnSuccess 구성에 대해 대상이 제공되지 않으므로 SAM은 묵시적으로 SQS 대기열을 생성하고 필요한 권한을 추가합니다. 또한 이 예제에서는 템플릿 파일에 선언된 Lambda 리소스에 대한 대상 람다 함수가 OnFailure 구성에 지정되어 있으므로 SAM은 대상 Lambda 함수를 직접 호출하는 데 필요한 권한을 이 Lambda 함수에 추가합니다.

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
```

```
      Type: SQS
    OnFailure:
      Type: Lambda
      Destination: !GetAtt DestinationLambda.Arn  # Arn of a Lambda function declared
 in the template file.
```

SNS 대상을 사용한 EventInvoke 구성 예제

이 예제에서는 OnSuccess 구성의 템플릿 파일에 선언된 SNS 주제에 대하여 대상이 제공됩니다.

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SNS
      Destination:
        Ref: DestinationSNS       # Arn of an SNS topic declared in the tempate file
```

OnSuccess

성공적으로 처리된 이벤트의 대상입니다.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  Destination: String
  Type: String
```

속성

`Destination`

대상 리소스의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 조건부

CloudFormation호환성: 이 속성은 `OnSuccess` 리소스의 `AWS::Lambda::EventInvokeConfig` 속성과 유사합니다. SAM은 이 속성에서 참조되는 리소스에 액세스하는 데 필요한 권한을 이 함수와 관련된 자동 생성 IAM 역할에 추가합니다.

추가 참고 사항: 유형이 Lambda/EventBridge인 경우 대상이 필요합니다.

`Type`

대상에서 참조되는 리소스의 유형입니다. 지원되는 유형은 `SQS`, `SNS`, `S3`, `Lambda` 및 `EventBridge`입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

추가 참고 사항: 유형이 SQS/SNS이고 `Destination` 속성이 비어 있는 경우 SAM이 SQS/SNS 리소스를 자동 생성합니다. 리소스를 참조하려면 SQS의 경우 `<function-logical-id>.DestinationQueue`를, SNS의 경우 `<function-logical-id>.DestinationTopic`를 사용합니다. 유형이 Lambda/EventBridge인 경우 `Destination`이 필수입니다.

예시

SQS 및 Lambda 대상을 사용한 EventInvoke 구성 예제

이 예제에서는 SQS OnSuccess 구성에 대해 대상이 제공되지 않으므로 SAM은 묵시적으로 SQS 대기열을 생성하고 필요한 권한을 추가합니다. 또한 이 예제에서는 템플릿 파일에 선언된 Lambda 리소스에 대한 대상 람다 함수가 OnFailure 구성에 지정되어 있으므로 SAM은 대상 Lambda 함수를 직접 호출하는 데 필요한 권한을 이 Lambda 함수에 추가합니다.

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SQS
    OnFailure:
      Type: Lambda
      Destination: !GetAtt DestinationLambda.Arn  # Arn of a Lambda function declared
 in the template file.
```

SNS 대상을 사용한 EventInvoke 구성 예제

이 예제에서는 OnSuccess 구성의 템플릿 파일에 선언된 SNS 주제에 대하여 대상이 제공됩니다.

YAML

```
EventInvokeConfig:
  DestinationConfig:
    OnSuccess:
      Type: SNS
      Destination:
        Ref: DestinationSNS       # Arn of an SNS topic declared in the tempate file
```

EventSource

함수를 트리거하는 이벤트의 소스를 설명하는 객체입니다. 각 이벤트는 유형과 해당 유형에 따라 달라지는 속성 집합으로 구성됩니다. 각 이벤트 소스의 속성에 대한 자세한 내용은 해당 유형에 적용되는 주제를 참조하세요.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  Properties: AlexaSkill | Api | CloudWatchEvent | CloudWatchLogs | Cognito
 | DocumentDB | DynamoDB | EventBridgeRule | HttpApi | IoTRule | Kinesis | MQ | MSK
 | S3 | Schedule | ScheduleV2 | SelfManagedKafka | SNS | SQS
  Type: String
```

속성

`Properties`

이 이벤트 매핑의 속성을 설명하는 객체입니다. 속성 세트는 정의된 유형을 준수해야 합니다.

유형: AlexaSkill | Api | CloudWatchEvent | CloudWatchLogs | Cognito | DocumentDB | DynamoDB | EventBridgeRule | HttpApi | IoTRule | Kinesis | MQ | MSK | S3 | Schedule | ScheduleV2 | SelfManagedKafka | SNS | SQS

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Type`

이벤트 유형.

유효한 값: `AlexaSkill`, `Api`, `CloudWatchEvent`, `CloudWatchLogs`, `Cognito`, `DocumentDB`, `DynamoDB`, `EventBridgeRule`, `HttpApi`, `IoTRule`, `Kinesis`, `MQ`, `MSK`, `S3`, `Schedule`, `ScheduleV2`, `SelfManagedKafka`, `SNS`, `SQS`

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

APIEvent

API 이벤트 사용 예제

YAML

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

AlexaSkill

`AlexaSkill` 이벤트 소스 유형을 설명하는 객체.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
SkillId: String
```

속성

SkillId

귀하의 Alexa 스킬에 대한 Alexa 스킬 ID. 스킬 ID에 대한 자세한 내용은 Alexa 스킬 키트 설명서에서 Lambda 함수에 대한 트리거 구성을 참조하세요.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

AlexaSkillTrigger

알렉사 스킬 이벤트 예제

YAML

```
AlexaSkillEvent:
  Type: AlexaSkill
```

Api

Api 이벤트 소스 유형을 설명하는 객체. AWS::Serverless::Api 리소스가 정의된 경우 경로 및 메서드 값은 API의 OpenAPI 정의에 있는 작업과 일치해야 합니다.

정의된 AWS::Serverless::Api이 없는 경우, 함수 입력 및 출력은 HTTP 요청 및 HTTP 응답을 나타냅니다.

예를 들어, JavaScript API를 사용하면 StatusCode 및 body 키가 있는 객체를 반환하여 응답의 상태 코드와 본문을 제어할 수 있습니다.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
Auth: ApiFunctionAuth
Method: String
Path: String
RequestModel: RequestModel
RequestParameters: List of [ String | RequestParameter ]
RestApiId: String
TimeoutInMillis: Integer
```

속성

`Auth`

이 특정 Api+Path+Method에 대한 인증 구성입니다.

지정된 `DefaultAuthorizer`가 없는 경우, 개별 경로에 대한 API의 `DefaultAuthorizer` 설정 인증 구성을 재정의하거나 기본 `ApiKeyRequired` 설정을 재정의하는 데 유용합니다.

유형: API/함수/권한부여

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Method`

이 함수가 간접 호출되는 HTTP 메서드입니다. 옵션에는 `DELETE`, `GET`, `HEAD`, `OPTIONS`, `PATCH`, `POST`, `PUT` 및 `ANY`이(가) 포함됩니다. 자세한 내용은 API Gateway 개발자 가이드의 HTTP 메서드 설정을 참조하세요.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Path`

이 함수가 간접 호출되는 Uri 경로입니다. /로 시작해야 합니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`RequestModel`

이 특정 API+Path+메서드에 사용할 요청 모델입니다. 이것은 AWS::Serverless::Api 리소스의 `Models` 섹션에 지정된 모델 이름을 참조해야 합니다.

유형: 요청 모델

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`RequestParameters`

이 특정 API+Path+메서드에 대한 파라미터 구성을 요청합니다. 모든 파라미터 이름은 `method.request`로 시작해야 하며, `method.request.header`, `method.request.querystring`, 혹은 `method.request.path`로 제한되어야 합니다.

목록에는 매개변수 이름 문자열과 RequestParameter 객체가 모두 포함될 수 있습니다. 문자열의 경우 `Required` 및 `Caching` 속성은 `false`에 기본적으로 설정됩니다.

유형: [문자열 | 요청 파라미터] 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`RestApiId`

지정된 경로와 메서드를 가진 작업을 포함해야 하는 RestAPI 리소스의 식별자. 이것은 일반적으로 이 템플릿에 정의된 AWS::Serverless::Api 리소스를 참조하도록 설정됩니다.

이 속성을 정의하지 않는 경우 AWS SAM은 생성된 AWS::Serverless::Api 문서를 사용하여 기본 `OpenApi` 리소스를 만듭니다. 해당 리소스에는 `RestApiId`를 지정하지 않은 동일한 템플릿의 `Api` 이벤트에 의해 정의된 모든 경로와 메서드가 통합되어 있습니다.

이것은 다른 템플릿에 정의된 AWS::Serverless::Api 리소스를 참조할 수 없습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`TimeoutInMillis`

50~29,000밀리초 사이의 제한 시간 사용자 지정입니다.

> **Note**
>
> 이 속성을 지정하면, AWS SAM은 OpenAPI 정의를 변경합니다. OpenAPI 정의는 `DefinitionBody` 속성을 사용하여 인라인으로 지정해야 합니다.

유형: 정수

필수 항목 여부: 아니요

기본값: 29,000밀리초(29초)입니다.

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

기본 예제

YAML

```
Events:
  ApiEvent:
    Type: Api
    Properties:
      Path: /path
```

```
      Method: get
      RequestParameters:
        - method.request.header.Authorization
        - method.request.querystring.keyword:
            Required: true
            Caching: false
```

ApiFunctionAuth

특정 API, 경로 및 메서드에 대해 이벤트 수준에서 권한 부여를 구성합니다.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  ApiKeyRequired: Boolean
  AuthorizationScopes: List
  Authorizer: String
  InvokeRole: String
  OverrideApiAuth: Boolean
  ResourcePolicy: ResourcePolicyStatement
```

속성

`ApiKeyRequired`

이 API, 경로 및 메서드에 대한 API 키가 필요합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`AuthorizationScopes`

이 API, 경로 및 메서드에 적용할 수 있는 권한 부여 범위.

지정한 범위는 해당 속성을 지정한 경우 `DefaultAuthorizer` 속성에서 적용한 모든 범위를 재정의합니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Authorizer`

특정 함수를 위한 `Authorizer`

귀하의 `AWS::Serverless::Api` 리소스에 글로벌 권한 부여자가 지정된 경우, `Authorizer`를 `NONE`으로 설정하여 권한 부여자를 재정의할 수 있습니다. 문제 해결 예는 Amazon API Gateway REST API에 대한 글로벌 권한 부여자 재정의을(를) 참조하세요.

> **Note**
>
> `AWS::Serverless::Api` 리소스의 `DefinitionBody` 속성을 사용하여 API를 설명하는 경우 `Authorizer`와 함께 `OverrideApiAuth`을 사용하여 귀하의 글로벌 권한 부여자를 재정의해야 합니다. 자세한 정보는 `OverrideApiAuth`을 참조하세요

유효한 값: `AWS_IAM`, `NONE`, 또는 AWS SAM 템플릿에 정의된 권한 부여자의 논리적 ID

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`InvokeRole`

`AWS_IAM` 권한 부여에 사용할 `InvokeRole`를 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

기본값: `CALLER_CREDENTIALS`

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

추가 참고 사항: CALLER_CREDENTIALS는 arn:aws:iam::*:<user>/*을 매핑하며, 이는 호출자 보안 인증을 사용하여 엔드포인트를 호출합니다.

`OverrideApiAuth`

귀하의 `AWS::Serverless::Api`리소스의 글로벌 권한 부여자 구성을 재정의하도록 `true`을 지정합니다. 이 속성은 글로벌 권한 부여자를 지정하고 `AWS::Serverless::Api` 리소스의 `DefinitionBody` 속성을 사용하여 API를 설명하는 경우에만 필요합니다.

> **Note**
>
> `OverrideApiAuth`을 `true`로 지정하면, AWS SAM은 귀하의 글로벌 권한 부여자를 `ApiKeyRequired`, `Authorizer`, 혹은 `ResourcePolicy`에 제공된 값으로 재정의합니다. 따라서 `OverrideApiAuth`를 사용할 때 이러한 속성들 중에서 적어도 하나가 지정되어야 합니다. 예시는 AWS::서버리스::Api용 DefinitionBody가 지정된 경우 글로벌 권한 부여자를 재정의합니다. 섹션을 참조하세요.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`ResourcePolicy`

API에서 이 경로에 대한 리소스 정책을 구성합니다.

유형: 리소스 정책 설명

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

함수-권한 부여

다음 예제에서는 함수 수준에서 권한 부여를 지정합니다.

YAML

```
Auth:
  ApiKeyRequired: true
  Authorizer: NONE
```

Amazon API Gateway REST API에 대한 글로벌 권한 부여자 재정의

`AWS::Serverless::Api` 리소스의 글로벌 권한 부여자를 지정할 수 있습니다. 다음은 글로벌 기본 권한 부여자를 구성하는 예입니다.

```
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
```

귀하의 AWS Lambda 함수의 기본 권한 부여자를 재정의하려면 귀하는 `Authorizer`를 NONE으로 지정할 수 있습니다. 다음은 예제입니다.

```
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
```

AWS::서버리스::Api용 DefinitionBody가 지정된 경우 글로벌 권한 부여자를 재정의합니다.

`DefinitionBody`속성을 사용하여 `AWS::Serverless::Api` 리소스를 설명하는 경우 이전 재정의 방법이 작동하지 않습니다. 다음은 `AWS::Serverless::Api` 리소스에 `DefinitionBody` 속성을 사용하는 예제입니다.

```
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
```

글로벌 권한 부여자를 재정의하려면 `OverrideApiAuth` 속성을 사용합니다. 다음은 `Authorizer`에 제공된 값으로 글로벌 권한 부여자를 재정의하는 데 `OverrideApiAuth`을 사용하는 예제입니다.

```
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

API의 모든 메서드와 경로에 대한 리소스 정책을 구성합니다. 리소스 정책에 대한 자세한 내용은 API Gateway 개발자 안내서의 API Gateway 리소스 정책을 사용하는 액세스 제어를 참조하세요.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  AwsAccountBlacklist: List
  AwsAccountWhitelist: List
```

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

속성

`AwsAccountBlacklist`

차단할 AWS 계정.

유형: 문자열 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`AwsAccountWhitelist`

허용할 AWS 계정. 이 속성의 예제 사용 예제를 알아보려면 이 페이지 하단의 예제을 참조하세요

유형: 문자열 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`CustomStatements`

이 API에 적용할 사용자 지정 리소스 정책 설명 목록. 이 속성의 예제 사용 예제를 알아보려면 이 페이지 하단의 예제을 참조하세요

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`IntrinsicVpcBlacklist`

차단할 Virtual Private Cloud(VPC))목록입니다. 여기서 각 VPC는 동적 참조 또는 `Ref` 내장 함수와 같은 참조로 지정됩니다. 이 속성의 예제 사용 예제를 알아보려면 이 페이지 하단의 예제을 참조하세요

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`IntrinsicVpcWhitelist`

허용할 VPC 목록. 여기서 각 VPC는 동적 참조 또는 `Ref` 내장 함수와 같은 참조로 지정됩니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`IntrinsicVpceBlacklist`

차단할 VPC 엔드포인트 목록. 여기서 각 VPC 엔드포인트는 동적 참조 또는 `Ref` 내장 함수와 같은 참조로 지정됩니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`IntrinsicVpceWhitelist`

허용할 VPC 엔드포인트 목록. 여기서 각 VPC 엔드포인트는 동적 참조 또는 `Ref` 내장 함수와 같은 참조로 지정됩니다. 이 속성의 예제 사용 예제를 알아보려면 이 페이지 하단의 예제을 참조하세요

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`IpRangeBlacklist`

차단할 IP 주소 또는 주소 범위. 이 속성의 예제 사용 예제를 알아보려면 이 페이지 하단의 예제을 참조하세요

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`IpRangeWhitelist`

허용할 IP 주소 또는 주소 범위.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`SourceVpcBlacklist`

차단할 소스 VPC 또는 VPC 엔드포인트. 소스 VPC 이름은 `"vpc-"`로 시작하고 소스 VPC 엔드포인트 이름은 `"vpce-"`로 시작해야 합니다. 이 속성의 예제 사용 예제를 알아보려면 이 페이지 하단의 예제을 참조하세요

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`SourceVpcWhitelist`

허용할 소스 VPC 또는 VPC 엔드포인트. 소스 VPC 이름은 `"vpc-"`로 시작하고 소스 VPC 엔드포인트 이름은 `"vpce-"`로 시작해야 합니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

리소스 정책 예시

다음 예시에서는 두 개의 IP 주소와 한 개의 소스 VPC를 차단하고 한 AWS계정을 허용합니다.

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

특정 Api+Path+Method에 대한 요청 모델을 구성합니다.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
Model: String
Required: Boolean
ValidateBody: Boolean
ValidateParameters: Boolean
```

속성

`Model`

AWS::Serverless::Api의 Models 속성에 정의된 모델 이름입니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Required`

주어진 API 엔드포인트에 대한 OpenAPI 정의의 매개변수 섹션에 `required` 속성을 추가합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`ValidateBody`

API Gateway가 요청 본문을 검증하기 위해 `Model`을 사용할지 여부를 지정합니다. 자세한 내용은 API Gateway 개발자 가이드 내 API 게이트웨이에서 요청 검증 활성화를 참조하세요.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`ValidateParameters`

API 게이트웨이에서 `Model`를 사용하여 요청 경로 매개변수, 쿼리 문자열 및 헤더를 검증할지 여부를 지정합니다. 자세한 내용은 [API Gateway 개발자 가이드](#) 내 API 게이트웨이에서 요청 검증 활성화를 참조하세요.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

요청 모델

요청 모델 예제

YAML

```
RequestModel:
  Model: User
  Required: true
  ValidateBody: true
  ValidateParameters: true
```

RequestParameter

특정 Api+Path+Method에 대한 요청 매개변수를 구성합니다.

요청 매개변수에 대해 `Required` 또는 `Caching` 속성을 지정해야 합니다

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
Caching: Boolean
Required: Boolean
```

속성

Caching

API 게이트웨이 OpenAPI 정의에 `cacheKeyParameters` 섹션 추가

유형: 부울

필수 항목 여부: 조건부

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

Required

이 필드는 매개변수가 필요한지를 지정합니다

유형: 부울

필수 항목 여부: 조건부

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

요청 매개변수

요청 매개변수 설정 예제

YAML

```
RequestParameters:
  - method.request.header.Authorization:
      Required: true
      Caching: true
```

CloudWatchEvent

CloudWatchEvent 이벤트 소스 유형을 설명하는 객체.

AWS Serverless Application Model(AWS SAM)은 이 이벤트 유형이 설정되면 AWS::Events::Rule리소스를 생성합니다.

중요 참고: EventBridgeRule는 CloudWatchEvent 대신 사용할 우선적인 이벤트 소스 유형입니다. EventBridgeRule 및 CloudWatchEvent는 동일한 기본 서비스, API, CloudFormation 리소스를 사용합니다. 하지만 AWS SAM은 오직 EventBridgeRule에 대해서만 새 기능에 대한 지원을 추가할 예정입니다.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
Enabled: Boolean
EventBusName: String
Input: String
InputPath: String
Pattern: EventPattern
State: String
```

속성

Enabled

규칙을 활성화할지를 나타냅니다.

규칙을 비활성화하려면 이 속성을 false로 설정합니다.

> **Note**
>
> Enabled 또는 State 속성을 지정할 수 있지만, 두 속성을 함께 지정할 수는 없습니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [State](#) 리소스의 AWS::Events::Rule 속성과 유사합니다. 이 속성이 true로 설정되면 AWS SAM가 ENABLED로 전달되고, 그렇지 않으면 DISABLED로 전달됩니다.

EventBusName

이 규칙과 연결할 이벤트 버스입니다. 이 속성을 생략하면 AWS SAM은 기본 사항으로서 이벤트 버스를 사용합니다.

유형: 문자열

필수 항목 여부: 아니요

기본값: 기본 이벤트 버스

CloudFormation호환성: 이 속성은 AWS::Events::Rule 리소스의 EventBusName 속성으로 직접 전달됩니다.

Input

대상으로 전달되는 유효한 JSON 텍스트입니다. 이 속성을 사용하면 이벤트 텍스트 자체의 어떤 것도 대상으로 전달되지 않습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS::Events::Rule Target 리소스의 Input 속성으로 직접 전달됩니다.

InputPath

일치된 이벤트 전체를 전달하지 않으려는 경우 InputPath 속성을 사용하여 이벤트의 어떤 부분이 전달되어야 하는지 설명하세요.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS::Events::Rule Target 리소스의 InputPath 속성으로 직접 전달됩니다.

`Pattern`

어떤 이벤트가 지정된 대상으로 라우팅되는지를 설명합니다. 자세한 내용은 Amazon EventBridge 사용자 가이드의 [EventBridge의 이벤트 및 이벤트 패턴]을 참조하세요.

유형: [이벤트패턴]

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 [`EventPattern`] 리소스의 `AWS::Events::Rule` 속성으로 직접 전달됩니다.

`State`

규칙의 상태입니다.

허용되는 값: `DISABLED | ENABLED`

> **Note**
>
> `Enabled` 또는 `State` 속성을 지정할 수 있지만, 두 속성을 함께 지정할 수는 없습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [`State`] 리소스의 `AWS::Events::Rule` 속성으로 직접 전달됩니다.

예시

CloudWatchEvent

다음은 `CloudWatchEvent` 이벤트 소스 유형의 한 예제입니다.

YAML

```
CWEvent:
  Type: CloudWatchEvent
  Properties:
    Enabled: false
```

```
    Input: '{"Key": "Value"}'
    Pattern:
      detail:
        state:
          - running
```

CloudWatchLogs

CloudWatchLogs 이벤트 소스 유형을 설명하는 객체.

AWS::Logs::SubscriptionFilter 리소스는 구독 필터를 지정하여 특정 로그 그룹과 연결합니다.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  FilterPattern: String
  LogGroupName: String
```

속성

FilterPattern

대상 AWS 리소스로 전달되는 항목을 제한하는 필터링 표현식입니다. 필터 패턴 구문에 대한 자세한 내용은 필터 및 패턴 구문을 참조하세요.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `FilterPattern` 리소스의 `AWS::Logs::SubscriptionFilter` 속성으로 직접 전달됩니다.

LogGroupName

구독 필터와 연결되는 로그 그룹입니다. 필터 패턴이 로그 이벤트와 일치하는 경우 이 로그 그룹으로 업로드되는 모든 로그 이벤트가 필터링되어 지정된 AWS리소스로 전달됩니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 LogGroupName 리소스의 AWS::Logs::SubscriptionFilter 속성으로 직접 전달됩니다.

예시

Cloudwatchlogs 구독 필터

Cloudwatchlogs 구독 필터 예제

YAML

```
CWLog:
  Type: CloudWatchLogs
  Properties:
    LogGroupName:
      Ref: CloudWatchLambdaLogsGroup
    FilterPattern: My pattern
```

Cognito

Cognito 이벤트 소스 유형을 설명하는 객체.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  Trigger: List
  UserPool: String
```

속성

Trigger

새로운 사용자 풀에 대한 Lambda 트리거 구성 정보입니다.

유형: 목록

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `LambdaConfig` 리소스의 `AWS::Cognito::UserPool` 속성으로 직접 전달됩니다.

`UserPool`

동일한 템플릿에 정의된 UserPool에 대한 참조

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

Cognito 이벤트

Cognito 이벤트 예

YAML

```
CognitoUserPoolPreSignup:
  Type: Cognito
  Properties:
    UserPool:
      Ref: MyCognitoUserPool
    Trigger: PreSignUp
```

DocumentDB

`DocumentDB` 이벤트 소스 유형을 설명하는 객체. 자세한 내용은 AWS Lambda 개발자 안내서의 Amazon DocumentDB 의AWS Lambda 사용을 참조하세요.

구문

귀하의 AWS SAM 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
BatchSize: Integer
```

```
Cluster: String
CollectionName: String
DatabaseName: String
Enabled: Boolean
FilterCriteria: FilterCriteria
FullDocument: String
KmsKeyArn: String
MaximumBatchingWindowInSeconds: Integer
SecretsManagerKmsKeyId: String
SourceAccessConfigurations: List
StartingPosition: String
StartingPositionTimestamp: Double
```

속성

`BatchSize`

한 번의 배치에서 검색하는 최대 항목 수입니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `BatchSize` 속성으로 직접 전달됩니다.

`Cluster`

Amazon DocumentDB 클러스터의 Amazon 리소스 이름(ARN).

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `EventSourceArn` 리소스의 `AWS::Lambda::EventSourceMapping` 속성으로 직접 전달됩니다.

`CollectionName`

데이터베이스 내에서 사용할 컬렉션의 이름입니다. 컬렉션을 지정하지 않으면 Lambda는 모든 컬렉션을 사용합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 CollectionName AWS::Lambda::EventSourceMapping 데이터 유형의 DocumentDBEventSourceConfig 속성에 직접 전달됩니다.

DatabaseName

Amazon DocumentDB 클러스터에서 사용할 데이터베이스의 이름.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 DatabaseName AWS::Lambda::EventSourceMapping 데이터 유형의 DocumentDBEventSourceConfig 속성에 직접 전달됩니다.

Enabled

true이면, 이벤트 소스 매핑이 활성 상태입니다. 폴링 및 간접 호출을 일시 중지하려면 false로 설정합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS::Lambda::EventSourceMapping 리소스의 Enabled 속성으로 직접 전달됩니다.

FilterCriteria

Lambda가 이벤트를 처리해야 하는지 결정하는 기준을 정의하는 객체입니다. 자세한 내용은 AWS Lambda개발자 안내서의 Lambda 이벤트 필터링을 참조하세요.

유형: FilterCriteria

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS::Lambda::EventSourceMapping 리소스의 FilterCriteria 속성으로 직접 전달됩니다.

FullDocument

문서 업데이트 작업 중에 Amazon DocumentDB가 귀하의 이벤트 스트림으로 보내는 내용을 결정합니다. UpdateLookup로 설정된 경우, Amazon DocumentDB는 전체 문서의 복사본과 함께 변경 내용을 설명하는 델타를 보냅니다. 그렇지 않으면 Amazon DocumentDB는 단지 변경 내용이 포함된 부분적 문서만 전송합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 FullDocument AWS::Lambda::EventSourceMapping 데이터 유형의 DocumentDBEventSourceConfig 속성에 직접 전달됩니다.

KmsKeyArn

이 이벤트와 관련된 정보를 암호화하는 키의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS::Lambda::EventSourceMapping 리소스의 KmsKeyArn 속성으로 직접 전달됩니다.

MaximumBatchingWindowInSeconds

함수를 호출하기 전에 기록을 수집할 최대 기간(단위: 초)입니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS::Lambda::EventSourceMapping 리소스의 MaximumBatchingWindowInSeconds 속성으로 직접 전달됩니다.

SecretsManagerKmsKeyId

AWS Secrets Manager에서 제공하는 고객 관리 키의 AWS Key Management Service(AWS KMS) 키 ID. kms:Decrypt 권한이 포함되지 않은 Lambda 실행 역할과 함께 Secrets Manager의 고객 관리형 키를 사용할 때 필요합니다.

이 속성의 값은 UUID입니다. 예를 들어 1abc23d4-567f-8ab9-cde0-1fab234c5d67입니다.

유형: 문자열

필수 여부: 조건부

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`SourceAccessConfigurations`

인증 프로토콜 또는 가상 호스트의 배열입니다. SourceAccessConfigurations 데이터 유형을 사용하여 이를 지정하십시오.

`DocumentDB`이벤트 소스 유형의 경우 유효한 구성 유형은 `BASIC_AUTH`뿐입니다.

- `BASIC_AUTH` - 귀하의 브로커 자격 증명을 저장하는 Secrets Manager 보안 암호입니다. 이 유형의 자격 증명은 다음 `{"username": "your-username", "password": "your-password"}` 형식이어야 합니다. `BASIC_AUTH` 유형의 객체는 오직 한 개만 허용됩니다.

유형: 목록

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `SourceAccessConfigurations` 리소스의 `AWS::Lambda::EventSourceMapping` 속성으로 직접 전달됩니다.

`StartingPosition`

읽기를 시작하는 스트림 내의 위치입니다.

- `AT_TIMESTAMP` - 기록 읽기를 시작할 시간을 지정합니다.
- `LATEST` – 새 기록만 읽습니다.
- `TRIM_HORIZON` – 사용 가능한 모든 기록을 처리합니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `StartingPosition` 리소스의 `AWS::Lambda::EventSourceMapping` 속성으로 직접 전달됩니다.

`StartingPositionTimestamp`

읽기를 시작하는 시간(유닉스 시간 초 단위)입니다. `StartingPositionTimestamp` 언제 `StartingPosition`를 `AT_TIMESTAMP`으로 지정할지 정의합니다.

유형: Double

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `StartingPositionTimestamp` 리소스의 `AWS::Lambda::EventSourceMapping` 속성으로 직접 전달됩니다.

예시

Amazon DocumentDB 이벤트 소스

```
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

DynamoDB 이벤트 소스 유형을 설명하는 객체. 자세한 내용은 AWS Lambda개발자 안내서의 Amazon DynamoDBAWS Lambda 사용을 참조하세요.

이 이벤트 유형이 설정되면 AWS SAM이 AWS::Lambda::EventSourceMapping리소스를 생성합니다.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  BatchSize: Integer
  BisectBatchOnFunctionError: Boolean
  DestinationConfig: DestinationConfig
```

```
  Enabled: Boolean
  FilterCriteria: FilterCriteria
  FunctionResponseTypes: List
  KmsKeyArn: String
  MaximumBatchingWindowInSeconds: Integer
  MaximumRecordAgeInSeconds: Integer
  MaximumRetryAttempts: Integer
  MetricsConfig: MetricsConfig
  ParallelizationFactor: Integer
  StartingPosition: String
  StartingPositionTimestamp: Double
  Stream: String
  TumblingWindowInSeconds: Integer
```

속성

`BatchSize`

한 번의 배치에서 검색하는 최대 항목 수입니다.

유형: 정수

필수 항목 여부: 아니요

기본 = 100

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `BatchSize` 속성으로 직접 전달됩니다.

최소: 1

최대: `1000`

`BisectBatchOnFunctionError`

함수가 오류를 제시하면 비치를 2개로 분할해서 다시 시도합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `BisectBatchOnFunctionError` 속성으로 직접 전달됩니다.

`DestinationConfig`

폐기된 기록을 위한 Amazon Simple Queue Service(Amazon SQS) 대기열 또는 Amazon Simple Notification Service(SNS) 주제 대상입니다.

유형: DestinationConfig

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `DestinationConfig` 속성으로 직접 전달됩니다.

`Enabled`

이벤트 소스 매핑을 비활성화하여 폴링 및 간접 호출을 일시 중지합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `Enabled` 속성으로 직접 전달됩니다.

`FilterCriteria`

Lambda가 이벤트를 처리해야 하는지 결정하는 기준을 정의하는 객체입니다. 자세한 내용은 AWS Lambda 개발자 가이드의 AWS Lambda 이벤트 필터링을 참조하세요.

유형: FilterCriteria

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `FilterCriteria` 속성으로 직접 전달됩니다.

`FunctionResponseTypes`

이벤트 소스 매핑에 현재 적용된 응답 유형의 목록입니다. 자세한 내용은 AWS Lambda Developer Guide의 Reporting batch item failures를 참조하세요.

유효한 값: `ReportBatchItemFailures`

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `FunctionResponseTypes` 속성으로 직접 전달됩니다.

`KmsKeyArn`

이 이벤트와 관련된 정보를 암호화하는 키의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `KmsKeyArn` 속성으로 직접 전달됩니다.

`MaximumBatchingWindowInSeconds`

함수를 호출하기 전에 기록을 수집할 최대 기간(단위: 초)입니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `MaximumBatchingWindowInSeconds` 속성으로 직접 전달됩니다.

`MaximumRecordAgeInSeconds`

Lambda가 처리를 위해 함수에 보내는 기록의 최대 사용 기간입니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `MaximumRecordAgeInSeconds` 속성으로 직접 전달됩니다.

`MaximumRetryAttempts`

함수가 오류를 반환할 때 재시도하는 최대 횟수입니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `MaximumRetryAttempts` 속성으로 직접 전달됩니다.

MetricsConfig

처리 단계별로 캡처하는 이벤트 소스 매핑에 대한 향상된 지표를 얻기 위한 옵트인 구성입니다. 문제 해결 예는 MetricsConfig 이벤트을(를) 참조하세요.

유형: MetricsConfig

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS::Lambda::EventSourceMapping 리소스의 MetricsConfig 속성으로 직접 전달됩니다.

ParallelizationFactor

각 샤드에서 동시에 처리할 배치의 수입니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS::Lambda::EventSourceMapping 리소스의 ParallelizationFactor 속성으로 직접 전달됩니다.

StartingPosition

읽기를 시작하는 스트림 내의 위치입니다.

- AT_TIMESTAMP - 기록 읽기를 시작할 시간을 지정합니다.
- LATEST – 새 기록만 읽습니다.
- TRIM_HORIZON – 사용 가능한 모든 기록을 처리합니다.

유효한 값: AT_TIMESTAMP | LATEST | TRIM_HORIZON

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS::Lambda::EventSourceMapping 리소스의 StartingPosition 속성으로 직접 전달됩니다.

StartingPositionTimestamp

읽기를 시작하는 시간(유닉스 시간 초 단위)입니다. StartingPositionTimestamp 언제 StartingPosition를 AT_TIMESTAMP으로 지정할지 정의합니다.

유형: Double

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 [StartingPositionTimestamp](#) 속성으로 직접 전달됩니다.

`Stream`

DynamoDB 스트림의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 [EventSourceArn](#) 속성으로 직접 전달됩니다.

`TumblingWindowInSeconds`

처리 윈도우 기간(초 단위). 유효한 범위는 1 ~ 900입니다(15분).

자세한 내용은 [개발자 가이드](#)의 AWS LambdaTumbling windows를 참조하세요.

유형: 정수

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 [TumblingWindowInSeconds](#) 속성으로 직접 전달됩니다.

예시

MetricsConfig 이벤트

다음은 이벤트 소스 매핑에 대한 처리 단계별 캡처를 위해 `MetricsConfig` 속성을 사용하는 리소스의 예시입니다.

```
Resources:
  FilteredEventsFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://sam-demo-bucket/metricsConfig.zip
```

```
      Handler: index.handler
      Runtime: nodejs16.x
      Events:
        KinesisStream:
          Type: Kinesis
          Properties:
            Stream: !GetAtt KinesisStream.Arn
            StartingPosition: LATEST
            MetricsConfig:
              Metrics:
              - EventCount
```

기존 DynamoDB 테이블의 DynamoDB 이벤트 소스

AWS 계정에 이미 있는 DynamoDB 테이블의 DynamoDB 이벤트 소스입니다.

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
```

템플릿에 선언된 DynamoDB 테이블에 대한 DynamoDB 이벤트

동일한 템플릿 파일에 선언된 DynamoDB 테이블에 대한 DynamoDB 이벤트입니다.

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

서버리스 함수를 Amazon EventBridge 규칙의 대상으로 설정하는 `EventBridgeRule`이벤트 소스 유형을 설명하는 객체입니다. 자세한 내용은 [Amazon EventBridge 사용자 가이드](#)의 Amazon EventBridge란?을 참조하세요.

AWS SAM은 이 이벤트 유형이 설정되어 있으면 [AWS::Events::Rule](#) 리소스를 생성합니다. AWS SAM은 `EventBridgeRule`이 Lambda를 직접적으로 호출할 수 있도록 필요한 `AWS::Lambda::Permission` 리소스도 생성합니다.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  DeadLetterConfig: DeadLetterConfig
  EventBusName: String
  Input: String
  InputPath: String
  InputTransformer: InputTransformer
  Pattern: EventPattern
  RetryPolicy: RetryPolicy
  RuleName: String
  State: String
  Target: Target
```

속성

`DeadLetterConfig`

대상 호출이 실패한 경우 EventBridge가 이벤트를 전송하는 Amazon Simple Queue Service(Amazon SQS) 대기열을 구성합니다. 예를 들어 존재하지 않는 Lambda 함수로 이벤트를 전송하거나 EventBridge에 Lambda 함수를 간접 호출할 권한이 충분하지 않은 경우 간접 호출이 실패할 수 있습니다. 자세한 내용은 Amazon EventBridge 사용자 가이드의 [이벤트 재시도 정책 및 DLQ(Dead Letter Queue) 사용](#)을 참조하세요.

> **Note**
>
> AWS::Serverless::Function 리소스 유형에는 유사한 데이터 유형인DeadLetterQueue가 있으며, 이는 대상 Lambda 함수를 성공적으로 호출한 후 발생하는 장애를 처리합니다. 이러한 유형의 실패의 예로는 Lambda 제한 또는 Lambda 대상 함수에서 표시되는 오류가 있습니다. 함수 DeadLetterQueue 속성에 대한 자세한 내용은 AWS Lambda 개발자 가이드의 DLQ(Dead Letter Queue)를 참조하세요.

유형: DeadLetterConfig

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `DeadLetterConfig` 데이터 유형의 `AWS::Events::Rule Target` 속성과 유사합니다. DLQ를 자동으로 AWS SAM 생성하고자 하는 경우에 대비하여 이 속성의 AWS SAM 버전에는 추가 하위 속성이 포함되어 있습니다.

`EventBusName`

이 규칙과 연결할 이벤트 버스입니다. 이 속성을 생략하면 AWS SAM은 기본 사항으로서 이벤트 버스를 사용합니다.

유형: 문자열

필수 항목 여부: 아니요

기본값: 기본 이벤트 버스

CloudFormation호환성: 이 속성은 `AWS::Events::Rule` 리소스의 `EventBusName` 속성으로 직접 전달됩니다.

`Input`

대상으로 전달되는 유효한 JSON 텍스트입니다. 이 속성을 사용하면 이벤트 텍스트 자체의 어떤 것도 대상으로 전달되지 않습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Events::Rule Target` 리소스의 `Input` 속성으로 직접 전달됩니다.

`InputPath`

일치된 이벤트 전체를 전달하지 않으려는 경우 `InputPath` 속성을 사용하여 이벤트의 어떤 부분이 전달되어야 하는지 설명하세요.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Events::Rule Target` 리소스의 `InputPath` 속성으로 직접 전달됩니다.

`InputTransformer`

특정 이벤트 데이터를 기반으로 대상에 사용자 지정 입력을 제공할 수 있게 하는 설정입니다. 이벤트에서 하나 이상의 키-값 페어를 추출한 후 이 데이터를 사용하여 대상에 사용자 지정 입력을 전송할 수 있습니다. 자세한 내용은 Amazon EventBridge 사용자 가이드의 Amazon EventBridge input transformation를 참조하세요.

유형: InputTransformer

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `InputTransformer` `AWS::Events::Rule` 데이터 유형의 `Target` 속성에 직접 전달됩니다.

`Pattern`

어떤 이벤트가 지정된 대상으로 라우팅되는지를 설명합니다. 자세한 내용은 Amazon EventBridge 사용자 가이드의 Amazon EventBridge 이벤트 및 EventBridge 이벤트 패턴을 참조하세요.

유형: 이벤트패턴

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `AWS::Events::Rule` 리소스의 `EventPattern` 속성으로 직접 전달됩니다.

`RetryPolicy`

재시도 정책 설정에 대한 정보가 포함된 `RetryPolicy` 객체입니다. 자세한 내용은 Amazon EventBridge 사용자 가이드의 이벤트 재시도 정책 및 DLQ(Dead Letter Queue) 사용을 참조하세요.

유형: [RetryPolicy]

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `RetryPolicy` `AWS::Events::Rule` 데이터 유형의 `Target` 속성에 직접 전달됩니다.

`RuleName`

규칙의 이름입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Events::Rule` 리소스의 `Name` 속성으로 직접 전달됩니다.

`State`

규칙의 상태입니다.

허용되는 값: `DISABLED` | `ENABLED` | `ENABLED_WITH_ALL_CLOUDTRAIL_MANAGEMENT_EVENTS`

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Events::Rule` 리소스의 `State` 속성으로 직접 전달됩니다.

`Target`

규칙이 트리거될 경우 EventBridge가 간접 호출하는 AWS 리소스입니다. 이 속성을 사용하여 대상의 논리적 ID를 지정할 수 있습니다. 이 속성을 지정하지 않으면 AWS SAM이 대상의 논리적 ID를 생성합니다.

유형: [Target]

필수 항목 여부: 아니요

CloudFormation 호환성: 이 속성은 `AWS::Events::Rule` 리소스의 `Targets` 속성과 유사합니다. `Amazon EC2 RebootInstances API call`은 대상 속성의 예입니다. 이 속성의 AWS SAM 버전에서는 단일 대상의 논리적 ID만 지정할 수 있습니다.

예제

EventBridgeRule

다음은 `EventBridgeRule` 이벤트 소스 유형의 예입니다.

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

대상 호출 실패 후 EventBridge가 이벤트를 전송하는 Amazon Simple Queue Service(Amazon SQS) 대기열을 지정하는 데 사용되는 객체입니다. 예를 들어 존재하지 않는 Lambda 함수로 이벤트를 전송하거나 Lambda 함수를 호출할 권한이 충분하지 않은 경우 간접 호출이 실패할 수 있습니다. 자세한 내용은 Amazon EventBridge 사용자 가이드의 이벤트 재시도 정책 및 DLQ(Dead Letter Queue) 사용을 참조하세요.

> **Note**
>
> AWS::Serverless::Function 리소스 유형에는 유사한 데이터 유형인`DeadLetterQueue`가 있으며, 이는 대상 Lambda 함수를 성공적으로 호출한 후 발생하는 장애를 처리합니다. 이러한 유형의 실패의 예로는 Lambda 제한 또는 Lambda 대상 함수에서 반환되는 오류가 있습니다. 함수 `DeadLetterQueue` 속성에 대한 자세한 내용은 AWS Lambda 개발자 가이드의 DLQ(Dead Letter Queue)를 참조하세요.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
Arn: String
QueueLogicalId: String
Type: String
```

속성

`Arn`

DLQ(Dead Letter Queue)의 대상으로 지정된 Amazon SQS 대기열의 Amazon 리소스 이름(ARN)입니다.

> **Note**
> Type 속성 또는 Arn 속성 중 하나만 지정해야 하며, 둘 다 지정할 수는 없습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `Arn` `AWS::Events::Rule` 데이터 유형의 `DeadLetterConfig` 속성에 직접 전달됩니다.

`QueueLogicalId`

Type가 지정된 경우 AWS SAM이 생성하는 DLQ의 사용자 지정 이름입니다.

> **Note**
> Type 속성이 설정되지 않은 경우 이 속성은 무시됩니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

Type

대기열 유형. 이 속성이 설정되면 AWS SAM은 DLQ를 자동으로 만들고 허가를 부여하기 위해 필요한 [리소스 기반 정책]을 규칙 리소스에 첨부하여 해당 큐에 이벤트를 전송합니다.

> **Note**
>
> `Type` 속성 또는 `Arn` 속성 중 하나만 지정해야 하며, 둘 다 지정할 수는 없습니다.

유효한 값: `SQS`

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예제

DeadLetterConfig

DeadLetterConfig

YAML

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

Target

규칙이 트리거될 경우 EventBridge가 호출하는 AWS 리소스를 구성합니다.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
Id: String
```

속성

Id

대상의 논리적 ID.

Id의 값은 영숫자 문자, 마침표(.), 하이픈(-), 밑줄(_)을 포함할 수 있습니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 Id AWS::Events::Rule 데이터 유형의 Target 속성에 직접 전달됩니다.

예제

대상

YAML

```
EBRule:
  Type: EventBridgeRule
  Properties:
    Target:
      Id: MyTarget
```

HttpApi

HttpAPI 유형의 이벤트 소스를 설명하는 객체입니다.

지정된 경로 및 메서드에 대한 OpenAPI 정의가 API에 있는 경우 SAM은 Lambda 통합 및 보안 섹션(해당하는 경우)을 자동으로 추가합니다.

API에 지정된 경로와 메서드에 대한 OpenApi 정의가 없는 경우 SAM이 이 정의를 자동으로 생성합니다.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
ApiId: String
Auth: HttpApiFunctionAuth
Method: String
Path: String
PayloadFormatVersion: String
RouteSettings: RouteSettings
TimeoutInMillis: Integer
```

속성

`ApiId`

이 템플릿에 정의된 AWS::Serverless::HttpApi 리소스의 식별자입니다.

정의되지 않은 경우, AWS::Serverless::HttpApi을 정의하지 않는 이 템플릿에 정의된 Api 이벤트에 의해 정의된 모든 경로와 메서드의 집합을 포함하는 생성된 OpenAPI 문서를 사용하여 `ServerlessHttpApi`라는 명칭의 기본 `ApiId` 리소스가 생성됩니다.

이것은 다른 템플릿에 정의된 AWS::Serverless::HttpApi 리소스를 참조할 수 없습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Auth`

이 특정 Api+Path+Method에 대한 인증 구성입니다.

지정된 `DefaultAuthorizer`가 없는 경우, API의 `DefaultAuthorizer`을 재정의하거나 개별 경로에 인증 구성을 설정하는 데 유용합니다.

유형: http://API/FunctionAuth

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Method`

이 함수가 간접 호출되는 HTTP 메서드입니다.

`Path`이나 `Method` 어느 것도 지정되지 않은 경우, SAM은 다른 엔드포인트에 매핑되지 않는 모든 요청을 이 Lambda 함수로 라우팅하는 기본 API 경로를 생성합니다. API당 이러한 기본 경로 중 하나만 존재할 수 있습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Path`

이 함수가 호출되는 Uri 경로입니다. `/`로 시작해야 합니다.

`Path`이나 `Method` 어느 것도 지정되지 않은 경우, SAM은 다른 엔드포인트에 매핑되지 않는 모든 요청을 이 Lambda 함수로 라우팅하는 기본 API 경로를 생성합니다. API당 이러한 기본 경로 중 하나만 존재할 수 있습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`PayloadFormatVersion`

통합으로 전송되는 페이로드의 형식을 지정합니다.

참고: PayloadFormatVersion을 사용하려면 SAM이 OpenAPI 정의를 수정해야 하므로 `DefinitionBody` 속성에 정의된 인라인 OpenApi와만 작동합니다.

유형: 문자열

필수 항목 여부: 아니요

기본값: 2.0

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`RouteSettings`

이 HTTP API의 경로별 경로 설정입니다. 경로 설정에 대한 자세한 내용은 API 게이트웨이 개발자 가이드의 AWS::API 게이트웨이V2::StageRouteSettings을 참조하세요.

참고: RouteSettings가 HttpApi 리소스와 이벤트 소스 모두에 지정된 경우 AWS SAM은 이들을 우선 순위가 있는 이벤트 소스 속성과 병합합니다.

유형: RouteSettings

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::ApiGatewayV2::Stage` 리소스의 `RouteSettings` 속성으로 직접 전달됩니다.

`TimeoutInMillis`

50~29,000밀리초 사이의 제한 시간 사용자 지정입니다.

참고: TimeoutInMillis를 사용하려면 SAM이 OpenAPI 정의를 수정해야 하므로 `DefinitionBody` 속성에 정의된 인라인 OpenApi와만 작동합니다.

유형: 정수

필수 항목 여부: 아니요

기본값: 5000

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

기본 HttpApi 이벤트

기본 경로를 사용하는 HttpApi 이벤트입니다. 이 API의 매핑되지 않은 모든 경로와 메서드는 이 엔드포인트로 라우팅됩니다.

YAML

```
Events:
  HttpApiEvent:
```

```
    Type: HttpApi
```

HttpApi

특정 경로와 메서드를 사용하는 HttpApi 이벤트입니다.

YAML

```
Events:
  HttpApiEvent:
    Type: HttpApi
    Properties:
      Path: /
      Method: GET
```

HttpApi 승인

권한 부여자를 사용하는HttpApi 이벤트입니다.

YAML

```
Events:
  HttpApiEvent:
    Type: HttpApi
    Properties:
      Path: /authenticated
      Method: GET
      Auth:
        Authorizer: OpenIdAuth
        AuthorizationScopes:
          - scope1
          - scope2
```

HttpApiFunctionAuth

이벤트 수준에서 권한 부여를 구성합니다.

특정 API + 경로 + 방법에 대한 권한 부여 구성

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
AuthorizationScopes: List
Authorizer: String
```

속성

AuthorizationScopes

이 API, 경로 및 메서드에 적용할 수 있는 권한 부여 범위.

여기에 나열된 범위는 DefaultAuthorizer이(존재한다면) 적용한 어떤 범위보다도 우선합니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

Authorizer

특정 함수를 위한 Authorizer IAM 인증을 사용하려면 AWS_IAM을 지정하고, 템플릿의 true 섹션에서 EnableIamAuthorizer에 관하여 Globals를 지정하십시오.

API에서 글로벌 권한 부여자를 지정한 상태에서 특정 함수를 공개하려면 Authorizer을 NONE로 설정하여 변경하십시오.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

함수-권한 부여

함수 수준에서 권한 부여

YAML

```
Auth:
```

```
  Authorizer: OpenIdAuth
  AuthorizationScopes:
    - scope1
    - scope2
```

IAM 권한 부여

이벤트 수준에서 IAM 권한 부여를 지정합니다. 이벤트 수준에서 AWS_IAM 권한 부여를 사용하려면 템플릿의 true 섹션에서 EnableIamAuthorizer에 관하여 Globals를 지정해야 합니다. 자세한 내용은 AWS SAM 템플릿의 Globals 섹션을 참조하십시오.

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

IoTRule 이벤트 소스 유형을 설명하는 객체.

AWS::IoT::TopicRule규칙을 선언하기 위한 AWS IoT리소스를 만듭니다. 자세한 내용은 CloudFormation 설명서를 참조하세요

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
AwsIotSqlVersion: String
Sql: String
```

속성

`AwsIotSqlVersion`

규칙을 평가할 때 사용할 SQL 규칙의 버전입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AwsIotSqlVersion` 리소스의 `AWS::IoT::TopicRule` `TopicRulePayload` 속성으로 직접 전달됩니다.

`Sql`

주제에 대한 쿼리에 사용되는 SQL 문입니다. 자세한 내용을 알아보려면 AWS IoT개발자 가이드의 AWS IoTSQL 참조에서 참조하세요.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `Sql` 리소스의 `AWS::IoT::TopicRule` `TopicRulePayload` 속성으로 직접 전달됩니다.

예시

IOT 규칙

IOT 규칙 예제

YAML

```
IoTRule:
```

```
  Type: IoTRule
  Properties:
    Sql: SELECT * FROM 'topic/test'
```

Kinesis

Kinesis 이벤트 소스 유형을 설명하는 객체. 자세한 내용은 개발자 가이드의 AWS LambdaAmazon Kinesis와AWS Lambda 함께 사용하기를 참조하세요.

이 이벤트 유형이 설정되면 AWS SAM이 AWS::Lambda::EventSourceMapping리소스를 생성합니다.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  BatchSize: Integer
  BisectBatchOnFunctionError: Boolean
  DestinationConfig: DestinationConfig
  Enabled: Boolean
  FilterCriteria: FilterCriteria
  FunctionResponseTypes: List
  KmsKeyArn: String
  MaximumBatchingWindowInSeconds: Integer
  MaximumRecordAgeInSeconds: Integer
  MaximumRetryAttempts: Integer
  MetricsConfig: MetricsConfig
  ParallelizationFactor: Integer
  StartingPosition: String
  StartingPositionTimestamp: Double
  Stream: String
  TumblingWindowInSeconds: Integer
```

속성

BatchSize

한 번의 배치에서 검색하는 최대 항목 수입니다.

유형: 정수

필수 항목 여부: 아니요

기본 = 100

CloudFormation호환성: 이 속성은 AWS::Lambda::EventSourceMapping 리소스의 BatchSize 속성으로 직접 전달됩니다.

최소: 1

최대: 10000

BisectBatchOnFunctionError

함수가 오류를 제시하면 비치를 2개로 분할해서 다시 시도합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS::Lambda::EventSourceMapping 리소스의 BisectBatchOnFunctionError 속성으로 직접 전달됩니다.

DestinationConfig

폐기된 기록을 위한 Amazon Simple Queue Service(Amazon SQS) 대기열 또는 Amazon Simple Notification Service(SNS) 주제 대상입니다.

유형: DestinationConfig

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS::Lambda::EventSourceMapping 리소스의 DestinationConfig 속성으로 직접 전달됩니다.

Enabled

이벤트 소스 매핑을 비활성화하여 폴링 및 간접 호출을 일시 중지합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS::Lambda::EventSourceMapping 리소스의 Enabled 속성으로 직접 전달됩니다.

`FilterCriteria`

Lambda가 이벤트를 처리해야 하는지 결정하는 기준을 정의하는 객체입니다. 자세한 내용은 AWS Lambda 개발자 가이드의 AWS Lambda 이벤트 필터링을 참조하세요.

유형: FilterCriteria

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `FilterCriteria` 속성으로 직접 전달됩니다.

`FunctionResponseTypes`

이벤트 소스 매핑에 현재 적용된 응답 유형의 목록입니다. 자세한 내용은 AWS Lambda Developer Guide의 Reporting batch item failures를 참조하세요.

유효한 값: `ReportBatchItemFailures`

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `FunctionResponseTypes` 속성으로 직접 전달됩니다.

`KmsKeyArn`

이 이벤트와 관련된 정보를 암호화하는 키의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `KmsKeyArn` 속성으로 직접 전달됩니다.

`MaximumBatchingWindowInSeconds`

함수를 호출하기 전에 기록을 수집할 최대 기간(단위: 초)입니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 [`MaximumBatchingWindowInSeconds`] 속성으로 직접 전달됩니다.

`MaximumRecordAgeInSeconds`

Lambda가 처리를 위해 함수에 보내는 기록의 최대 사용 기간입니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 [`MaximumRecordAgeInSeconds`] 속성으로 직접 전달됩니다.

`MaximumRetryAttempts`

함수가 오류를 반환할 때 재시도하는 최대 횟수입니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 [`MaximumRetryAttempts`] 속성으로 직접 전달됩니다.

`MetricsConfig`

처리 단계별로 캡처하는 이벤트 소스 매핑에 대한 향상된 지표를 얻기 위한 옵트인 구성입니다. 문제 해결 예는 [MetricsConfig 이벤트]을(를) 참조하세요.

유형: [MetricsConfig]

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 [`MetricsConfig`] 속성으로 직접 전달됩니다.

`ParallelizationFactor`

각 샤드에서 동시에 처리할 배치의 수입니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS::Lambda::EventSourceMapping 리소스의 [ParallelizationFactor] 속성으로 직접 전달됩니다.

StartingPosition

읽기를 시작하는 스트림 내의 위치입니다.

- AT_TIMESTAMP - 기록 읽기를 시작할 시간을 지정합니다.
- LATEST – 새 기록만 읽습니다.
- TRIM_HORIZON – 사용 가능한 모든 기록을 처리합니다.

유효한 값: AT_TIMESTAMP | LATEST | TRIM_HORIZON

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS::Lambda::EventSourceMapping 리소스의 [StartingPosition] 속성으로 직접 전달됩니다.

StartingPositionTimestamp

읽기를 시작하는 시간(유닉스 시간 초 단위)입니다. StartingPositionTimestamp 언제 StartingPosition를 AT_TIMESTAMP으로 지정할지 정의합니다.

유형: Double

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS::Lambda::EventSourceMapping 리소스의 [StartingPositionTimestamp] 속성으로 직접 전달됩니다.

Stream

데이터 스트림 혹은 스트림 컨슈머의 Amazon 리소스 이름(ARN).

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS::Lambda::EventSourceMapping 리소스의 [EventSourceArn] 속성으로 직접 전달됩니다.

TumblingWindowInSeconds

처리 윈도우 기간(초 단위). 유효한 범위는 1 ~ 900입니다(15분).

자세한 내용은 개발자 가이드의 AWS LambdaTumbling windows를 참조하세요.

유형: 정수

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `TumblingWindowInSeconds` 속성으로 직접 전달됩니다.

예제

MetricsConfig 이벤트

다음은 이벤트 소스 매핑에 대한 처리 단계별 캡처를 위해 `MetricsConfig` 속성을 사용하는 리소스의 예시입니다.

```
Resources:
  FilteredEventsFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://sam-demo-bucket/metricsConfig.zip
      Handler: index.handler
      Runtime: nodejs16.x
      Events:
        KinesisStream:
          Type: Kinesis
          Properties:
            Stream: !GetAtt KinesisStream.Arn
            StartingPosition: LATEST
            MetricsConfig:
              Metrics:
              - EventCount
```

Kinesis 이벤트 소스

다음은 Kinesis 이벤트 소스의 예입니다.

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

MQ 이벤트 소스 유형을 설명하는 객체. 자세한 내용은 개발자 가이드의 AWS Lambda Amazon MQ로 Lambda 이용하기를 참조하세요.

AWS Serverless Application Model(AWS SAM)은 이 이벤트 유형이 설정되면 AWS::Lambda::EventSourceMapping리소스를 생성합니다.

> **Note**
>
> 공개 네트워크의 Lambda 함수에 연결되는 Virtual Private Cloud(VPC) 에 Amazon MQ 대기열을 두려면 함수의 실행 역할에 다음 권한이 포함되어야 합니다.
>
> - `ec2:CreateNetworkInterface`
> - `ec2:DeleteNetworkInterface`
> - `ec2:DescribeNetworkInterfaces`
> - `ec2:DescribeSecurityGroups`
> - `ec2:DescribeSubnets`
> - `ec2:DescribeVpcs`
>
> 자세한 내용은 개발자 가이드의 AWS Lambda 실행 역할 권한을 참조하세요.

구문

귀하의 AWS SAM 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  BatchSize: Integer
```

```
  Broker: String
  DynamicPolicyName: Boolean
  Enabled: Boolean
  FilterCriteria: FilterCriteria
  KmsKeyArn: String
  MaximumBatchingWindowInSeconds: Integer
  Queues: List
  SecretsManagerKmsKeyId: String
  SourceAccessConfigurations: List
```

속성

BatchSize

한 번의 배치에서 검색하는 최대 항목 수입니다.

유형: 정수

필수 항목 여부: 아니요

기본 = 100

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `BatchSize` 속성으로 직접 전달됩니다.

최소: 1

최대: `10000`

Broker

Amazon MQ 브로커의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `EventSourceArn` 리소스의 `AWS::Lambda::EventSourceMapping` 속성으로 직접 전달됩니다.

DynamicPolicyName

기본적으로 AWS Identity and Access Management(IAM) 정책 이름은 `SamAutoGeneratedAMQPolicy`이며 이전 버전과의 호환성을 위한 것입니다. IAM 정책을 위해

자동 생성된 이름을 사용하도록 `true`을 지정합니다. 이 이름에는 Amazon MQ 이벤트 소스 논리 ID가 포함됩니다.

> **Note**
>
> 복수의 Amazon MQ 이벤트 소스를 사용하는 경우, IAM 정책 이름이 중복되지 않도록 `true`을 지정하십시오.

유형: 부울

필수 항목 여부: 아니요

기본값: `false`

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Enabled`

`true`이면, 이벤트 소스 매핑이 활성 상태입니다. 폴링 및 간접 호출을 일시 중지하려면 `false`로 설정합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `Enabled` 속성으로 직접 전달됩니다.

`FilterCriteria`

Lambda가 이벤트를 처리해야 하는지 결정하는 기준을 정의하는 객체입니다. 자세한 내용은 AWS Lambda 개발자 가이드의 AWS Lambda 이벤트 필터링을 참조하세요.

유형: FilterCriteria

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `FilterCriteria` 속성으로 직접 전달됩니다.

`KmsKeyArn`

이 이벤트와 관련된 정보를 암호화하는 키의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `KmsKeyArn` 속성으로 직접 전달됩니다.

`MaximumBatchingWindowInSeconds`

함수를 호출하기 전에 기록을 수집할 최대 기간(단위: 초)입니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `MaximumBatchingWindowInSeconds` 속성으로 직접 전달됩니다.

`Queues`

소비할 Amazon MQ 브로커 대상 대기열의 이름입니다.

유형: 목록

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `Queues` 리소스의 `AWS::Lambda::EventSourceMapping` 속성으로 직접 전달됩니다.

`SecretsManagerKmsKeyId`

AWS Key Management Service에서 고객 관리 키의 AWS KMS(AWS Secrets Manager) 키 ID를 입력합니다. `kms:Decrypt` 권한이 포함되지 않은 Lambda 실행 역할과 함께 Secrets Manager의 고객 관리형 키를 사용할 때 필요합니다.

이 속성의 값은 UUID입니다. 예를 들어 `1abc23d4-567f-8ab9-cde0-1fab234c5d67`입니다.

유형: 문자열

필수 항목 여부: 조건부

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`SourceAccessConfigurations`

권한 부여 프로토콜 또는 가상 호스트의 배열입니다. SourceAccessConfigurations 데이터 유형을 사용하여 이를 지정하십시오.

MQ 이벤트 소스 유형의 경우 유효한 구성 유형은 BASIC_AUTH 및 VIRTUAL_HOST뿐입니다.

- **BASIC_AUTH** - 귀하의 브로커 자격 증명을 저장하는 Secrets Manager 보안 암호입니다. 이 유형의 자격 증명은 다음 {"username": "your-username", "password": "your-password"} 형식이어야 합니다. BASIC_AUTH 유형의 객체는 오직 한 개만 허용됩니다.
- **VIRTUAL_HOST** - RabbitMQ 브로커에 있는 가상 호스트의 이름입니다. Lambda는 이 Rabbit MQ의 호스트를 이벤트 소스로 사용합니다. VIRTUAL_HOST 유형의 객체는 오직 한 개만 허용됩니다.

유형: 목록

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `SourceAccessConfigurations` 리소스의 `AWS::Lambda::EventSourceMapping` 속성으로 직접 전달됩니다.

예시

Amazon MQ 이벤트 소스

다음은 Amazon MQ 브로커의 MQ 이벤트 소스 유형 예시입니다.

YAML

```
Events:
  MQEvent:
    Type: MQ
    Properties:
      Broker: arn:aws:mq:us-
east-2:123456789012:broker:MyBroker:b-1234a5b6-78cd-901e-2fgh-3i45j6k178l9
      Queues: List of queues
      SourceAccessConfigurations:
        - Type: BASIC_AUTH
          URI: arn:aws:secretsmanager:us-east-1:01234567890:secret:MyBrokerSecretName
      BatchSize: 200
      Enabled: true
```

MSK

MSK 이벤트 소스 유형을 설명하는 객체. 자세한 내용은 AWS Lambda 개발자 안내서의 Amazon MSK AWS Lambda 에서 사용을 참조하세요.

AWS Serverless Application Model (AWS SAM)는이 이벤트 유형이 설정되면 AWS::Lambda::EventSourceMapping 리소스를 생성합니다.

Schema Registry를 사용하려면 함수에 대한 특정 IAM 역할 권한을 정의해야 합니다. 필수 구성의 예시는 IAM 역할로 설정 완료하기를 참조하세요.

구문

AWS SAM 템플릿에서이 개체를 선언하려면 다음 구문을 사용합니다.

YAML

```
BatchSize: Integer
BisectBatchOnFunctionError: Boolean
ConsumerGroupId: String
DestinationConfig: DestinationConfig
Enabled: Boolean
FilterCriteria: FilterCriteria
FunctionResponseTypes: List
KmsKeyArn: String
MaximumBatchingWindowInSeconds: Integer
MaximumRecordAgeInSeconds: Integer
MaximumRetryAttempts: Integer
ProvisionedPollerConfig: ProvisionedPollerConfig
SchemaRegistryConfig: SchemaRegistryConfig
SourceAccessConfigurations: SourceAccessConfigurations
StartingPosition: String
StartingPositionTimestamp: Double
Stream: String
Topics: List
```

속성

`BatchSize`

Lambda가 스트림 또는 대기열에서 폴링하여 함수로 보내는 각 배치의 최대 레코드 수입니다. Lambda는 동기식 간접 호출에 대한 페이로드 제한(6MB)까지 단일 호출로 배치의 모든 레코드를 함수로 전달합니다.

기본값: 100

유효 범위: 최소값 1. 최대값은 10,000입니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 [`BatchSize`] 속성으로 직접 전달됩니다.

`BisectBatchOnFunctionError`

함수가 오류를 제시하면 비치를 2개로 분할해서 다시 시도합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 [`BisectBatchOnFunctionError`] 속성으로 직접 전달됩니다.

`ConsumerGroupId`

Kafka 주제에서 이벤트를 읽는 방법을 구성하는 문자열입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 [`AmazonManagedKafkaConfiguration`] 속성으로 직접 전달됩니다.

`DestinationConfig`

Lambda가 이벤트를 처리한 후 이벤트의 대상을 지정하는 구성 객체입니다.

이 속성을 사용하여 Amazon MSK 이벤트 소스에서 실패한 간접 호출의 대상을 지정합니다.

유형: [DestinationConfig]

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 [`DestinationConfig`] 속성으로 직접 전달됩니다.

`Enabled`

이벤트 소스 매핑을 비활성화하여 폴링 및 간접 호출을 일시 중지합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `Enabled` 속성으로 직접 전달됩니다.

`FilterCriteria`

Lambda가 이벤트를 처리해야 하는지 결정하는 기준을 정의하는 객체입니다. 자세한 내용은 AWS Lambda 개발자 가이드의 AWS Lambda 이벤트 필터링을 참조하세요.

유형: FilterCriteria

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `FilterCriteria` 속성으로 직접 전달됩니다.

`FunctionResponseTypes`

이벤트 소스 매핑에 현재 적용된 응답 유형의 목록입니다. 자세한 내용은 AWS Lambda Developer Guide의 Reporting batch item failures를 참조하세요.

유효한 값: `ReportBatchItemFailures`

유형: 목록

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `FunctionResponseTypes` 속성으로 직접 전달됩니다.

`KmsKeyArn`

이 이벤트와 관련된 정보를 암호화하는 키의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 [`KmsKeyArn`] 속성으로 직접 전달됩니다.

`MaximumBatchingWindowInSeconds`

함수를 호출하기 전에 기록을 수집할 최대 기간(단위: 초)입니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 [`MaximumBatchingWindowInSeconds`] 속성으로 직접 전달됩니다.

`MaximumRecordAgeInSeconds`

Lambda가 처리를 위해 함수에 보내는 기록의 최대 사용 기간입니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 [`MaximumRecordAgeInSeconds`] 속성으로 직접 전달됩니다.

`MaximumRetryAttempts`

함수가 오류를 반환할 때 재시도하는 최대 횟수입니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 [`MaximumRetryAttempts`] 속성으로 직접 전달됩니다.

`ProvisionedPollerConfig`

이벤트 소스 매핑을 컴퓨팅하는 데 사용되는 폴러의 양을 늘리기 위한 구성입니다. 이 구성은 최소 1개의 폴러와 최대 2,000개의 폴러를 허용합니다. 예시는 ProvisionedPollerConfig 예시에서 확인하세요.

유형: ProvisionedPollerConfig

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 [`ProvisionedPollerConfig`] 속성으로 직접 전달됩니다.

`SchemaRegistryConfig`

Kafka 이벤트 소스와 스키마 레지스트리를 사용하기 위한 구성입니다.

> **Note**
> 이 기능에는 `ProvisionedPollerConfig` 구성이 필요합니다.

유형: SchemaRegistryConfig

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `AmazonManagedKafkaEventSourceConfig` 속성으로 직접 전달됩니다.

`SourceAccessConfigurations`

이벤트 소스를 보호하고 정의하는 데 사용되는 인증 프로토콜, VPC 구성 요소 또는 가상 호스트의 배열입니다.

유효한 값: `CLIENT_CERTIFICATE_TLS_AUTH`

유형: SourceAccessConfiguration 목록

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 AmazonManagedKafkaEventSourceConfig 속성의 일부입니다.

`StartingPosition`

읽기를 시작하는 스트림 내의 위치입니다.

- `AT_TIMESTAMP` - 기록 읽기를 시작할 시간을 지정합니다.
- `LATEST` – 새 기록만 읽습니다.
- `TRIM_HORIZON` – 사용 가능한 모든 기록을 처리합니다.

유효한 값: AT_TIMESTAMP | LATEST | TRIM_HORIZON

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 AWS::Lambda::EventSourceMapping 리소스의 StartingPosition 속성으로 직접 전달됩니다.

StartingPositionTimestamp

읽기를 시작하는 시간(유닉스 시간 초 단위)입니다. StartingPositionTimestamp 언제 StartingPosition를 AT_TIMESTAMP으로 지정할지 정의합니다.

유형: Double

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 AWS::Lambda::EventSourceMapping 리소스의 StartingPositionTimestamp 속성으로 직접 전달됩니다.

Stream

데이터 스트림 혹은 스트림 컨슈머의 Amazon 리소스 이름(ARN).

유형: 문자열

필수 항목 여부: 예

CloudFormation 호환성:이 속성은 AWS::Lambda::EventSourceMapping 리소스의 EventSourceArn 속성으로 직접 전달됩니다.

Topics

Kafka 주제의 이름입니다.

유형: 목록

필수 항목 여부: 예

CloudFormation 호환성:이 속성은 AWS::Lambda::EventSourceMapping 리소스의 Topics 속성으로 직접 전달됩니다.

예제

IAM 역할로 설정 완료

다음 예시는 Schema Registry 사용을 위해 필요한 IAM 역할 구성을 포함한 완료된 설정을 보여줍니다.

```
Parameters:
  PreCreatedSubnetOne:
    Type: String
  PreCreatedSubnetTwo:
    Type: String
  MskClusterName4:
    Type: String

Resources:
  MyLambdaExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Action: [sts:AssumeRole]
          Effect: Allow
          Principal:
            Service: [lambda.amazonaws.com]
      Policies:
      - PolicyName: KafkaClusterPermissions
        PolicyDocument:
          Statement:
          - Action: [kafka:DescribeClusterV2, kafka:GetBootstrapBrokers]
            Effect: Allow
            Resource: 'arn:aws:kafka:us-east-1:123456789012:cluster/*'
      - PolicyName: KafkaAuthPolicy
        PolicyDocument:
          Statement:
          - Action: [secretsmanager:GetSecretValue, kms:Decrypt]
            Effect: "Allow"
            Resource: ['arn:aws:secretsmanager:us-
west-2:123456789012:secret:kafkaSecret-******',
                        'arn:aws:kms:us-west-2:123456789012:key/keyId']
      - PolicyName: ENIPolicy
        PolicyDocument:
          Statement:
          - Action: [ec2:CreateNetworkInterface,
```

```
              ec2:DescribeNetworkInterfaces, ec2:DescribeVpcs,
 ec2:DeleteNetworkInterface,
              ec2:DescribeSubnets, ec2:DescribeSecurityGroups]
            Effect: Allow
            Resource: '*'
      - PolicyName: SchemaRegistryPolicy
        PolicyDocument:
          Statement:
          - Action: [glue:GetRegistry]
            Effect: Allow
            Resource: 'arn:aws:glue:{region}:{account-id}:registry/{registry-name}'
      - PolicyName: SchemaVersionsPolicy
        PolicyDocument:
          Statement:
          - Action: [glue:GetSchemaVersions]
            Effect: Allow
            Resource: '*'
      ManagedPolicyArns:
      - !Sub arn:${AWS::Partition}:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole
      Tags:
      - {Value: SAM, Key: lambda:createdBy}

  MyMskCluster:
    Type: AWS::MSK::Cluster
    Properties:
      BrokerNodeGroupInfo:
        ClientSubnets:
        - Ref: PreCreatedSubnetOne
        - Ref: PreCreatedSubnetTwo
        InstanceType: kafka.t3.small
        StorageInfo:
          EBSStorageInfo:
            VolumeSize: 1
      ClusterName:
        Ref: MskClusterName4
      KafkaVersion: 3.8.x
      NumberOfBrokerNodes: 2

  MyMskStreamProcessor:
    Type: AWS::Serverless::Function
    Properties:
      Runtime: nodejs18.x
      Handler: index.handler
```

```
      CodeUri: ${codeuri}
      Role:
        Fn::GetAtt: [MyLambdaExecutionRole, Arn]
      Events:
        MyMskEvent:
          Type: MSK
          Properties:
            StartingPosition: LATEST
            Stream:
              Ref: MyMskCluster
            SourceAccessConfigurations:
            - Type: SASL_SCRAM_512_AUTH
              URI: !Sub arn:${AWS::Partition}:secretsmanager:us-
west-2:123456789012:secret:my-path/my-secret-name-1a2b3c
            Topics:
            - SchemaRegistryTestTopic
            ProvisionedPollerConfig:
              MinimumPollers: 1
            SchemaRegistryConfig:
              AccessConfigs:
              - Type: BASIC_AUTH
                URI: !Sub arn:${AWS::Partition}:secretsmanager:us-
west-2:123456789012:secret:my-path/my-secret-name-1a2b3c
              SchemaValidationConfigs:
              - Attribute: KEY
              EventRecordFormat: JSON
              SchemaRegistryURI: !Sub arn:${AWS::Partition}:glue:us-
west-2:123456789012:registry/myregistry
```

ProvisionedPollerConfig 예시

```
ProvisionedPollerConfig:
  MinimumPollers: 1
  MaximumPollers: 200
```

기존 클러스터를 위한 Amazon MSK 예제

다음은 이미 AWS 계정에 존재하는 Amazon MSK 클러스터의 MSK 이벤트 소스 유형의 예입니다.

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
```

동일한 템플릿에 선언된 클러스터에 대한 Amazon MSK 예제

다음은 동일한 템플릿 파일에 선언된 Amazon MSK 클러스터의 MSK 이벤트 소스 유형입니다.

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
```

Schema Registry를 사용한 MSK 이벤트 소스

다음은 Schema Registry로 구성된 MSK 이벤트 소스 유형의 예시입니다.

```
Events:
  MSKEvent:
    Type: MSK
    Properties:
      StartingPosition: LATEST
      Stream:
        Ref: MyMskCluster
      Topics:
        - SchemaRegistryTestTopic
      ProvisionedPollerConfig:
        MinimumPollers: 1
      SchemaRegistryConfig:
        SchemaRegistryURI: !Sub arn:${AWS::Partition}:glue:us-
west-2:123456789012:registry/myregistry
```

```
        EventRecordFormat: JSON
        SchemaValidationConfigs:
          - Attribute: KEY
          - Attribute: VALUE
```

Confluent Schema Registry를 사용한 MSK 이벤트 소스

다음은 Confluent Schema Registry로 구성된 MSK 이벤트 소스 유형의 예시입니다.

```
Events:
  MSKEvent:
    Type: MSK
    Properties:
      StartingPosition: LATEST
      Stream:
        Ref: MyMskCluster
      Topics:
        - SchemaRegistryTestTopic
      ProvisionedPollerConfig:
        MinimumPollers: 1
      SchemaRegistryConfig:
        SchemaRegistryURI: https://my-schema-registry.confluent.cloud
        AccessConfigs:
          - Type: BASIC_AUTH
            URI: !Sub arn:${AWS::Partition}:secretsmanager:us-
west-2:123456789012:secret:my-secret
        EventRecordFormat: JSON
        SchemaValidationConfigs:
          - Attribute: KEY
          - Attribute: VALUE
```

S3

S3 이벤트 소스 유형을 설명하는 객체.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  Bucket: String
```

```
  Events: String | List
  Filter: NotificationFilter
```

속성

`Bucket`

S3 버킷 이름 이 버킷은 같은 템플릿 내에 있어야 합니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `AWS::S3::Bucket` 리소스의 `BucketName` 속성과 유사합니다. 이것은 SAM에 속한 필수적 필드입니다. 이 필드는 이 템플릿에서 생성된 S3 버킷에 대한 참조만 허용합니다.

`Events`

Lambda 함수의 호출 목적이 되는 Amazon S3 버킷 이벤트입니다. 유효한 값 목록은 Amazon S3 지원 이벤트 유형을 참조하세요

형식: 문자열 | 목록

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `Event` `AWS::S3::Bucket` 데이터 유형의 `LambdaConfiguration` 속성에 직접 전달됩니다.

`Filter`

어느 Amazon S3 객체가 Lambda 함수를 호출할 것인지 결정하는 필터링 규칙입니다. Amazon S3 키 이름 필터링에 대한 자세한 내용은 Amazon Simple Storage Service 개발자 안내서의 Amazon S3 이벤트 알림 구성을 참조하세요.

유형: NotificationFilter

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `Filter` `AWS::S3::Bucket` 데이터 유형의 `LambdaConfiguration` 속성에 직접 전달됩니다.

예시

S3-이벤트

S3 이벤트의 예시.

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

`Schedule` 이벤트 소스 유형을 설명하는 객체로, 이는 귀하의 서버리스 함수를 일정에 따라 트리거되는 Amazon EventBridge 규칙의 대상으로 설정합니다. 자세한 내용은 Amazon EventBridge 사용자 가이드의 What Is Amazon EventBridge?(Amazon EventBridge란?)을 참조하십시오.

AWS Serverless Application Model(AWS SAM)은 이 이벤트 유형이 설정되면 AWS::Events::Rule리소스를 생성합니다.

> **Note**
>
> EventBridge는 이제 새로운 예약 기능인 Amazon EventBridgeScheduler를 제공합니다. Amazon EventBridge Scheduler은 하나의 중앙 관리형 서비스에서 작업을 생성, 실행 및 관리할 수 있는 서버리스 스케줄러입니다. EventBridge Scheduler은 고도로 사용자 지정이 가능하며, 더 광범위한 대상 API 작업 세트 및 AWS 서비스를 갖춤으로써 EventBridge의 예정된 규칙보다 향상된 확장성을 제공합니다.
> 일정에 따라 EventBridge Scheduler을 사용하여 대상을 간접 호출하는 것이 좋습니다. 귀하의 AWS SAM 템플릿에서 이 이벤트 소스 유형을 정의하려면 ScheduleV2를 참조하세요.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
DeadLetterConfig: DeadLetterConfig
Description: String
Enabled: Boolean
Input: String
Name: String
RetryPolicy: RetryPolicy
Schedule: String
State: String
```

속성

`DeadLetterConfig`

대상 호출이 실패한 경우 EventBridge가 이벤트를 전송하는 Amazon Simple Queue Service(Amazon SQS) 대기열을 구성합니다. 예를 들어 존재하지 않는 Lambda 함수로 이벤트를 전송하거나 EventBridge에 Lambda 함수를 호출할 권한이 충분하지 않은 경우 호출이 실패할 수 있습니다. 자세한 내용은 Amazon EventBridge 사용자 가이드의 이벤트 재시도 정책 및 DLQ(Dead Letter Queue) 사용을 참조하세요.

> **Note**
>
> AWS::Serverless::Function 리소스 유형에는 유사한 데이터 유형인`DeadLetterQueue`가 있으며, 이는 대상 Lambda 함수를 성공적으로 호출한 후 발생하는 장애를 처리합니다. 이러한 유형의 실패의 예로는 Lambda 제한 또는 Lambda 대상 함수에서 표시되는 오류가 있습니다. 함수 `DeadLetterQueue` 속성에 대한 자세한 내용은 AWS Lambda 개발자 가이드의 DLQ(Dead Letter Queue)를 참조하세요.

유형: DeadLetterConfig

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 DeadLetterConfig 데이터 유형의 AWS::Events::Rule Target 속성과 유사합니다. DLQ를 자동으로 AWS SAM 생성하고자 하는 경우에 대비하여 이 속성의 AWS SAM 버전에는 추가 하위 속성이 포함되어 있습니다.

`Description`

규칙에 대한 설명.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 Description 리소스의 AWS::Events::Rule 속성으로 직접 전달됩니다.

`Enabled`

규칙을 활성화할지를 나타냅니다.

규칙을 비활성화하려면 이 속성을 false로 설정합니다.

> **Note**
> Enabled 또는 State 속성을 지정할 수 있지만, 두 속성을 함께 지정할 수는 없습니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 State 리소스의 AWS::Events::Rule 속성과 유사합니다. 이 속성이 true로 설정되면 AWS SAM가 ENABLED로 전달되고, 그렇지 않으면 DISABLED로 전달됩니다.

`Input`

대상으로 전달되는 유효한 JSON 텍스트입니다. 이 속성을 사용하면 이벤트 텍스트 자체의 어떤 것도 대상으로 전달되지 않습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [Input](#) 리소스의 `AWS::Events::Rule Target` 속성으로 직접 전달됩니다.

`Name`

규칙의 이름입니다. 이름을 지정하지 않은 경우 CloudFormation은 고유한 물리적 ID를 생성한 후 규칙 이름에 해당 ID를 사용합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `Name` 리소스의 `AWS::Events::Rule` 속성으로 직접 전달됩니다.

`RetryPolicy`

재시도 정책 설정에 대한 정보가 포함된 `RetryPolicy` 객체입니다. 자세한 내용은 Amazon EventBridge 사용자 가이드의 이벤트 재시도 정책 및 DLQ(Dead Letter Queue) 사용을 참조하세요.

유형: RetryPolicy

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `RetryPolicy` `AWS::Events::Rule` 데이터 유형의 `Target` 속성에 직접 전달됩니다.

`Schedule`

규칙 실행 시기 및 방법을 결정하는 스케줄링 표현식입니다. 자세한 내용은 규칙에 대한 예약 표현식을 참조하세요.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `ScheduleExpression` 리소스의 `AWS::Events::Rule` 속성으로 직접 전달됩니다.

`State`

규칙의 상태입니다.

허용되는 값: `DISABLED | ENABLED`

> **Note**
>
> Enabled 또는 State 속성을 지정할 수 있지만, 두 속성을 함께 지정할 수는 없습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `State` 리소스의 `AWS::Events::Rule` 속성으로 직접 전달됩니다.

예시

CloudWatch Schedule Event

CloudWatch Schedule Events

YAML

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

DeadLetterConfig

대상 호출 실패 후 EventBridge가 이벤트를 전송하는 Amazon Simple Queue Service(Amazon SQS) 대기열을 지정하는 데 사용되는 객체입니다. 예를 들어 존재하지 않는 Lambda 함수로 이벤트를 전송하거나 Lambda 함수를 호출할 권한이 충분하지 않은 경우 간접 호출이 실패할 수 있습니다. 자세한 내용은 Amazon EventBridge 사용자 가이드의 이벤트 재시도 정책 및 DLQ(Dead Letter Queue) 사용을 참조하세요.

> **Note**
>
> AWS::Serverless::Function 리소스 유형에는 유사한 데이터 유형인`DeadLetterQueue`가 있으며, 이는 대상 Lambda 함수를 성공적으로 호출한 후 발생하는 장애를 처리합니다. 이러한 유형의 실패의 예로는 Lambda 제한 또는 Lambda 대상 함수에서 반환되는 오류가 있습니다. 함

수 DeadLetterQueue 속성에 대한 자세한 내용은 AWS Lambda 개발자 가이드의 DLQ(Dead Letter Queue)를 참조하세요.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
Arn: String
QueueLogicalId: String
Type: String
```

속성

`Arn`

DLQ(Dead Letter Queue)의 대상으로 지정된 Amazon SQS 대기열의 Amazon 리소스 이름(ARN)입니다.

> **Note**
>
> Type 속성 또는 Arn 속성 중 하나만 지정해야 하며, 둘 다 지정할 수는 없습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 Arn AWS::Events::Rule 데이터 유형의 DeadLetterConfig 속성에 직접 전달됩니다.

`QueueLogicalId`

Type가 지정된 경우 AWS SAM이 생성하는 DLQ의 사용자 지정 이름입니다.

> **Note**
>
> Type 속성이 설정되지 않은 경우 이 속성은 무시됩니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

Type

대기열 유형. 이 속성이 설정되면 AWS SAM은 DLQ를 자동으로 만들고 허가를 부여하기 위해 필요한 리소스 기반 정책을 규칙 리소스에 첨부하여 해당 큐에 이벤트를 전송합니다.

> **Note**
>
> Type 속성 또는 Arn 속성 중 하나만 지정해야 하며, 둘 다 지정할 수는 없습니다.

유효한 값: SQS

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예제

DeadLetterConfig

DeadLetterConfig

YAML

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

ScheduleV2

ScheduleV2 이벤트 소스 유형을 설명하는 객체로, 이는 귀하의 서버리스 함수를 일정에 따라 트리거되는 Amazon EventBridge Scheduler 이벤트의 대상으로 설정합니다. 자세한 내용은 EventBridge Scheduler 사용 설명서의 Amazon EventBridge Scheduler란 무엇인가요?를 참조하세요.

AWS Serverless Application Model(AWS SAM)은 이 이벤트 유형이 설정되면 AWS::Scheduler::Schedule리소스를 생성합니다.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

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
```

속성

`DeadLetterConfig`

대상 호출이 실패한 경우 EventBridge가 이벤트를 전송하는 Amazon Simple Queue Service(Amazon SQS) 대기열을 구성합니다. 예를 들어 존재하지 않는 Lambda 함수로 이벤트를 전송하거나 EventBridge에 Lambda 함수를 간접 호출할 권한이 충분하지 않은 경우 간접 호출이 실패할 수 있습니다. 자세한 내용은 EventBridge 스케줄러 사용자 가이드의 EventBridge 스케줄러용 DLQ 대기열 구성을 참조하세요.

> **Note**
>
> AWS::Serverless::Function 리소스 유형에는 유사한 데이터 유형인`DeadLetterQueue`가 있으며, 이는 대상 Lambda 함수를 성공적으로 호출한 후 발생하는 장애를 처리합니다. 이

러한 유형의 실패의 예로는 Lambda 제한 또는 Lambda 대상 함수에서 표시되는 오류가 있습니다. 함수 `DeadLetterQueue` 속성에 대한 자세한 내용은 AWS Lambda 개발자 가이드의 DLQ(Dead Letter Queue)를 참조하세요.

유형: DeadLetterConfig

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `DeadLetterConfig` 데이터 유형의 `AWS::Scheduler::Schedule` `Target` 속성과 유사합니다. DLQ를 자동으로 AWS SAM 생성하고자 하는 경우에 대비하여 이 속성의 AWS SAM 버전에는 추가 하위 속성이 포함되어 있습니다.

`Description`

일정에 대한 설명입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `Description` 리소스의 `AWS::Scheduler::Schedule` 속성으로 직접 전달됩니다.

`EndDate`

일정이 대상을 간접 호출할 수 있는 날짜(UTC 기준)입니다. 일정의 반복 식에 따라 사용자가 지정하는 EndDate 또는 그 이전에 호출이 중지될 수 있습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Scheduler::Schedule` 리소스의 `EndDate` 속성으로 직접 전달됩니다.

`FlexibleTimeWindow`

일정을 간접 호출할 수 있는 창을 구성할 수 있습니다.

유형: FlexibleTimeWindow

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS::Scheduler::Schedule 리소스의 [FlexibleTimeWindow](#) 속성으로 직접 전달됩니다.

`GroupName`

이 일정과 연계하기 위한 일정 그룹의 이름입니다. 정의되지 않은 경우 기본 그룹이 사용됩니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS::Scheduler::Schedule 리소스의 [GroupName](#) 속성으로 직접 전달됩니다.

`Input`

대상으로 전달되는 유효한 JSON 텍스트입니다. 이 속성을 사용하면 이벤트 텍스트 자체의 어떤 것도 대상으로 전달되지 않습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS::Scheduler::Schedule Target 리소스의 [Input](#) 속성으로 직접 전달됩니다.

`KmsKeyArn`

고객 데이터를 암호화하는 데 사용되는 KMS 키의 ARN.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS::Scheduler::Schedule 리소스의 [KmsKeyArn](#) 속성으로 직접 전달됩니다.

`Name`

일정의 이름입니다. 이름을 지정하지 않으면 AWS SAM이 *Function-Logical-IDEvent-Source-Name*의 형식으로 이름을 생성하고 일정 이름에 해당 ID를 사용합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Scheduler::Schedule` 리소스의 [Name](#) 속성으로 직접 전달됩니다.

`OmitName`

기본적으로는 AWS SAM이 *<Function-logical-ID><event-source-name>*의 형식으로 일정의 이름을 생성 및 사용합니다. `true`가 고유한 물리적 ID를 생성하도록 이 속성을 CloudFormation에 설정하고 이 ID를 스케줄 이름으로 대신 사용하십시오.

유형: 부울

필수 항목 여부: 아니요

기본값: `false`

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`PermissionsBoundary`

역할에 대한 권한 경계 설정에 사용되는 정책의 ARN입니다.

> **Note**
>
> `PermissionsBoundary`가 정의된 경우 AWS SAM는 스케줄러 일정의 대상 IAM 역할에 동일한 경계를 적용합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [PermissionsBoundary](#) 리소스의 `AWS::IAM::Role` 속성으로 직접 전달됩니다.

`RetryPolicy`

재시도 정책 설정에 대한 정보가 포함된 RetryPolicy 객체입니다.

유형: [RetryPolicy](#)

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `RetryPolicy` `AWS::Scheduler::Schedule` 데이터 유형의 `Target` 속성에 직접 전달됩니다.

`RoleArn`

스케줄이 호출될 때 EventBridge 스케줄러가 대상에 사용할 IAM 역할의 ARN입니다.

유형: RoleArn

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `RoleArn` `AWS::Scheduler::Schedule` 데이터 유형의 `Target` 속성에 직접 전달됩니다.

`ScheduleExpression`

스케줄러 일정 이벤트가 실행되는 시기와 빈도를 결정하는 스케줄링 표현식입니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `ScheduleExpression` 리소스의 `AWS::Scheduler::Schedule` 속성으로 직접 전달됩니다.

`ScheduleExpressionTimezone`

스케줄링 표현식이 평가되는 시간대입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Scheduler::Schedule` 리소스의 `ScheduleExpressionTimezone` 속성으로 직접 전달됩니다.

`StartDate`

일정이 대상의 간접 호출을 시작할 수 있는 날짜(UTC 기준)입니다. 일정의 반복 식에 따라 사용자가 지정하는 StartDate 또는 그 이후에 호출이 발생할 수 있습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Scheduler::Schedule` 리소스의 [`StartDate`] 속성으로 직접 전달됩니다.

`State`

스케줄러 일정의 상태.

허용되는 값: `DISABLED | ENABLED`

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [`State`] 리소스의 `AWS::Scheduler::Schedule` 속성으로 직접 전달됩니다.

예시

ScheduleV2 리소스를 정의하는 기본 예제

```
Resources:
  Function:
    Properties:
      ...
      Events:
        ScheduleEvent:
          Type: ScheduleV2
          Properties:
            ScheduleExpression: "rate(1 minute)"
        ComplexScheduleEvent:
          Type: ScheduleV2
          Properties:
            ScheduleExpression: rate(1 minute)
            FlexibleTimeWindow:
              Mode: FLEXIBLE
              MaximumWindowInMinutes: 5
            StartDate: '2022-12-28T12:00:00.000Z'
            EndDate: '2023-01-28T12:00:00.000Z'
            ScheduleExpressionTimezone: UTC
            RetryPolicy:
              MaximumRetryAttempts: 5
```

```
        MaximumEventAgeInSeconds: 300
      DeadLetterConfig:
        Type: SQS
```

> **Note**
>
> ScheduleV2의 생성된 물리적 ID에는 스택 이름이 포함되지 않습니다.

SelfManagedKafka

`SelfManagedKafka` 이벤트 소스 유형을 설명하는 객체. 자세한 내용은 AWS Lambda 개발자 안내서의 [자체 관리형 Apache Kafka AWS Lambda 에서 사용을](#) 참조하세요.

AWS Serverless Application Model (AWS SAM)는이 이벤트 유형이 설정되면 [AWS::Lambda::EventSourceMapping](#) 리소스를 생성합니다.

Schema Registry를 사용하려면 함수에 대한 특정 IAM 역할 권한을 정의해야 합니다. 필수 구성의 예시는 [IAM 역할로 설정 완료하기](#)를 참조하세요.

구문

AWS SAM 템플릿에서이 개체를 선언하려면 다음 구문을 사용합니다.

YAML

```
  BatchSize: Integer
  BisectBatchOnFunctionError: Boolean
  ConsumerGroupId: String
  DestinationConfig: DestinationConfig
  Enabled: Boolean
  FilterCriteria: FilterCriteria
  KafkaBootstrapServers: List
  FunctionResponseTypes: List
  KmsKeyArn: String
  MaximumRecordAgeInSeconds: Integer
  MaximumRetryAttempts: Integer
  ProvisionedPollerConfig: ProvisionedPollerConfig
  SchemaRegistryConfig: SchemaRegistryConfig
  SourceAccessConfigurations: SourceAccessConfigurations
  StartingPosition: String
```

```
  StartingPositionTimestamp: Double
  Topics: List
```

속성

BatchSize

Lambda가 귀하의 스트림으로부터 풀링하여 귀하의 함수로 보내는 각 배치의 최대 기록 수.

유형: 정수

필수 항목 여부: 아니요

기본값: 100

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 [`BatchSize`] 속성으로 직접 전달됩니다.

최소: 1

최대: `10000`

BisectBatchOnFunctionError

함수가 오류를 제시하면 비치를 2개로 분할해서 다시 시도합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 [`BisectBatchOnFunctionError`] 속성으로 직접 전달됩니다.

ConsumerGroupId

Kafka 주제에서 이벤트를 읽는 방법을 구성하는 문자열입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 [`SelfManagedKafkaConfiguration`] 속성으로 직접 전달됩니다.

`DestinationConfig`

Lambda가 이벤트를 처리한 후 이벤트의 대상을 지정하는 구성 객체입니다.

이 속성을 사용하여 자체 관리형 Kafka 이벤트 소스에서 실패한 간접 호출의 대상을 지정합니다.

유형: DestinationConfig

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `DestinationConfig` 속성으로 직접 전달됩니다.

`Enabled`

이벤트 소스 매핑을 비활성화하여 폴링 및 간접 호출을 일시 중지합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `Enabled` 속성으로 직접 전달됩니다.

`FilterCriteria`

Lambda가 이벤트를 처리해야 하는지 결정하는 기준을 정의하는 객체입니다. 자세한 내용은 AWS Lambda 개발자 가이드의 AWS Lambda 이벤트 필터링을 참조하세요.

유형: FilterCriteria

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `FilterCriteria` 속성으로 직접 전달됩니다.

`KafkaBootstrapServers`

귀하의 Kafka 브로커의 부트스트랩 서버 목록. 포트를 포함시킵니다(예: `broker.example.com:xxxx`)

유형: 목록

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은에 고유 AWS SAM 하며 CloudFormation 동등한 속성이 없습니다.

`FunctionResponseTypes`

이벤트 소스 매핑에 현재 적용된 응답 유형의 목록입니다. 자세한 내용은 AWS Lambda Developer Guide의 [Reporting batch item failures](https://docs.aws.amazon.com/lambda/latest/dg/with-kinesis.html#services-kinesis-batchfailurereporting)를 참조하세요.

유효한 값: `ReportBatchItemFailures`

유형: 목록

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 [`FunctionResponseTypes`](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-lambda-eventsourcemapping.html#cfn-lambda-eventsourcemapping-functionresponsetypes) 속성으로 직접 전달됩니다.

`KmsKeyArn`

이 이벤트와 관련된 정보를 암호화하는 키의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 [`KmsKeyArn`](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-lambda-eventsourcemapping.html#cfn-lambda-eventsourcemapping-kmskeyarn) 속성으로 직접 전달됩니다.

`MaximumRecordAgeInSeconds`

Lambda가 처리를 위해 함수에 보내는 기록의 최대 사용 기간입니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 [`MaximumRecordAgeInSeconds`](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-lambda-eventsourcemapping.html#cfn-lambda-eventsourcemapping-maximumrecordageinseconds) 속성으로 직접 전달됩니다.

`MaximumRetryAttempts`

함수가 오류를 반환할 때 재시도하는 최대 횟수입니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 [`MaximumRetryAttempts`] 속성으로 직접 전달됩니다.

`ProvisionedPollerConfig`

이벤트 소스 매핑을 컴퓨팅하는 데 사용되는 폴러의 양을 늘리기 위한 구성입니다. 이 구성은 최소 1개의 폴러와 최대 2,000개의 폴러를 허용합니다. 예시는 ProvisionedPollerConfig 예시에서 확인하세요

유형: ProvisionedPollerConfig

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `ProvisionedPollerConfig` 속성으로 직접 전달됩니다.

`SchemaRegistryConfig`

자체 관리형 Kafka 이벤트 소스와 스키마 레지스트리를 사용하기 위한 구성입니다.

> **Note**
>
> 이 기능에는 `ProvisionedPollerConfig` 구성이 필요합니다.

유형: SchemaRegistryConfig

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `SelfManagedKafkaEventSourceConfig` 속성으로 직접 전달됩니다.

`SourceAccessConfigurations`

이벤트 소스를 보호하고 정의하는 데 사용되는 인증 프로토콜, VPC 구성 요소 또는 가상 호스트의 배열입니다.

유효한 값: `BASIC_AUTH | CLIENT_CERTIFICATE_TLS_AUTH | SASL_SCRAM_256_AUTH | SASL_SCRAM_512_AUTH | SERVER_ROOT_CA_CERTIFICATE`

유형: SourceAccessConfiguration 목록

필수 항목 여부: 예

CloudFormation 호환성:이 속성은 AWS::Lambda::EventSourceMapping 리소스의 SelfManagedKafkaEventSourceConfig 속성의 일부입니다.

StartingPosition

읽기를 시작하는 스트림 내의 위치입니다.

- AT_TIMESTAMP - 기록 읽기를 시작할 시간을 지정합니다.
- LATEST – 새 기록만 읽습니다.
- TRIM_HORIZON – 사용 가능한 모든 기록을 처리합니다.

유효한 값: AT_TIMESTAMP | LATEST | TRIM_HORIZON

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 AWS::Lambda::EventSourceMapping 리소스의 StartingPosition 속성으로 직접 전달됩니다.

StartingPositionTimestamp

읽기를 시작하는 시간(유닉스 시간 초 단위)입니다. StartingPositionTimestamp 언제 StartingPosition를 AT_TIMESTAMP으로 지정할지 정의합니다.

유형: Double

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 AWS::Lambda::EventSourceMapping 리소스의 StartingPositionTimestamp 속성으로 직접 전달됩니다.

Topics

Kafka 주제의 이름입니다.

유형: 목록

필수 항목 여부: 예

CloudFormation 호환성:이 속성은 AWS::Lambda::EventSourceMapping 리소스의 Topics 속성으로 직접 전달됩니다.

예제

IAM 역할로 설정 완료

다음 예시는 Schema Registry 사용을 위해 필요한 IAM 역할 구성을 포함한 완료된 설정을 보여줍니다.

```
Parameters:
  PreCreatedSubnetOne:
    Type: String
  PreCreatedSubnetTwo:
    Type: String

Resources:
  MyLambdaExecutionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
        - Action: [sts:AssumeRole]
          Effect: Allow
          Principal:
            Service: [lambda.amazonaws.com]
      Policies:
      - PolicyName: KafkaAuthPolicy
        PolicyDocument:
          Statement:
          - Action: [secretsmanager:GetSecretValue, kms:Decrypt]
            Effect: "Allow"
            Resource: ['arn:aws:secretsmanager:us-
west-2:123456789012:secret:kafkaSecret-******',
                        'arn:aws:kms:us-west-2:123456789012:key/keyId']
      - PolicyName: ENIPolicy
        PolicyDocument:
          Statement:
          - Action: [ec2:CreateNetworkInterface,
              ec2:DescribeNetworkInterfaces, ec2:DescribeVpcs,
 ec2:DeleteNetworkInterface,
              ec2:DescribeSubnets, ec2:DescribeSecurityGroups]
            Effect: Allow
            Resource: '*'
      - PolicyName: SchemaRegistryPolicy
        PolicyDocument:
          Statement:
```

```
          - Action: [glue:GetRegistry]
            Effect: Allow
            Resource: 'arn:aws:glue:{region}:{account-id}:registry/{registry-name}'
      - PolicyName: SchemaVersionsPolicy
        PolicyDocument:
          Statement:
          - Action: [glue:GetSchemaVersions]
            Effect: Allow
            Resource: '*'
      ManagedPolicyArns:
      - !Sub arn:${AWS::Partition}:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole
      Tags:
      - {Value: SAM, Key: lambda:createdBy}

  MyKafkaProcessor:
    Type: AWS::Serverless::Function
    Properties:
      Runtime: nodejs18.x
      Handler: index.handler
      CodeUri: ${codeuri}
      Role:
        Fn::GetAtt: [MyLambdaExecutionRole, Arn]
      Events:
        SelfManagedKafkaEvent:
          Type: SelfManagedKafka
          Properties:
            KafkaBootstrapServers:
              - my-kafka-broker-1:9092
              - my-kafka-broker-2:9092
            Topics:
              - SchemaRegistryTestTopic
            StartingPosition: LATEST
            SourceAccessConfigurations:
              - Type: VPC_SUBNET
                URI: subnet:subnet-12345678
              - Type: VPC_SECURITY_GROUP
                URI: security_group:sg-12345678
              - Type: BASIC_AUTH
                URI: !Sub arn:${AWS::Partition}:secretsmanager:us-
west-2:123456789012:secret:my-path/my-secret-name-1a2b3c
            ProvisionedPollerConfig:
              MinimumPollers: 1
            SchemaRegistryConfig:
```

```
              AccessConfigs:
              - Type: BASIC_AUTH
                URI: !Sub arn:${AWS::Partition}:secretsmanager:us-
west-2:123456789012:secret:my-path/my-secret-name-1a2b3c
              SchemaValidationConfigs:
              - Attribute: KEY
              EventRecordFormat: JSON
              SchemaRegistryURI: !Sub arn:${AWS::Partition}:glue:us-
west-2:123456789012:registry/myregistry
```

ProvisionedPollerConfig 예시

```
ProvisionedPollerConfig:
  MinimumPollers: 1
  MaximumPollers: 200
```

자체 관리형 Kafka 이벤트 소스

다음은 `SelfManagedKafka` 이벤트 소스 유형의 한 예제입니다.

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
```

AWS Glue 스키마 레지스트리를 사용하는 자체 관리형 Kafka 이벤트 소스

다음은 AWS Glue Schema Registry로 구성된 `SelfManagedKafka` 이벤트 소스 유형의 예입니다.

```
Events:
```

```
  SelfManagedKafkaEvent:
    Type: SelfManagedKafka
    Properties:
      KafkaBootstrapServers:
        - abc.xyz.com:9092
      Topics:
        - SchemaRegistryTestTopic
      StartingPosition: LATEST
      ProvisionedPollerConfig:
        MinimumPollers: 1
      SchemaRegistryConfig:
        SchemaRegistryURI: !Sub arn:${AWS::Partition}:glue:us-
west-2:123456789012:registry/myregistry
        EventRecordFormat: JSON
        SchemaValidationConfigs:
          - Attribute: KEY
          - Attribute: VALUE
      SourceAccessConfigurations:
        - Type: VPC_SUBNET
          URI: subnet:subnet-12345678
        - Type: VPC_SECURITY_GROUP
          URI: security_group:sg-12345678
```

Confluent Schema Registry를 사용한 자체 관리형 Kafka 이벤트 소스

다음은 Confluent Schema Registry로 구성된 `SelfManagedKafka` 이벤트 소스 유형의 예시입니다.

```
Events:
  SelfManagedKafkaEvent:
    Type: SelfManagedKafka
    Properties:
      KafkaBootstrapServers:
        - abc.xyz.com:9092
      Topics:
        - SchemaRegistryTestTopic
      StartingPosition: LATEST
      ProvisionedPollerConfig:
        MinimumPollers: 1
      SchemaRegistryConfig:
        SchemaRegistryURI: https://my-schema-registry.confluent.cloud
        AccessConfigs:
          - Type: BASIC_AUTH
            URI: !Sub arn:${AWS::Partition}:secretsmanager:us-
west-2:123456789012:secret:my-secret
```

```
        EventRecordFormat: JSON
        SchemaValidationConfigs:
          - Attribute: KEY
          - Attribute: VALUE
      SourceAccessConfigurations:
        - Type: VPC_SUBNET
          URI: subnet:subnet-12345678
        - Type: VPC_SECURITY_GROUP
          URI: security_group:sg-12345678
        - Type: BASIC_AUTH
          URI: !Sub arn:${AWS::Partition}:secretsmanager:us-
west-2:123456789012:secret:kafka-secret
```

SNS

SNS 이벤트 소스 유형을 설명하는 객체.

SAM은 이 이벤트 유형이 설정되면 AWS::SNS::Subscription 리소스를 생성합니다.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  FilterPolicy: SnsFilterPolicy
  FilterPolicyScope: String
  RedrivePolicy: Json
  Region: String
  SqsSubscription: Boolean | SqsSubscriptionObject
  Topic: String
```

속성

`FilterPolicy`

구독에 할당된 필터 정책 JSON입니다. 자세한 내용은 Amazon Simple Notification Service API 참조에서 GetSubscriptionAttributes을 참조하세요.

유형: SnsFilterPolicy

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [FilterPolicy]() 리소스의 AWS::SNS::Subscription 속성으로 직접 전달됩니다.

`FilterPolicyScope`

이 속성을 사용하면 다음 문자열 값 유형 중 하나를 사용하여 필터링 범위를 선택할 수 있습니다.

- `MessageAttributes` - 필터가 메시지 속성에 적용됩니다.
- `MessageBody` - 필터가 메시지 본문에 적용됩니다.

유형: 문자열

필수 항목 여부: 아니요

기본값: `MessageAttributes`

CloudFormation호환성: 이 속성은 [FilterPolicyScope]() 리소스의 `AWS::SNS::Subscription` 속성으로 직접 전달됩니다.

`RedrivePolicy`

지정되면, 전송할 수 없는 메시지를 지정된 Amazon SQS DLQ(Dead Letter Queue)로 보냅니다. 클라이언트 오류(예: 구독 엔드포인트에 연결할 수 없는 경우) 또는 서버 오류(예: 구독 엔드포인트에 전원을 공급하는 서비스를 사용할 수 없게 된 경우)로 인해 전송할 수 없는 메시지는 추가 분석 또는 재처리를 위해 DLQ(Dead Letter Queue)에 보관됩니다.

리드라이브 정책 및 DLQ(Dead Letter Queue)에 대한 자세한 내용은 [Amazon Simple Queue Service 개발자 가이드]()의 Amazon SQS DLQ를 참조하세요.

유형: Json

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [RedrivePolicy]() 리소스의 `AWS::SNS::Subscription` 속성으로 직접 전달됩니다.

`Region`

교차 리전 구독의 경우 주제가 상주하는 리전입니다.

리전을 지정하지 않을 경우 CloudFormation이 기본적으로 호출자의 리전을 사용합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [`Region`] 리소스의 `AWS::SNS::Subscription` 속성으로 직접 전달됩니다.

`SqsSubscription`

이 속성을 true로 설정하거나 `SqsSubscriptionObject`을 지정하여 SQS 대기열에서 SNS 토픽 알림 일괄 처리를 사용하도록 설정합니다. 이 속성을 `true`로 설정하면 새 SQS 대기열이 만들어지고, `SqsSubscriptionObject`를 지정하면 기존 SQS 대기열이 사용됩니다.

유형: Boolean | [SqsSubscriptionObject]

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Topic`

구독할 주제의 ARN입니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 [`TopicArn`] 리소스의 `AWS::SNS::Subscription` 속성으로 직접 전달됩니다.

예시

SNS 이벤트 소스 예제

SNS 이벤트 소스 예제

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

기존 SQS 대기열 옵션을 SNS 이벤트에 지정

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  BatchSize: String
  Enabled: Boolean
  QueueArn: String
  QueuePolicyLogicalId: String
  QueueUrl: String
```

속성

`BatchSize`

SQS 대기열을 위한 한 번의 배치에서 검색하는 최대 항목 수.

유형: 문자열

필수 항목 여부: 아니요

기본값: 10

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Enabled`

이벤트 소스 매핑을 비활성화하여 폴링 및 호출을 일시 중지합니다.

유형: 부울

필수 항목 여부: 아니요

기본값: True

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`QueueArn`

기존 SQS 대기열 arn을 지정합니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`QueuePolicyLogicalId`

[AWS::SQS::QueuePolicy](#) 리소스에 대한 사용자 지정 logicalId 이름을 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`QueueUrl`

`QueueArn` 속성과 관련된 대기열 URL을 지정합니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

SNS 이벤트용 기존 SQS

SNS 주제 구독을 위한 기존 SQS 대기열을 추가하는 예.

YAML

```
QueuePolicyLogicalId: CustomQueuePolicyLogicalId
```

```
QueueArn:
  Fn::GetAtt: MyCustomQueue.Arn
QueueUrl:
  Ref: MyCustomQueue
BatchSize: 5
```

SQS

SQS 이벤트 소스 유형을 설명하는 객체. 자세한 내용은 AWS Lambda 개발자 안내서의 [Amazon SQS AWS Lambda 에서 사용을] 참조하세요.

SAM은 이 이벤트 유형이 설정되면 [AWS::Lambda::EventSourceMapping] 리소스를 생성합니다.

구문

AWS Serverless Application Model (AWS SAM) 템플릿에서이 개체를 선언하려면 다음 구문을 사용합니다.

YAML

```
  BatchSize: Integer
  Enabled: Boolean
  FilterCriteria: FilterCriteria
  FunctionResponseTypes: List
  KmsKeyArn: String
  MaximumBatchingWindowInSeconds: Integer
  MetricsConfig: MetricsConfig
  ProvisionedPollerConfig: ProvisionedPollerConfig
  Queue: String
  ScalingConfig: ScalingConfig
```

속성

`BatchSize`

한 번의 배치에서 검색하는 최대 항목 수입니다.

유형: 정수

필수 항목 여부: 아니요

기본값: 10

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `BatchSize` 속성으로 직접 전달됩니다.

최소: 1

최대: `10000`

`Enabled`

이벤트 소스 매핑을 비활성화하여 폴링 및 간접 호출을 일시 중지합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `Enabled` 속성으로 직접 전달됩니다.

`FilterCriteria`

Lambda가 이벤트를 처리해야 하는지 결정하는 기준을 정의하는 객체입니다. 자세한 내용은 AWS Lambda 개발자 가이드의 AWS Lambda 이벤트 필터링을 참조하세요.

유형: FilterCriteria

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `FilterCriteria` 속성으로 직접 전달됩니다.

`FunctionResponseTypes`

이벤트 소스 매핑에 현재 적용된 응답 유형의 목록입니다. 자세한 내용은 AWS Lambda Developer Guide의 Reporting batch item failures를 참조하세요.

유효한 값: `ReportBatchItemFailures`

유형: 목록

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `FunctionResponseTypes` 속성으로 직접 전달됩니다.

`KmsKeyArn`

이 이벤트와 관련된 정보를 암호화하는 키의 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `KmsKeyArn` 속성으로 직접 전달됩니다.

`MaximumBatchingWindowInSeconds`

함수를 간접 호출하기 전에 레코드를 수집할 최대 시간(단위: 초)입니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `MaximumBatchingWindowInSeconds` 속성으로 직접 전달됩니다.

`MetricsConfig`

처리 단계별로 캡처하는 이벤트 소스 매핑에 대한 향상된 지표를 얻기 위한 옵트인 구성입니다. 예제는 MetricsConfig 이벤트 섹션을 참조하세요.

유형: MetricsConfig

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `MetricsConfig` 속성으로 직접 전달됩니다.

`ProvisionedPollerConfig`

이벤트 소스 매핑을 컴퓨팅하는 데 사용되는 폴러의 양을 늘리기 위한 구성입니다. 이 구성은 최소 2개의 폴러와 최대 2,000개의 폴러를 허용합니다. 예시는 ProvisionedPollerConfig 예시에서 확인하세요.

유형: ProvisionedPollerConfig

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::EventSourceMapping` 리소스의 `ProvisionedPollerConfig` 속성으로 직접 전달됩니다.

`Queue`

대기열의 ARN입니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation 호환성:이 속성은 AWS::Lambda::EventSourceMapping 리소스의 EventSourceArn 속성으로 직접 전달됩니다.

ScalingConfig

SQS 폴러의 구성 규모를 조정하여 간접 호출 속도를 제어하고 최대 동시 간접 호출값을 설정합니다.

유형: ScalingConfig

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 AWS::Lambda::EventSourceMapping 리소스의 ScalingConfig 속성으로 직접 전달됩니다.

예제

MetricsConfig 이벤트

다음은 이벤트 소스 매핑에 대한 처리 단계별 캡처를 위해 MetricsConfig 속성을 사용하는 리소스의 예시입니다.

```
Resources:
  FilteredEventsFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://sam-demo-bucket/metricsConfig.zip
      Handler: index.handler
      Runtime: nodejs16.x
      Events:
        KinesisStream:
          Type: Kinesis
          Properties:
            Stream: !GetAtt KinesisStream.Arn
            StartingPosition: LATEST
            MetricsConfig:
              Metrics:
              - EventCount
```

기본 SQS 이벤트

```
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"]}'
```

SQS 대기열에 대한 부분별 배치 리포트 구성

```
Events:
  SQSEvent:
    Type: SQS
    Properties:
      Enabled: true
      FunctionResponseTypes:
        - ReportBatchItemFailures
      Queue: !GetAtt MySqsQueue.Arn
      BatchSize: 10
```

규모 조정이 구성화된 SQS 이벤트가 포함된 Lambda 함수

```
MyFunction:
  Type: AWS::Serverless::Function
  Properties:
    ...
    Events:
      MySQSEvent:
        Type: SQS
        Properties:
          ...
          ScalingConfig:
            MaximumConcurrency: 10
```

ProvisionedPollerConfig 예시

```
MyFunction:
```

```
  Type: AWS::Serverless::Function
  Properties:
    Handler: index.handler
    Runtime: nodejs18.x
    Timeout: 30
    Events:
      SQSEvent:
        Type: SQS
        Properties:
          Queue: !GetAtt MyQueue.Arn
          BatchSize: 10
          Enabled: True
          ProvisionedPollerConfig:
            MaximumPollers: 300
            MinimumPollers: 10
```

FunctionCode

Lambda 함수의 배포 패키지입니다.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

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

속성

`Bucket`

함수와 동일한 AWS 리전에 있는 Amazon S3 버킷입니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `S3Bucket` `AWS::Lambda::Function` 데이터 유형의 `Code` 속성에 직접 전달됩니다.

`Key`

배포 패키지의 Amazon S3 키입니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 [S3Key](#) `AWS::Lambda::Function` 데이터 유형의 `Code` 속성에 직접 전달됩니다.

`Version`

버전이 지정된 객체의 경우 사용할 배포 패키지 객체의 버전입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [S3ObjectVersion](#) `AWS::Lambda::Function` 데이터 유형의 `Code` 속성에 직접 전달됩니다.

예제

FunctionCode

`CodeUri`: 함수 코드 예제

YAML

```
CodeUri:
  Bucket: sam-s3-demo-bucket-name
  Key: mykey-name
  Version: 121212
```

FunctionUrlConfig

지정된 구성 매개변수를 사용하여 AWS Lambda 함수 URL을 생성합니다. Lambda 함수 URL은 함수를 간접 호출하는 데 사용할 수 있는 HTTPS 엔드포인트입니다.

기본적으로 귀하가 생성한 함수 URL은 Lambda 함수 `$LATEST` 버전을 사용합니다. Lambda 함수에 대해 `AutoPublishAlias`를 지정하는 경우 엔드포인트는 지정된 함수 별칭에 연결됩니다.

자세한 내용은 개발자 가이드 내 AWS LambdaLambda 함수 URL를 참조하세요.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
AuthType: String
Cors: Cors
InvokeMode: String
```

속성

`AuthType`

함수 URL에서 사용하는 인증 유형입니다. AWS Identity and Access Management(IAM)을 사용하여 요청을 승인하려면 `AWS_IAM`로 설정합니다. 오픈 액세스의 경우 `NONE`으로 설정합니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `AWS::Lambda::Url` 리소스의 `AuthType` 속성으로 직접 전달됩니다.

`Cors`

함수 URL에 대한 교차 오리진 리소스 공유(CORS) 설정입니다.

유형: Cors

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Lambda::Url` 리소스의 `Cors` 속성으로 직접 전달됩니다.

`InvokeMode`

함수 URL이 간접 호출되는 모드입니다. 호출 완료 후 함수가 응답을 반환하도록 하려면 `BUFFERED`로 설정합니다. 함수가 응답을 스트리밍하도록 하려면 `RESPONSE_STREAM`로 설정합니다. 기본값은 `BUFFERED`입니다.

유효한 값: `BUFFERED` 또는 `RESPONSE_STREAM`

유형: 문자열

필수 항목 여부: 아니요

AWS CloudFormation호환성: 이 속성은 `InvokeMode` 리소스의 `AWS::Lambda::Url` 속성으로 직접 전달됩니다.

예제

함수 URL

다음 예제에서는 함수 URL이 있는 Lambda 함수를 생성합니다. 함수 URL은 IAM 인증을 사용합니다.

YAML

```
HelloWorldFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: hello_world/
    Handler: index.handler
    Runtime: nodejs20.x
    FunctionUrlConfig:
      AuthType: AWS_IAM
      InvokeMode: RESPONSE_STREAM

Outputs:
  MyFunctionUrlEndpoint:
      Description: "My Lambda Function URL Endpoint"
      Value:
        Fn::GetAtt: HelloWorldFunctionUrl.FunctionUrl
```

CapacityProviderConfig

함수의 게시된 버전을 연결할 용량 공급자를 구성합니다. 이렇게 하면 Lambda에서 관리하는 고객 소유 EC2 인스턴스에서 함수를 실행할 수 있습니다.

> **Note**
> 이 구성은 함수의 컴퓨팅 유형을 결정하며 첫 번째 함수 배포 중에 지정해야 합니다. 함수 리소스가 생성된 후에는 추가하거나 제거할 수 없습니다.

구문

AWS Serverless Application Model (AWS SAM) 템플릿에서이 개체를 선언하려면 다음 구문을 사용합니다.

YAML

```
Arn: String
ExecutionEnvironmentMemoryGiBPerVCpu: Float
PerExecutionEnvironmentMaxConcurrency: Integer
```

속성

`Arn`

이 함수에 사용할 용량 공급자의 ARN입니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation 호환성:이 속성은 SAM에 고유합니다.

`ExecutionEnvironmentMemoryGiBPerVCpu`

각 실행 환경에 대한 vCPU 대비 메모리(GiB)의 비율입니다.

> **Note**
>
> CPU당 메모리 비율은 함수의 총 메모리인 2048MB를 초과할 수 없습니다. 지원되는 memory-to-CPU 비율은 CPU당 2GB, 4GB 또는 8GB입니다.

유형: 부동 소수점

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 `ExecutionEnvironmentMemoryGiBPerVCpu` 속성으로 직접 전달됩니다.

`PerExecutionEnvironmentMaxConcurrency`

실행 환경(샌드박스)당 최대 동시 실행 수입니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 `PerExecutionEnvironmentMaxConcurrency` 속성으로 직접 전달됩니다.

예제

용량 공급자 구성

다음 예제는 용량 공급자 리소스를 참조하는 용량 공급자 구성을 보여줍니다.

```
CapacityProviderConfig:
  Arn: !GetAtt MyCapacityProvider.Arn
  ExecutionEnvironmentMemoryGiBPerVCpu: 4.0
  PerExecutionEnvironmentMaxConcurrency: 100
```

FunctionScalingConfig

생성할 수 있는 실행 환경(샌드박스) 수를 제어하여 Lambda 함수 버전의 조정 동작을 구성합니다. 이 구성은 $LATEST.PUBLISHED 및 숫자 함수 버전 모두에 적용됩니다.

구문

AWS Serverless Application Model (AWS SAM) 템플릿에서이 개체를 선언하려면 다음 구문을 사용합니다.

YAML

```
MinExecutionEnvironments: Integer
MaxExecutionEnvironments: Integer
```

속성

`MinExecutionEnvironments`

함수 버전에 대해 유지할 최소 실행 환경 수입니다.

유형: 정수

필수 항목 여부: 아니요

기본값: 3

최소: `0`

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 `MinExecutionEnvironments` 속성으로 직접 전달됩니다.

`MaxExecutionEnvironments`

함수 버전에 대해 생성할 수 있는 최대 실행 환경 수입니다.

유형: 정수

필수 항목 여부: 아니요

기본값: `3`

최소: `0`

CloudFormation 호환성:이 속성은 `AWS::Lambda::Function` 리소스의 `MaxExecutionEnvironments` 속성으로 직접 전달됩니다.

예제

함수 조정 구성

다음 예제에서는 최소 및 최대 실행 환경의 함수 조정 구성을 보여줍니다.

```
FunctionScalingConfig:
  MinExecutionEnvironments: 5
  MaxExecutionEnvironments: 100
```

AWS::Serverless::GraphQLApi

AWS Serverless Application Model(AWS SAM) `AWS::Serverless::GraphQLApi` 리소스 유형을 사용하여 서버리스 애플리케이션용 AWS AppSync GraphQL API를 만들고 구성합니다.

AWS AppSync대한 자세한 내용은 AWS AppSync 개발자 안내서에서 AWS AppSync이란 무엇입니까?를 참조하세요.

구문

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
```

속성

`ApiKeys`

API 키가 필요한 GraphQL 작업을 수행하는 데 사용할 수 있는 고유 키를 생성합니다.

유형: APIKey

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Auth`

GraphQLAPI에 대한 인증을 구성하십시오.

유형: 인증

필수 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Cache`

`CreateApiCache` 작업의 입력.

유형: AWS::AppSync::ApiCache

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS::AppSync::ApiCache 리소스로 직접 전달됩니다.

`DataSources`

AWS AppSync 안에서 연결할 함수의 데이터 소스를 생성합니다. AWS SAM는 Amazon DynamoDB 및AWS Lambda 데이터 소스를 지원합니다.

유형: DataSource

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`DomainName`

GraphQLAPI에 대한 사용자 지정 도메인 이름.

유형: AWS::AppSync::DomainName

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS::AppSync::DomainName 리소스로 직접 전달됩니다. AWS SAM는 AWS::AppSync::DomainNameApiAssociation 리소스를 자동으로 생성합니다.

`Functions`

특정 작업을 수행하도록 GraphQL API의 함수를 구성합니다.

유형: 함수

필수 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Logging`

GraphQL API에 대한 Amazon CloudWatch 로깅을 구성합니다.

이 속성을 지정하지 않으면 다음 값이 AWS SAM이 `CloudWatchLogsRoleArn`를 생성하고 다음 값을 설정합니다.

- `ExcludeVerboseContent: true`
- `FieldLogLevel: ALL`

로깅을 배제하려면 다음을 지정합니다.

```
Logging: false
```

유형: LogConfig

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `LogConfig` 리소스의 `AWS::AppSync::GraphQLApi` 속성으로 직접 전달됩니다.

`LogicalId`

GraphQL API의 고유한 이름.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `Name` 리소스의 `AWS::AppSync::GraphQLApi` 속성으로 직접 전달됩니다.

`Name`

귀하의 GraphQLAPI의 이름입니다. 이 속성을 지정하여 `LogicalId` 값을 재정의합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `Name` 리소스의 `AWS::AppSync::GraphQLApi` 속성으로 직접 전달됩니다.

`Resolvers`

귀하의 GraphQL API 필드에 대한 해석기를 구성하십시오. AWS SAM는 JavaScript파이프라인 리졸버를 지원합니다.

유형: 해석기

필수 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`SchemaInline`

SDL 형식의 GraphQL 스키마 텍스트 표시.

유형: 문자열

필수 항목 여부: 조건부. `SchemaInline` 또는`SchemaUri` 를 지정해야 합니다.

CloudFormation호환성: 이 속성은 [`Definition`]() 리소스의 `AWS::AppSync::GraphQLSchema` 속성으로 직접 전달됩니다.

`SchemaUri`

스키마의 Amazon Simple Storage Service(S3) 버킷 URI 또는 로컬 폴더 경로

로컬 폴더 경로를 지정하는 경우 CloudFormation은 배포하기 전에 먼저 파일을 Amazon S3에 업로드해야 합니다. AWS SAM CLI를 사용하여 이 프로세스를 용이하게 할 수 있습니다. 자세한 내용은 [배포 시 AWS SAM이 로컬 파일을 업로드하는 방법]()을 참조하십시오

유형: 문자열

필수 항목 여부: 조건부. `SchemaInline` 또는`SchemaUri` 를 지정해야 합니다.

CloudFormation호환성: 이 속성은 [`DefinitionS3Location`]() 리소스의 `AWS::AppSync::GraphQLSchema` 속성으로 직접 전달됩니다.

`Tags`

이 GraphQL API에 대한 태그(키-값 페어)입니다. 태그를 사용하여 리소스를 식별하고 범주화하십시오.

유형: [태그]()의 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [`Tag`]() 리소스의 `AWS::AppSync::GraphQLApi` 속성으로 직접 전달됩니다.

`XrayEnabled`

이 리소스에 대해 [AWSX-Ray 추적]()을 사용할지 여부를 표시하십시오.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `XrayEnabled` 리소스의 `AWS::AppSync::GraphQLApi` 속성으로 직접 전달됩니다.

반환 값

반환 값 목록은 CloudFormation 사용 설명서의 AWS::Serverless::GraphQLApi을(를) 참조하세요.

예시

GraphQL APIDynamoDB 데이터 소스 사용

이 예시에서는 DynamoDB 테이블을 데이터 소스로 사용하는 GraphQL API를 생성합니다.

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;
}
```

해석기 코드는 다음과 같습니다.

getPost.js

```
export function request(ctx) {
  return {};
}

export function response(ctx) {
  return ctx.prev.result;
}
```

GraphQLLambda 함수를 데이터 소스로 사용하는 API

이 예시에서는 Lambda 함수를 데이터 소스로 사용하는 GraphQL API를 생성합니다.

template.yaml

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyLambdaFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: nodejs20.x
      CodeUri: ./lambda

  MyGraphQLAPI:
    Type: AWS::Serverless::GraphQLApi
    Properties:
      Name: MyApi
      SchemaUri: ./gql/schema.gql
      Auth:
        Type: API_KEY
      ApiKeys:
        MyApiKey:
          Description: my api key
      DataSources:
        Lambda:
          MyLambdaDataSource:
            FunctionArn: !GetAtt MyLambdaFunction.Arn
      Functions:
        lambdaInvoker:
          Runtime:
            Name: APPSYNC_JS
            Version: 1.0.0
          DataSource: MyLambdaDataSource
          CodeUri: ./gql/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
}
```

함수는 다음과 같습니다.

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

API 키가 필요한 GraphQL 작업을 수행하는 데 사용할 수 있는 고유 키를 생성합니다.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
LogicalId:
  ApiKeyId: String
  Description: String
  ExpiresOn: Double
```

속성

`ApiKeyId`

귀하의 API 키의 고유한 이름. `LogicalId` 값을 재정의하도록 지정합니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `ApiKeyId` 리소스의 `AWS::AppSync::ApiKey` 속성으로 직접 전달됩니다.

`Description`

귀하의 API 키의 고유한 설명.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::AppSync::ApiKey` 리소스의 `Description` 속성으로 직접 전달됩니다.

`ExpiresOn`

API 키가 만료되는 시간입니다. 날짜는 Epoch 이후 경과한 초로 표시되며 가장 가까운 시간으로 반올림됩니다.

유형: Double

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::AppSync::ApiKey` 리소스의 `Expires` 속성으로 직접 전달됩니다.

`LogicalId`

귀하의 API 키의 고유한 이름.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `ApiKeyId` 리소스의 `AWS::AppSync::ApiKey` 속성으로 직접 전달됩니다.

Auth

귀하의 GraphQL API에 대한 권한 부여를 구성합니다.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
Additional:
- AuthProvider
LambdaAuthorizer: LambdaAuthorizerConfig
OpenIDConnect: OpenIDConnectConfig
Type: String
UserPool: UserPoolConfig
```

속성

`Additional`

귀하의 GraphQLAPI에 대한 추가 인증 유형 목록.

유형: AuthProvider 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`LambdaAuthorizer`

Lambda 함수 권한 부여자에 대한 선택적 권한 부여 구성을 지정합니다. `Type`이 `AWS_LAMBDA`로 지정된 경우 이 선택적 속성을 구성할 수 있습니다.

유형: [LambdaAuthorizerConfig](#)

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::AppSync::GraphQLApi` 리소스의 [`LambdaAuthorizerConfig`](#) 속성으로 직접 전달됩니다.

`OpenIDConnect`

귀하의 OpenID Connect 규정 준수 서비스를 위한 선택적 권한 부여 구성을 지정합니다. `Type`이 `OPENID_CONNECT`로 지정된 경우 이 선택적 속성을 구성할 수 있습니다.

유형: [OpenIDConnectConfig](#)

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::AppSync::GraphQLApi` 리소스의 [`OpenIDConnectConfig`](#) 속성으로 직접 전달됩니다.

`Type`

애플리케이션과 귀하의 AWS AppSync GraphQL API 간의 기본 인증 유형입니다.

허용된 값의 목록과 설명은 AWS AppSync개발자 안내서의 [권한 부여 및 인증을](#) 참조하세요.

Lambda 권한 부여자(`AWS_LAMBDA`)를 지정하면, AWS SAM가 귀하의 GraphQLAPI와 Lambda 함수 사이에 권한을 프로비저닝하는 AWS Identity and Access Management(IAM) 정책을 생성합니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 [`AuthenticationType`](#) 리소스의 `AWS::AppSync::GraphQLApi` 속성으로 직접 전달됩니다.

`UserPool`

Amazon Cognito 사용자 풀을 사용하기 위한 선택적 권한 부여 구성을 지정합니다. `Type`이 `AMAZON_COGNITO_USER_POOLS`로 지정된 경우 이 선택적 속성을 구성할 수 있습니다.

유형: UserPoolConfig

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `UserPoolConfig` 리소스의 `AWS::AppSync::GraphQLApi` 속성으로 직접 전달됩니다.

예시

기본 및 추가 권한 부여 유형을 구성합니다.

이 예시에서는 먼저 Lambda 권한 부여자를 GraphQL API의 기본 권한 부여 유형으로 구성합니다.

```
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
```

다음으로 AWS SAM 템플릿에 다음을 추가하여 GraphQL API에 대한 추가 권한 부여 유형을 구성합니다.

```
        Additional:
        - Type: AWS_IAM
        - Type: API_KEY
        - Type: OPENID_CONNECT
          OpenIDConnect:
            AuthTTL: 10
            ClientId: myId
```

```
            IatTTL: 10
            Issuer: prod
```

이 결과 다음과 같은 AWS SAM 템플릿이 생성됩니다.

```
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

귀하의 추가적인 GraphQL API 권한 부여 유형을 위한 선택적 권한 부여 구성.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
LambdaAuthorizer: LambdaAuthorizerConfig
OpenIDConnect: OpenIDConnectConfig
Type: String
UserPool: UserPoolConfig
```

속성

`LambdaAuthorizer`

귀하의 AWS Lambda 함수 권한 부여자의 선택적 권한 부여 구성을 지정합니다. `Type`이 `AWS_LAMBDA`로 지정된 경우 이 선택적 속성을 구성할 수 있습니다.

유형: [LambdaAuthorizerConfig](#)

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [LambdaAuthorizerConfig](#) `AWS::AppSync::GraphQLApi` 객체의 [AdditionalAuthenticationProvider](#) 속성에 직접 전달됩니다.

`OpenIDConnect`

귀하의 OpenID Connect 규정 준수 서비스를 위한 선택적 권한 부여 구성을 지정합니다. `Type`이 `OPENID_CONNECT`로 지정된 경우 이 선택적 속성을 구성할 수 있습니다.

유형: [OpenIDConnectConfig](#)

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [OpenIDConnectConfig](#) `AWS::AppSync::GraphQLApi` 객체의 [AdditionalAuthenticationProvider](#) 속성에 직접 전달됩니다.

`Type`

애플리케이션과 귀하의 AWS AppSync GraphQL API 간의 기본 인증 유형입니다.

허용된 값의 목록과 설명은 AWS AppSync개발자 안내서의 [권한 부여 및 인증을](#) 참조하십시오.

Lambda 권한 부여자(`AWS_LAMBDA`)를 지정하면, AWS SAM가 귀하의 GraphQLAPI와 Lambda 함수 사이에 권한을 프로비저닝하는 AWS Identity and Access Management(IAM) 정책을 생성합니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 [AuthenticationType](#) `AWS::AppSync::GraphQLApi` 객체의 [AdditionalAuthenticationProvider](#) 속성에 직접 전달됩니다.

`UserPool`

Amazon Cognito 사용자 풀을 사용하기 위한 선택적 권한 부여 구성을 지정합니다. `Type`이 `AMAZON_COGNITO_USER_POOLS`로 지정된 경우 이 선택적 속성을 구성할 수 있습니다.

유형: [UserPoolConfig]

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [`UserPoolConfig`] `AWS::AppSync::GraphQLApi` 객체의 [`AdditionalAuthenticationProvider`] 속성에 직접 전달됩니다.

데이터 소스

GraphQL API 해석기가 연결할 수 있는 데이터 소스를 구성하십시오. AWS Serverless Application Model(AWS SAM) 템플릿을 사용하여 다음 데이터 소스에 대한 연결을 구성할 수 있습니다.

- Amazon DynamoDB
- AWS Lambda

데이터 소스에 대해 자세히 알아보려면 AWS AppSync개발자 안내서의 [데이터 소스 연결]을 참조하세요.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
DynamoDb: DynamoDb
Lambda: Lambda
```

속성

`DynamoDb`

DynamoDB 테이블을 GraphQL API 해석기의 데이터 소스로 구성합니다.

유형: [DynamoDB]

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Lambda`

Lambda 함수를 귀하의 GraphQL API 해석기의 데이터 소스로 구성합니다.

유형: Lambda

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

DynamoDb

Amazon DynamoDB 테이블을 귀하의 GraphQLAPI 해석기의 데이터 소스로 구성합니다.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
LogicalId:
  DeltaSync: DeltaSyncConfig
  Description: String
  Name: String
  Permissions: List
  Region: String
  ServiceRoleArn: String
  TableArn: String
  TableName: String
  UseCallerCredentials: Boolean
  Versioned: Boolean
```

속성

`DeltaSync`

델타 동기화 구성에 대해 설명합니다.

유형: [DeltaSyncConfig]

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `DeltaSyncConfig` `AWS::AppSync::DataSource` `DynamoDBConfig` 객체의 속성에 직접 전달됩니다.

`Description`

귀하의 데이터 소스에 대한 설명.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::AppSync::DataSource` 리소스의 `Description` 속성으로 직접 전달됩니다.

`LogicalId`

귀하의 데이터 소스의 고유한 이름.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `Name` 리소스의 `AWS::AppSync::DataSource` 속성으로 직접 전달됩니다.

`Name`

귀하의 데이터 소스의 이름. 이 속성을 지정하여 `LogicalId` 값을 재정의합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::AppSync::DataSource` 리소스의 `Name` 속성으로 직접 전달됩니다.

`Permissions`

AWS SAM 커넥터를 사용하여 데이터 소스에 권한을 제공합니다. 다음 값 중 하나를 목록에 제공할 수 있습니다.

- `Read` – 귀하의 리졸버가 데이터 소스를 읽을 수 있도록 허용합니다.

- Write – 귀하의 리졸버가 데이터 소스에 쓸 수 있도록 허용합니다.

AWS SAM은 배포 시 변환된 AWS::Serverless::Connector 리소스를 사용하여 귀하의 권한을 프로비저닝합니다. 생성된 리소스에 대한 자세한 내용은을 참조하세요CloudFormation을 지정한 경우 생성되는 AWS::Serverless::Connector 리소스 .

> **Note**
>
> Permissions 또는 ServiceRoleArn을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다. 둘 중 어느 것도 지정되지 않으면 AWS SAM은 기본값인 Read 및 Write을 생성합니다. 데이터 소스에 대한 액세스 권한을 취소하려면 귀하의 AWS SAM 템플릿에서 DynamoDB 객체를 제거합니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다. 이것은 AWS::Serverless::Connector 리소스의 Permissions 속성과 유사합니다.

Region

귀하의 DynamoDB 테이블의 AWS 리전. 이것을 지정하지 않으면, AWS SAM이 AWS::Region를 사용합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AwsRegion AWS::AppSync::DataSource DynamoDBConfig 객체의 속성에 직접 전달됩니다.

ServiceRoleArn

데이터 소스에 대한 AWS Identity and Access Management(IAM) 서비스 역할 ARN입니다. 시스템은 데이터 소스에 액세스할 때 이 역할을 사용합니다.

Permissions 또는 ServiceRoleArn을 지정할 수 있지만 둘 다 함께 지정할 수는 없습니다.

유형: 문자열

필수 여부: 아니요. 지정하지 않을 경우 AWS SAM는 Permissions의 기본값을 적용합니다.

CloudFormation호환성: 이 속성은 `AWS::AppSync::DataSource` 리소스의 [`ServiceRoleArn`](#) 속성으로 직접 전달됩니다.

`TableArn`

DynamoDB 테이블의 ARN.

유형: 문자열

필수 항목 여부: 조건부. `ServiceRoleArn`를 지정하지 않는 경우 `TableArn`가 필수입니다.

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`TableName`

테이블 이름.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 [`TableName`](#) `AWS::AppSync::DataSource` `DynamoDBConfig` 객체의 속성에 직접 전달됩니다.

`UseCallerCredentials`

이 데이터 소스로 IAM을 사용하려면 `true`로 설정합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [`UseCallerCredentials`](#) `AWS::AppSync::DataSource` `DynamoDBConfig` 객체의 속성에 직접 전달됩니다.

`Versioned`

이 데이터 소스로 [충돌 감지 및 충돌 해결 및 동기화](#)를 사용하려면 `true`로 설정합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [`Versioned`](#) `AWS::AppSync::DataSource` `DynamoDBConfig` 객체의 속성에 직접 전달됩니다.

Lambda

귀하의 GraphQL API 해석기의 데이터 소스로 AWS Lambda함수를 구성합니다.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
LogicalId:
  Description: String
  FunctionArn: String
  Name: String
  ServiceRoleArn: String
```

속성

`Description`

귀하의 데이터 소스에 대한 설명.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::AppSync::DataSource` 리소스의 `Description` 속성으로 직접 전달됩니다.

`FunctionArn`

Lambda 함수의 ARN입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `LambdaFunctionArn` `AWS::AppSync::DataSource` `LambdaConfig` 객체의 속성에 직접 전달됩니다.

`LogicalId`

귀하의 데이터 소스의 고유한 이름.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 Name 리소스의 `AWS::AppSync::DataSource` 속성으로 직접 전달됩니다.

`Name`

귀하의 데이터 소스의 이름. 이 속성을 지정하여 `LogicalId` 값을 재정의합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::AppSync::DataSource` 리소스의 Name 속성으로 직접 전달됩니다.

`ServiceRoleArn`

데이터 소스에 대한 AWS Identity and Access Management(IAM) 서비스 역할 ARN입니다. 시스템은 데이터 소스에 액세스할 때 이 역할을 사용합니다.

> **Note**
> 데이터 소스에 대한 액세스 권한을 취소하려면 귀하의 AWS SAM템플릿에서 Lambda 객체를 제거하십시오.

유형: 문자열

필수 여부: 아니요. 지정하지 않으면 AWS SAM는 AWS SAM 커넥터를 사용하여 `Write` 권한을 제공합니다.

CloudFormation호환성: 이 속성은 `ServiceRoleArn` 리소스의 `AWS::AppSync::DataSource` 속성으로 직접 전달됩니다.

함수

특정 작업을 수행하도록 GraphQL API의 함수를 구성합니다.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
LogicalId:
  CodeUri: String
  DataSource: String
  Description: String
  Id: String
  InlineCode: String
  MaxBatchSize: Integer
  Name: String
  Runtime: Runtime
  Sync: SyncConfig
```

속성

CodeUri

함수 코드의 Amazon Simple Storage Service(S3) URI 또는 로컬 폴더 경로.

로컬 폴더 경로를 지정하는 경우 CloudFormation은 배포하기 전에 먼저 파일을 Amazon S3에 업로드해야 합니다. AWS SAM CLI를 사용하여 이 프로세스를 용이하게 할 수 있습니다. 자세한 내용은 [배포 시 AWS SAM이 로컬 파일을 업로드하는 방법](#)을 참조하십시오

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `CodeS3Location` 리소스의 `AWS::AppSync::FunctionConfiguration` 속성으로 직접 전달됩니다.

DataSource

이 함수가 연결되는 데이터 원본의 이름입니다.

- `AWS::Serverless::GraphQLApi` 리소스 내 데이터 소스를 참조하려면 해당 데이터 소스의 논리적 ID를 지정합니다.
- `AWS::Serverless::GraphQLApi` 리소스 외부의 데이터 소스를 참조하려면 `Fn::GetAtt` 내장 함수를 사용하여 해당 `Name` 속성을 제공합니다. 예를 들어 `!GetAtt MyLambdaDataSource.Name`입니다.
- 다른 스택의 데이터 소스를 참조하려면 `Fn::ImportValue`를 사용합니다.

`[NONE | None | none]`의 변형이 지정된 경우 AWS SAM은 `AWS::AppSync::DataSource` `Type` 객체에 대한 `None` 값을 생성합니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `DataSourceName` 리소스의 `AWS::AppSync::FunctionConfiguration` 속성으로 직접 전달됩니다.

`Description`

함수에 대한 설명.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `Description` 리소스의 `AWS::AppSync::FunctionConfiguration` 속성으로 직접 전달됩니다.

`Id`

`AWS::Serverless::GraphQLApi` 리소스 외부에 있는 함수의 함수 ID입니다.

- 동일한 AWS SAM 템플릿 내에서 함수를 참조하려면 `Fn::GetAtt` 내장 함수를 사용합니다. 예: `Id: !GetAtt createPostItemFunc.FunctionId`.
- 다른 스택의 함수를 참조하려면 `Fn::ImportValue`를 사용합니다.

`Id`를 사용하는 경우 다른 모든 속성은 허용되지 않습니다. AWS SAM은 참조된 함수의 함수 ID를 자동으로 전달합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`InlineCode`

요청 및 응답 함수가 포함된 코드입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `Code` 리소스의 `AWS::AppSync::FunctionConfiguration` 속성으로 직접 전달됩니다.

`LogicalId`

함수의 고유한 이름.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 Name 리소스의 `AWS::AppSync::FunctionConfiguration` 속성으로 직접 전달됩니다.

`MaxBatchSize`

`BatchInvoke` 작업에서 단일 AWS Lambda 함수에 보낼 최대 해석기 요청 입력 수입니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation 호환성: 이 속성은 `AWS::AppSync::FunctionConfiguration` 리소스의 MaxBatchSize 속성으로 직접 전달됩니다.

`Name`

함수의 이름입니다. `LogicalId` 값을 재정의하도록 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 Name 리소스의 `AWS::AppSync::FunctionConfiguration` 속성으로 직접 전달됩니다.

`Runtime`

AWS AppSync 파이프라인 해석기 또는 AWS AppSync 함수에서 사용하는 런타임을 설명합니다. 사용할 런타임의 이름과 버전을 지정합니다.

유형: 런타임

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다. 이것은 `Runtime` 리소스의 `AWS::AppSync::FunctionConfiguration` 속성과 유사합니다.

`Sync`

해석기에 대한 동기화 구성에 대해 설명합니다.

해석기가 호출될 때 사용할 충돌 감지 전략과 해결 전략을 지정합니다.

유형: SyncConfig

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `SyncConfig` 리소스의 `AWS::AppSync::FunctionConfiguration` 속성으로 직접 전달됩니다.

런타임

파이프라인 해석기 또는 함수의 런타임 사용할 런타임의 이름과 버전을 지정합니다.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
Name: String
Version: String
```

속성

`Name`

사용할 런타임의 이름. 현재, 유일하게 허용되는 값은 `APPSYNC_JS`입니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `Name` `AWS::AppSync::FunctionConfiguration` `AppSyncRuntime` 객체의 속성에 직접 전달됩니다.

`Version`

사용할 런타임의 버전. 현재 유일하게 허용되는 버전은 `1.0.0`입니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `RuntimeVersion` `AWS::AppSync::FunctionConfiguration AppSyncRuntime` 객체의 속성에 직접 전달됩니다.

해석기

GraphQL API 필드에 대한 해석기를 구성합니다. AWS Serverless Application Model(AWS SAM)는 JavaScript 파이프라인 해석기를 지원합니다.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
OperationType:
  LogicalId:
    Caching: CachingConfig
    CodeUri: String
    FieldName: String
    InlineCode: String
    MaxBatchSize: Integer
    Pipeline: List
    Runtime: Runtime
    Sync: SyncConfig
```

속성

`Caching`

캐싱이 활성화된 해석기에 대한 캐싱 구성입니다.

Type: CachingConfig

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `CachingConfig` 리소스의 `AWS::AppSync::Resolver` 속성으로 직접 전달됩니다.

`CodeUri`

해석기 함수 코드의 Amazon Simple Storage Service(S3) URI 또는 로컬 폴더 경로.

로컬 폴더 경로를 지정하는 경우 CloudFormation은 배포하기 전에 먼저 파일을 Amazon S3에 업로드해야 합니다. AWS SAM CLI를 사용하여 이 프로세스를 용이하게 할 수 있습니다. 자세한 내용은 배포 시 AWS SAM이 로컬 파일을 업로드하는 방법 섹션을 참조하세요.

`CodeUri` 또는 `InlineCode`가 제공되지 않은 경우 AWS SAM는 요청을 첫 번째 파이프라인 함수로 리디렉션하고 마지막 파이프라인 함수로부터 응답을 받는 `InlineCode`를 생성합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `CodeS3Location` 리소스의 `AWS::AppSync::Resolver` 속성으로 직접 전달됩니다.

`FieldName`

해석기의 이름. 이 속성을 지정하여 `LogicalId` 값을 재정의합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `FieldName` 리소스의 `AWS::AppSync::Resolver` 속성으로 직접 전달됩니다.

`InlineCode`

요청 및 응답 함수가 포함된 해석기 코드입니다.

`CodeUri` 또는 `InlineCode`가 제공되지 않은 경우 AWS SAM는 요청을 첫 번째 파이프라인 함수로 리디렉션하고 마지막 파이프라인 함수로부터 응답을 받는 `InlineCode`를 생성합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `Code` 리소스의 `AWS::AppSync::Resolver` 속성으로 직접 전달됩니다.

`LogicalId`

해석기의 고유 이름입니다. GraphQL 스키마에서 해석기 이름은 사용된 필드 이름과 일치해야 합니다. `LogicalId`에 같은 필드 이름을 사용합니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`MaxBatchSize`

`BatchInvoke` 작업에서 단일 AWS Lambda 함수에 보낼 최대 해석기 요청 입력 수입니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [`MaxBatchSize`] 리소스의 `AWS::AppSync::Resolver` 속성으로 직접 전달됩니다.

`OperationType`

해석기와 관련된 GraphQL 작업 유형입니다. 예: `Query`, `Mutation` 또는 `Subscription`. 단일 `OperationType` 내에 `LogicalId`으로 여러 개의 해석기를 중첩할 수 있습니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 [`TypeName`] 리소스의 `AWS::AppSync::Resolver` 속성으로 직접 전달됩니다.

`Pipeline`

파이프라인 해석기와 연결된 함수. 목록에서 논리적 ID로 함수를 지정합니다.

유형: 목록

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다. 이것은 [`PipelineConfig`] 리소스의 `AWS::AppSync::Resolver` 속성과 유사합니다.

`Runtime`

파이프라인 해석기 또는 함수의 런타임 사용할 런타임의 이름과 버전을 지정합니다.

유형: [런타임]

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다. 이것은 [Runtime](#) 리소스의 `AWS::AppSync::Resolver` 속성과 유사합니다.

`Sync`

해석기에 대한 동기화 구성에 대해 설명합니다.

해석기가 호출될 때 사용할 충돌 감지 전략과 해결 전략을 지정합니다.

유형: [SyncConfig](#)

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [SyncConfig](#) 리소스의 `AWS::AppSync::Resolver` 속성으로 직접 전달됩니다.

예시

AWS SAM 생성 해석기 함수 코드를 사용하고 필드를 변수로 저장합니다.

예제의 GraphQL 스키마는 다음과 같습니다.

```
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
}
```

AWS SAM 템플릿 스니펫은 다음과 같습니다.

```
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
```

AWS SAM 템플릿에서는 `CodeUri` 또는 `InlineCode`를 지정하지 않습니다. 배포 시, AWS SAM은 해석기에 대해 다음과 같은 인라인 코드를 자동으로 생성합니다.

```
export function request(ctx) {
  return {};
}

export function response(ctx) {
  return ctx.prev.result;
}
```

이 디폴트 해석기 코드는 요청을 첫 번째 파이프라인 함수로 리디렉션하고 마지막 파이프라인 함수로부터 응답을 받습니다.

첫 번째 파이프라인 함수에서는 제공된 `args` 필드를 사용하여 요청 객체를 파싱하고 변수를 생성할 수 있습니다. 그러면 함수 내에서 이러한 변수를 사용할 수 있습니다. 다음은 `preprocessPostItem` 함수의 예입니다.

```
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;
}
```

런타임

파이프라인 해석기 또는 함수의 런타임 사용할 런타임의 이름과 버전을 지정합니다.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
Name: String
Version: String
```

속성

Name

사용할 런타임의 이름. 현재, 유일하게 허용되는 값은 APPSYNC_JS입니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 Name AWS::AppSync::Resolver AppSyncRuntime 객체의 속성에 직접 전달됩니다.

Version

사용할 런타임의 버전. 현재 유일하게 허용되는 버전은 1.0.0입니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `RuntimeVersion` `AWS::AppSync::Resolver` `AppSyncRuntime` 객체의 속성에 직접 전달됩니다.

AWS::Serverless::HttpApi

Amazon API Gateway HTTP API를 생성함으로써 귀하는 REST API보다 지연 시간이 짧고 비용이 저렴한 RESTful API를 생성할 수 있습니다. 자세한 내용을 알아보려면 API Gateway 개발자 안내서의 HTTP API로 작업하기를 참조하세요.

CloudFormation 후크 또는 IAM 정책을 사용하여 API Gateway 리소스에 액세스를 제어할 권한 부여자가 연결되어 있는지 확인하는 것이 좋습니다.

CloudFormation 후크 사용에 대한 자세한 내용은 CLI 사용자 가이드의 CloudFormation후크 등록 및 apigw-enforce-authorizer GitHub 리포지토리를 참조하세요.

IAM 정책 사용에 대한 자세한 내용은 API 게이트웨이 개발자 가이드의 API 경로에 권한 부여 필요를 참조하세요.

> **Note**
>
> AWS CloudFormation에 배포하면 AWS SAM는 귀하의 AWS SAM 리소스를 CloudFormation 리소스로 변환됩니다. 자세한 내용은 AWS SAM용으로 생성된 CloudFormation 리소스 섹션을 참조하십시오.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

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
```

속성

`AccessLogSettings`

단계의 로깅 액세스에 대한 설정입니다.

유형: AccessLogSettings

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::ApiGatewayV2::Stage` 리소스의 `AccessLogSettings` 속성으로 직접 전달됩니다.

`Auth`

API Gateway HTTP API에 대한 액세스를 제어하기 위한 권한 부여를 구성합니다.

자세한 내용은 API Gateway 개발자 가이드에서 JWT 권한 부여자를 사용하여 HTTP API에 대한 액세스 제어를 참조하십시오.

유형: HTTP/APIAuth

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`CorsConfiguration`

귀하의 모든 API Gateway HTTP API를 위하여 교차 오리진 리소스 공유(CORS)를 관리합니다. 허용할 도메인을 문자열로 지정하거나 `HttpApiCorsConfiguration` 객체를 지정합

니다. 단, CORS는 AWS SAM로 하여금 OpenAPI 정의를 수정하도록 해야 하므로 CORS는 `DefinitionBody` 속성이 지정된 경우에만 작동합니다.

자세한 내용은 API Gateway 개발자 안내서의 [HTTP API에 대한 CORS 구성]을 참조하세요.

> **Note**
>
> `CorsConfiguration`이 OpenAPI 정의와 속성 수준에서 모두 설정된 경우 AWS SAM는 두 구성 소스를 우선 순위가 있는 속성과 병합합니다. 이 속성을 `true`로 설정하면 모든 오리진이 허용됩니다.

유형: 문자열 |HttpApiCorsConfiguration

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`DefaultRouteSettings`

이 HTTP API에 대한 기본 라우팅 설정입니다. 이 설정은 특정 경로에 대해 `RouteSettings` 속성에서 재정의하지 않는 한 모든 경로에 적용됩니다.

유형: RouteSettings

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::ApiGatewayV2::Stage` 리소스의 `RouteSettings` 속성으로 직접 전달됩니다.

`DefinitionBody`

HTTP API를 설명하는 OpenAPI 정의. 귀하가 `DefinitionUri` 또는 `DefinitionBody` 를 지정하지 않으면 AWS SAM이 귀하의 템플릿 구성을 기반으로 귀하의 `DefinitionBody`를 자동으로 생성합니다.

유형: JSON

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::ApiGatewayV2::Api` 리소스의 `Body` 속성과 유사합니다. 특정 속성이 제공되는 경우, AWS SAM은 CloudFormation에 전달되기 전에

DefinitionBody에 콘텐츠를 삽입하거나 수정할 수 있습니다. 속성들에는 Auth 및 해당 AWS::Serverless::Function 속성을 위한 HttpApi 유형의 EventSource가 포함됩니다.

DefinitionUri

HTTP API를 정의할 수 있는 OpenAPI 정의의 Amazon Simple Storage Service(S3) URI이거나 로컬 파일 경로이거나 위치 객체입니다. 이 속성이 참조하는 Amazon S3 객체는 유효한 OpenAPI 정의 파일이어야 합니다. 귀하가 DefinitionUri 또는 DefinitionBody를 지정하지 않으면 AWS SAM이 귀하의 템플릿 구성을 기반으로 귀하의 DefinitionBody를 생성합니다.

귀하가 로컬 파일 경로를 제공하는 경우 템플릿은 sam deploy 또는 sam package 명령이 포함된 워크플로를 거쳐야 정의가 제대로 변환됩니다.

귀하가 DefinitionUri로 참조하는 외부 OpenAPI 정의 파일에서는 내장 함수가 지원되지 않습니다. OpenAPI 정의를 템플릿으로 들여오려면 해당 DefinitionBody 속성을 Include 변환과 함께 사용합니다.

유형: 문자열 | HttpApiDefinition

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS::ApiGatewayV2::Api 리소스의 BodyS3Location 속성과 유사합니다. 중첩된 Amazon S3 속성은 다르게 지정됩니다.

Description

HTTP API 리소스 그룹에 대한 설명입니다.

귀하가 Description을 지정하면 AWS SAM는 description필드를 설정하여 HTTP API 리소스의 OpenAPI 정의를 수정합니다. 다음 시나리오에서는 오류가 발생합니다.

- DefinitionBody 속성이 Open API 정의에 설정된 description 필드로 지정됨 – 이로 인해 description 필드 충돌이 발생하지만 AWS SAM가 이를 해결하지 못합니다.
- DefinitionUri 속성이 지정되었으므로 AWS SAM는 Amazon S3에서 검색된 오픈 API 정의를 수정하지 않습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`DisableExecuteApiEndpoint`

클라이언트가 기본 execute-api 엔드포인트https://{api_id}.execute-api.{region}.amazonaws.com를 사용하여 HTTP API를 호출할 수 있는지 여부를 지정합니다. 기본적으로 클라이언트는 기본 엔드포인트로 API를 호출할 수 있습니다. 클라이언트가 단지 사용자 지정 도메인 이름을 사용하여 API를 호출하도록 요구하려면 기본 엔드포인트를 비활성화합니다.

이 속성을 사용하려면 DefinitionUri 속성 대신 DefinitionBody 속성을 지정하거나 OpenAPI 정의에서 x-amazon-apigateway-endpoint-configuration을 disableExecuteApiEndpoint로 정의해야 합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS::ApiGatewayV2::Api 리소스의 DisableExecuteApiEndpoint 속성과 유사합니다. 이것은 x-amazon-apigateway-endpoint-configuration 확장의 disableExecuteApiEndpoint 속성으로 직접 전달되며, 이것은 AWS::ApiGatewayV2::Api 리소스의 Body 속성에 추가됩니다.

`Domain`

이 API Gateway HTTP API의 사용자 지정 도메인을 구성합니다.

유형: HttpApiDomainConfiguration

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`FailOnWarnings`

경고가 발생할 때 HTTP API 생성을 롤백할 것인지(true), 아니면 롤백하지 않을 것인지(false) 지정합니다. 기본값은 false입니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS::ApiGatewayV2::Api 리소스의 FailOnWarnings 속성으로 직접 전달됩니다.

`Name`

HTTP API 리소스의 이름입니다.

귀하가 `Name`을 지정하면 AWS SAM는 `title`필드를 설정하여 HTTP API 리소스의 OpenAPI 정의를 수정합니다. 다음 시나리오에서는 오류가 발생합니다.

- `DefinitionBody` 속성이 Open API 정의에 설정된 `title` 필드로 지정됨 – 이로 인해 `title` 필드 충돌이 발생하지만 AWS SAM가 이를 해결하지 못합니다.
- `DefinitionUri` 속성이 지정되었으므로 AWS SAM는 Amazon S3에서 검색된 오픈 API 정의를 수정하지 않습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`PropagateTags`

`Tags`속성의 태그를 AWS::Serverless::HttpApi 생성된 리소스로 전달할지 여부를 지정합니다. 귀하의 생성된 리소스에 태그를 전파하도록 `True`을 지정합니다.

유형: 부울

필수 항목 여부: 아니요

기본값: `False`

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`RouteSettings`

이 HTTP API의 경로별 경로 설정입니다. 자세한 내용을 알아보려면 API Gateway 개발자 안내서의 HTTP API에 대한 루트 작업을 참조하세요.

유형: RouteSettings

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::ApiGatewayV2::Stage` 리소스의 `RouteSettings` 속성으로 직접 전달됩니다.

`StageName`

API 단계의 이름. 이름을 지정하지 않은 경우 AWS SAM은 API Gateway로부터 `$default` 스테이지를 사용합니다.

유형: 문자열

필수 항목 여부: 아니요

기본값: $default

CloudFormation호환성: 이 속성은 `AWS::ApiGatewayV2::Stage` 리소스의 [`StageName`] 속성으로 직접 전달됩니다.

`StageVariables`

단계 변수를 정의하는 맵입니다. 변수 이름은 영숫자와 밑줄 문자를 사용할 수 있습니다. 값은 [A-Za-z0-9-._~:/?#&=,]+와 일치해야 합니다.

유형: [Json]

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::ApiGatewayV2::Stage` 리소스의 [`StageVariables`] 속성으로 직접 전달됩니다.

`Tags`

이 API Gateway 단계에 추가할 태그를 지정하는 맵(문자열에 문자열)입니다. 키는 길이가 1~128자인 유니코드 문자이며 `aws:`로 시작할 수 없습니다. 유니코드 문자 세트, 숫자, 공백, _, ., /, =, +, - 등의 문자를 무엇이든 사용할 수 있습니다. 값은 길이가 1~256자인 유니코드 문자입니다.

유형: 맵

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

추가 참고 사항: `Tags` 속성은 AWS SAM에게 OpenAPI 정의를 수정할 것을 요구하므로 `DefinitionBody`속성이 지정된 경우에만 태그가 추가되고 `DefinitionUri`속성이 지정된 경우에는 태그가 추가되지 않습니다. AWS SAM은 `httpapi:createdBy:SAM`태그를 자동으로 추가합니다. `AWS::ApiGatewayV2::Stage`리소스와 `AWS::ApiGatewayV2::DomainName`리소스 (`DomainName`이 지정된 경우)에도 태그가 추가됩니다.

반환 값

Ref

이 리소스의 논리적 ID를 내장 `Ref` 함수에 전달하면 `Ref`가 기저의 `AWS::ApiGatewayV2::Api` 리소스의 API ID를 반환합니다(예:`a1bcdef2gh`)

`Ref` 함수의 사용에 대한 자세한 내용은 AWS CloudFormation사용자 안내서의 [`Ref`] 항목을 참조하십시오.

예시

단순 HTTP API

다음 예는 Lambda 함수가 지원하는 HTTP API 엔드포인트를 설정하는 데 필요한 최소값을 보여줍니다. 이 예제에서는 AWS SAM이 생성하는 기본 HTTP API를 사용합니다.

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
```

권한 부여가 있는 HttpApi

다음 예에서는 HTTP API 엔드포인트에서 권한 부여를 설정하는 방법을 보여드립니다.

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"
```

OpenAPI 정의를 포함하는 HttpApi

다음 예에서는 OpenAPI 정의를 템플릿에 추가하는 방법을 보여드립니다.

참고로, AWS SAM은 이 HTTP API를 참조하는 HttpApi 이벤트에 대해 누락된 Lambda 통합을 모두 채웁니다. AWS SAM 역시 HttpApi 이벤트가 참조하는 누락된 경로도 모두 추가합니다.

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
```

구성 설정을 포함하는 HttpApi

다음 예에서는 템플릿에 HTTP API 및 단계 구성을 추가하는 방법을 보여드립니다.

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

권한 부여를 구성하여 Amazon API Gateway HTTP API에 대한 액세스를 제어합니다.

HTTP API에 대한 액세스 구성에 대한 자세한 내용은 API Gateway 개발자 안내서의 API Gateway의 HTTP API에 대한 액세스 제어 및 관리 섹션을 참조하세요.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
Authorizers: OAuth2Authorizer | LambdaAuthorizer
DefaultAuthorizer: String
EnableIamAuthorizer: Boolean
```

속성

`Authorizers`

API Gateway API에 대한 액세스를 제어하는 데 사용되는 권한 부여자.

유형: OAuth2Authorizer | LambdaAuthorizer

필수 항목 여부: 아니요

기본값: 없음

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

추가 참고 사항: AWS SAM은 OpenAPI 정의에 권한 부여자를 추가합니다.

`DefaultAuthorizer`

API Gateway API에 대한 API 호출을 승인하는 데 사용할 기본 권한 부여자를 지정합니다. `EnableIamAuthorizer`가 `true`로 설정된 경우 `AWS_IAM`을 기본 권한 부여자로 지정할 수 있습니다. 그렇지 않은 경우 `Authorizers`에서 정의한 권한 부여자를 지정합니다.

유형: 문자열

필수 항목 여부: 아니요

기본값: 없음

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`EnableIamAuthorizer`

API 경로에 IAM 인증을 사용할지 여부를 지정합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

OAuth 2.0 권한 부여자

OAuth 2.0 권한 부여자 예시

YAML

```
Auth:
  Authorizers:
    OAuth2Authorizer:
      AuthorizationScopes:
        - scope1
        - scope2
      JwtConfiguration:
        issuer: "https://www.example.com/v1/connect/oauth2"
        audience:
          - MyApi
      IdentitySource: "$request.querystring.param"
  DefaultAuthorizer: OAuth2Authorizer
```

IAM 권한 부여자

IAM 권한 부여자 예시

YAML

```
Auth:
  EnableIamAuthorizer: true
  DefaultAuthorizer: AWS_IAM
```

LambdaAuthorizer

AWS Lambda 함수를 사용하여 Amazon API Gateway HTTP API에 대한 액세스를 제어하도록 Lambda 권한 부여자를 구성합니다.

자세한 내용 및 예시는 API Gateway 개발자 안내서의HTTP API용 AWS Lambda 권한 부여자 작업을 참조하세요.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
AuthorizerPayloadFormatVersion: String
EnableFunctionDefaultPermissions: Boolean
EnableSimpleResponses: Boolean
FunctionArn: String
FunctionInvokeRole: String
Identity: LambdaAuthorizationIdentity
```

속성

`AuthorizerPayloadFormatVersion`

HTTP API Lambda 권한 부여자에게 전송되는 페이로드의 형식을 지정합니다. HTTP API Lambda 권한 부여자에 필요합니다.

이는 OpenAPI 정의의 `securitySchemes` 섹션에 있는 `x-amazon-apigateway-authorizer`의 `authorizerPayloadFormatVersion` 섹션으로 전달됩니다.

유효한 값: `1.0` 또는 `2.0`

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`EnableFunctionDefaultPermissions`

기본적으로 HTTP API 리소스에는 Lambda 권한 부여자를 호출할 수 있는 권한이 부여되지 않습니다. HTTP API 리소스와 Lambda 권한 부여자 간에 권한을 자동으로 생성하도록 이 속성을 `true`로 지정합니다.

유형: 부울

필수 항목 여부: 아니요

기본 값: `false`

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`EnableSimpleResponses`

Lambda 권한 부여자가 간단한 형식으로 응답을 반환하는지 여부를 지정합니다. 기본적으로 Lambda 권한 부여자는 AWS Identity and Access Management(IAM 정책)을 반환해야 합니다. 사용하도록 설정되면 Lambda 권한 부여자가 IAM 정책 대신 부울 값을 반환할 수 있습니다.

이는 OpenAPI 정의의 `securitySchemes` 섹션에 있는 `x-amazon-apigateway-authorizer`의 `enableSimpleResponses` 섹션으로 전달됩니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`FunctionArn`

API에 대한 인증을 제공하는 Lambda 함수의 Amazon 리소스 이름(ARN)입니다.

이는 OpenAPI 정의의 `securitySchemes` 섹션에 있는 `x-amazon-apigateway-authorizer`의 `authorizerUri` 섹션으로 전달됩니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`FunctionInvokeRole`

API Gateway가 권한 부여자 함수를 호출하는 데 필요한 보안 인증 정보가 있는 IAM 역할의 ARN입니다. 함수의 리소스 기반 정책이 API Gateway `lambda:InvokeFunction` 권한을 부여하지 않는 경우, 이 파라미터를 지정합니다.

이는 OpenAPI 정의의 `securitySchemes` 섹션에 있는 `x-amazon-apigateway-authorizer`의 `authorizerCredentials` 섹션으로 전달됩니다.

자세한 내용은 API Gateway 개발자 안내서의 [Lambda 권한 부여자 만들기](#) 섹션을 참조하세요.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Identity`

권한 부여자에 대한 수신 요청에 `IdentitySource`를 지정합니다.

이는 OpenAPI 정의의 `securitySchemes` 섹션에 있는 `x-amazon-apigateway-authorizer`의 `identitySource` 섹션으로 전달됩니다.

유형: LambdaAuthorizationIdentity

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

LambdaAuthorizer

LambdaAuthorizer 예

YAML

```
Auth:
  Authorizers:
    MyLambdaAuthorizer:
      AuthorizerPayloadFormatVersion: 2.0
      FunctionArn:
        Fn::GetAtt:
          - MyAuthFunction
          - Arn
      FunctionInvokeRole:
        Fn::GetAtt:
          - LambdaAuthInvokeRole
          - Arn
      Identity:
        Headers:
          - Authorization
```

LambdaAuthorizationIdentity

Lambda 권한 부여자에 대한 수신 요청에서 IdentitySource를 지정하는 데 속성을 사용할 수 있습니다. ID 소스에 대한 자세한 내용은 API Gateway 개발자 안내서의 ID 소스를 참조하세요.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

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

속성

`Context`

지정된 컨텍스트 문자열을 `$context.contextString` 형식의 매핑 표현식 목록으로 변환합니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Headers`

헤더를 `$request.header.name` 형식의 매핑 표현식 목록으로 변환합니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`QueryStrings`

지정된 쿼리 문자열을 `$request.querystring.queryString` 형식의 매핑 표현식 목록으로 변환합니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`ReauthorizeEvery`

API Gateway가 권한 부여자 결과를 캐싱하는 기간을 지정하는 TTL(Time to Live) 기간(초)입니다. 0보다 큰 값을 지정한 경우 API Gateway에서 권한 부여자 응답을 캐싱합니다. 최대 값은 3600 또는 1시간입니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`StageVariables`

지정된 단계 변수를 `$stageVariables.stageVariable` 형식의 매핑 표현식 목록으로 변환합니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

LambdaRequestIdentity

Lambda 요청 자격 증명 예제

YAML

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

OAuth2Authorizer

OAuth 2.0 권한 부여자에 대한 정의로, authorizer(JWT) 권한 부여자라고도 합니다.

자세한 내용은 API Gateway 개발자 가이드에서 [JWT 권한 부여자를 사용하는 HTTP API에 대한 액세스 제어]를 참조하십시오.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  AuthorizationScopes: List
  IdentitySource: String
  JwtConfiguration: Map
```

속성

`AuthorizationScopes`

이 권한 부여자의 권한 부여 범위 목록.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`IdentitySource`

이 권한 부여자의 ID 소스 표현식.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`JwtConfiguration`

이 권한 부여자의 JWT 구성.

이는 OpenAPI 정의의 `securitySchemes` 섹션에 있는 `x-amazon-apigateway-authorizer`의 `jwtConfiguration` 섹션으로 전달됩니다.

> **Note**
>
> 속성 `issuer` `audience`는 대소문자를 구분하지 않으므로 OpenAPI에서처럼 소문자를 사용하거나 AWS::ApiGatewayV2::Authorizer에서 `Issuer` 및 `Audience`를 대문자로 사용할 수 있습니다.

유형: 맵

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

OAuth 2.0 권한 부여자

OAuth 2.0 권한 부여자 예시

YAML

```
Auth:
  Authorizers:
    OAuth2Authorizer:
```

```
      AuthorizationScopes:
        - scope1
      JwtConfiguration:
        issuer: "https://www.example.com/v1/connect/oauth2"
        audience:
          - MyApi
      IdentitySource: "$request.querystring.param"
  DefaultAuthorizer: OAuth2Authorizer
```

HttpApiCorsConfiguration

HTTP API에 대한 교차 오리진 리소스 공유(CORS)을 관리합니다. 허용할 도메인을 문자열로 지정하거나 추가 Cors 구성을 사용하여 사전을 지정합니다. 참고: Cors는 OpenAPI 정의를 수정하기 위해 SAM이 필요하므로 `DefinitionBody` 속성에 정의된 인라인 OpenAPI에서만 작동합니다.

HTTP API에 대한 자세한 내용을 알아보려면 API Gateway 개발자 안내서의 HTTP API를 참조하세요.

참고: HttpApiCorsConfiguration이 OpenAPI와 속성 수준에서 모두 설정된 경우 AWS SAM은 이들을 우선하는 속성과 병합합니다.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  AllowCredentials: Boolean
  AllowHeaders: List
  AllowMethods: List
  AllowOrigins: List
  ExposeHeaders: List
  MaxAge: Integer
```

속성

`AllowCredentials`

보안 인증이 CORS 요청에 포함되는지 여부를 지정합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`AllowHeaders`

허용된 헤더의 모음을 나타냅니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`AllowMethods`

허용된 HTTP 메서드의 모음을 나타냅니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`AllowOrigins`

허용된 오리진의 모음을 나타냅니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`ExposeHeaders`

노출된 헤더의 모음을 나타냅니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`MaxAge`

브라우저가 사전 요청 결과를 캐시해야 하는 시간(초)을 지정합니다.

유형: 정수

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

HttpApiCorsConfiguration

HTTP API Cors 구성 예제.

YAML

```
CorsConfiguration:
  AllowOrigins:
    - "https://example.com"
  AllowHeaders:
    - x-apigateway-header
  AllowMethods:
    - GET
  MaxAge: 600
  AllowCredentials: true
```

HttpApiDefinition

API를 정의하는 OpenAPI 문서.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  Bucket: String
```

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

속성

`Bucket`

OpenAPI 파일이 저장되는 Amazon S3 버킷의 이름입니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `Bucket` `AWS::ApiGatewayV2::Api` 데이터 유형의 `BodyS3Location` 속성에 직접 전달됩니다.

`Key`

OpenAPI 파일의 Amazon S3 키.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `Key` `AWS::ApiGatewayV2::Api` 데이터 유형의 `BodyS3Location` 속성에 직접 전달됩니다.

`Version`

버전이 지정된 객체의 경우 OpenAPI 파일의 버전.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `Version` `AWS::ApiGatewayV2::Api` 데이터 유형의 `BodyS3Location` 속성에 직접 전달됩니다.

예시

Uri 정의 예제

API 정의 예

YAML

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

HttpApiDomainConfiguration

API의 사용자 지정 도메인을 구성합니다.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  BasePath: List
  CertificateArn: String
  DomainName: String
  EndpointConfiguration: String
  MutualTlsAuthentication: MutualTlsAuthentication
  OwnershipVerificationCertificateArn: String
  Route53: Route53Configuration
  SecurityPolicy: String
```

속성

`BasePath`

Amazon API Gateway 도메인 이름을 사용하여 구성할 basepath 목록입니다.

유형: 목록

필수 항목 여부: 아니요

기본값: /

CloudFormation 호환성: 이 속성은 `AWS::ApiGatewayV2::ApiMapping` 리소스의 `ApiMappingKey` 속성과 유사합니다. AWS SAM은 이 속성에 지정된 값당 하나씩 여러 `AWS::ApiGatewayV2::ApiMapping` 리소스를 만듭니다.

`CertificateArn`

이 도메인 이름 엔드포인트에 대한 AWS 관리형 인증서의Amazon 리소스 이름(ARN)입니다. 유일하게 지원되는 소스는 AWS Certificate Manager입니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `CertificateArn` 리소스의 `AWS::ApiGateway2::DomainName DomainNameConfiguration` 속성으로 직접 전달됩니다.

`DomainName`

Amazon API Gateway의 API에 대한 사용자 지정 도메인 이름입니다. 대문자는 지원되지 않습니다.

AWS SAM은 이 속성이 설정되면 `AWS::ApiGatewayV2::DomainName` 리소스를 생성합니다. 이 시나리오에 대한 자세한 내용은 DomainName 속성이 지정되었습니다. 섹션을 참조하세요. CloudFormation 리소스 태그 지정에 대한 자세한 내용은 AWS SAM용으로 생성된 CloudFormation 리소스 섹션을 참조하세요.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `DomainName` 리소스의 `AWS::ApiGateway2::DomainName` 속성으로 직접 전달됩니다.

`EndpointConfiguration`

사용자 지정 도메인에 매핑할 API Gateway 엔드포인트의 유형을 정의합니다. 이 속성의 값에 따라 `CertificateArn` 속성이 CloudFormation에 매핑되는 방식이 결정됩니다.

HTTP API의 유일한 유효 값은 `REGIONAL`입니다.

유형: 문자열

필수 항목 여부: 아니요

기본값: `REGIONAL`

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`MutualTlsAuthentication`

사용자 지정 도메인 이름에 대한 상호 전송 계층 보안(TLS) 인증 구성입니다.

유형: MutualTlsAuthentication

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `MutualTlsAuthentication` 리소스의 `AWS::ApiGatewayV2::DomainName` 속성으로 직접 전달됩니다.

`OwnershipVerificationCertificateArn`

사용자 지정 도메인의 소유권을 확인하기 위해 ACM에서 발급한 공인 인증서의 ARN입니다. 상호 TLS를 구성하고 `CertificateArn`에 대해 ACM 가져오기 또는 사설 CA 인증서 ARN을 지정하는 경우에만 필요합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `OwnershipVerificationCertificateArn` `AWS::ApiGatewayV2::DomainName` 데이터 유형의 `DomainNameConfiguration` 속성에 직접 전달됩니다.

`Route53`

Amazon Route 53 구성을 정의합니다.

유형: Route53Configuration

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`SecurityPolicy`

이 도메인 이름에 대한 보안 정책의 TLS(전송 계층 보안) 버전입니다.

HTTP API의 유일한 유효 값은 `TLS_1_2`입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `SecurityPolicy` `AWS::ApiGatewayV2::DomainName` 데이터 유형의 `DomainNameConfiguration` 속성에 직접 전달됩니다.

예시

DomainName

DomainName 예제

YAML

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

Route53Configuration

API에 대한 Route53 레코드 세트를 구성합니다.

구문

AWS Serverless Application Model (AWS SAM) 템플릿에서이 개체를 선언하려면 다음 구문을 사용합니다.

YAML

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

속성

`DistributionDomainName`

API 사용자 지정 도메인 이름의 사용자 지정 배포를 구성합니다.

유형: 문자열

필수 항목 여부: 아니요

기본값: API Gateway 배포를 사용합니다.

CloudFormation 호환성:이 속성은 `AWS::Route53::RecordSetGroup AliasTarget` 리소스의 `DNSName` 속성으로 직접 전달됩니다.

추가 참고 사항: CloudFront 배포의 도메인 이름입니다.

`EvaluateTargetHealth`

EvaluateTargetHealth가 true인 경우 별칭 레코드는 ELB 로드 밸런서 또는 호스팅 영역의 다른 레코드와 같이 참조된 AWS 리소스의 상태를 상속합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Route53::RecordSetGroup AliasTarget` 리소스의 `EvaluateTargetHealth` 속성으로 직접 전달됩니다.

추가 참고 사항: 별칭 대상이 CloudFront 배포인 경우, EvaluateTargetHealth를 true로 설정할 수 없습니다.

`HostedZoneId`

기록을 생성하려는 호스팅 영역의 ID입니다.

`HostedZoneName` 또는 `HostedZoneId` 중 하나를 지정하며 둘 다 지정하지 않습니다. 동일한 도메인 이름을 가진 호스팅 영역이 여러 개 있는 경우 `HostedZoneId`를 사용하여 호스팅 영역을 지정해야 합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Route53::RecordSetGroup RecordSet` 리소스의 [`HostedZoneId`](https://docs.aws.amazon.com/) 속성으로 직접 전달됩니다.

`HostedZoneName`

기록을 생성하려는 호스팅 영역의 이름입니다. 후행 점(예: `www.example.com.`)을 `HostedZoneName`의 일부로 포함해야 합니다.

`HostedZoneName` 또는 `HostedZoneId` 중 하나를 지정하며 둘 다 지정하지 않습니다. 동일한 도메인 이름을 가진 호스팅 영역이 여러 개 있는 경우 `HostedZoneId`를 사용하여 호스팅 영역을 지정해야 합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Route53::RecordSetGroup RecordSet` 리소스의 `HostedZoneName` 속성으로 직접 전달됩니다.

`IpV6`

이 속성이 설정되면는 `AWS::Route53::RecordSet` 리소스를 AWS SAM 생성하고 제공된 HostedZone에 AAAA 대해 유형을 로 설정합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은에 고유 AWS SAM 하며 CloudFormation 동등한 속성이 없습니다.

`Region`

지연 시간 기반 리소스 기록 세트만 해당: 이 리소스 기록 세트가 참조하는 리소스를 생성한 Amazon EC2 리전입니다. 리소스는 일반적으로 EC2 인스턴스 또는 ELB 로드 밸런서와 같은 AWS 리소스이며 레코드 유형에 따라 IP 주소 또는 DNS 도메인 이름으로 참조됩니다.

Amazon Route 53에서 지연 리소스 기록 세트에 대해 생성된 도메인 이름 및 유형에 대한 DNS 쿼리를 수신한 경우 Route 53은 최종 사용자와 연결된 Amazon EC2 리전 간에 지연 시간이 가장 짧은 지연 리소스 기록 세트를 선택합니다. 그 다음에 Route 53은 선택된 리소스 기록 세트와 연결된 값을 반환합니다.

다음 사항에 유의하세요.

- 지연 리소스 기록 세트별로 `ResourceRecord`를 1개씩만 지정할 수 있습니다.
- 각 Amazon EC2 리전에 지연 시간 리소스 기록 세트를 1개씩만 생성할 수 있습니다.
- 모든 Amazon EC2 리전에 지연 시간 리소스 기록 세트를 생성할 필요는 없습니다. Route 53은 지연 시간 리소스 기록 세트를 생성할 리전 중에서 지연 시간이 가장 좋은 리전을 선택합니다.
- `Name` 및 `Type` 요소 값이 지연 시간 리소스 기록 세트와 같은 비-지연 시간 리소스 기록 세트를 생성할 수 없습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Route53::RecordSetGroup RecordSet` 데이터 유형의 [`Region`] 속성으로 직접 전달됩니다.

`SetIdentifier`

단순하지 않은 라우팅 정책이 있는 리소스 기록 세트: 이름과 유형의 조합이 동일한 여러 리소스 기록 세트(이름이 acme.example.com이고 유형이 A인 여러 가중치 기반 리소스 기록 세트) 사이에서 차별화되는 식별자입니다. 이름과 유형이 동일한 리소스 기록 세트 그룹에서 각 리소스 기록 세트의 `SetIdentifier` 값은 고유해야 합니다.

라우팅 정책에 대한 자세한 내용을 알아보려면 Amazon Route 53 개발자 안내서의 라우팅 정책 선택을 참조하세요.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation 호환성:이 속성은 `AWS::Route53::RecordSetGroup RecordSet` 데이터 유형의 `SetIdentifier` 속성으로 직접 전달됩니다.

예제

Route 53 구성 예제

이 예제에서는 Route 53를 구성하는 방법을 보여줍니다.

YAML

```
Domain:
```

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

AWS::Serverless::LayerVersion

Lambda 함수에 필요한 라이브러리 또는 런타임 코드를 포함하는 Lambda LayerVersion을 생성합니다.

또한 AWS::Serverless::LayerVersion 리소스는 `Metadata` 리소스 속성을 지원하므로 애플리케이션에 포함된 레이어를 구축하도록 AWS SAM에게 지시할 수 있습니다. 레이어 구축에 대한 자세한 내용은 에서 Lambda 계층 빌드 AWS SAM 섹션을 참조하세요.

중요 참고 사항: CloudFormation에서 UpdateReplacePolicy 리소스 속성이 릴리스된 이후 AWS::Lambda::LayerVersion(권장)은 AWS::Serverless::LayerVersion과 동일한 이점을 제공합니다.

서버리스 LayerVersion이 변환되면 SAM은 리소스의 논리적 ID도 변환하여 리소스가 업데이트될 때 CloudFormation이 이전 LayerVersion을 자동으로 삭제하지 않도록 합니다.

> **Note**
>
> AWS CloudFormation에 배포하면 AWS SAM는 귀하의 AWS SAM 리소스를 CloudFormation 리소스로 변환됩니다. 자세한 내용은 AWS SAM용으로 생성된 CloudFormation 리소스을 참조하십시오

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
Type: AWS::Serverless::LayerVersion
Properties:
  CompatibleArchitectures: List
```

```
  CompatibleRuntimes: List
  ContentUri: String | LayerContent
  Description: String
  LayerName: String
  LicenseInfo: String
  PublishLambdaVersion: Boolean
  RetentionPolicy: String
```

속성

`CompatibleArchitectures`

레이어 버전에 지원되는 명령어 세트 아키텍처를 지정합니다.

이 속성에 대한 자세한 내용은 AWS Lambda 개발자 가이드의 Lambda 명령 세트 아키텍처를 참조하세요.

유효한 값: `x86_64`, `arm64`

유형: 목록

필수 항목 여부: 아니요

기본값: `x86_64`

CloudFormation호환성: 이 속성은 `CompatibleArchitectures` 리소스의 `AWS::Lambda::LayerVersion` 속성으로 직접 전달됩니다.

`CompatibleRuntimes`

이 LayerVersion과 호환되는 런타임 목록.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `CompatibleRuntimes` 리소스의 `AWS::Lambda::LayerVersion` 속성으로 직접 전달됩니다.

`ContentUri`

Amazon S3 Uri, 로컬 폴더 경로 또는 레이어 코드의 레이어 콘텐츠 객체.

Amazon S3 Uri 또는 LayerContent 객체가 제공되는 경우 참조되는 Amazon S3 객체는 Lambda 레이어의 콘텐츠를 포함하는 유효한 ZIP 아카이브여야 합니다.

로컬 폴더 경로가 제공되는 경우 콘텐츠가 제대로 변환되려면 템플릿은 sam build 및 그 다음으로 sam deploy 또는 sam package 중 하나가 포함된 워크플로를 거쳐야 합니다. 기본적으로 상대 경로는 AWS SAM 템플릿 위치를 기준으로 확인됩니다.

유형: 문자열 | 레이어 콘텐츠

필수 여부: 예

CloudFormation호환성: 이 속성은 `AWS::Lambda::LayerVersion` 리소스의 `Content` 속성과 유사합니다. 중첩된 Amazon S3 속성은 다르게 지정됩니다.

`Description`

이 레이어에 대한 설명.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `Description` 리소스의 `AWS::Lambda::LayerVersion` 속성으로 직접 전달됩니다.

`LayerName`

레이어의 이름 또는 Amazon 리소스 이름(ARN)입니다.

유형: 문자열

필수 항목 여부: 아니요

기본값: 리소스 논리적 ID

CloudFormation호환성: 이 속성은 `AWS::Lambda::LayerVersion` 리소스의 `LayerName` 속성과 유사합니다. 이름을 지정하지 않으면 리소스의 논리적 ID를 이름으로 사용합니다.

`LicenseInfo`

이 Layer Version의 라이선스에 대한 정보.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 LicenseInfo 리소스의 AWS::Lambda::LayerVersion 속성으로 직접 전달됩니다.

PublishLambdaVersion

참조된 LayerVersion 리소스에 변경 사항이 있을 때마다 새 Lambda 버전을 생성하는 옵트인 속성입니다. 연결된 Lambda 함수에서 AutoPublishAlias 및 AutoPublishAliasAllProperties이(가) 활성화되면, LayerVersion 리소스에 대한 변경 사항이 있을 때마다 새 Lambda 버전이 생성됩니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

RetentionPolicy

이 속성은 리소스를 삭제할 때 LayerVersion의 이전 버전을 유지할지, 아니면 삭제할지 여부를 지정합니다. 리소스를 업데이트하거나 교체할 때 LayerVersion의 이전 버전을 유지해야 하는 경우 UpdateReplacePolicy 속성을 활성화해야 합니다. 이를 수행하는 방법에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 UpdateReplacePolicy 속성을 참조하세요.

유효한 값: Retain 또는 Delete

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

추가 참고 사항: 귀하가 Retain를 지정하면 AWS SAM은 DeletionPolicy: Retain의 AWS SAM에서 지원되는 리소스 속성를 변환된 AWS::Lambda::LayerVersion 리소스에 추가합니다.

반환 값

Ref

Ref 내장 함수에 이 리소스의 논리적 ID를 입력하면 기저의 Lambda LayerVertion의 리소스 ARN을 반환합니다.

Ref 함수의 사용에 대한 자세한 내용은 AWS CloudFormation 사용자 가이드의 Ref 섹션을 참조하세요.

예시

LayerVersionExample

레이어 버전 예제

YAML

```
Properties:
  LayerName: MyLayer
  Description: Layer description
  ContentUri: 's3://sam-s3-demo-bucket/my-layer.zip'
  CompatibleRuntimes:
    - nodejs10.x
    - nodejs12.x
  LicenseInfo: 'Available under the MIT-0 license.'
  RetentionPolicy: Retain
```

LayerContent

Lambda 계층의 내용을 포함하는 ZIP 아카이브입니다.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

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

속성

`Bucket`

계층 아카이브의 Amazon S3 버킷입니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 [S3Bucket](#) `AWS::Lambda::LayerVersion` 데이터 유형의 `Content` 속성에 직접 전달됩니다.

`Key`

계층 아카이브의 Amazon S3 키입니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 [S3Key](#) `AWS::Lambda::LayerVersion` 데이터 유형의 `Content` 속성에 직접 전달됩니다.

`Version`

버전이 지정된 객체의 경우 사용할 레이어 아카이브 객체의 버전입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [S3ObjectVersion](#) `AWS::Lambda::LayerVersion` 데이터 유형의 `Content` 속성에 직접 전달됩니다.

예시

LayerContent

계층 콘텐츠 예제

YAML

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

AWS::Serverless::SimpleTable

단일 속성 프라이머리 키를 사용하여 DynamoDB 테이블을 생성합니다. 프라이머리 키를 통해서만 데이터에 액세스해야 하는 경우에 유용합니다.

고급 기능을 사용하려면 CloudFormation에서 AWS::DynamoDB::Table 리소스를 사용하세요. 이러한 리소스는 AWS SAM에서 사용할 수 있습니다. 포괄적이며 key schema 및 resource policy 사용자 지정을 포함하여 추가적인 사용자 지정 기능을 제공합니다.

> **Note**
>
> AWS CloudFormation에 배포하면 AWS SAM는 귀하의 AWS SAM 리소스를 CloudFormation 리소스로 변환됩니다. 자세한 내용은 AWS SAM용으로 생성된 CloudFormation 리소스을 참조하십시오

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
Type: AWS::Serverless::SimpleTable
Properties:
  PointInTimeRecoverySpecification: PointInTimeRecoverySpecification
  PrimaryKey: PrimaryKeyObject
  ProvisionedThroughput: ProvisionedThroughputObject
  SSESpecification: SSESpecification
  TableName: String
  Tags: Map
```

속성

`PointInTimeRecoverySpecification`

특정 시점으로 복구를 활성화하는 데 사용되는 설정입니다.

유형: PointInTimeRecoverySpecification

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `PointInTimeRecoverySpecification` 리소스의 `AWS::DynamoDB::Table` 속성으로 직접 전달됩니다.

`PrimaryKey`

테이블의 프라이머리 키로 사용할 속성 이름 및 유형입니다. 지정하지 않을 경우 기본 키는 값이 `String`인 `id`가 됩니다.

> **Note**
> 이 리소스가 생성된 후에는 이 속성의 값을 수정할 수 없습니다.

유형: PrimaryKeyObject

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`ProvisionedThroughput`

처리량 프로비저닝 정보를 읽고 씁니다.

`ProvisionedThroughput`이 지정되지 않은 경우 `BillingMode`가 `PAY_PER_REQUEST`로 지정됩니다.

유형: ProvisionedThroughputObject

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `ProvisionedThroughput` 리소스의 `AWS::DynamoDB::Table` 속성으로 직접 전달됩니다.

`SSESpecification`

서버 측 암호화를 활성화하도록 설정합니다.

유형: SSESpecification

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [SSESpecification] 리소스의 AWS::DynamoDB::Table 속성으로 직접 전달됩니다.

`TableName`

DynamoDB 테이블의 이름입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [TableName] 리소스의 AWS::DynamoDB::Table 속성으로 직접 전달됩니다.

`Tags`

SimpleTable에 추가할 태그를 지정하는 문자열을 문자열로 변환하는 맵입니다. 태그의 유효한 키와 값에 대한 자세한 내용은 사용자 가이드의 AWS CloudFormation리소스 태그를 참조하세요.

유형: 맵

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS::DynamoDB::Table 리소스의 [Tags] 속성과 유사합니다. SAM의 태그 속성은 키:값의 쌍으로 구성되고, CloudFormation에서는 태그 객체 목록으로 구성됩니다.

반환 값

Ref

Ref 내장 함수에 이 리소스의 논리적 ID를 입력하면 이 함수는 기본 DynamoDB 테이블의 리소스 이름을 반환합니다.

`Ref` 함수의 사용에 대한 자세한 내용은 AWS CloudFormation 사용자 가이드의 `Ref` 섹션을 참조하세요.

예시

SimpleTableExample

SimpleTable의 예제

YAML

```
Properties:
  TableName: my-table
  PrimaryKey:
    Name: MyPrimaryKey
    Type: String
  ProvisionedThroughput:
    ReadCapacityUnits: 5
    WriteCapacityUnits: 5
  Tags:
    Department: Engineering
    AppType: Serverless
```

PrimaryKeyObject

프라이머리 키의 속성을 설명하는 객체.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  Name: String
  Type: String
```

속성

Name

프라이머리 키의 속성 이름.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `AttributeName` `AWS::DynamoDB::Table` 데이터 유형의 `AttributeDefinition` 속성에 직접 전달됩니다.

추가 참고 사항: 이 속성은 `AWS::DynamoDB::Table KeySchema` 데이터 유형의 AttributeName 속성에도 전달됩니다.

`Type`

프라이머리 키의 데이터 유형.

유효한 값: `String`, `Number`, `Binary`

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `AttributeType` `AWS::DynamoDB::Table` 데이터 유형의 `AttributeDefinition` 속성에 직접 전달됩니다.

예시

primaryKeys

프라이머리 키 예시.

YAML

```
Properties:
  PrimaryKey:
    Name: MyPrimaryKey
    Type: String
```

ProvisionedThroughputObject

프로비저닝된 처리량의 속성을 설명하는 객체입니다.

구문

AWS Serverless Application Model (AWS SAM) 템플릿에서이 개체를 선언하려면 다음 구문을 사용합니다.

YAML

```
  ReadCapacityUnits: Integer
  WriteCapacityUnits: Integer
```

속성

`ReadCapacityUnits`

DynamoDB가 `ThrottlingException`을 반환하기 전에 초당 사용되는 Strongly Consistent Read의 최대 수입니다.

유형: 정수

필수 항목 여부: 예

CloudFormation 호환성:이 속성은 `AWS::DynamoDB::Table` `ProvisionedThroughput` 데이터 유형의 [`ReadCapacityUnits`] 속성으로 직접 전달됩니다.

`WriteCapacityUnits`

DynamoDB가 `ThrottlingException`을 반환하기 전에 초당 사용되는 최대 쓰기 수입니다.

유형: 정수

필수 항목 여부: 예

CloudFormation 호환성:이 속성은 `AWS::DynamoDB::Table` `ProvisionedThroughput` 데이터 유형의 [`WriteCapacityUnits`] 속성으로 직접 전달됩니다.

예제

ProvisionedThroughput

프로비저닝된 처리량 예시입니다.

YAML

```
Properties:
   ProvisionedThroughput:
     ReadCapacityUnits: 5
     WriteCapacityUnits: 5
```

AWS::Serverless::StateMachine

AWS Step Functions 함수 및 기타 AWS Lambda 리소스를 오케스트레이션하여 복잡하고 강력한 워크플로를 구성하는 데 사용할 수 있는 AWS 상태 기기를 만듭니다.

계단 함수에 대한 자세한 내용은 AWS Step Functions 개발자 가이드를 참조하세요.

> **Note**
>
> AWS CloudFormation에 배포하면 AWS SAM는 귀하의 AWS SAM 리소스를 CloudFormation 리소스로 변환됩니다. 자세한 내용은 AWS SAM용으로 생성된 CloudFormation 리소스 섹션을 참조하십시오.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
Type: AWS::Serverless::StateMachine
Properties:
  AutoPublishAlias: String
  UseAliasAsEventTarget: Boolean
  Definition: Map
  DefinitionSubstitutions: Map
  DefinitionUri: String | S3Location
  DeploymentPreference: DeploymentPreference
  Events: EventSource
  Logging: LoggingConfiguration
  Name: String
  PermissionsBoundary: String
  Policies: String | List | Map
  PropagateTags: Boolean
  RolePath: String
  Role: String
  Tags: Map
  Tracing: TracingConfiguration
  Type: String
```

속성

`AutoPublishAlias`

상태 기기 별칭의 이름입니다. 계단 함수 상태 기기 별칭을 사용하는 방법에 대해 자세히 알아보려면 개발자 가이드의 AWS Step Functions 버전 및 별칭을 사용한 지속적 배포 관리를 참조하세요.

별칭에 대한 배포 기본 설정을 구성하는데 DeploymentPreference을 사용합니다. DeploymentPreference을 지정하지 않으면 AWS SAM가 트래픽을 최신 상태 기기 버전으로 한꺼번에 전환되도록 구성합니다.

기본 사항으로 AWS SAM이 버전의 DeletionPolicy 및 UpdateReplacePolicy을 Retain로 설정합니다. 이전 버전은 자동으로 삭제되지 않습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS::StepFunctions::StateMachineAlias 리소스의 Name 속성으로 직접 전달됩니다.

UseAliasAsEventTarget

AutoPublishAlias 속성을 사용하여 생성된 별칭을 이벤트로 정의된 이벤트 소스의 대상에 전달할지 여부를 지정합니다.

별칭을 이벤트의 대상으로 사용하려면 True를 지정합니다.

유형: 부울

필수 항목 여부: 아니요

기본값: False

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

Definition

상태 기기 정의는 객체이며, 이 객체의 형식은 AWS SAM 템플릿 파일(예: JSON 또는 YAML)의 형식과 일치합니다. 상태 기기 정의는 Amazon States Language를 따릅니다.

인라인 스테이트 기기 정의의 예는 예시을 참조하세요.

Definition 또는 DefinitionUri를 제공해야 합니다.

유형: 맵

필수 항목 여부: 조건부

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`DefinitionSubstitutions`

상태 시스템 정의의 자리 표시자 변수에 대한 매핑을 지정하는 문자열-문자열 맵입니다. 이를 통해 런타임에 얻은 값(예: 내장 함수)을 상태 기기 정의에 삽입할 수 있습니다.

유형: 맵

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [`DefinitionSubstitutions`] 리소스의 `AWS::StepFunctions::StateMachine` 속성과 유사합니다. 인라인 스테이트 기기 정의에 내장 함수가 지정된 경우 AWS SAM은 이 속성에 항목을 추가하여 스테이트 기기 정의에 삽입합니다.

`DefinitionUri`

Amazon Simple Storage Service(S3) URI 또는 [Amazon States Language] 내 기재된 상태 기기 정의의 로컬 파일 경로입니다.

로컬 파일 경로를 제공하는 경우 템플릿은 `sam deploy` 또는 `sam package` 명령이 포함된 워크플로를 거쳐야 정의를 올바르게 변환할 수 있습니다. 이렇게 하려면 AWS SAM CLI의 버전 0.52.0 이상을 사용해야 합니다.

`Definition` 또는 `DefinitionUri`를 제공해야 합니다.

유형: 문자열 | [S3Location]

필수 항목 여부: 조건부

CloudFormation호환성: 이 속성은 `AWS::StepFunctions::StateMachine` 리소스의 [`DefinitionS3Location`] 속성으로 직접 전달됩니다.

`DeploymentPreference`

점진적 상태 기기 배포를 활성화하고 구성하는 설정입니다. 계단 함수의 점진적 배포에 대해 자세히 알아보려면 [개발자 가이드]의 AWS Step Functions 버전 및 별칭을 사용한 지속적 배포 관리를 참조하세요.

이 속성을 구성하기 전에 `AutoPublishAlias`을 지정하십시오. `DeploymentPreference` 설정이 `AutoPublishAlias`와 함께 지정된 별칭에 적용됩니다.

DeploymentPreference을 지정하면 AWS SAM가 StateMachineVersionArn 하위 속성 값을 자동으로 생성합니다.

유형: DeploymentPreference

필수 항목 여부: 아니요

CloudFormation호환성: AWS SAM은 StateMachineVersionArn속성 값을 생성하여 DeploymentPreference 리소스의 속성에 연결하고 DeploymentPreference 리소스의 DeploymentPreference 속성에 AWS::StepFunctions::StateMachineAlias전달합니다.

Events

이 상태 기기를 촉발하는 이벤트를 지정합니다. 이벤트는 유형 및 각 유형에 따라 달라지는 속성 집합으로 구성됩니다.

유형: EventSource

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

Logging

어떤 실행 기록 이벤트가 로그되는지, 또한 어디에서 로그되는지 정의합니다.

유형: LoggingConfiguration

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS::StepFunctions::StateMachine 리소스의 LoggingConfiguration 속성으로 직접 전달됩니다.

Name

상태 기기의 이름입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS::StepFunctions::StateMachine 리소스의 StateMachineName 속성으로 직접 전달됩니다.

`PermissionsBoundary`

이 상태 기기의 실행 역할에 사용할 권한 경계의 ARN. 이 속성은 역할이 자동으로 생성된 경우에만 작동합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::IAM::Role` 리소스의 [`PermissionsBoundary`] 속성으로 직접 전달됩니다.

`Policies`

이 상태 기기의 권한 정책입니다. 정책은 스테이트 기기의 기본 AWS Identity and Access Management(IAM) 실행 역할에 추가됩니다.

이 속성은 단일 값 또는 값 목록을 허용합니다. 허용되는 값은 다음과 같습니다.

- [AWS SAM 정책 템플릿].
- ARN 관리형 정책 [AWS] 또는 [고객 관리형 정책]의.
- 다음 AWS 목록에 있는 [관리형 정책]의 이름입니다.
- [맵 형식]의 YAML 인라인 IAM 정책입니다.

> **Note**
>
> `Role` 속성을 설정하면 이 속성은 무시됩니다.

유형: 문자열 | 목록 | 맵

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`PropagateTags`

`Tags`속성의 태그를 [AWS::Serverless::StateMachine] 생성된 리소스로 전달할지 여부를 지정합니다. 귀하의 생성된 리소스에 태그를 전파하도록 `True`을 지정합니다.

유형: 부울

필수 항목 여부: 아니요

기본값: `False`

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Role`

이 상태 기기의 실행 역할로 사용할 IAM 역할의 ARN.

유형: 문자열

필수 항목 여부: 조건부

CloudFormation호환성: 이 속성은 `AWS::StepFunctions::StateMachine` 리소스의 `RoleArn` 속성으로 직접 전달됩니다.

`RolePath`

상태 기기의 IAM 실행 역할 경로입니다.

역할이 자동으로 생성될 때 이 속성을 사용하십시오. `Role` 속성에 역할이 지정된 경우에는 사용하지 마십시오.

유형: 문자열

필수 항목 여부: 조건부

CloudFormation호환성: 이 속성은 `AWS::IAM::Role` 리소스의 `Path` 속성으로 직접 전달됩니다.

`Tags`

상태 시스템에 추가된 태그와 해당 실행 역할을 지정하는 문자열-문자열 맵입니다. 태그의 유효한 키와 값에 대한 자세한 내용은 리소스의 AWS::StepFunctions::StateMachine태그 속성을 참조하세요.

유형: 맵

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `Tags` 리소스의 `AWS::StepFunctions::StateMachine` 속성과 유사합니다. AWS SAM가 이 리소스와 해당 리소스에 대해 생성된 기본 역할에 `stateMachine:createdBy:SAM` 태그를 자동으로 추가합니다.

`Tracing`

상태 기기에 대하여 AWS X-Ray을 활성화할 것인지를 선택합니다. Step Functions와 함께 X-Ray를 사용하는 방법에 대한 자세한 내용은 AWS X-Ray개발자 가이드의 AWS Step Functions계단 함수를 참조하세요.

유형: TracingConfiguration

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::StepFunctions::StateMachine` 리소스의 `TracingConfiguration` 속성으로 직접 전달됩니다.

`Type`

상태 기기의 이름입니다.

유효한 값: `STANDARD` 또는 `EXPRESS`

유형: 문자열

필수 항목 여부: 아니요

기본값: `STANDARD`

CloudFormation호환성: 이 속성은 `StateMachineType` 리소스의 `AWS::StepFunctions::StateMachine` 속성으로 직접 전달됩니다.

반환 값

Ref

이 리소스의 논리적 ID를 Ref 내장 함수에 전달하면 Ref가 기저의 `AWS::StepFunctions::StateMachine` 리소스의 Amazon 리소스 이름(ARN)을 반환합니다.

`Ref` 함수에 대한 자세한 내용은 `Ref` 사용자 가이드의 AWS CloudFormation를 참조하십시오.

Fn::GetAtt

`Fn::GetAtt`은 이 유형의 지정된 속성에 대한 값을 반환합니다. 다음은 사용 가능한 속성과 반환되는 샘플 값.

`Fn::GetAtt`의 사용에 대한 자세한 내용은 AWS CloudFormation 사용자 가이드의 `Fn::GetAtt` 섹션을 참조하세요.

`Name`

`HelloWorld-StateMachine`과 같은 상태 기기의 이름을 반환합니다.

예시

스테이트 기기 정의 파일

다음은 Lambda 함수가 상태 시스템을 호출할 수 있도록 허용하는 인라인 상태 시스템 정의의 예입니다. 이 예제에서는 `Role` 속성이 호출을 허용하는 적절한 정책으로 구성되어 있다고 가정합니다. `my_state_machine.asl.json`파일은 Amazon States Language 로 작성해야 합니다.

이 예제에서는 `DefinitionSubstitution` 항목들은 상태 기기가 AWS SAM 템플릿 파일에 선언된 리소스를 포함할 수 있도록 합니다.

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
```

인라인 상태 기기 정의

다음은 인라인 상태 기기 정의의 예입니다.

이 예제에서는 AWS SAM 템플릿 파일이 YAML로 작성되므로 상태 기기 정의도 YAML로 작성됩니다. 인라인 상태 기기 정의를 JSON으로 선언하려면 AWS SAM 템플릿 파일을 JSON으로 작성하십시오.

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

스테이트 머신을 트리거하는 이벤트의 소스를 설명하는 객체. 각 이벤트는 유형과 해당 유형에 따라 달라지는 속성 집합으로 구성됩니다. 각 이벤트 소스의 속성에 대한 자세한 내용은 해당 유형에 해당하는 하위 주제를 참조하세요.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
Properties: Schedule | ScheduleV2 | CloudWatchEvent | EventBridgeRule | Api
Type: String
```

속성

`Properties`

이 이벤트 매핑의 속성을 설명하는 객체. 속성 집합은 정의된 `Type`과 일치해야 합니다.

유형: 일정 | ScheduleV2 |CloudWatchEvent|EventBridgeRule규칙 | Api

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

Type

이벤트 유형.

유효한 값 Api Schedule, ScheduleV2, CloudWatchEvent, EventBridgeRule

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

API

다음은 API 유형의 이벤트 예입니다.

YAML

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

Api

Api 이벤트 소스 유형을 설명하는 객체. AWS::Serverless::Api 리소스가 정의된 경우 경로 및 메서드 값은 API의 OpenAPI 정의에 있는 작업과 일치해야 합니다.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  Auth: ApiStateMachineAuth
```

```
  Method: String
  Path: String
  RestApiId: String
  UnescapeMappingTemplate: Boolean
```

속성

`Auth`

이 API, 경로 및 메서드의 권한 부여 구성.

이 속성을 사용하면 `DefaultAuthorizer`가 지정되지 않은 경우 개별 경로에 대한 API의 `DefaultAuthorizer` 설정을 재정의하거나 기본 `ApiKeyRequired` 설정을 재정의할 수 있습니다.

유형: ApiStateMachineAuth

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Method`

이 함수가 호출되는 HTTP 메서드.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Path`

이 함수가 호출되는 Uri 경로입니다. /로 값을 시작합니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`RestApiId`

`RestApi` 리소스의 식별자로, 지정된 경로와 메서드를 사용하는 작업이 포함되어야 합니다. 일반적으로 이 템플릿에 정의된 AWS::Serverless::Api 리소스를 참조하도록 설정됩니다.

이 속성을 정의하지 않는 경우 AWS SAM은 생성된 AWS::Serverless::Api 문서를 사용하여 기본 `OpenApi` 리소스를 만듭니다. 해당 리소스에는 `RestApiId`를 지정하지 않은 동일한 템플릿의 `Api` 이벤트에 의해 정의된 모든 경로와 메서드가 통합되어 있습니다.

이 속성은 다른 템플릿에 정의된 AWS::Serverless::Api 리소스를 참조할 수 없습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`UnescapeMappingTemplate`

상태 시스템에 전달되는 입력에서 `\'`를 `'`로 대체하여 이스케이프 해제합니다. 입력에 작은따옴표가 포함된 경우 사용합니다.

> **Note**
> `False`로 설정되었고 입력에 작은따옴표가 포함된 경우 오류가 발생합니다.

유형: 부울

필수 항목 여부: 아니요

기본값: False

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

ApiEvent

다음은 `Api` 유형의 이벤트 예입니다.

YAML

```
Events:
  ApiEvent:
    Type: Api
    Properties:
      Path: /path
      Method: get
```

ApiStateMachineAuth

특정 API, 경로 및 메서드에 대해 이벤트 수준에서 권한 부여를 구성합니다.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  ApiKeyRequired: Boolean
  AuthorizationScopes: List
  Authorizer: String
  ResourcePolicy: ResourcePolicyStatement
```

속성

`ApiKeyRequired`

이 API, 경로 및 메서드에 대한 API 키가 필요합니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`AuthorizationScopes`

이 API, 경로 및 메서드에 적용할 수 있는 권한 부여 범위.

지정한 범위는 해당 속성을 지정한 경우 `DefaultAuthorizer` 속성에서 적용한 모든 범위를 재정의합니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Authorizer`

특정 상태 시스템용 `Authorizer`.

API에 대한 글로벌 권한 부여자를 지정하고 이 상태 시스템을 공개하려면 `Authorizer`를 `NONE`으로 설정하여 전역 권한 부여자를 재정의합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`ResourcePolicy`

이 API 및 경로에 대한 리소스 정책을 구성합니다.

유형: [리소스 정책 설명]

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

StateMachine-Auth

다음 예에서는 상태 시스템 수준에서 권한 부여를 지정합니다.

YAML

```
Auth:
  ApiKeyRequired: true
```

```
  Authorizer: NONE
```

ResourcePolicyStatement

API의 모든 메서드와 경로에 대한 리소스 정책을 구성합니다. 리소스 정책에 대한 자세한 내용은 API Gateway 개발자 안내서의 API Gateway 리소스 정책을 사용하는 액세스 제어를 참조하세요.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

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

속성

`AwsAccountBlacklist`

차단할 AWS 계정.

유형: 문자열 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`AwsAccountWhitelist`

허용할 AWS 계정. 이 속성의 예제 사용 예제를 알아보려면 이 페이지 하단의 예제을 참조하세요

유형: 문자열 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`CustomStatements`

이 API에 적용할 사용자 지정 리소스 정책 설명 목록. 이 속성의 예제 사용 예제를 알아보려면 이 페이지 하단의 예제을 참조하세요

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`IntrinsicVpcBlacklist`

차단할 Virtual Private Cloud(VPC))목록입니다. 여기서 각 VPC는 동적 참조 또는 `Ref` 내장 함수와 같은 참조로 지정됩니다. 이 속성의 예제 사용 예제를 알아보려면 이 페이지 하단의 예제을 참조하세요

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`IntrinsicVpcWhitelist`

허용할 VPC 목록. 여기서 각 VPC는 동적 참조 또는 `Ref` 내장 함수와 같은 참조로 지정됩니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`IntrinsicVpceBlacklist`

차단할 VPC 엔드포인트 목록. 여기서 각 VPC 엔드포인트는 동적 참조 또는 Ref 내장 함수와 같은 참조로 지정됩니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`IntrinsicVpceWhitelist`

허용할 VPC 엔드포인트 목록. 여기서 각 VPC 엔드포인트는 동적 참조 또는 Ref 내장 함수와 같은 참조로 지정됩니다. 이 속성의 예제 사용 예제를 알아보려면 이 페이지 하단의 예제을 참조하세요

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`IpRangeBlacklist`

차단할 IP 주소 또는 주소 범위. 이 속성의 예제 사용 예제를 알아보려면 이 페이지 하단의 예제을 참조하세요

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`IpRangeWhitelist`

허용할 IP 주소 또는 주소 범위.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`SourceVpcBlacklist`

차단할 소스 VPC 또는 VPC 엔드포인트. 소스 VPC 이름은 `"vpc-"`로 시작하고 소스 VPC 엔드포인트 이름은 `"vpce-"`로 시작해야 합니다. 이 속성의 예제 사용 예제를 알아보려면 이 페이지 하단의 예제을 참조하세요

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`SourceVpcWhitelist`

허용할 소스 VPC 또는 VPC 엔드포인트. 소스 VPC 이름은 `"vpc-"`로 시작하고 소스 VPC 엔드포인트 이름은 `"vpce-"`로 시작해야 합니다.

유형: 목록

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

리소스 정책 예시

다음 예시에서는 두 개의 IP 주소와 한 개의 소스 VPC를 차단하고 한 AWS계정을 허용합니다.

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

`CloudWatchEvent` 이벤트 소스 유형을 설명하는 객체.

AWS Serverless Application Model(AWS SAM)은 이 이벤트 유형이 설정되면 AWS::Events::Rule리소스를 생성합니다.

중요 참고: EventBridgeRule는 `CloudWatchEvent` 대신 사용할 우선적인 이벤트 소스 유형입니다. `EventBridgeRule` 및 `CloudWatchEvent`는 동일한 기본 서비스, API, CloudFormation 리소스를 사용합니다. 하지만 AWS SAM은 오직 `EventBridgeRule`에 대해서만 새 기능에 대한 지원을 추가할 예정입니다.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  EventBusName: String
  Input: String
```

```
  InputPath: String
  Pattern: EventPattern
```

속성

EventBusName

이 규칙과 연결할 이벤트 버스입니다. 이 속성을 생략하면 AWS SAM은 기본 사항으로서 이벤트 버스를 사용합니다.

유형: 문자열

필수 항목 여부: 아니요

기본값: 기본 이벤트 버스

CloudFormation호환성: 이 속성은 `AWS::Events::Rule` 리소스의 [EventBusName](#) 속성으로 직접 전달됩니다.

Input

대상으로 전달되는 유효한 JSON 텍스트입니다. 이 속성을 사용하면 이벤트 텍스트 자체의 어떤 것도 대상으로 전달되지 않습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Events::Rule Target` 리소스의 [Input](#) 속성으로 직접 전달됩니다.

InputPath

일치된 이벤트 전체를 전달하지 않으려는 경우 `InputPath` 속성을 사용하여 이벤트의 어떤 부분이 전달되어야 하는지 설명하세요.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Events::Rule Target` 리소스의 [InputPath](#) 속성으로 직접 전달됩니다.

`Pattern`

어떤 이벤트가 지정된 대상으로 라우팅되는지를 설명합니다. 자세한 내용은 Amazon EventBridge 사용자 가이드의 EventBridge의 이벤트 및 이벤트 패턴을 참조하세요.

유형: 이벤트패턴

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `EventPattern` 리소스의 `AWS::Events::Rule` 속성으로 직접 전달됩니다.

예시

CloudWatchEvent

다음은 `CloudWatchEvent` 이벤트 소스 유형의 한 예제입니다.

YAML

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

EventBridgeRule

상태 머신을 Amazon EventBridge 규칙의 대상으로 설정하는 `EventBridgeRule`이벤트 소스 유형을 설명하는 객체. 자세한 내용은 Amazon EventBridge User Guide(Amazon EventBridge 사용 설명서)의 What Is Amazon EventBridge?(Amazon EventBridge란?)를 참조하십시오.

이 이벤트 유형이 설정되면 AWS SAM이 AWS::Events::Rule리소스를 생성합니다.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
DeadLetterConfig: DeadLetterConfig
EventBusName: String
Input: String
InputPath: String
InputTransformer: InputTransformer
Pattern: EventPattern
RetryPolicy: RetryPolicy
RuleName: String
State: String
Target: Target
```

속성

`DeadLetterConfig`

대상 호출이 실패한 경우 EventBridge가 이벤트를 전송하는 Amazon Simple Queue Service(Amazon SQS) 대기열을 구성합니다. 예를 들어 존재하지 않는 Lambda 함수로 이벤트를 전송하거나 EventBridge에 Lambda 함수를 호출할 권한이 충분하지 않은 경우 호출이 실패할 수 있습니다. 자세한 내용은 Amazon EventBridge 사용자 가이드의 이벤트 재시도 정책 및 DLQ(Dead Letter Queue) 사용을 참조하세요.

유형: DeadLetterConfig

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `DeadLetterConfig` 데이터 유형의 `AWS::Events::Rule Target` 속성과 유사합니다. DLQ를 자동으로 AWS SAM 생성하고자 하는 경우에 대비하여 이 속성의 AWS SAM 버전에는 추가 하위 속성이 포함되어 있습니다.

`EventBusName`

이 규칙과 연결할 이벤트 버스입니다. 이 속성을 생략하면 AWS SAM은 기본 사항으로서 이벤트 버스를 사용합니다.

유형: 문자열

필수 항목 여부: 아니요

기본값: 기본 이벤트 버스

CloudFormation호환성: 이 속성은 `AWS::Events::Rule` 리소스의 `EventBusName` 속성으로 직접 전달됩니다.

`Input`

대상으로 전달되는 유효한 JSON 텍스트입니다. 이 속성을 사용하면 이벤트 텍스트 자체의 어떤 것도 대상으로 전달되지 않습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Events::Rule Target` 리소스의 `Input` 속성으로 직접 전달됩니다.

`InputPath`

일치된 이벤트 전체를 전달하지 않으려는 경우 `InputPath` 속성을 사용하여 이벤트의 어떤 부분이 전달되어야 하는지 설명하세요.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Events::Rule Target` 리소스의 `InputPath` 속성으로 직접 전달됩니다.

`InputTransformer`

특정 이벤트 데이터를 기반으로 대상에 사용자 지정 입력을 제공할 수 있게 하는 설정입니다. 이벤트에서 하나 이상의 키-값 페어를 추출한 후 이 데이터를 사용하여 대상에 사용자 지정 입력을 전송할 수 있습니다. 자세한 내용은 Amazon EventBridge 사용자 가이드의 Amazon EventBridge 입력 변환을 참조하세요.

유형: InputTransformer

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `InputTransformer` `AWS::Events::Rule` 데이터 유형의 `Target` 속성에 직접 전달됩니다.

`Pattern`

어떤 이벤트가 지정된 대상으로 라우팅되는지를 설명합니다. 자세한 내용은 Amazon EventBridge 사용자 가이드의 EventBridge의 이벤트 및 이벤트 패턴을 참조하세요.

유형: [이벤트패턴]

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `EventPattern` 리소스의 `AWS::Events::Rule` 속성으로 직접 전달됩니다.

`RetryPolicy`

재시도 정책 설정에 대한 정보가 포함된 `RetryPolicy` 객체입니다. 자세한 내용은 Amazon EventBridge 사용자 가이드의 이벤트 재시도 정책 및 DLQ(Dead Letter Queue) 사용을 참조하세요.

유형: RetryPolicy

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `RetryPolicy` `AWS::Events::Rule` 데이터 유형의 `Target` 속성에 직접 전달됩니다.

`RuleName`

규칙의 이름입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Events::Rule` 리소스의 `Name` 속성으로 직접 전달됩니다.

`State`

규칙의 상태입니다.

유효한 값: `[ DISABLED | ENABLED ]`

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Events::Rule` 리소스의 `State` 속성으로 직접 전달됩니다.

`Target`

규칙이 트리거될 경우 EventBridge가 호출하는 AWS 리소스입니다. 이 속성을 사용하여 대상의 논리적 ID를 지정할 수 있습니다. 이 속성을 지정하지 않으면 AWS SAM이 대상의 논리적 ID를 생성합니다.

유형: [Target]

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Events::Rule` 리소스의 [`Targets`] 속성과 유사합니다. 이 속성의 AWS SAM 버전에서는 단일 대상의 논리적 ID만 지정할 수 있습니다.

예시

EventBridgeRule

다음은 `EventBridgeRule` 이벤트 소스 유형의 예입니다.

YAML

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

DeadLetterConfig

대상 호출 실패 후 EventBridge가 이벤트를 전송하는 Amazon Simple Queue Service(Amazon SQS) 대기열을 지정하는 데 사용되는 객체입니다. 예를 들어 존재하지 않는 상태 시스템으로 이벤트를 전송하거나 상태 시스템을 호출할 권한이 충분하지 않은 경우 간접 호출이 실패할 수 있습니다. 자세한 내용은 Amazon EventBridge 사용자 가이드의 [이벤트 재시도 정책 및 DLQ(Dead Letter Queue) 사용]을 참조하세요.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
Arn: String
QueueLogicalId: String
Type: String
```

속성

Arn

DLQ(Dead Letter Queue)의 대상으로 지정된 Amazon SQS 대기열의 Amazon 리소스 이름(ARN)입니다.

> **Note**
>
> Type 속성 또는 Arn 속성 중 하나만 지정해야 하며, 둘 다 지정할 수는 없습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 Arn AWS::Events::Rule 데이터 유형의 DeadLetterConfig 속성에 직접 전달됩니다.

QueueLogicalId

Type가 지정된 경우 AWS SAM이 생성하는 DLQ의 사용자 지정 이름입니다.

> **Note**
>
> Type 속성이 설정되지 않은 경우 이 속성은 무시됩니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

Type

대기열 유형. 이 속성이 설정되면 AWS SAM은 DLQ를 자동으로 만들고 허가를 부여하기 위해 필요한 리소스 기반 정책을 규칙 리소스에 첨부하여 해당 큐에 이벤트를 전송합니다.

> **Note**
>
> `Type` 속성 또는 `Arn` 속성 중 하나만 지정해야 하며, 둘 다 지정할 수는 없습니다.

유효한 값: SQS

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

DeadLetterConfig

DeadLetterConfig

YAML

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

Target

규칙이 트리거될 경우 EventBridge가 호출하는 AWS 리소스를 구성합니다.

구문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
Id: String
```

속성

Id

대상의 논리적 ID.

Id의 값은 영숫자 문자, 마침표(.), 하이픈(-), 밑줄(_)을 포함할 수 있습니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 Id AWS::Events::Rule 데이터 유형의 Target 속성에 직접 전달됩니다.

예시

대상

YAML

```
EBRule:
  Type: EventBridgeRule
  Properties:
    Target:
      Id: MyTarget
```

Schedule

Schedule이벤트 소스 유형을 설명하는 객체로, 이 객체는 귀하의 상태 머신을 일정에 따라 트리거되는 EventBridge 규칙의 대상으로 설정합니다. 자세한 내용은 Amazon EventBridge User Guide(Amazon EventBridge 사용 설명서)의 What Is Amazon EventBridge?(Amazon EventBridge 란?)를 참조하십시오.

AWS Serverless Application Model(AWS SAM)은 이 이벤트 유형이 설정되면 AWS::Events::Rule리소스를 생성합니다.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
DeadLetterConfig: DeadLetterConfig
Description: String
Enabled: Boolean
Input: String
Name: String
RetryPolicy: RetryPolicy
RoleArn: String
Schedule: String
State: String
Target: Target
```

속성

`DeadLetterConfig`

대상 호출이 실패한 경우 EventBridge가 이벤트를 전송하는 Amazon Simple Queue Service(Amazon SQS) 대기열을 구성합니다. 예를 들어 존재하지 않는 Lambda 함수로 이벤트를 전송하거나 EventBridge에 Lambda 함수를 호출할 권한이 충분하지 않은 경우 호출이 실패할 수 있습니다. 자세한 내용은 Amazon EventBridge 사용자 가이드의 이벤트 재시도 정책 및 DLQ(Dead Letter Queue) 사용을 참조하세요.

유형: DeadLetterConfig

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `DeadLetterConfig` 데이터 유형의 `AWS::Events::Rule Target` 속성과 유사합니다. DLQ를 자동으로 AWS SAM 생성하고자 하는 경우에 대비하여 이 속성의 AWS SAM 버전에는 추가 하위 속성이 포함되어 있습니다.

`Description`

규칙에 대한 설명.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [Description] 리소스의 AWS::Events::Rule 속성으로 직접 전달됩니다.

Enabled

규칙을 활성화할지를 나타냅니다.

규칙을 비활성화하려면 이 속성을 false로 설정합니다.

> **Note**
>
> Enabled 또는 State 속성을 지정할 수 있지만, 두 속성을 함께 지정할 수는 없습니다.

유형: 부울

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [State] 리소스의 AWS::Events::Rule 속성과 유사합니다. 이 속성이 true로 설정되면 AWS SAM가 ENABLED로 전달되고, 그렇지 않으면 DISABLED로 전달됩니다.

Input

대상으로 전달되는 유효한 JSON 텍스트입니다. 이 속성을 사용하면 이벤트 텍스트 자체의 어떤 것도 대상으로 전달되지 않습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [Input] 리소스의 AWS::Events::Rule Target 속성으로 직접 전달됩니다.

Name

규칙의 이름입니다. 이름을 지정하지 않은 경우 CloudFormation은 고유한 물리적 ID를 생성한 후 규칙 이름에 해당 ID를 사용합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [Name](#) 리소스의 `AWS::Events::Rule` 속성으로 직접 전달됩니다.

`RetryPolicy`

재시도 정책 설정에 대한 정보가 포함된 `RetryPolicy` 객체입니다. 자세한 내용은 Amazon EventBridge 사용자 가이드의 이벤트 재시도 정책 및 DLQ(Dead Letter Queue) 사용을 참조하세요.

유형: RetryPolicy

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `RetryPolicy` `AWS::Events::Rule` 데이터 유형의 `Target` 속성에 직접 전달됩니다.

`RoleArn`

스케줄이 호출될 때 EventBridge 스케줄러가 대상에 사용할 IAM 역할의 ARN입니다.

유형: RoleArn

필수 여부: 아니요. 제공되지 않으면 새 역할이 생성되어 사용됩니다.

CloudFormation호환성: 이 속성은 `RoleArn` `AWS::Scheduler::Schedule` 데이터 유형의 `Target` 속성에 직접 전달됩니다.

`Schedule`

규칙 실행 시기 및 방법을 결정하는 스케줄링 표현식입니다. 자세한 내용은 규칙에 대한 예약 표현식을 참조하세요.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `ScheduleExpression` 리소스의 `AWS::Events::Rule` 속성으로 직접 전달됩니다.

`State`

규칙의 상태입니다.

허용되는 값: `DISABLED | ENABLED`

> **Note**
>
> `Enabled` 또는 `State` 속성을 지정할 수 있지만, 두 속성을 함께 지정할 수는 없습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Events::Rule` 리소스의 [`State`] 속성으로 직접 전달됩니다.

`Target`

규칙이 트리거될 경우 EventBridge가 호출하는 AWS 리소스입니다. 이 속성을 사용하여 대상의 논리적 ID를 지정할 수 있습니다. 이 속성을 지정하지 않으면 AWS SAM이 대상의 논리적 ID를 생성합니다.

유형: [Target]

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Events::Rule` 리소스의 [`Targets`] 속성과 유사합니다. 이 속성의 AWS SAM 버전에서는 단일 대상의 논리적 ID만 지정할 수 있습니다.

예시

CloudWatch Schedule Event

CloudWatch Schedule Events

YAML

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

DeadLetterConfig

대상 호출 실패 후 EventBridge가 이벤트를 전송하는 Amazon Simple Queue Service(Amazon SQS) 대기열을 지정하는 데 사용되는 객체입니다. 예를 들어 존재하지 않는 상태 시스템으로 이벤트를 전송하거나 상태 시스템을 호출할 권한이 충분하지 않은 경우 간접 호출이 실패할 수 있습니다. 자세한 내용은 Amazon EventBridge 사용자 가이드의 이벤트 재시도 정책 및 DLQ(Dead Letter Queue) 사용을 참조하세요.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
Arn: String
QueueLogicalId: String
Type: String
```

속성

`Arn`

DLQ(Dead Letter Queue)의 대상으로 지정된 Amazon SQS 대기열의 Amazon 리소스 이름(ARN)입니다.

> **Note**
>
> `Type` 속성 또는 `Arn` 속성 중 하나만 지정해야 하며, 둘 다 지정할 수는 없습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `Arn` `AWS::Events::Rule` 데이터 유형의 `DeadLetterConfig` 속성에 직접 전달됩니다.

`QueueLogicalId`

`Type`가 지정된 경우 AWS SAM이 생성하는 DLQ의 사용자 지정 이름입니다.

> **Note**
>
> `Type` 속성이 설정되지 않은 경우 이 속성은 무시됩니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`Type`

대기열 유형. 이 속성이 설정되면 AWS SAM은 DLQ를 자동으로 만들고 허가를 부여하기 위해 필요한 리소스 기반 정책을 규칙 리소스에 첨부하여 해당 큐에 이벤트를 전송합니다.

> **Note**
>
> `Type` 속성 또는 `Arn` 속성 중 하나만 지정해야 하며, 둘 다 지정할 수는 없습니다.

유효한 값: `SQS`

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

예시

DeadLetterConfig

DeadLetterConfig

YAML

```
DeadLetterConfig:
```

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

Target

규칙이 트리거될 경우 EventBridge가 호출하는 AWS 리소스를 구성합니다.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

YAML

```
  Id: String
```

속성

`Id`

대상의 논리적 ID.

`Id`의 값은 영숫자 문자, 마침표(`.`), 하이픈(`-`), 밑줄(_)을 포함할 수 있습니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `Id` `AWS::Events::Rule` 데이터 유형의 `Target` 속성에 직접 전달됩니다.

예시

대상

YAML

```
EBRule:
  Type: Schedule
  Properties:
```

```
    Target:
      Id: MyTarget
```

ScheduleV2

`ScheduleV2` 이벤트 소스 유형을 설명하는 객체로, 이 객체는 상태 머신을 일정에 따라 트리거되는 Amazon EventBridge Scheduler 이벤트의 대상으로 설정합니다. 자세한 내용은 EventBridge Scheduler 사용 설명서의 [Amazon EventBridge Scheduler란 무엇인가요?](#)를 참조하세요.

AWS Serverless Application Model(AWS SAM)은 이 이벤트 유형이 설정되면 [AWS::Scheduler::Schedule](#)리소스를 생성합니다.

명령문

귀하의 AWS Serverless Application Model(AWS SAM) 템플릿에서 이 객체를 선언하려면 다음 명령문을 사용합니다.

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
```

속성

`DeadLetterConfig`

대상 호출이 실패한 경우 EventBridge가 이벤트를 전송하는 Amazon Simple Queue Service(Amazon SQS) 대기열을 구성합니다. 예를 들어 존재하지 않는 Lambda 함수로 이벤트를

전송하거나 EventBridge에 Lambda 함수를 간접 호출할 권한이 충분하지 않은 경우 간접 호출이 실패할 수 있습니다. 자세한 내용은 EventBridge 스케줄러 사용자 가이드의 EventBridge 스케줄러용 DLQ 대기열 구성을 참조하세요.

유형: DeadLetterConfig

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `DeadLetterConfig` 데이터 유형의 `AWS::Scheduler::Schedule` `Target` 속성과 유사합니다. DLQ를 자동으로 AWS SAM 생성하고자 하는 경우에 대비하여 이 속성의 AWS SAM 버전에는 추가 하위 속성이 포함되어 있습니다.

`Description`

일정에 대한 설명입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `Description` 리소스의 `AWS::Scheduler::Schedule` 속성으로 직접 전달됩니다.

`EndDate`

일정이 대상을 간접 호출할 수 있는 날짜(UTC 기준)입니다. 일정의 반복 식에 따라 사용자가 지정하는 EndDate 또는 그 이전에 호출이 중지될 수 있습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Scheduler::Schedule` 리소스의 `EndDate` 속성으로 직접 전달됩니다.

`FlexibleTimeWindow`

일정을 간접 호출할 수 있는 창을 구성할 수 있습니다.

유형: FlexibleTimeWindow

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Scheduler::Schedule` 리소스의 [`FlexibleTimeWindow`] 속성으로 직접 전달됩니다.

`GroupName`

이 일정과 연계하기 위한 일정 그룹의 이름입니다. 정의되지 않은 경우 기본 그룹이 사용됩니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Scheduler::Schedule` 리소스의 [`GroupName`] 속성으로 직접 전달됩니다.

`Input`

대상으로 전달되는 유효한 JSON 텍스트입니다. 이 속성을 사용하면 이벤트 텍스트 자체의 어떤 것도 대상으로 전달되지 않습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Scheduler::Schedule Target` 리소스의 [`Input`] 속성으로 직접 전달됩니다.

`KmsKeyArn`

고객 데이터를 암호화하는 데 사용되는 KMS 키의 ARN.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Scheduler::Schedule` 리소스의 [`KmsKeyArn`] 속성으로 직접 전달됩니다.

`Name`

일정의 이름입니다. 이름을 지정하지 않으면 AWS SAM이 *`StateMachine-Logical-IDEvent-Source-Name`*의 형식으로 이름을 생성하고 일정 이름에 해당 ID를 사용합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Scheduler::Schedule` 리소스의 [Name](#) 속성으로 직접 전달됩니다.

`OmitName`

기본적으로는 AWS SAM이 스케줄의 명칭을 *<State-machine-logical-ID><event-source-name>*의 형식으로 일정의 이름을 생성 및 사용합니다. `true`가 고유한 물리적 ID를 생성하도록 이 속성을 CloudFormation에 설정하고 이 ID를 스케줄 이름으로 대신 사용하십시오.

유형: 부울

필수 항목 여부: 아니요

기본값: `false`

CloudFormation호환성: 이 속성은 AWS SAM에 고유하며 CloudFormation 동등한 속성이 없습니다.

`PermissionsBoundary`

역할에 대한 권한 경계 설정에 사용되는 정책의 ARN입니다.

> **Note**
>
> `PermissionsBoundary`가 정의된 경우 AWS SAM는 스케줄러 일정의 대상 IAM 역할에 동일한 경계를 적용합니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [`PermissionsBoundary`](#) 리소스의 `AWS::IAM::Role` 속성으로 직접 전달됩니다.

`RetryPolicy`

재시도 정책 설정에 대한 정보가 포함된 `RetryPolicy` 객체입니다.

유형: [RetryPolicy](#)

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `RetryPolicy` `AWS::Scheduler::Schedule` 데이터 유형의 `Target` 속성에 직접 전달됩니다.

`RoleArn`

스케줄이 호출될 때 EventBridge 스케줄러가 대상에 사용할 IAM 역할의 ARN입니다.

유형: RoleArn

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `RoleArn` `AWS::Scheduler::Schedule` 데이터 유형의 `Target` 속성에 직접 전달됩니다.

`ScheduleExpression`

일정 실행 시기 및 방법을 결정하는 예약 표현식입니다.

유형: 문자열

필수 항목 여부: 예

CloudFormation호환성: 이 속성은 `ScheduleExpression` 리소스의 `AWS::Scheduler::Schedule` 속성으로 직접 전달됩니다.

`ScheduleExpressionTimezone`

스케줄링 표현식이 평가되는 시간대입니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Scheduler::Schedule` 리소스의 `ScheduleExpressionTimezone` 속성으로 직접 전달됩니다.

`StartDate`

일정이 대상의 간접 호출을 시작할 수 있는 날짜(UTC 기준)입니다. 일정의 반복 식에 따라 사용자가 지정하는 StartDate 또는 그 이후에 호출이 발생할 수 있습니다.

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 `AWS::Scheduler::Schedule` 리소스의 [StartDate](#) 속성으로 직접 전달됩니다.

`State`

일정 상태

허용되는 값:`DISABLED | ENABLED`

유형: 문자열

필수 항목 여부: 아니요

CloudFormation호환성: 이 속성은 [State](#) 리소스의 `AWS::Scheduler::Schedule` 속성으로 직접 전달됩니다.

예시

ScheduleV2 리소스를 정의하는 기본 예제

```
StateMachine:
  Type: AWS::Serverless::StateMachine
  Properties:
    Name: MyStateMachine
    Events:
      ScheduleEvent:
        Type: ScheduleV2
        Properties:
          ScheduleExpression: "rate(1 minute)"
      ComplexScheduleEvent:
        Type: ScheduleV2
        Properties:
          ScheduleExpression: rate(1 minute)
          FlexibleTimeWindow:
            Mode: FLEXIBLE
            MaximumWindowInMinutes: 5
          StartDate: '2022-12-28T12:00:00.000Z'
          EndDate: '2023-01-28T12:00:00.000Z'
          ScheduleExpressionTimezone: UTC
          RetryPolicy:
            MaximumRetryAttempts: 5
```

```
            MaximumEventAgeInSeconds: 300
          DeadLetterConfig:
            Type: SQS
    DefinitionUri:
      Bucket: sam-sam-s3-demo-bucket
      Key: my-state-machine.asl.json
      Version: 3
    Policies:
      - LambdaInvokePolicy:
          FunctionName: !Ref MyFunction
```

AWS SAM용으로 생성된 CloudFormation 리소스

이 섹션에서는 AWS SAM이 AWS 템플릿을 처리할 때 생성되는 CloudFormation 리소스에 대한 세부 정보를 제공합니다. AWS SAM이 생성하는 CloudFormation 리소스 세트는 지정하는 시나리오에 따라 달라집니다. 시나리오는 템플릿 파일에 지정된 AWS SAM 리소스와 속성의 조합입니다. 템플릿 파일에서 명시적으로 선언한 CloudFormation 리소스를 참조하는 것과 마찬가지로 템플릿 파일 내 다른 곳에서 생성된 리소스를 참조할 수 있습니다.

예를 들어 `AWS::Serverless::Function` 템플릿 파일에 AWS SAM리소스를 지정하면 항상 AWS SAM 기본 리소스가 `AWS::Lambda::Function`생성됩니다. 선택적 `AutoPublishAlias` 속성도 지정하면 AWS SAM 가 `AWS::Lambda::Alias`과 `AWS::Lambda::Version` 리소스를 추가로 생성합니다.

이 섹션에서는 시나리오와 각 시나리오에서 생성되는 CloudFormation 리소스를 나열하고 CloudFormation 템플릿 파일에서 생성된 AWS SAM 리소스를 참조하는 방법을 보여줍니다.

생성된 CloudFormation 리소스 참조

CloudFormation 템플릿 파일 내에서 생성된 AWS SAM 리소스를 `LogicalId` 또는 참조 가능한 속성별로 참조하는 두 가지 옵션이 있습니다.

LogicalID로 생성된 CloudFormation 리소스 참조

CloudFormation가 생성하는 AWS SAM 리소스에는 각각 템플릿 파일 내에서 고유한 영숫자(A-Z, a-z, 0-9) 식별자인 [`LogicalId`]가 있습니다. AWS SAM는 템플릿 파일에 있는 `LogicalIds` 리소스의 AWS SAM을 사용하여 템플릿 파일이 생성하는 `LogicalIds` 리소스의 CloudFormation을 구성합니다. 귀하가 명시적으로 선언한 `LogicalId` 리소스의 경우와 마찬가지로 생성된 CloudFormation 리소스의 CloudFormation을 사용하여 귀하의 템플릿 파일 내에서 해당 리소스의 속성에 액세스할 수 있습

니다. `LogicalIds` 및 CloudFormation 템플릿 내 AWS SAM에 대한 자세한 내용은 [사용자 가이드](#)의 AWS CloudFormation리소스를 참조하세요.

> **Note**
>
> 생성된 일부 리소스의 `LogicalIds`에는 네임스페이스 충돌을 방지하기 위한 고유한 해시 값이 포함되어 있습니다. 이러한 리소스의 `LogicalIds`는 스택이 생성될 때 파생됩니다. AWS Management 콘솔이나 AWS CLI, 아니면 AWS SDK 중 하나를 사용하여 스택을 생성한 후에만 해당 스택을 검색할 수 있습니다. 해시 값이 변경될 수 있으므로 이러한 `LogicalId` 리소스를 참조하지 않는 것이 좋습니다.

참조 가능한 속성별 생성된 CloudFormation 리소스 참조

생성된 일부 리소스의 경우 AWS SAM은 AWS SAM 리소스의 참조 가능한 속성을 제공합니다. 이 속성을 사용하여 귀하의 CloudFormation 템플릿 파일 내에서 생성된 AWS SAM 리소스와 해당 속성을 참조할 수 있습니다.

> **Note**
>
> 생성된 모든 CloudFormation 리소스에 참조 가능한 속성이 있는 것은 아닙니다. 이러한 리소스의 경우 `LogicalId`를 사용해야 합니다.

생성된 CloudFormation 리소스 시나리오

다음 표에는 AWS SAM 리소스 생성 시나리오를 구성하는 CloudFormation 리소스 및 속성이 요약되어 있습니다. 시나리오 열의 주제는 해당 시나리오를 위해 CloudFormation가 생성하는 추가 AWS SAM 리소스에 대한 세부 정보를 제공합니다.

AWS SAM 리소스	기본 CloudFormation 리소스	시나리오
[`AWS::Serverless::Api`](#)	[`AWS::ApiGateway::RestApi`](#)	• [DomainName 속성이 지정되었습니다.](#) • [UsagePlan 속성 지정됨](#)

AWS SAM 리소스	기본 CloudFormation 리소스	시나리오
[AWS::Serverless::Application](https://docs.aws.amazon.com/)	[AWS::CloudFormation::Stack](https://docs.aws.amazon.com/)	• 기본 CloudFormation 리소스를 생성하는 것 외에 이 서버리스 리소스에 대한 추가 시나리오는 없습니다.
AWS::Serverless::Function	AWS::Lambda::Function	• AutoPublishalias 속성 지정됨 • 역할 속성이 지정되지 않음 • DeploymentPreference 속성 지정됨 • Api 이벤트 소스 지정됨 • HttpApi 이벤트 소스 지정됨 • 스트리밍 이벤트 소스 지정됨 • 이벤트 브리지(또는 이벤트 버스) 이벤트 소스 지정됨 • IoTRule 이벤트 소스 지정됨 • Amazon SNS 이벤트에 대하여 OnSuccess (또는 OnFailure) 속성이 지정됩니다 • Amazon SQS 이벤트에 대하여 OnSuccess (또는 OnFailure) 속성이 지정됩니다
AWS::Serverless::HttpApi	AWS::ApiGatewayV2::Api	• 스테이지 이름 속성이 지정되었습니다. • StageName 속성이 지정되지 않았습니다. • DomainName 속성이 지정되었습니다.

AWS SAM 리소스	기본 CloudFormation 리소스	시나리오
AWS::Serverless::LayerVersion	AWS::Lambda::LayerVersion	• 기본 CloudFormation 리소스를 생성하는 것 외에 이 서버리스 리소스에 대한 추가 시나리오는 없습니다.
AWS::Serverless::SimpleTable	AWS::DynamoDB::Table	• 기본 CloudFormation 리소스를 생성하는 것 외에 이 서버리스 리소스에 대한 추가 시나리오는 없습니다.
AWS::Serverless::StateMachine	AWS::StepFunctions::StateMachine	• 역할 속성이 지정되지 않음 • API 이벤트 소스 지정됨 • 이벤트 브리지(또는 이벤트 버스) 이벤트 소스 지정됨

주제

- CloudFormation가 지정된 경우 생성되는 AWS::Serverless::Api 리소스
- CloudFormation가 지정된 경우 생성되는 AWS::Serverless::Application 리소스
- CloudFormation을 지정한 경우 생성되는 AWS::Serverless::Connector 리소스
- CloudFormation가 지정된 경우 생성되는 AWS::Serverless::Function 리소스
- CloudFormation가 지정된 경우 생성되는 AWS::Serverless::GraphQLApi 리소스
- CloudFormationAWS::서버리스::HTTP API가 지정될 때 생성되는 리소스
- CloudFormation가 지정된 경우 생성되는 AWS::Serverless::LayerVersion 리소스
- CloudFormation가 지정된 경우 생성되는 AWS::Serverless::SimpleTable 리소스
- CloudFormation가 지정된 경우 생성되는 AWS::Serverless::StateMachine 리소스

CloudFormation가 지정된 경우 생성되는 AWS::Serverless::Api 리소스

`AWS::Serverless::Api`이 지정되면 AWS Serverless Application Model(AWS SAM)은 항상 `AWS::ApiGateway::RestApi` 기본 CloudFormation 리소스를 생성합니다. 또한 항상 `AWS::ApiGateway::Stage` 및 `AWS::ApiGateway::Deployment` 리소스도 생성합니다.

`AWS::ApiGateway::RestApi`

LogicalId: *<api-LogicalId>*

참조 가능한 속성: 해당 없음(이 `LogicalId` 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

`AWS::ApiGateway::Stage`

LogicalId: *<api-LogicalId><stage-name>*`Stage`

*<stage-name>*은 `StageName` 속성이 설정된 문자열입니다. 예를 들어 `StageName`을 `Gamma`로 설정하면, `LogicalId`은 *MyRestApiGamma*`Stage`가 됩니다.

참조 가능한 속성: *<api-LogicalId>*`.Stage`

`AWS::ApiGateway::Deployment`

LogicalId: *<api-LogicalId>*`Deployment`*<sha>*

*<sha>*은 스택이 만들어질 때 생성되는 고유한 해시 값입니다. 예를 들어 *MyRestApi*`Deployment`*926eeb5ff1*입니다.

참조 가능한 속성: *<api-LogicalId>*`.Deployment`

이러한 CloudFormation 리소스 외에도 `AWS::Serverless::Api`가 지정되면, AWS SAM은 다음 시나리오들에 대하여 추가적인 CloudFormation 리소스를 생성합니다.

시나리오

- DomainName 속성이 지정되었습니다.
- UsagePlan 속성 지정됨

DomainName 속성이 지정되었습니다.

`DomainName`의 `Domain` 속성의 `AWS::Serverless::Api` 속성이 지정되면 AWS SAM가 `AWS::ApiGateway::DomainName` CloudFormation 리소스를 생성합니다.

AWS::ApiGateway::DomainName

LogicalId: ApiGatewayDomainName*<sha>*

*<sha>*은 스택이 만들어질 때 생성되는 고유한 해시 값입니다. 예를 들어 ApiGatewayDomainName*926eeb5ff1*입니다.

참조 가능한 속성: *<api-LogicalId>*.DomainName

UsagePlan 속성 지정됨

UsagePlan의 Auth 속성의 AWS::Serverless::Api 속성이 지정되면 AWS SAM가 다음의 CloudFormation 리소스, 즉 AWS::ApiGateway::UsagePlan, AWS::ApiGateway::UsagePlanKey, 및 AWS::ApiGateway::ApiKey를 생성합니다.

AWS::ApiGateway::UsagePlan

LogicalId: *<api-LogicalId>*UsagePlan

참조 가능한 속성: *<api-LogicalId>*.UsagePlan

AWS::ApiGateway::UsagePlanKey

LogicalId: *<api-LogicalId>*UsagePlanKey

참조 가능한 속성: *<api-LogicalId>*.UsagePlanKey

AWS::ApiGateway::ApiKey

LogicalId: *<api-LogicalId>*ApiKey

참조 가능한 속성: *<api-LogicalId>*.ApiKey

CloudFormation가 지정된 경우 생성되는 AWS::Serverless::Application 리소스

AWS::Serverless::Application이 지정되면 AWS Serverless Application Model(AWS SAM)는 AWS::CloudFormation::Stack 기본 CloudFormation 리소스를 생성합니다.

AWS::CloudFormation::Stack

LogicalId: *<application-LogicalId>*

참조 가능한 속성: 해당 없음(이 LogicalId 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

CloudFormation을 지정한 경우 생성되는 AWS::Serverless::Connector 리소스

> **Note**
>
> 임베디드 Connectors 속성을 통해 커넥터를 정의하면 이러한 리소스가 생성되기 전에 먼저 AWS::Serverless::Connector 리소스로 변환됩니다.

AWS::Serverless::Connector 템플릿에서 AWS SAM 리소스를 지정하면 AWS SAM이 필요에 따라 다음과 같은 AWS CloudFormation 리소스가 생성됩니다.

AWS::IAM::ManagedPolicy

LogicalId:*<connector-LogicalId>*Policy

참조 가능한 속성: 해당 없음(이 CloudFormation 리소스를 참조하려면 LogicalId를 사용해야 합니다.)

AWS::SNS::TopicPolicy

LogicalId:*<connector-LogicalId>*TopicPolicy

참조 가능한 속성: 해당 없음(이 CloudFormation 리소스를 참조하려면 LogicalId를 사용해야 합니다.)

AWS::SQS::QueuePolicy

LogicalId:*<connector-LogicalId>*QueuePolicy

참조 가능한 속성: 해당 없음(이 CloudFormation 리소스를 참조하려면 LogicalId를 사용해야 합니다.)

AWS::Lambda::Permission

LogicalId:*<connector-LogicalId><permission>*LambdaPermission

*<permission>*은 Permissions 속성에 의해 지정된 권한입니다. 예를 들어 Write입니다.

참조 가능한 속성: 해당 없음(이 CloudFormation 리소스를 참조하려면 `LogicalId`를 사용해야 합니다.)

CloudFormation가 지정된 경우 생성되는 AWS::Serverless::Function 리소스

`AWS::Serverless::Function`이 지정되면 AWS Serverless Application Model(AWS SAM)는 항상 `AWS::Lambda::Function` 기본 CloudFormation 리소스를 만듭니다.

`AWS::Lambda::Function`

`LogicalId`: *`<function-LogicalId>`*

참조 가능한 속성: 해당 없음(이 CloudFormation 리소스를 참조하려면 `LogicalId`를 사용해야 합니다.)

이 CloudFormation 리소스 외에도 `AWS::Serverless::Function`를 지정하면 AWS SAM은 다음 시나리오에 대한 CloudFormation 리소스도 생성합니다.

시나리오

- 핵심 함수 속성
- 이벤트 소스
- 이벤트 구성

핵심 함수 속성

다음 시나리오는 핵심 함수 속성을 기반으로 CloudFormation 리소스를 생성합니다.

역할 속성이 지정되지 않음

`Role`의 `AWS::Serverless::Function` 속성이 지정되지 않은 경우 AWS SAM는 `AWS::IAM::Role` CloudFormation 리소스를 생성합니다.

`AWS::IAM::Role`

`LogicalId`: *`<function-LogicalId>`*`Role`

참조 가능한 속성: 해당 없음(이 CloudFormation 리소스를 참조하려면 `LogicalId`를 사용해야 합니다.)

AutoPublishalias 속성 지정됨

`AutoPublishAlias`의 `AWS::Serverless::Function` 속성을 지정하면 AWS SAM은 다음과 같은 CloudFormation 리소스, 즉 `AWS::Lambda::Alias` 및 `AWS::Lambda::Version`을 생성합니다.

`AWS::Lambda::Alias`

LogicalId: *<function-LogicalId>*Alias*<alias-name>*

*<alias-name>*은 `AutoPublishAlias`가 설정된 문자열입니다. 예를 들어 `AutoPublishAlias`을 `live`로 설정하면 `LogicalId`은 *MyFunction*Alias*live*입니다.

참조 가능한 속성: *<function-LogicalId>*.Alias

`AWS::Lambda::Version`

LogicalId: *<function-LogicalId>*Version*<sha>*

*<sha>*은 스택이 만들어질 때 생성되는 고유한 해시 값입니다. 예를 들어 *MyFunction*버전*926eeb5ff1*입니다.

참조 가능한 속성: *<function-LogicalId>*.Version

`AutoPublishAlias` 속성에 대한 자세한 내용은 AWS::Serverless::Function의 속성 섹션을 참조하세요.

DeploymentPreference 속성 지정됨

`DeploymentPreference`의 `AWS::Serverless::Function` 속성을 지정하면 AWS SAM은 다음과 같은 CloudFormation 리소스, 즉 `AWS::CodeDeploy::Application` 및 `AWS::CodeDeploy::DeploymentGroup`을 생성합니다. 또한, `Role` 객체의 `DeploymentPreference` 속성이 지정되지 않은 경우, AWS SAM는 `AWS::IAM::Role` CloudFormation 리소스를 생성합니다.

`AWS::CodeDeploy::Application`

LogicalId: `ServerlessDeploymentApplication`

참조 가능한 속성: 해당 없음(이 `LogicalId` 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

AWS::CodeDeploy::DeploymentGroup

LogicalId: *<function-LogicalId>*DeploymentGroup

참조 가능한 속성: 해당 없음(이 LogicalId 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

AWS::IAM::Role

LogicalId: CodeDeployServiceRole

참조 가능한 속성: 해당 없음(이 LogicalId 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

FunctionUrlConfig 속성이 지정됨

FunctionUrlConfig 속성이 지정되면, AWS SAM은(는) AuthType을(를) 기반으로 다른 CloudFormation 리소스를 생성합니다.

AuthType: NONE이(가) 지정되면 AWS SAM은(는) 다음 CloudFormation 리소스를 생성합니다.

AWS::Lambda::Permission(액세스 간접 호출)

LogicalId: *<function-LogicalId>*URLInvokeAllowPublicAccess

참조 가능한 속성: 해당 없음(이 LogicalId 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

AWS::Lambda::Permission(퍼블릭 액세스)

LogicalId: *<function-LogicalId>*UrlPublicPermissions

참조 가능한 속성: 해당 없음(이 LogicalId 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

AWS::Lambda::Url

LogicalId: *<function-LogicalId>*Url

참조 가능한 속성: 해당 없음(이 LogicalId 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

AuthType: AWS_IAM이(가) 지정되면 AWS SAM은(는) 다음만 생성합니다.

AWS::Lambda::Url

LogicalId: *<function-LogicalId>*Url

참조 가능한 속성: 해당 없음(이 LogicalId 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

FunctionUrlConfig 속성에 대한 추가 정보는 FunctionUrlConfig를 참조하세요.

이벤트 소스

다음 시나리오에서는 이벤트 소스를 기반으로 CloudFormation 리소스를 생성합니다.

Api 이벤트 소스 지정됨

Event의 AWS::Serverless::Function 속성이 Api으로 설정되었지만 RestApiId 속성이 지정되지 않은 경우 AWS SAM이 AWS::ApiGateway::RestApi CloudFormation 리소스를 생성합니다.

AWS::ApiGateway::RestApi

LogicalId: ServerlessRestApi

참조 가능한 속성: 해당 없음(이 LogicalId 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

HttpApi 이벤트 소스 지정됨

Event의 AWS::Serverless::Function 속성이 HttpApi으로 설정되었지만 ApiId 속성이 지정되지 않은 경우 AWS SAM이 AWS::ApiGatewayV2::Api CloudFormation 리소스를 생성합니다.

AWS::ApiGatewayV2::Api

LogicalId: ServerlessHttpApi

참조 가능한 속성: 해당 없음(이 LogicalId 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

스트리밍 이벤트 소스 지정됨

Event의 AWS::Serverless::Function 속성이 스트리밍 타입의 하나로 설정된 경우, AWS SAM이 AWS::Lambda::EventSourceMapping CloudFormation 리소스를 생성합니다. 이는 DynamoDB, Kinesis, MQ, MSK, 및 SQS 유형에 적용됩니다.

AWS::Lambda::EventSourceMapping

LogicalId: *<function-LogicalId><event-LogicalId>*

참조 가능한 속성: 해당 없음(이 LogicalId 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

이벤트 브리지(또는 이벤트 버스) 이벤트 소스 지정됨

Event의 AWS::Serverless::Function 속성이 이벤트 브리지 (또는 이벤트 버스) 유형 중 하나로 설정되면 AWS SAM이 AWS::Events::Rule CloudFormation 리소스를 생성합니다. 이는EventBridgeRule, Schedule, 및 CloudWatchEvents 유형에 적용됩니다.

AWS::Events::Rule

LogicalId: *<function-LogicalId><event-LogicalId>*

참조 가능한 속성: 해당 없음(이 LogicalId 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

IoTRule 이벤트 소스 지정됨

Event의 AWS::Serverless::Function 속성이 IoTrule로 설정되면 AWS SAM이 AWS::IoT::TopicRule CloudFormation 리소스를 생성합니다.

AWS::IoT::TopicRule

LogicalId: *<function-LogicalId><event-LogicalId>*

참조 가능한 속성: 해당 없음(이 LogicalId 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

이벤트 구성

다음 시나리오에서는 이벤트 구성을 기반으로 CloudFormation 리소스를 생성합니다.

Amazon SNS 이벤트에 대하여 OnSuccess(또는 OnFailure) 속성이 지정됩니다

`OnSuccess`의 `OnFailure` 속성의 `DestinationConfig` 속성의 `EventInvokeConfig`(또는 `AWS::Serverless::Function`) 속성이 지정되고 대상 유형이 `SNS`이지만, 대상 ARN이 지정되지 않은 경우, AWS SAM이 다음 CloudFormation 리소스, 즉 `AWS::Lambda::EventInvokeConfig` 및 `AWS::SNS::Topic`를 생성합니다.

`AWS::Lambda::EventInvokeConfig`

`LogicalId`: *`<function-LogicalId>`*`EventInvokeConfig`

참조 가능한 속성: 해당 없음(이 `LogicalId` 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

`AWS::SNS::Topic`

`LogicalId`: *`<function-LogicalId>`*`OnSuccessTopic` (또는 *`<function-LogicalId>`*`OnFailureTopic`)

참조 가능한 속성: *`<function-LogicalId>`*`.DestinationTopic`

`OnSuccess`및 `OnFailure` 가 모두 Amazon SNS 이벤트에 관하여 지정된 경우, 생성된 리소스 간에 구분하려면 `LogicalId`를 사용해야 합니다.

Amazon SQS 이벤트에 대하여 OnSuccess(또는 OnFailure) 속성이 지정됩니다

`OnSuccess`의 `OnFailure` 속성의 `DestinationConfig` 속성의 `EventInvokeConfig`(또는 `AWS::Serverless::Function`) 속성이 지정되고 대상 유형이 `SQS`이지만, 대상 ARN이 지정되지 않은 경우, AWS SAM이 다음 CloudFormation 리소스, 즉 `AWS::Lambda::EventInvokeConfig` 및 `AWS::SQS::Queue`를 생성합니다.

`AWS::Lambda::EventInvokeConfig`

`LogicalId`: *`<function-LogicalId>`*`EventInvokeConfig`

참조 가능한 속성: 해당 없음(이 `LogicalId` 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

`AWS::SQS::Queue`

`LogicalId`: *`<function-LogicalId>`*`OnSuccessQueue` (또는 *`<function-LogicalId>`*`OnFailureQueue`)

참조 가능한 속성: *<function-LogicalId>*.DestinationQueue

OnSuccess및 OnFailure 가 모두 Amazon SQS 이벤트에 관하여 지정된 경우, 생성된 리소스 간에 구분하려면 LogicalId를 사용해야 합니다.

CloudFormation가 지정된 경우 생성되는 AWS::Serverless::GraphQLApi 리소스

AWS::Serverless::GraphQLApi 리소스를 AWS Serverless Application Model(AWS SAM) 템플릿에서 지정하면 AWS SAM는 항상 다음과 같은 기본 AWS CloudFormation 리소스를 생성합니다.

AWS::AppSync::DataSource

LogicalId: *<graphqlapi-LogicalId><datasource-RelativeId><datasource-Type>*DataSource

참조 가능한 속성: 해당 없음(이 LogicalId 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

AWS::AppSync::FunctionConfiguration

LogicalId: *<graphqlapi-LogicalId><function-RelativeId>*

참조 가능한 속성: 해당 없음(이 LogicalId 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

AWS::AppSync::GraphQLApi

LogicalId: *<graphqlapi-LogicalId>*

참조 가능한 속성: 해당 없음(이 LogicalId 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

AWS::AppSync::GraphQLSchema

LogicalId: *<graphqlapi-LogicalId>*Schema

참조 가능한 속성: 해당 없음(이 LogicalId 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

AWS::AppSync::Resolver

LogicalId: *<graphqlapi-LogicalId><OperationType><resolver-RelativeId>*

참조 가능한 속성: 해당 없음(이 `LogicalId` 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

이러한 CloudFormation 리소스 외에도 `AWS::Serverless::GraphQLApi`이 지정된 경우 AWS SAM은 CloudFormation 리소스도 생성할 수 있습니다.

`AWS::AppSync::ApiCache`

`LogicalId`: *`<graphqlapi-LogicalId>`*`ApiCache`

참조 가능한 속성: 해당 없음(이 `LogicalId` 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

`AWS::AppSync::ApiKey`

`LogicalId`: *`<graphqlapi-LogicalId><apikey-RelativeId>`*

참조 가능한 속성: 해당 없음(이 `LogicalId` 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

`AWS::AppSync::DomainName`

`LogicalId`: *`<graphqlapi-LogicalId>`*`DomainName`

참조 가능한 속성: 해당 없음(이 `LogicalId` 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

`AWS::AppSync::DomainNameApiAssociation`

`LogicalId`: *`<graphqlapi-LogicalId>`*`DomainNameApiAssociation`

참조 가능한 속성: 해당 없음(이 `LogicalId` 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

AWS SAM은 `AWS::Serverless::Connector` 리소스를 사용하여 권한을 부여할 수도 있습니다. 자세한 내용은 [CloudFormation을 지정한 경우 생성되는 AWS::Serverless::Connector 리소스]을 참조하십시오.

CloudFormationAWS::서버리스::HTTP API가 지정될 때 생성되는 리소스

`AWS::Serverless::HttpApi`이 지정되면 AWS Serverless Application Model(AWS SAM)은 `AWS::ApiGatewayV2::Api` 기본 CloudFormation 리소스를 생성합니다.

AWS::ApiGatewayV2::Api

LogicalId: *<httpapi-LogicalId>*

참조 가능한 속성: 해당 없음(이 LogicalId 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

이 CloudFormation 리소스 외에도 AWS::Serverless::HttpApi를 지정하면 AWS SAM은 다음 시나리오에 대한 CloudFormation 리소스도 생성합니다.

시나리오

- 스테이지 이름 속성이 지정되었습니다.
- StageName 속성이 지정되지 않았습니다.
- DomainName 속성이 지정되었습니다.

스테이지 이름 속성이 지정되었습니다.

AWS::Serverless::HttpApi의 StageName 속성이 지정되면 AWS SAM이 AWS::ApiGatewayV2::Stage CloudFormation 리소스를 생성합니다.

AWS::ApiGatewayV2::Stage

LogicalId: *<httpapi-LogicalId><stage-name>*Stage

*<stage-name>*은 StageName 속성이 설정된 문자열입니다. 예를 들어, StageName을 Gamma로 설정하면 LogicalId은 *myHTTPAPIGamma*Stage입니다.

참조 가능한 속성: *<httpapi-LogicalId>*.Stage

StageName 속성이 지정되지 않았습니다.

AWS::Serverless::HttpApi의 StageName 속성이 지정되지 않은 경우 AWS SAM가 AWS::ApiGatewayV2::Stage CloudFormation 리소스를 생성합니다.

AWS::ApiGatewayV2::Stage

LogicalId: *<httpapi-LogicalId>*ApiGatewayDefaultStage

참조 가능한 속성: *<httpapi-LogicalId>*.Stage

DomainName 속성이 지정되었습니다.

DomainName의 Domain 속성의 AWS::Serverless::HttpApi 속성이 지정되면 AWS SAM가 AWS::ApiGatewayV2::DomainName CloudFormation 리소스를 생성합니다.

AWS::ApiGatewayV2::DomainName

LogicalId: ApiGatewayDomainNameV2*<sha>*

*<sha>*은 스택이 만들어질 때 생성되는 고유한 해시 값입니다. ApiGatewayDomainNameV2*926eeb5ff1*을 예로 들 수 있습니다.

참조 가능한 속성: *<httpapi-LogicalId>*.DomainName

CloudFormation가 지정된 경우 생성되는 AWS::Serverless::LayerVersion 리소스

AWS::Serverless::LayerVersion이 지정되면 AWS Serverless Application Model(AWS SAM)은 AWS::Lambda::LayerVersion 기본 CloudFormation 리소스를 생성합니다.

AWS::Lambda::LayerVersion

LogicalId: *<layerversion-LogicalId>*

참조 가능한 속성: 해당 없음(이 LogicalId 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

CloudFormation가 지정된 경우 생성되는 AWS::Serverless::SimpleTable 리소스

AWS::Serverless::SimpleTable이 지정되면 AWS Serverless Application Model(AWS SAM)은 AWS::DynamoDB::Table 기본 CloudFormation 리소스를 생성합니다.

AWS::DynamoDB::Table

LogicalId: *<simpletable-LogicalId>*

참조 가능한 속성: 해당 없음(이 LogicalId 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

CloudFormation가 지정된 경우 생성되는 AWS::Serverless::StateMachine 리소스

AWS::Serverless::StateMachine이 지정되면 AWS Serverless Application Model(AWS SAM)은 AWS::StepFunctions::StateMachine 기본 CloudFormation 리소스를 생성합니다.

AWS::StepFunctions::StateMachine

LogicalId: *<statemachine-LogicalId>*

참조 가능한 속성: 해당 없음(이 LogicalId 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

이 CloudFormation 리소스 외에도 AWS::Serverless::StateMachine를 지정하면 AWS SAM은 다음 시나리오에 대한 CloudFormation 리소스도 생성합니다.

시나리오

- 역할 속성이 지정되지 않음
- API 이벤트 소스 지정됨
- 이벤트 브리지(또는 이벤트 버스) 이벤트 소스 지정됨

역할 속성이 지정되지 않음

Role의 AWS::Serverless::StateMachine 속성이 지정되지 않은 경우 AWS SAM는 AWS::IAM::Role CloudFormation 리소스를 생성합니다.

AWS::IAM::Role

LogicalId: *<statemachine-LogicalId>*Role

참조 가능한 속성: 해당 없음(이 LogicalId 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

API 이벤트 소스 지정됨

Event의 AWS::Serverless::StateMachine 속성이 Api으로 설정되었지만 RestApiId 속성이 지정되지 않은 경우 AWS SAM이 AWS::ApiGateway::RestApi CloudFormation 리소스를 생성합니다.

`AWS::ApiGateway::RestApi`

`LogicalId`: `ServerlessRestApi`

참조 가능한 속성: 해당 없음(이 `LogicalId` 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

이벤트 브리지(또는 이벤트 버스) 이벤트 소스 지정됨

`Event`의 `AWS::Serverless::StateMachine` 속성이 이벤트 브리지 (또는 이벤트 버스) 유형 중 하나로 설정되면 AWS SAM이 `AWS::Events::Rule` CloudFormation 리소스를 생성합니다. 이는`EventBridgeRule`, `Schedule`, 및 `CloudWatchEvents` 유형에 적용됩니다.

`AWS::Events::Rule`

`LogicalId`: *`<statemachine-LogicalId><event-LogicalId>`*

참조 가능한 속성: 해당 없음(이 `LogicalId` 리소스를 참조하려면 CloudFormation를 사용해야 합니다.)

AWS SAM에서 지원되는 리소스 속성

리소스 속성은 추가 동작 및 관계를 제어하기 위해 AWS SAM 및 CloudFormation 리소스에 추가할 수 있는 속성입니다. 리소스 속성에 대한 자세한 내용은 AWS CloudFormation 사용 설명서의 리소스 속성 참조를 참조하세요.

AWS SAM은 CloudFormation에서 정의한 리소스 속성의 하위 집합을 지원합니다. 지원되는 리소스 속성 중 일부는 해당 CloudFormation 리소스의 기본 생성 AWS SAM 리소스에만 복사되고 일부는 해당 CloudFormation 리소스에서 생성된 모든 AWS SAM 리소스에 복사됩니다. 해당 CloudFormation 리소스에서 생성된 AWS SAM 리소스에 대한 자세한 내용은 AWS SAM용으로 생성된 CloudFormation 리소스을 참조하세요.

다음 표에는 아래 AWS SAM 목록에 따른 예외에 의한 리소스 속성 지원이 요약되어 있습니다.

Resource attributes	대상 생성 리소스
DependsOn 메타데이터 [1, 2]	기본 CloudFormation 생성 리소스에 한정합니다. AWS SAM리소스와 기본 CloudFormation 리소스 간의 매핑에 대한 자세한 내용은 생성된 CloudFormation 리소스 시나리오 섹션을 참조하세요.

Resource attributes	대상 생성 리소스
조건 DeletionPolicy UpdateReplacePolicy	해당 CloudFormation 리소스에서 생성된 모든 AWS SAM 리소스입니다. 생성된 CloudFormation 리소스의 시나리오에 대한 자세한 내용은 생성된 CloudFormation 리소스 시나리오 섹션을 참조하세요.

참고:

1. `Metadata`리소스 유형으로 `AWS::Serverless::Function` 리소스 속성을 사용하는 방법에 대한 자세한 내용은 AWS SAM에서 사용자 지정 런타임을 사용하여 Lambda 함수 빌드 섹션을 참조하세요.
2. `Metadata`리소스 유형으로 `AWS::Serverless::LayerVersion` 리소스 속성을 사용하는 방법에 대한 자세한 내용은 에서 Lambda 계층 빌드 AWS SAM 섹션을 참조하세요.

예외

앞서 설명한 리소스 속성 규칙에는 몇 가지 예외가 있습니다.

- `AWS::Lambda::LayerVersion`의 경우, AWS SAM만 있는 사용자 지정 필드는 생성된 `RetentionPolicy` 리소스를 위해 `DeletionPolicy`를 CloudFormation 설정합니다. 이것은 `DeletionPolicy` 자체보다 우선 순위가 높습니다. 둘 다 설정되지 않은 경우 기본 사항으로 `DeletionPolicy`이 `Retain`에 설정됩니다.
- `AWS::Lambda::Version`의 경우, `DeletionPolicy`를 지정하지 않으면 기본값으로 `Retain`이 지정됩니다.
- `DeploymentPreferences`이 서버리스 함수로 지정된 시나리오의 경우 리소스 속성은 다음과 같이 생성된 CloudFormation 리소스에 복사되지 않습니다.
  - `AWS::CodeDeploy::Application`
  - `AWS::CodeDeploy::DeploymentGroup`
  - 이 시나리오에 대해 생성된 `AWS::IAM::Role`라는 이름의 `CodeDeployServiceRole`
- 귀하의 AWS SAM템플릿에 묵시적으로 생성된 API 이벤트 소스가 있는 여러 함수가 포함된 경우 이 함수들은 생성된 `AWS::ApiGateway::RestApi` 리소스를 공유합니다. 이 시나리오에서 함수의 리

소스 속성이 서로 다른 경우, 생성된 `AWS::ApiGateway::RestApi` 리소스에 대해 AWS SAM는 다음 우선순위 목록에 따라 리소스 속성을 복사합니다:

- `UpdateReplacePolicy`:
  1. `Retain`
  2. `Snapshot`
  3. `Delete`
- `DeletionPolicy`:
  1. `Retain`
  2. `Delete`

AWS SAM의 API Gateway 확장

AWS을(를) 위해 특별히 설계된 API Gateway 확장 기능은 API를 설계하고 관리하기 위한 추가적인 사용자 지정 및 기능을 제공합니다. API 게이트웨이 확장은 AWS 해당 인증 및 API 게이트웨이별 API 통합을 지원하는 OpenAPI 사양의 확장입니다. API Gateway 확장에 대한 자세한 내용은 OpenAPI로 확장된 API 게이트웨이를 참조하세요.

AWS SAM은 API 게이트웨이 확장의 하위 집합을 지원합니다. AWS SAM이 어떤 API Gateway 확장을 지원하는지 확인하려면 다음 표를 참조하세요.

API 게이트웨이 확장	지원 제공자 AWS SAM
x-amazon-apigateway-any-method 객체	예
x-amazon-apigateway-api-key-source 속성	아니요
x-amazon-apigateway-auth 객체	예
x-amazon-apigateway-authorizer 객체	예
x-amazon-apigateway-authtype 속성	예
x-amazon-apigateway-binary-media-types 속성	예
x-amazon-apigateway-documentation 객체	아니요

x-amazon-apigateway-endpoint-configuration 객체	아니요
x-amazon-apigateway-gateway-responses 객체	예
x-amazon-apigateway-gateway-responses.gatewayResponse 객체	예
x-amazon-apigateway-gateway-responses.responseParameters 객체	예
x-amazon-apigateway-gateway-responses.responseTemplates 객체	예
x-amazon-apigateway-integration 객체	예
x-amazon-apigateway-integration.requestTemplates 객체	예
x-amazon-apigateway-integration.requestParameters 객체	아니요
x-amazon-apigateway-integration.responses 객체	예
x-amazon-apigateway-integration.response 객체	예
x-amazon-apigateway-integration.responseTemplates 객체	예
x-amazon-apigateway-integration.responseParameters 객체	예
x-amazon-apigateway-request-validator 속성	아니요
x-amazon-apigateway-request-validators 객체	아니요
x-amazon-apigateway-request-validators.requestValidator 객체	아니요

AWS SAM의 내장 함수

내장 함수는 런타임 시에만 사용할 수 있는 속성에 값을 할당하는 기본 제공 함수입니다. AWS SAM은 특정 내장 함수 속성에 대한 지원이 제한적이므로 일부 내장 함수를 해결할 수 없습니다. 따라서 이 문제를 해결하려면 `AWS::LanguageExtensions` 변환을 추가하는 것이 좋습니다. `AWS::LanguageExtensions`는 기본적으로 CloudFormation에 포함되지 않은 내장 함수와 기타 기능을 사용할 수 있도록 CloudFormation에서 호스팅하는 매크로입니다.

```
Transform:
  - AWS::LanguageExtensions
```

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

Note

CodeUri 속성에서 내장 함수를 사용하는 경우 AWS SAM은 값을 올바르게 구문 분석할 수 없습니다. 대신 `AWS::LanguageExtensions` 변환을 사용하는 것이 좋습니다.
자세한 내용은 AWS::Serverless::Function의 속성 섹션을 참조하세요.

내장 함수에 관한 자세한 정보는 사용자 가이드의 AWS CloudFormation내장 함수 참조를 확인하십시오.

AWS SAM용 서버리스 애플리케이션의 예

이 섹션에는 DynamoDB 이벤트를 처리하는 애플리케이션과 Amazon S3 이벤트를 처리하는 애플리케이션의 두 가지 애플리케이션 예제가 포함되어 있습니다. 각 예제에서는 애플리케이션을 생성하는 단계별 프로세스를 안내합니다. 또한 두 예제 모두 이벤트 소스 및 AWS 리소스 구성에 대한 세부 정보를 포함합니다. 두 예제 모두 시작하기 전에 완료해야 하는 작업을 알려준 다음 단계에 따라 애플리케이션 초기화, 테스트, 패키징 및 배포를 수행합니다.

주제

- AWS SAM에서 DynamoDB 이벤트 처리
- AWS SAM에서 Amazon S3 이벤트 처리

AWS SAM에서 DynamoDB 이벤트 처리

이 예제 애플리케이션을 사용하면 개요 및 Quick Start 가이드에서 학습한 내용을 토대로 다른 예제 애플리케이션을 설치합니다. 이 애플리케이션은 DynamoDB 테이블 이벤트 소스에서 간접 호출되는 Lambda 함수로 구성되어 있습니다. Lambda 함수는 매우 간단합니다. 즉, 이벤트 소스 메시지를 통해 전달된 데이터를 기록합니다.

이 연습에서는 Lambda 함수가 간접 호출될 때 Lambda 함수로 전달되는 이벤트 소스 메시지를 모방하는 방법을 보여줍니다.

시작하기 전 준비 사항

AWS SAM CLI 설치에서 필요한 설정을 완료했는지 확인하십시오.

1단계: 애플리케이션 초기화

이 단원에서 귀하는 AWS SAM 템플릿과 애플리케이션 코드로 구성된 애플리케이션 패키지를 다운로드합니다.

애플리케이션의 초기화

1. AWS SAM CLI 명령 프롬프트에서 다음 명령을 실행합니다.

```
sam init \
```

```
--location gh:aws-samples/cookiecutter-aws-sam-dynamodb-python \
--no-input
```

참고로 위 명령어 내 gh:는 GitHub URL https://github.com/로 확장됩니다.

2. 명령을 통해 생성된 디렉터리(dynamodb_event_reader/)의 내용을 검토합니다.

   - template.yaml – 읽기 DynamoDB 애플리케이션이 필요로 하는 두 개의 AWS 리소스, 즉 Lambda 함수와 DynamoDB 테이블을 정의합니다. 또한 템플릿은 두 리소스 간의 매핑을 정의합니다.
   - read_dynamodb_event/ 디렉터리 - DynamoDB 애플리케이션 코드를 포함합니다.

2단계: 애플리케이션 로컬 테스트

로컬 테스트의 경우 AWS SAM CLI를 사용하여 샘플 DynamoDB 이벤트를 생성하고 Lambda 함수를 간접 호출합니다.

```
sam local generate-event dynamodb update | sam local invoke --event - ReadDynamoDBEvent
```

이 generate-event 명령은 모든 구성 요소가 AWS 클라우드에 배포될 때 생성되는 메시지와 같은 테스트 이벤트 소스 메시지를 생성합니다. 이 이벤트 소스 메시지는 Lambda 함수 ReadDynamodBevent로 파이프됩니다.

app.py 내 소스 코드를 기반으로 예상 메시지가 콘솔에 인쇄되는지 확인합니다.

3단계: 애플리케이션 패키징

애플리케이션을 로컬에서 테스트한 후 AWS SAM CLI를 사용하여 애플리케이션을 AWS 클라우드에 배포하는 데 사용할 배포 패키지를 생성합니다.

Lambda 배포 패키지 생성하기

1. 패키징된 코드를 저장할 위치에 S3 버킷을 생성합니다. 기존 S3 버킷을 사용하려면 이 단계를 건너뜁니다.

```
aws s3 mb s3://bucketname
```

2. 다음 package CLI 명령을 명령 프롬프트에서 실행하여 배포 패키지를 생성합니다.

```
sam package \
```

```
    --template-file template.yaml \
    --output-template-file packaged.yaml \
    --s3-bucket bucketname
```

귀하는 다음 단계에서 애플리케이션을 배포할 때 새 템플릿 파일인 packaged.yaml을 지정합니다.

4단계: 애플리케이션 배포

이제 배포 패키지를 생성했으므로 이 패키지를 사용하여 AWS 클라우드에 애플리케이션을 배포합니다. 그런 다음 애플리케이션을 테스트합니다.

AWS 클라우드에 서버리스 애플리케이션을 배포하려면

- AWS SAM CLI에서는 deploy CLI 명령을 사용하여 템플릿에서 귀하가 정의한 모든 리소스를 배포합니다.

```
sam deploy \
    --template-file packaged.yaml \
    --stack-name sam-app \
    --capabilities CAPABILITY_IAM \
    --region us-east-1
```

  명령에서 --capabilities 파라미터를 사용하면 AWS CloudFormation이 IAM 역할을 생성할 수 있습니다.

  CloudFormation은 템플릿에 정의된 AWS 리소스를 생성합니다. 귀하는 CloudFormation 콘솔에서 이러한 리소스의 이름에 액세스할 수 있습니다.

AWS 클라우드에서 서버리스 애플리케이션을 테스트하려면

1. DynamoDB 콘솔을 엽니다.
2. 방금 귀하가 생성한 테이블에 기록을 삽입합니다.
3. 표의 지표 탭으로 이동하여 모든 CloudWatch 지표 보기를 선택합니다. CloudWatch 콘솔에서 로그를 선택하여 로그 출력을 볼 수 있도록 합니다.

다음 단계

AWS SAM에 대해 계속 알아보고자 한다면 다음 리소스를 참조하세요.

- AWS SAM 예제 애플리케이션 – AWS SAM GitHub 리포지토리에는 다운로드하여 실험해 볼 수 있는 추가 예제 애플리케이션이 포함되어 있습니다.
- 서버리스 랜드 - AWS SAM 애플리케이션을 위한 서버리스 패턴, 코드 예시 및 학습 리소스를 모아 놓은 사이트입니다.

AWS SAM에서 Amazon S3 이벤트 처리

이 예제 애플리케이션을 사용하면 이전 예제에서 학습한 내용을 기반으로 더 복잡한 애플리케이션을 설치합니다. 이 애플리케이션은 Amazon S3 객체 업로드 이벤트 소스에서 간접 호출되는 Lambda 함수로 구성되어 있습니다. 이 연습은 Lambda 함수를 통해 AWS 리소스에 액세스하고 AWS 서비스를 호출하는 방법을 보여줍니다.

이 샘플 서버리스 애플리케이션은 Amazon S3에서 객체 생성 이벤트를 처리합니다. Amazon S3는 버킷에 업로드된 각 이미지에 대해 객체 생성 이벤트를 감지하고 Lambda 함수를 간접 호출합니다. Lambda 함수는 Amazon Rekognition을 간접 호출하여 이미지에 있는 텍스트를 탐지합니다. 그런 다음 Amazon Rekognition에서 반환한 결과를 DynamoDB 테이블에 저장합니다.

> **Note**
>
> 이 예제 애플리케이션에서는 이전 예제와 약간 다른 순서로 단계를 수행합니다. 그 이유는 이 예제를 실행하려면 Lambda 함수를 로컬에서 테스트하기 전에 먼저 AWS 리소스를 생성하고 IAM 권한을 구성해야 하기 때문입니다. 리소스를 생성하고 사용자 대신 권한을 구성하는데 CloudFormation을 활용할 것입니다. 그렇지 않으면 Lambda 함수를 로컬에서 테스트하기 전에 이 작업을 수동으로 수행해야 합니다.
> 이 예제는 더 복잡하므로 이 예제를 실행하기 전에 이전 예제 애플리케이션을 설치하는 데 익숙해야 합니다.

시작하기 전 준비 사항

AWS SAM CLI 설치에서 필요한 설정을 완료했는지 확인하십시오.

1단계: 애플리케이션 초기화

이 단원에서는 AWS SAM 템플릿과 애플리케이션 코드로 구성된 샘플 애플리케이션을 다운로드합니다.

애플리케이션의 초기화

1. AWS SAM CLI 명령 프롬프트에서 다음 명령을 실행합니다.

```
sam init \
--location https://github.com/aws-samples/cookiecutter-aws-sam-s3-rekognition-
dynamodb-python \
--no-input
```

2. 명령을 통해 생성된 디렉터리(`aws_sam_ocr/`)의 내용을 검토합니다.

   - `template.yaml` – Amazon S3 애플리케이션에 필요한 세 가지 AWS 리소스, 즉 Lambda 함수, Amazon S3 버킷, DynamoDB 테이블을 정의합니다. 또한 템플릿은 이 리소스들 간의 매핑 및 권한을 정의합니다.
   - `src/` 디렉터리 – Amazon S3 애플리케이션 코드를 포함합니다.
   - `SampleEvent.json` – 로컬 테스트에 사용되는 샘플 이벤트 소스.

2단계: 애플리케이션 패키징

이 애플리케이션을 로컬에서 테스트하려면 귀하는 먼저 AWS SAM CLI를 사용하여 애플리케이션을 AWS 클라우드에 배포하는 데 사용하는 배포 패키지를 만들어야 합니다. 이 배포를 통해 애플리케이션을 로컬에서 테스트하는 데 필요한 AWS 리소스와 권한이 생성됩니다.

Lambda 배포 패키지 생성하기

1. 패키징된 코드를 저장할 위치에 S3 버킷을 생성합니다. 기존 S3 버킷을 사용하려면 이 단계를 건너뜁니다.

```
aws s3 mb s3://bucketname
```

2. 다음 `package` CLI 명령을 명령 프롬프트에서 실행하여 배포 패키지를 생성합니다.

```
sam package \
    --template-file template.yaml \
    --output-template-file packaged.yaml \
```

```
    --s3-bucket bucketname
```

귀하는 다음 단계에서 애플리케이션을 배포할 때 새 템플릿 파일인 `packaged.yaml`을 지정합니다.

3단계: 애플리케이션 배포

이제 배포 패키지를 생성했으므로 이 패키지를 사용하여 AWS 클라우드에 애플리케이션을 배포합니다. 그런 다음 AWS 클라우드에서 애플리케이션을 호출하여 테스트합니다.

AWS 클라우드에 서버리스 애플리케이션을 배포하려면

- AWS SAM CLI에서 `deploy` 명령을 사용하여 귀하가 템플릿에서 정의한 모든 리소스를 배포합니다.

```
sam deploy \
    --template-file packaged.yaml \
    --stack-name aws-sam-ocr \
    --capabilities CAPABILITY_IAM \
    --region us-east-1
```

명령에서 `--capabilities` 파라미터를 사용하면 CloudFormation에게 IAM 역할을 생성하도록 허용합니다.

CloudFormation은 템플릿에 정의된 AWS 리소스를 생성합니다. 귀하는 CloudFormation 콘솔에서 이러한 리소스의 이름에 액세스할 수 있습니다.

AWS 클라우드에서 서버리스 애플리케이션을 테스트하려면

1. 이 샘플 애플리케이션용으로 생성한 Amazon S3 버킷에 이미지를 업로드합니다.
2. DynamoDB 콘솔을 열고 생성된 테이블을 찾습니다. Amazon Rekognition에서 반환한 결과는 표를 참조하세요.
3. Amazon Rekognition이 업로드한 이미지에서 찾은 텍스트를 포함하는 새 기록이 DynamoDB 테이블에 포함되어 있는지 확인하십시오.

2단계: 애플리케이션 로컬 테스트

애플리케이션을 로컬에서 테스트하려면 먼저 CloudFormation에 의해 생성된 AWS 리소스의 이름을 검색해야 합니다.

- CloudFormation에서 Amazon S3 키 이름 및 버킷 이름을 검색합니다. 객체 키, 버킷 이름, 버킷 ARN의 값을 대체하여 `SampleEvent.json` 파일을 수정합니다.
- DynamoDB 테이블 이름을 검색합니다. 이 이름은 다음 `sam local invoke` 명령에 사용됩니다.

AWS SAM CLI를 사용하여 샘플 Amazon S3 이벤트를 생성하고 Lambda 함수를 간접 호출합니다.

```
TABLE_NAME=Table name obtained from CloudFormation console sam local invoke --event
 SampleEvent.json
```

이 `TABLE_NAME=` 부분은 DynamoDB 테이블 이름을 설정합니다. `--event` 파라미터는 Lambda 함수에 전달할 테스트 이벤트 메시지가 들어 있는 파일을 지정합니다.

이제 Amazon Rekognition에서 반환한 결과를 기반으로 예상 DynamoDB 기록이 생성되었는지 확인할 수 있습니다.

다음 단계

AWS SAM에 대해 계속 알아보고자 한다면 다음 리소스를 참조하세요.

- AWS SAM 예제 애플리케이션 – AWS SAM GitHub 리포지토리에는 다운로드하여 실험해 볼 수 있는 추가 예제 애플리케이션이 포함되어 있습니다.
- 서버리스 랜드 - AWS SAM 애플리케이션을 위한 서버리스 패턴, 코드 예시 및 학습 리소스를 모아 놓은 사이트입니다.

AWS SAM을 사용하여 서버리스 애플리케이션 개발

이 섹션에는 귀하의 AWS SAM 템플릿의 유효성을 검사하고 종속성을 사용하여 애플리케이션을 빌드하는 방법에 대한 항목이 포함되어 있습니다. 또한 Lambda 레이어 작업, 중첩 애플리케이션 사용, API Gateway API에 대한 액세스 제어, Step Functions를 사용한 AWS 리소스 조정, 애플리케이션 코드 서명 등 특정 사용 사례에 AWS SAM을 사용하는 방법에 대한 항목도 포함되어 있습니다. 애플리케이션 개발을 위해 완료해야 하는 세 가지 주요 마일스톤은 다음과 같습니다.

주제

- 에서 애플리케이션 생성 AWS SAM
- AWS SAM을 사용하여 인프라 정의
- AWS SAM을 사용한 애플리케이션 빌드

에서 애플리케이션 생성 AWS SAM

시작하기를 완료하고 AWS Serverless Application Model(AWS SAM) 사용 방법 섹션을 읽은 후에는 개발자 환경에서 AWS SAM 프로젝트를 생성할 수 있습니다. AWS SAM 프로젝트는 서버리스 애플리케이션을 작성하는 출발점 역할을 합니다. `sam init` 명령 옵션 목록은 AWS SAMCLI 섹션을 참조하세요sam init.

AWS Serverless Application Model 명령줄 인터페이스(AWS SAMCLI) `sam init` 명령은 다음으로 구성된 새 서버리스 애플리케이션을 초기화하는 옵션을 제공합니다.

- 인프라 코드를 정의하는 AWS SAM 템플릿입니다.
- 애플리케이션을 구성하는 폴더 구조입니다.
- AWS Lambda 함수에 대한 구성입니다.

AWS SAM 프로젝트를 생성하려면이 섹션의 주제를 참조하세요.

> **Note**
>
> `sam init` 에는 지원되는 런타임(TypeScript, Python, Java)의 내구성 있는 함수를 위한 프로젝트 템플릿이 포함되어 있습니다. 이러한 템플릿은 상태 저장 서버리스 애플리케이션을 빌드하기 위한 스타터 코드와 구성을 제공합니다.

주제

- 새 서버리스 애플리케이션 초기화
- sam init에 대한 옵션
- 문제 해결
- 예제
- 자세히 알아보기
- 다음 단계

새 서버리스 애플리케이션 초기화

CLIAWS SAM을 사용하여 새 서버리스 애플리케이션을 초기화하려면

1. cd를 시작 디렉터리로.
2. 명령줄 프롬프트에 다음 명령을 실행합니다.

```
$ sam init
```

3. AWS SAMCLI는 새 서버리스 애플리케이션을 만들기 위한 대화형 흐름을 안내합니다.

> **Note**
>
> 자습서: AWS SAM에서 Hello World 애플리케이션 배포에 자세히 설명된 대로 이 명령은 서버리스 애플리케이션을 초기화하여 프로젝트 디렉터리를 생성합니다. 이 디렉터리에는 여러 파일과 폴더가 포함되어 있습니다. 가장 중요한 파일은 `template.yaml`입니다. 템플릿입니다 AWS SAM . python 버전은 sam init 명령이 생성한 `template.yaml` 파일에 나와 있는 python 버전과 일치해야 합니다.

시작 템플릿 선택

템플릿은 다음과 같이 구성되어 있습니다.

1. 인프라 코드에 대한 AWS SAM 템플릿입니다.
2. 프로젝트 파일을 구성하는 시작 프로젝트 디렉터리. 예를 들어, 여기에는 다음이 포함될 수 있습니다.
   a. Lambda 함수 코드 및 해당 종속성에 대한 구조.

b. 로컬 테스트용 테스트 이벤트가 포함된 `events` 폴더.

c. 유닛 테스트를 지원하는 `tests` 폴더.

d. 프로젝트 설정을 구성하는 `samconfig.toml` 파일.

e. `ReadMe` 파일 및 기타 기본 시작 프로젝트 파일.

다음은 시작 프로젝트 디렉터리의 예입니다.

```
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
```

사용 가능한 AWS 빠른 시작 템플릿 목록에서 선택하거나 사용자 지정 템플릿 위치를 제공할 수 있습니다.

AWS 빠른 시작 템플릿을 선택하려면

1. 메시지가 표시되면 AWS 빠른 시작 템플릿을 선택합니다.
2. 시작할 AWS 퀵 스타트 템플릿을 선택합니다. 다음은 예제입니다.

```
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
```

사용자 지정 템플릿 위치를 직접 선택하려면

1. 메시지가 표시되면 사용자 지정 템플릿 위치를 선택합니다.

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

2. AWS SAMCLI에 템플릿 위치를 제공하라는 메시지가 표시됩니다.

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

템플릿 .zip 파일 아카이브에 다음 위치 중 하나를 제공합니다.

- GitHub 리포지토리 - GitHub 리포지토리에 있는 .zip 파일의 경로입니다. 파일은 리포지토리의 루트에 있어야 합니다.
- Mercurial 리포지토리 - Mercurial 리포지토리에 있는 .zip 파일의 경로입니다. 파일은 리포지토리의 루트에 있어야 합니다.
- .zip 경로 -.zip 파일의 HTTPS 또는 로컬 경로입니다.

3. AWS SAMCLI는 사용자 지정 템플릿을 사용하여 서버리스 애플리케이션을 초기화합니다.

런타임 선택

AWS 빠른 시작 템플릿을 선택하면 AWS SAMCLI에 Lambda 함수의 런타임을 선택하라는 메시지가 표시됩니다. AWS SAMCLI에 표시되는 옵션 목록은 Lambda에서 기본적으로 지원하는 런타임입니다.

- 런타임은 실행 환경에서 실행되는 언어별 환경을 제공합니다.
- 에 배포되면 AWS 클라우드 Lambda 서비스는 실행 환경에서 함수를 호출합니다.

사용자 지정 런타임과 함께 다른 프로그래밍 언어를 사용할 수 있습니다. 이렇게 하려면 시작 애플리케이션 구조를 수동으로 만들어야 합니다. 그런 다음 `sam init`를 사용하여 사용자 지정 템플릿 위치를 구성하여 애플리케이션을 빠르게 초기화할 수 있습니다.

선택 항목에 따라 AWS SAMCLI는 Lambda 함수 코드 및 종속성에 대한 시작 디렉터리를 생성합니다.

Lambda가 런타임에 대해 여러 종속성 관리자를 지원하는 경우, 선호하는 종속성 관리자를 선택하라는 메시지가 표시됩니다.

패키지 유형 선택

AWS 빠른 시작 템플릿과 런타임을 선택하면 AWS SAMCLI에 패키지 유형을 선택하라는 메시지가 표시됩니다. 패키지 유형은 Lambda 서비스에서 사용하기 위해 Lambda 함수를 배포하는 방법을 결정합니다. 지원되는 두 가지 패키지 유형은 다음과 같습니다.

1. 컨테이너 이미지에는 기본 운영 체제, 런타임, Lambda 익스텐션, 애플리케이션 코드 및 해당 종속 항목이 포함됩니다.
2. .zip 파일 아카이브에는 애플리케이션 코드와 해당 종속 항목이 포함됩니다.

배포 패키지 유형에 대해 자세히 알아보려면 AWS Lambda 개발자 안내서의 Lambda 배포 패키지를 참조하세요.

다음은 컨테이너 이미지로 패키징된 Lambda 함수를 사용하는 애플리케이션의 예제 디렉터리 구조입니다. 는 이미지를 AWS SAMCLI 다운로드하고 함수의 디렉터리`Dockerfile`에를 생성하여 이미지를 지정합니다.

```
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
```

다음은 .zip 파일 아카이브로 패키징된 함수를 사용하는 애플리케이션의 예제 디렉터리 구조입니다.

```
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
```

AWS X-Ray 추적 구성

AWS X-Ray 추적을 활성화하도록 선택할 수 있습니다. 자세한 내용은 AWS X-Ray 개발자 안내서의 [What is AWS X-Ray?](#)를 참조하세요.

활성화하면가 AWS SAM 템플릿을 AWS SAMCLI 구성합니다. 다음은 예제입니다.

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

Amazon CloudWatch Application Insights를 사용한 모니터링 설정

Amazon CloudWatch Application Insights를 사용하여 모니터링을 활성화하도록 선택할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용 설명서의 [Amazon CloudWatch Application Insights](https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/cloudwatch-application-insights.html)를 참조하세요.

활성화하면가 AWS SAM 템플릿을 AWS SAMCLI 구성합니다. 다음은 예제입니다.

```
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
```

애플리케이션 이름을 지정합니다.

애플리케이션의 이름을 제공합니다. AWS SAMCLI는 이 이름을 사용하여 애플리케이션의 최상위 폴더를 만듭니다.

sam init에 대한 옵션

다음은 `sam init` 명령과 함께 사용할 수 있는 몇 가지 기본 옵션입니다. 콘텐츠 옵션 목록은 sam init 섹션을 참조하세요.

사용자 지정 템플릿 위치를 사용하여 애플리케이션 초기화

`--location` 옵션을 사용하고 지원되는 사용자 지정 템플릿 위치를 제공합니다. 다음은 예제입니다.

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

대화형 흐름 없이 애플리케이션 초기화

`--no-interactive` 옵션을 사용하고 명령줄에서 구성 선택 사항을 제공하여 대화형 흐름을 건너뜁니다. 다음은 예제입니다.

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

문제 해결

문제를 해결하려면 단원을 AWS SAMCLI참조하십시오AWS SAMCLI 문제 해결.

예제

Hello World AWS Starter 템플릿을 사용하여 새 서버리스 애플리케이션 초기화

이 예제는 자습서: Hello World 애플리케이션 배포의 1단계: Hello World 샘플 애플리케이션 초기화 섹션을 참조하세요.

사용자 지정 템플릿 위치를 사용하여 새 서버리스 애플리케이션 초기화

다음은 사용자 지정 템플릿에 GitHub 위치를 제공하는 예제입니다.

```
$ 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
```

다음은 로컬 파일 경로의 예제입니다:

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

다음은 HTTPS로 도달할 수 있는 경로의 예입니다.

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

자세히 알아보기

`sam init` 명령 사용에 대한 자세한 내용은 다음을 참조하세요.

- 학습 AWS SAM: sam init –의 Serverless Land "학습 AWS SAM" 시리즈YouTube.
- AWS SAM CLI와 함께 사용할 서버리스 애플리케이션 구조화(SAM S2E7 세션) – YouTube의 AWS SAM 시리즈 세션

다음 단계

프로젝트를 생성했으므로 이제 애플리케이션 작성을 시작할 준비가 AWS SAM 되었습니다. 이를 위해 완료해야 하는 작업에 대한 자세한 지침은 AWS SAM을 사용하여 인프라 정의 섹션을 참조하세요.

AWS SAM을 사용하여 인프라 정의

이제 프로젝트를 생성했으므로 AWS SAM을 사용하여 애플리케이션 인프라를 정의할 준비가 되었습니다. 이 작업을 수행하려면 AWS SAM 프로젝트에서 `template.yaml` 파일에 해당하는 AWS SAM 템플릿을 구성하여 애플리케이션의 리소스 및 속성을 정의합니다.

이 섹션의 주제에서는 AWS SAM 템플릿(`template.yaml` 파일)에서 인프라를 정의하는 것에 대해 다룹니다. 또한 Lambda 계층 작업, 중첩 애플리케이션 사용, API Gateway API에 대한 액세스 제어, Step Functions를 사용한 AWS 리소스 오케스트레이션, 애플리케이션 코드 서명, AWS SAM 템플릿 확인 등과 같은 구체적인 사용 사례에서 리소스를 정의하는 것에 대한 주제도 다룹니다.

주제

- AWS SAM 템플릿에서 애플리케이션 리소스 정의
- AWS SAM 템플릿에서 리소스 액세스 설정 및 관리
- AWS SAM 템플릿으로 API 액세스 제어

- AWS SAM에서 Lambda 계층을 사용하여 효율성 향상
- AWS SAM에서 중첩 애플리케이션을 사용하여 코드 및 리소스 재사용
- AWS SAM에서 EventBridge 스케줄러를 사용하여 시간 기반 이벤트 관리
- AWS Step Functions를 사용하여 AWS SAM 리소스를 오케스트레이션합니다.
- AWS SAM 애플리케이션에 대한 코드 서명 설정
- AWS SAM 템플릿 파일 검증

AWS SAM 템플릿에서 애플리케이션 리소스 정의

AWS SAM 템플릿의 `Resources` 섹션에서 서버리스 애플리케이션이 사용하는 AWS 리소스를 정의합니다. 리소스를 정의할 때 리소스의 용도, 리소스가 다른 리소스와 상호 작용하는 방식과 리소스에 액세스할 수 있는 방법(즉, 리소스의 권한)을 식별합니다.

AWS SAM 템플릿의 `Resources` 섹션에는 CloudFormation 리소스와 AWS SAM 리소스의 조합이 포함될 수 있습니다. 또한 다음 리소스에 대해 AWS SAM의 단축 구문을 사용할 수 있습니다.

AWS SAM 단축 구문	관련 AWS 리소스로 수행하는 작업
AWS::Serverless::Api	HTTPS 엔드포인트를 통해 호출할 수 있는 API Gateway 리소스 및 메서드 컬렉션을 생성합니다.
AWS::Serverless::Application	AWS Serverless Application Repository로부터, 또는 Amazon S3 버킷으로부터 서버리스 애플리케이션을 중첩 애플리케이션으로 내장합니다.
AWS::Serverless::Connector	두 리소스 간의 권한을 구성합니다. 커넥터에 대한 소개는 AWS SAM 커넥터를 사용한 리소스 권한 관리 섹션을 참조하세요.
AWS::Serverless::Function	AWS Lambda 함수, AWS Identity and Access Management(IAM) 실행 역할, 함수를 트리거하는 이벤트 소스 매핑 등을 생성합니다.
AWS::Serverless::GraphQLApi	서버리스 애플리케이션용 AWS AppSync GraphQL API를 생성하고 구성합니다.

AWS SAM 단축 구문	관련 AWS 리소스로 수행하는 작업
AWS::Serverless::HttpApi	Amazon API Gateway HTTP API를 생성함으로써 귀하는 REST API보다 지연 시간이 짧고 비용이 저렴한 RESTful API를 생성할 수 있습니다.
AWS::Serverless::LayerVersion	Lambda 함수에 필요한 라이브러리 또는 런타임 코드를 포함하는 Lambda LayerVersion을 생성합니다.
AWS::Serverless::SimpleTable	단일 속성 프라이머리 키를 사용하여 DynamoDB 테이블을 생성합니다.
AWS::Serverless::StateMachine	AWS Step Functions 함수 및 기타 AWS Lambda 리소스를 오케스트레이션하여 복잡하고 강력한 워크플로를 구성하는 데 사용할 수 있는 AWS 상태 기기를 만듭니다.

위의 리소스도 AWS SAM 리소스 및 속성에 나열되어 있습니다.

모든 AWS 리소스 및 속성 유형 CloudFormation과 AWS SAM 지원에 대한 참조 정보는 AWS사용자 가이드의 AWS CloudFormation 리소스 및 속성 유형 참조를 참조하세요.

AWS SAM 템플릿에서 리소스 액세스 설정 및 관리

AWS 리소스가 서로 상호 작용하려면 리소스 간에 적절한 액세스 및 권한이 구성되어 있어야 합니다. 이렇게 하려면 안전한 방식으로 상호 작용을 수행할 수 있도록 AWS Identity and Access Management(IAM) 사용자, 역할 및 정책을 구성해야 합니다.

이 섹션의 주제는 모두 템플릿에 정의된 리소스에 대한 액세스를 설정하는 것과 관련이 있습니다. 이 섹션은 일반적인 모범 사례로 시작합니다. 다음 두 주제에서는 서버리스 애플리케이션에서 참조되는 리소스 간의 액세스 및 권한을 설정하기 위해 사용하는 두 가지 옵션인 AWS SAM 커넥터와 AWS SAM 정책 템플릿을 검토합니다. 마지막 주제에서는 CloudFormation에서 사용자 관리에 사용하는 것과 동일한 메커니즘을 사용하여 사용자 액세스를 관리하기 위한 세부 정보를 제공합니다.

자세한 내용은 AWS CloudFormation 사용 설명서의 AWS Identity and Access Management를 사용한 액세스 제어를 참조하세요.

AWS Serverless Application Model(AWS SAM)는 서버리스 애플리케이션의 액세스 및 권한 관리를 단순화하는 두 가지 옵션을 제공합니다.

1. AWS SAM 커넥터
2. AWS SAM 정책 템플릿

AWS SAM 커넥터

커넥터는 두 리소스 간에 권한을 프로비저닝하는 방법입니다. AWS SAM 템플릿에서 서로 상호 작용하는 방식을 설명하여 이를 수행할 수 있습니다. `Connectors` 리소스 속성이나 `AWS::Serverless::Connector` 리소스 유형을 사용하여 정의할 수 있습니다. 커넥터는 AWS 리소스 조합 간에 데이터 및 이벤트의 `Read` 및 `Write` 액세스 권한을 프로비저닝할 수 있도록 지원합니다. AWS SAM 커넥터에 대한 자세한 내용은 AWS SAM 커넥터를 사용한 리소스 권한 관리 섹션을 참조하세요.

AWS SAM 정책 템플릿

AWS SAM 정책 템플릿은 AWS Lambda 함수, AWS Step Functions 상태 시스템, 상호 작용하는 리소스 간의 액세스 및 권한을 관리하기 위해 AWS SAM 템플릿에 추가할 수 있는 사전 정의된 권한 집합입니다. AWS SAM의 사용자 지정 템플릿에 대한 자세한 내용은 AWS SAM 정책 템플릿 섹션을 참조하세요.

AWS CloudFormation 메커니즘

CloudFormation 메커니즘에는 AWS 리소스 간 권한을 관리하기 위한 IAM 사용자, 역할 및 정책 구성이 포함됩니다. 자세한 내용은 CloudFormation 메커니즘을 사용하여 AWS SAM 권한 관리 섹션을 참조하세요.

모범 사례

서버리스 애플리케이션 전체에서 여러 방법을 사용하여 리소스 간에 권한을 구성할 수 있습니다. 따라서 각 시나리오에 가장 적합한 옵션을 선택하고 애플리케이션 전체에서 여러 옵션을 함께 사용할 수 있습니다. 가장 적합한 옵션을 선택할 때는 다음과 같은 몇 가지 고려할 사항이 있습니다.

- AWS SAM 커넥터와 정책 템플릿 모두 AWS 리소스 간의 안전한 상호 작용을 촉진하는 데 필요한 IAM 전문 지식을 줄여줍니다. 지원되는 경우 커넥터와 정책 템플릿을 사용하세요.
- AWS SAM 커넥터는 AWS SAM 템플릿에서 권한을 정의할 수 있는 간단하고 직관적인 단축 구문을 제공하며, IAM 전문 지식이 거의 필요하지 않습니다. AWS SAM 커넥터와 정책 템플릿이 모두 지원되는 경우 AWS SAM 커넥터를 사용하세요.

- AWS SAM 커넥터는 지원되는 AWS SAM 소스 및 대상 리소스 간에 데이터 및 이벤트의 `Read` 및 `Write` 액세스 권한을 프로비저닝할 수 있습니다. 지원되는 리소스 유형 목록은 AWS SAM 커넥터 참조 섹션을 참조하세요. 지원되는 경우 AWS SAM 커넥터를 사용하세요.
- AWS SAM 정책 템플릿은 Lambda 함수, Step Functions 상태 시스템 및 상호 작용하는 AWS 리소스 간의 권한으로 제한되지만, 정책 템플릿은 모든 CRUD 작업을 지원합니다. 지원되는 경우, 시나리오에 맞는 AWS SAM 정책 템플릿이 제공된다면 AWS SAM 정책 템플릿을 사용하세요. 사용 가능한 정책 템플릿 목록은 AWS SAM 정책 템플릿 섹션을 참조하세요.
- 다른 모든 시나리오의 경우 또는 세분성이 필요한 경우에는 CloudFormation 메커니즘을 사용하세요.

AWS SAM 커넥터를 사용한 리소스 권한 관리

커넥터는 AWS Serverless Application Model로 식별되는 AWS SAM(`AWS::Serverless::Connector`) 추상 리소스 유형으로, 서버리스 애플리케이션 리소스 간에 간단하고 범위가 적절한 권한을 제공합니다.

AWS SAM 커넥터의 이점

커넥터는 리소스 간에 적절한 액세스 정책을 자동으로 구성하므로 AWS 권한 부여 기능, 정책 언어 및 서비스별 보안 설정에 대한 전문 지식이 없어도 서버리스 애플리케이션을 작성하고 애플리케이션 아키텍처에 집중할 수 있습니다. 따라서 커넥터는 서버리스 개발을 처음 접하는 개발자나 개발 속도를 높이려는 노련한 개발자에게 큰 도움이 됩니다.

AWS SAM 커넥터 사용

`Connectors` 리소스 속성을 소스 리소스에 임베드하여 확인합니다. 그런 다음 대상 리소스를 정의하고 이러한 리소스들 간에 데이터나 이벤트가 어떻게 이동해야 하는지 설명하십시오. AWS SAM 그런 다음 요구되는 상호 작용을 촉진하는 데 필요한 액세스 정책을 작성합니다.

다음은 이 리소스 속성이 작성되는 방법에 대한 간략한 설명입니다.

```
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>
  ...
```

커넥터 작동 방식

> **Note**
>
> 이 섹션에서는 커넥터가 보이지 않는 곳에서 필요한 리소스를 제공하는 방법을 설명합니다. 이는 커넥터를 사용할 때 자동으로 발생합니다.

먼저, 탑재된 `Connectors` 리소스 속성이 `AWS::Serverless::Connector`리소스 유형으로 변환됩니다. 논리 ID는 *`<source-resource-logical-id><embedded-connector-logical-id>`*로 자동 생성됩니다.

예를 들어 탑재된 커넥터는 다음과 같습니다.

```
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
```

그러면 다음과 같은 `AWS::Serverless::Connector` 리소스가 생성됩니다.

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

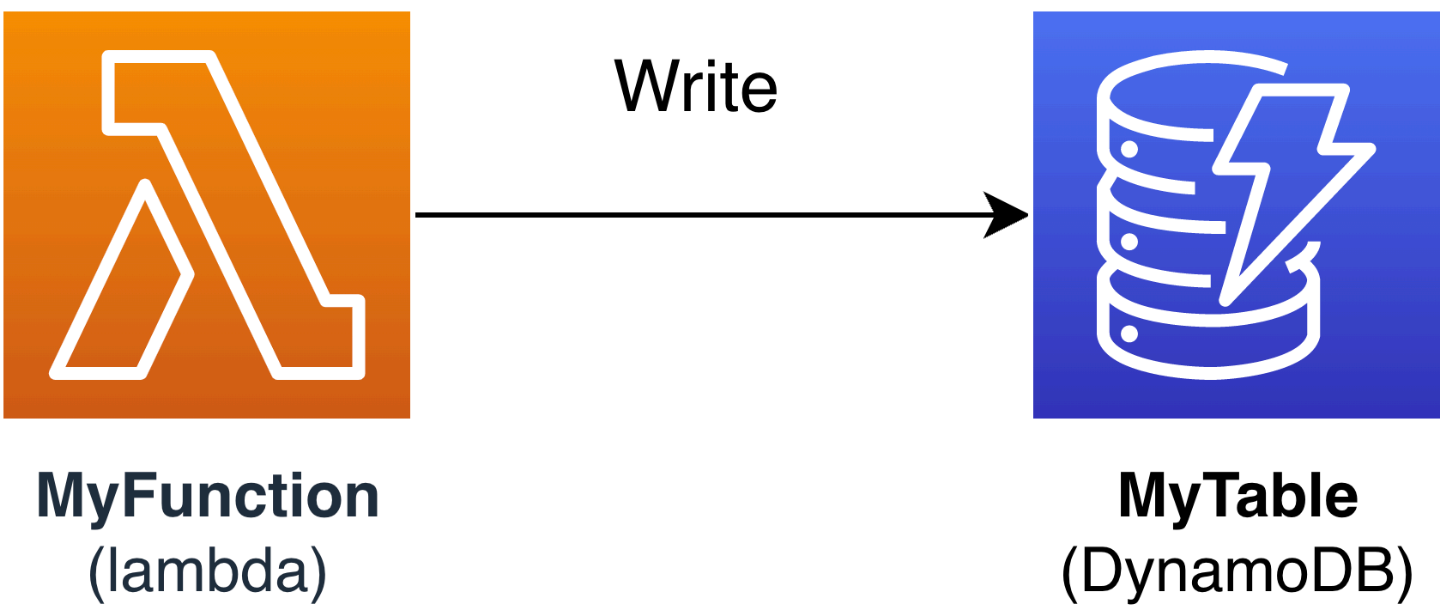
```
Resources:
  ...
  MyFunctionMyConn:
    Type: AWS::Serverless::Connector
    Properties:
      Source:
        Id: MyFunction
      Destination:
        Id: MyTable
      Permissions:
        - Read
        - Write
```

> **Note**
>
> 이 명령문을 사용하여 AWS SAM 템플릿에서 커넥터를 정의할 수도 있습니다. 소스 리소스가 커넥터와 별도의 템플릿에 정의된 경우 이 방법을 사용하는 것이 좋습니다.

그런 다음 이 연결에 필요한 액세스 정책이 자동으로 구성됩니다. 커넥터에서 생성된 리소스에 대한 자세한 내용은 CloudFormation을 지정한 경우 생성되는 AWS::Serverless::Connector 리소스 을 참조하세요.

커넥터의 예

다음 예제에서는 커넥터를 사용하여 AWS Lambda 함수의 데이터를 Amazon DynamoDB 테이블에 기록하는 방법을 보여줍니다.



```
Transform: AWS::Serverless-2016-10-31
Resources:
  MyTable:
    Type: AWS::Serverless::SimpleTable
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      MyConn:
        Properties:
          Destination:
            Id: MyTable
          Permissions:
            - Write
    Properties:
      Runtime: nodejs16.x
      Handler: index.handler
      InlineCode: |
        const AWS = require("aws-sdk");
        const docClient = new AWS.DynamoDB.DocumentClient();
        exports.handler = async (event, context) => {
          await docClient.put({
            TableName: process.env.TABLE_NAME,
            Item: {
              id: context.awsRequestId,
              event: JSON.stringify(event)
            }
          }).promise();
        }
      Environment:
        Variables:
          TABLE_NAME: !Ref MyTable
```

`Connectors` 리소스 속성은 Lambda 함수 소스 리소스 내에 내장되어 있습니다. DynamoDB 테이블은 `Id`속성을 사용하는 대상 리소스로 정의됩니다. 커넥터는 이 두 리소스 간의 `Write` 권한을 프로비저닝합니다.

AWS SAM 템플릿을 CloudFormation에 배포하면 AWS SAM은 이러한 연결이 작동하는 데 필요한 액세스 정책을 자동으로 작성합니다.

소스 및 대상 리소스 간 지원되는 연결

커넥터는 선별된 소스의 조합과 대상 리소스 연결 간 `Read` 및 `Write`데이터 및 이벤트 권한 유형을 지원합니다. 예를 들어 커넥터는 `Write` 소스 리소스와 `AWS::ApiGateway::RestApi` 대상 리소스 간의 `AWS::Lambda::Function` 연결을 지원합니다.

지원되는 속성의 조합을 사용하여 소스 및 대상 리소스를 정의할 수 있습니다. 속성 요구 사항은 생성하는 연결 방식과 리소스가 정의된 위치에 따라 달라집니다.

> **Note**
>
> 커넥터는 지원되는 서버리스 리소스 유형과 비서버리스 리소스 유형 간에 권한을 제공할 수 있습니다.

지원되는 리소스 연결 및 속성 요구 사항 목록은 [커넥터에 지원되는 소스 및 대상 리소스 유형](#)을 참조하세요.

AWS SAM에서 읽기 및 쓰기 권한 정의

AWS SAM, `Read` 및 `Write` 권한은 단일 커넥터 내에서 프로비저닝할 수 있습니다.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Lambda::Function
    Connectors:
      MyTableConn:
        Properties:
          Destination:
            Id: MyTable
          Permissions:
            - Read
            - Write
  MyTable:
    Type: AWS::DynamoDB::Table
```

커넥터 사용에 대한 자세한 내용은 [AWS SAM 커넥터 참조](#) 섹션을 참조하세요.

AWS SAM에서 지원되는 다른 속성을 사용하여 리소스 정의

소스 및 대상 리소스 모두에 대해 동일한 템플릿 내에 정의된 경우 Id 속성을 사용합니다. 선택적으로 Qualifier을 추가하여 정의된 리소스의 범위를 좁힐 수 있습니다. 리소스가 동일한 템플릿 내에 있지 않은 경우 지원되는 속성을 조합하여 사용합니다.

- 소스 및 대상 리소스에 지원되는 속성 조합 목록은 [커넥터에 지원되는 소스 및 대상 리소스 유형]을 참조하세요.
- 커넥터와 함께 사용할 수 있는 속성에 대한 설명은 [AWS::Serverless::Connector]을 참조하세요.

Id이 아닌 속성을 사용하여 소스 리소스를 정의하는 경우 SourceReference속성을 사용하십시오.

```
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>
```

다음은 Qualifier를 사용하여 Amazon API Gateway 리소스의 범위를 좁히는 예제입니다.

```
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
  ...
```

다음은 지원되는 Arn와 Type의 조합을 사용하여 다른 템플릿에서 대상 리소스를 정의하는 예제입니다.

```
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
  ...
```

커넥터 사용에 대한 자세한 내용은 AWS SAM 커넥터 참조 섹션을 참조하세요.

AWS SAM의 단일 소스에서 여러 커넥터 생성

소스 리소스 내에서 각각 대상 리소스가 다른 여러 커넥터를 정의할 수 있습니다.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      BucketConn:
        Properties:
          Destination:
            Id: amzn-s3-demo-bucket
          Permissions:
```

```
            - Read
            - Write
      SQSConn:
        Properties:
          Destination:
            Id: MyQueue
          Permissions:
            - Read
            - Write
      TableConn:
        Properties:
          Destination:
            Id: MyTable
          Permissions:
            - Read
            - Write
      TableConnWithTableArn:
        Properties:
          Destination:
            Type: AWS::DynamoDB::Table
            Arn: !GetAtt MyTable.Arn
          Permissions:
            - Read
            - Write
...
```

커넥터 사용에 대한 자세한 내용은 AWS SAM 커넥터 참조 섹션을 참조하세요.

AWS SAM에서 다중 대상 커넥터 생성

소스 리소스 내에서 여러 대상 리소스가 있는 단일 커넥터를 정의할 수 있습니다. Amazon Simple Storage Service(S3) 버킷과 DynamoDB 테이블에 연결된 Lambda 함수 소스 리소스의 예는 다음과 같습니다.

```
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
```

커넥터 사용에 대한 자세한 내용은 AWS SAM 커넥터 참조 섹션을 참조하세요.

AWS SAM에서 커넥터를 사용한 리소스 속성 정의

리소스에 대해서 리소스 속성을 정의하여 추가 행동 및 관계를 지정할 수 있습니다. 리소스 속성에 대한 자세한 내용은 사용자 가이드의 AWS CloudFormation리소스 속성 참조를 참조하세요.

커넥터 속성과 동일한 수준에서 리소스 속성을 정의하여 내장 커넥터에 리소스 속성을 추가할 수 있습니다. 배포 시 AWS SAM 템플릿이 변환되면 속성이 생성된 리소스에 이전됩니다.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  MyFunction:
    Type: AWS::Serverless::Function
    Connectors:
      MyConn:
        DeletionPolicy: Retain
        DependsOn: AnotherFunction
        Properties:
          ...
```

커넥터 사용에 대한 자세한 내용은 AWS SAM 커넥터 참조 섹션을 참조하세요.

자세히 알아보기

AWS SAM 커넥터 사용에 대한 자세한 내용은 다음 주제를 참조하세요.

- AWS::Serverless::Connector
- AWS SAM에서 읽기 및 쓰기 권한 정의
- AWS SAM에서 지원되는 다른 속성을 사용하여 리소스 정의

- AWS SAM의 단일 소스에서 여러 커넥터 생성
- AWS SAM에서 다중 대상 커넥터 생성
- AWS SAM에서 읽기 및 쓰기 권한 정의
- AWS SAM에서 커넥터를 사용한 리소스 속성 정의

피드백 제공

커넥터에 대한 피드백을 제공하려면 서버리스 애플리케이션 모델AWSGitHub 리포지토리에 새 문제를 제출하십시오.

AWS SAM 정책 템플릿

AWS Serverless Application Model(AWS SAM)를 사용하면 정책 템플릿 목록에서 선택하여 Lambda 함수 및 AWS Step Functions 상태 시스템의 권한 범위를 애플리케이션에서 사용하는 리소스로 지정할 수 있습니다.

정책 템플릿을 사용하는 AWS Serverless Application Repository의 AWS SAM 애플리케이션의 경우 AWS Serverless Application Repository에서 애플리케이션을 배포할 때 특별한 고객 승인이 필요하지 않습니다.

새 정책 템플릿 추가를 요청하려면 다음을 수행합니다.

1. AWS SAM GitHub 프로젝트의 `develop` 브랜치에 있는 policy_templates.json 소스 파일에 대해 풀 요청을 제출합니다. 소스 파일은 GitHub 웹사이트의 policy_templates.json에서 찾을 수 있습니다.
2. AWS SAM GitHub 프로젝트에 풀 요청의 이유와 요청 링크가 포함된 문제를 제출합니다. 이 링크를 사용하여 새 문제를 제출합니다. AWS Serverless Application Model: 문제.

명령문

AWS SAM 템플릿 파일에 지정하는 모든 정책 템플릿에는 항상 정책 템플릿의 자리 표시자 값이 포함된 객체를 지정해야 합니다. 정책 템플릿에 자리 표시자 값이 필요하지 않은 경우 빈 객체를 지정해야 합니다.

YAML

```
MyFunction:
  Type: AWS::Serverless::Function
  Properties:
    Policies:
```

```
    - PolicyTemplateName1:        # Policy template with placeholder value
        Key1: Value1
    - PolicyTemplateName2: {}     # Policy template with no placeholder value
```

> **Note**
>
> 일반 IAM 정책을 설정했거나 Lambda를 통해 정책을 관리한 경우 빈 객체를 사용하지 않고 정책 템플릿을 설정할 수 있습니다.

예시

예 1: 자리 표시자 값이 있는 정책 템플릿

다음 예는 SQSPollerPolicy 정책 템플릿에서 QueueName를 리소스로 예상한다는 것을 보여줍니다. AWS SAM 템플릿은 'MyQueue' Amazon SQS 대기열의 이름을 검색합니다. 이 대기열은 동일한 애플리케이션에서 생성하거나 애플리케이션에 대한 파라미터로 요청할 수 있습니다.

```
MyFunction:
  Type: 'AWS::Serverless::Function'
  Properties:
    CodeUri: ${codeuri}
    Handler: hello.handler
    Runtime: python2.7
    Policies:
      - SQSPollerPolicy:
          QueueName:
            !GetAtt MyQueue.QueueName
```

예 2: 자리 표시자 값이 없는 정책 템플릿

다음 예에는 자리 표시자 값이 없는 CloudWatchPutMetricPolicy 정책 템플릿이 포함되어 있습니다.

> **Note**
>
> 자리 표시자 값이 없더라도 빈 객체를 지정해야 합니다. 그렇지 않으면 오류가 발생합니다.

```
MyFunction:
  Type: 'AWS::Serverless::Function'
```

```
  Properties:
    CodeUri: ${codeuri}
    Handler: hello.handler
    Runtime: python2.7
    Policies:
      - CloudWatchPutMetricPolicy: {}
```

예 3: 자리 표시자 값과 일반 IAM 정책이 포함된 정책 템플릿

다음 예제에는 AmazonSQSFullAcess 정책 및 DynamoDBCrudPolicy 정책 템플릿이 포함되어 있습니다. AmazonSQSFullAccess 정책은 AWS SAM 정책이 아닌 IAM 정책이므로 정책이 CloudFormation에 직접 전달되므로 빈 객체를 지정할 필요가 없습니다.

```
MyFunction:
  Type: 'AWS::Serverless::Function'
  Properties:
    CodeUri: ${codeuri}
    Handler: hello.handler
    Runtime: python2.7
    Policies:
      - AmazonSQSFullAccess // IAM policy could be set without passing an empty object
      - DynamoDBCrudPolicy: // SAM specific policy, has a defined structure
          TableName:
            !Ref SampleTable
```

정책 템플릿 테이블

다음은 사용 가능한 정책 템플릿의 테이블입니다.

정책 템플릿	설명		
AcmGetCertificatePolicy	AWS Certificate Manager에서 인증서를 읽을 수 있는 권한을 부여합니다.		
AMIDescribePolicy	Amazon Machine Image(AMI)를 설명할 수 있는 권한을 부여합니다.		
AthenaQueryPolicy	Athena 쿼리를 실행할 권한을 부여합니다.		

정책 템플릿	설명
AWSSecretsManagerGetSecretValuePolicy	지정된 AWS Secrets Manager 보안 암호의 보안 암호 값을 가져올 수 있는 권한을 부여합니다.
AWSSecretsManagerRotationPolicy	AWS Secrets Manager에서 보안 암호를 교체할 수 있는 권한을 부여합니다.
CloudFormationDescribeStacksPolicy	CloudFormation 스택을 설명할 수 있는 권한을 부여합니다.
CloudWatchDashboardPolicy	CloudWatch 대시보드에서 작동할 지표를 배치할 권한을 부여합니다.
CloudWatchDescribeAlarmHistoryPolicy	CloudWatch 경보 기록을 설명할 수 있는 권한을 부여합니다.
CloudWatchPutMetricPolicy	CloudWatch에 지표를 전송할 수 있는 권한을 부여합니다.
CodeCommitCrudPolicy	특정 CodeCommit 리포지토리 내에서 객체를 생성/읽기/업데이트/삭제할 수 있는 권한을 부여합니다.
CodeCommitReadPolicy	특정 CodeCommit 리포지토리 내에서 객체를 읽을 수 있는 권한을 부여합니다.

정책 템플릿	설명
CodePipelineLambdaExecutionPolicy	CodePipeline에서 간접 호출한 Lambda 함수가 작업 상태를 보고하도록 권한을 부여합니다.
CodePipelineReadOnlyPolicy	CodePipeline 파이프라인의 세부 정보를 가져올 수 있는 읽기 권한을 부여합니다.
ComprehendBasicAccessPolicy	엔터티, 핵심 문구, 언어 및 감정을 탐지할 수 있는 권한을 부여합니다.
CostExplorerReadOnlyPolicy	청구 내역에 대한 읽기 전용 Cost Explorer API에 읽기 전용 권한을 부여합니다.
DynamoDBBackupFullAccessPolicy	테이블에 대한 DynamoDB 온디맨드 백업에 읽기 및 쓰기 권한을 부여합니다.
DynamoDBCrudPolicy	Amazon DynamoDB 테이블에 생성, 읽기, 업데이트 및 삭제 권한을 부여합니다.
DynamoDBReadPolicy	DynamoDB 테이블에 읽기 전용 권한 부여
DynamoDBReconfigurePolicy	DynamoDB 테이블을 재구성할 수 있는 권한을 부여합니다.
DynamoDBRestoreFromBackupPolicy	백업에서 테이블을 복원할 수 있는 권한을 부여합니다.

정책 템플릿	설명		
DynamoDBStreamReadPolicy	DynamoDB 스트림과 레코드를 설명하고 읽을 수 있는 권한을 부여합니다.		
DynamoDBWritePolicy	DynamoDB 테이블에 쓰기 전용 권한 부여		
EC2CopyImagePolicy	Amazon EC2 이미지를 복사할 수 있는 권한을 부여합니다.		
EC2DescribePolicy	Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스를 설명할 수 있는 권한을 부여합니다.		
EcsRunTaskPolicy	태스크 정의에 대해 새 태스크를 시작할 수 있는 권한을 부여합니다.		
EFSWriteAccessPolicy	쓰기 액세스 권한이 있는 Amazon EFS 파일 시스템을 마운트할 수 있는 권한을 부여합니다.		
EKSDescribePolicy	Amazon EKS 클러스터를 설명하거나 나열할 수 있는 권한을 부여합니다.		
ElasticMapReduceAddJobFlowStepsPolicy	실행 중인 클러스터에 새 단계를 추가할 수 있는 권한을 부여합니다.		
ElasticMapReduceCancelStepsPolicy	실행 중인 클러스터에서 대기 중 단계를 취소할 수 있는 권한을 부여합니다.		

정책 템플릿	설명
ElasticMapReduceModifyInstanceFleetPolicy	클러스터 내 인스턴스 플릿의 세부 정보를 나열하고 용량을 수정할 수 있는 권한을 부여합니다.
ElasticMapReduceModifyInstanceGroupsPolicy	클러스터 내 인스턴스 그룹의 세부 정보를 나열하고 설정을 수정할 수 있는 권한을 부여합니다.
ElasticMapReduceSetTerminationProtectionPolicy	클러스터에 대한 종료 보호를 설정할 수 있는 권한을 부여합니다.
ElasticMapReduceTerminateJobFlowsPolicy	클러스터를 종료할 수 있는 권한을 부여합니다.
ElasticsearchHttpPostPolicy	Amazon OpenSearch Service에 POST 권한을 부여합니다.
EventBridgePutEventsPolicy	EventBridge로 이벤트를 보낼 수 있는 권한을 부여합니다.
FilterLogEventsPolicy	지정된 로그 그룹에서 CloudWatch Logs 이벤트를 필터링할 수 있는 권한을 부여합니다.
FirehoseCrudPolicy	Firehose 전송 스트림을 생성, 쓰기, 업데이트 및 삭제할 수 있는 권한을 부여합니다.

정책 템플릿	설명		
FirehoseWritePolicy	Firehose 전송 스트림에 쓸 수 있는 권한을 부여합니다.		
KinesisCrudPolicy	Amazon Kinesis 스트림을 생성, 게시 및 삭제할 수 있는 권한을 부여합니다.		
KinesisStreamReadPolicy	Amazon Kinesis 스트림을 나열하고 읽을 수 있는 권한을 부여합니다.		
KMSDecryptPolicy	AWS Key Management Service(AWS KMS) 키로 복호화할 수 있는 권한을 부여합니다.		
KMSEncryptPolicy	AWS Key Management Service(AWS KMS) 키로 암호화할 수 있는 권한을 부여합니다.		
LambdaInvokePolicy	AWS Lambda 함수, 별칭 또는 버전을 간접 호출할 수 있는 권한을 부여합니다.		
MobileAnalyticsWriteOnlyAccessPolicy	모든 애플리케이션 리소스에 이벤트 데이터를 넣을 수 있는 쓰기 전용 권한을 부여합니다.		
OrganizationsListAccountsPolicy	하위 계정 이름 및 ID를 나열할 수 있는 읽기 전용 권한을 부여합니다.		
PinpointEndpointAccessPolicy	Amazon Pinpoint 애플리케이션의 엔드포인트를 가져오고 업데이트할 수 있는 권한을 부여합니다.		
PollyFullAccessPolicy	Amazon Polly 어휘 리소스에 대한 전체 액세스 권한을 부여합니다.		

정책 템플릿	설명		
RekognitionDetectOnlyPolicy	얼굴, 레이블, 텍스트를 감지할 수 있는 권한을 부여합니다.		
RekognitionFacesManagementPolicy	Amazon Rekognition 컬렉션에서 얼굴을 추가, 삭제 및 검색할 수 있는 권한을 부여합니다.		
RekognitionFacesPolicy	얼굴과 레이블을 비교하고 감지할 수 있는 권한을 부여합니다.		
RekognitionLabelsPolicy	객체 및 중재 레이블을 탐지할 수 있는 권한을 부여합니다.		
RekognitionNoDataAccessPolicy	얼굴과 레이블을 비교하고 감지할 수 있는 권한을 부여합니다.		
RekognitionReadPolicy	얼굴을 나열하고 검색할 수 있는 권한을 부여합니다.		
RekognitionWriteOnlyAccessPolicy	얼굴을 모으고 인덱싱할 수 있는 권한을 부여합니다.		
Route53ChangeResourceRecordSetsPolicy	Route 53에서 리소스 레코드 세트를 변경할 수 있는 권한을 부여합니다.		
S3CrudPolicy	Amazon S3 버킷의 객체에 대해 작업을 수행할 수 있는 생성, 읽기, 업데이트 및 삭제 권한을 부여합니다.		

정책 템플릿	설명		
S3FullAccessPolicy	Amazon S3 버킷의 객체에 대해 작업을 수행할 수 있는 전체 액세스 권한을 부여합니다.		
S3ReadPolicy	Amazon Simple Storage Service(S3) 버킷에 있는 객체를 읽을 수 있는 읽기 전용 권한을 부여합니다.		
S3WritePolicy	Amazon S3 버킷에 객체를 쓸 수 있는 쓰기 권한을 부여합니다.		
SageMakerCreateEndpointConfigPolicy	SageMaker AI에서 엔드포인트 구성을 생성할 수 있는 권한을 부여합니다.		
SageMakerCreateEndpointPolicy	SageMaker AI에서 엔드포인트를 생성할 수 있는 권한을 부여합니다.		
ServerlessRepoReadWriteAccessPolicy	AWS Serverless Application Repository 서비스에서 애플리케이션을 생성하고 나열할 수 있는 권한을 부여합니다.		
SESBulkTemplatedCrudPolicy	이메일, 템플릿 이메일, 템플릿 대량 이메일을 보내고 자격 증명을 확인할 수 있는 권한을 부여합니다.		
SESBulkTemplatedCrudPolicy_v2	Amazon SES 이메일, 템플릿 기반 이메일, 템플릿 형식의 대량 이메일을 보내고 ID를 확인할 수 있는 권한을 부여합니다.		
SESCrudPolicy	이메일을 보내고 ID를 확인할 수 있는 권한을 부여합니다.		

정책 템플릿	설명
SESEmailTemplateCrudPolicy	Amazon SES 이메일 템플릿을 만들고, 가져오고, 나열하고, 업데이트하고, 삭제할 수 있는 권한을 부여합니다.
SESSendBouncePolicy	Amazon Simple Email Service(Amazon SES) ID에 대한 SendBounce 권한을 부여합니다.
SNSCrudPolicy	Amazon SNS 주제를 만들고, 게시하고, 구독할 수 있는 권한을 부여합니다.
SNSPublishMessagePolicy	Amazon Simple Notification Service(Amazon SNS) 주제에 이벤트 메시지 게시
SQSPollerPolicy	Amazon Simple Queue Service(Amazon SQS) 대기열을 폴링할 수 있는 권한을 부여합니다.
SQSSendMessagePolicy	Amazon SQS 대기열에 메시지를 보낼 수 있는 권한을 부여합니다.
SSMParameterReadPolicy	Amazon EC2 Systems Manager(SSM) 파라미터 스토어의 파라미터에 액세스하여 이 계정에 보안 암호를 로드할 수 있는 권한을 부여합니다. 파라미터 이름에 슬래시 접두사가 없을 때 사용합니다.
SSMParameterWithSlashPrefixReadPolicy	Amazon EC2 Systems Manager(SSM) 파라미터 스토어의 파라미터에 액세스하여 이 계정에 보안 암호를 로드할 수 있는 권한을 부여합니다. 파라미터 이름에 슬래시 접두사가 있는 경우 사용합니다.
StepFunctionsExecutionPolicy	Step Functions 상태 머신 실행을 시작하기 위해 권한을 제공합니다.

정책 템플릿	설명
TextractDetectAnalyzePolicy	Amazon Textract를 사용하여 문서를 탐지하고 분석할 수 있는 액세스 권한을 제공합니다.
TextractGetResultPolicy	Amazon Textract에서 문서를 탐지하고 분석할 수 있는 액세스 권한을 제공합니다.
TextractPolicy	Amazon Textract에 대한 모든 액세스 권한을 부여합니다.
VPCAccessPolicy	탄력적 네트워크 인터페이스를 생성, 삭제, 설명 및 분리할 수 있는 액세스 권한을 제공합니다.

문제 해결

SAM CLI error: "Must specify valid parameter values for policy template '<policy-template-name>'"

`sam build`를 실행하면 다음 오류가 표시됩니다.

```
"Must specify valid parameter values for policy template '<policy-template-name>'"
```

즉, 자리 표시자 값이 없는 정책 템플릿을 선언할 때 빈 객체를 전달하지 않았습니다.

이 문제를 해결하려면 다음 CloudWatchPutMetricPolicy 예제와 같이 정책을 선언합니다.

```
MyFunction:
  Policies:
    - CloudWatchPutMetricPolicy: {}
```

AWS SAM 정책 템플릿 목록

다음은 사용 가능한 정책 템플릿과 각 템플릿에 적용되는 권한입니다. AWS Serverless Application Model(AWS SAM)는 자리 표시자 항목(예: AWS 리전 및 계정 ID)을 적절한 정보로 자동으로 채웁니다.

주제

- AcmGetCertificatePolicy
- AMIDescribePolicy
- AthenaQueryPolicy
- AWSSecretsManagerGetSecretValuePolicy
- AWSSecretsManagerRotationPolicy
- CloudFormationDescribeStacksPolicy
- CloudWatchDashboardPolicy
- CloudWatchDescribeAlarmHistoryPolicy
- CloudWatchPutMetricPolicy
- CodePipelineLambdaExecutionPolicy
- CodePipelineReadOnlyPolicy
- CodeCommitCrudPolicy
- CodeCommitReadPolicy
- ComprehendBasicAccessPolicy
- CostExplorerReadOnlyPolicy
- DynamoDBBackupFullAccessPolicy
- DynamoDBCrudPolicy
- DynamoDBReadPolicy
- DynamoDBReconfigurePolicy
- DynamoDBRestoreFromBackupPolicy
- DynamoDBStreamReadPolicy
- DynamoDBWritePolicy
- EC2CopyImagePolicy
- EC2DescribePolicy
- EcsRunTaskPolicy
- EFSWriteAccessPolicy
- EKSDescribePolicy
- ElasticMapReduceAddJobFlowStepsPolicy
- ElasticMapReduceCancelStepsPolicy
- ElasticMapReduceModifyInstanceFleetPolicy

- ElasticMapReduceModifyInstanceGroupsPolicy
- ElasticMapReduceSetTerminationProtectionPolicy
- ElasticMapReduceTerminateJobFlowsPolicy
- ElasticsearchHttpPostPolicy
- EventBridgePutEventsPolicy
- FilterLogEventsPolicy
- FirehoseCrudPolicy
- FirehoseWritePolicy
- KinesisCrudPolicy
- KinesisStreamReadPolicy
- KMSDecryptPolicy
- KMSEncryptPolicy
- LambdaInvokePolicy
- MobileAnalyticsWriteOnlyAccessPolicy
- OrganizationsListAccountsPolicy
- PinpointEndpointAccessPolicy
- PollyFullAccessPolicy
- RekognitionDetectOnlyPolicy
- RekognitionFacesManagementPolicy
- RekognitionFacesPolicy
- RekognitionLabelsPolicy
- RekognitionNoDataAccessPolicy
- RekognitionReadPolicy
- RekognitionWriteOnlyAccessPolicy
- Route53ChangeResourceRecordSetsPolicy
- S3CrudPolicy
- S3FullAccessPolicy
- S3ReadPolicy
- S3WritePolicy
- SageMakerCreateEndpointConfigPolicy

- SageMakerCreateEndpointPolicy
- ServerlessRepoReadWriteAccessPolicy
- SESBulkTemplatedCrudPolicy
- SESBulkTemplatedCrudPolicy_v2
- SESCrudPolicy
- SESEmailTemplateCrudPolicy
- SESSendBouncePolicy
- SNSCrudPolicy
- SNSPublishMessagePolicy
- SQSPollerPolicy
- SQSSendMessagePolicy
- SSMParameterReadPolicy
- SSMParameterWithSlashPrefixReadPolicy
- StepFunctionsExecutionPolicy
- TextractDetectAnalyzePolicy
- TextractGetResultPolicy
- TextractPolicy
- VPCAccessPolicy

AcmGetCertificatePolicy

AWS Certificate Manager에서 인증서를 읽을 수 있는 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "acm:GetCertificate"
    ],
    "Resource": {
      "Fn::Sub": [
        "${certificateArn}",
        {
          "certificateArn": {
            "Ref": "CertificateArn"
```

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

AMIDescribePolicy

Amazon Machine Image(AMI)를 설명할 수 있는 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeImages"
    ],
    "Resource": "*"
  }
]
```

AthenaQueryPolicy

Athena 쿼리를 실행할 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "athena:ListWorkGroups",
      "athena:GetExecutionEngine",
      "athena:GetExecutionEngines",
      "athena:GetNamespace",
      "athena:GetCatalogs",
      "athena:GetNamespaces",
      "athena:GetTables",
      "athena:GetTable"
    ],
    "Resource": "*"
  },
  {
    "Effect": "Allow",
    "Action": [
```

```
        "athena:StartQueryExecution",
        "athena:GetQueryResults",
        "athena:DeleteNamedQuery",
        "athena:GetNamedQuery",
        "athena:ListQueryExecutions",
        "athena:StopQueryExecution",
        "athena:GetQueryResultsStream",
        "athena:ListNamedQueries",
        "athena:CreateNamedQuery",
        "athena:GetQueryExecution",
        "athena:BatchGetNamedQuery",
        "athena:BatchGetQueryExecution",
        "athena:GetWorkGroup"
      ],
      "Resource": {
        "Fn::Sub": [
          "arn:${AWS::Partition}:athena:${AWS::Region}:${AWS::AccountId}:workgroup/
${workgroupName}",
          {
            "workgroupName": {
              "Ref": "WorkGroupName"
            }
          }
        ]
      }
    }
]
```

AWSSecretsManagerGetSecretValuePolicy

지정된 AWS Secrets Manager 보안 암호의 보안 암호 값을 가져올 수 있는 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "secretsmanager:GetSecretValue"
    ],
    "Resource": {
      "Fn::Sub": [
        "${secretArn}",
        {
          "secretArn": {
            "Ref": "SecretArn"
```

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

AWSSecretsManagerRotationPolicy

AWS Secrets Manager에서 보안 암호를 교체할 수 있는 권한을 부여합니다.

```
"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

CloudFormation 스택을 설명할 수 있는 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "cloudformation:DescribeStacks"
    ],
    "Resource": {
      "Fn::Sub": "arn:${AWS::Partition}:cloudformation:${AWS::Region}:
${AWS::AccountId}:stack/*"
    }
  }
]
```

CloudWatchDashboardPolicy

CloudWatch 대시보드에서 작동할 지표를 배치할 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "cloudwatch:GetDashboard",
      "cloudwatch:ListDashboards",
      "cloudwatch:PutDashboard",
      "cloudwatch:ListMetrics"
    ],
    "Resource": "*"
  }
]
```

CloudWatchDescribeAlarmHistoryPolicy

Amazon CloudWatch 경보 기록을 설명할 수 있는 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "cloudwatch:DescribeAlarmHistory"
    ],
    "Resource": "*"
  }
]
```

CloudWatchPutMetricPolicy

CloudWatch에 지표를 전송할 수 있는 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "cloudwatch:PutMetricData"
    ],
    "Resource": "*"
  }
]
```

CodePipelineLambdaExecutionPolicy

AWS CodePipeline에서 간접 호출한 Lambda 함수가 작업 상태를 보고할 수 있는 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "codepipeline:PutJobSuccessResult",
      "codepipeline:PutJobFailureResult"
    ],
    "Resource": "*"
  }
]
```

CodePipelineReadOnlyPolicy

CodePipeline 파이프라인의 세부 정보를 가져올 수 있는 읽기 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "codepipeline:ListPipelineExecutions"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:codepipeline:${AWS::Region}:${AWS::AccountId}:
${pipelinename}",
        {
          "pipelinename": {
            "Ref": "PipelineName"
          }
        }
      ]
    }
  }
]
```

CodeCommitCrudPolicy

특정 CodeCommit 리포지토리 내에서 객체를 만들고, 읽고, 업데이트하고, 삭제할 수 있는 권한을 부여합니다.

```
"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

특정 CodeCommit 리포지토리 내에서 객체를 읽을 수 있는 권한을 부여합니다.

```
"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

엔터티, 핵심 문구, 언어 및 감정을 탐지할 수 있는 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "comprehend:BatchDetectKeyPhrases",
      "comprehend:DetectDominantLanguage",
      "comprehend:DetectEntities",
      "comprehend:BatchDetectEntities",
      "comprehend:DetectKeyPhrases",
```

```
        "comprehend:DetectSentiment",
        "comprehend:BatchDetectDominantLanguage",
        "comprehend:BatchDetectSentiment"
      ],
      "Resource": "*"
    }
]
```

CostExplorerReadOnlyPolicy

청구 내역에 대한 읽기 전용 AWS Cost Explorer(Cost Explorer) API에 읽기 전용 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ce:GetCostAndUsage",
      "ce:GetDimensionValues",
      "ce:GetReservationCoverage",
      "ce:GetReservationPurchaseRecommendation",
      "ce:GetReservationUtilization",
      "ce:GetTags"
    ],
    "Resource": "*"
  }
]
```

DynamoDBBackupFullAccessPolicy

테이블에 대한 DynamoDB 온디맨드 백업에 읽기 및 쓰기 권한을 부여합니다.

```
"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

Amazon DynamoDB 테이블에 생성, 읽기, 업데이트 및 삭제 권한을 부여합니다.

```
"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

DynamoDB 테이블에 읽기 전용 권한 부여

```
"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

DynamoDB 테이블을 재구성할 수 있는 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "dynamodb:UpdateTable"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:dynamodb:${AWS::Region}:${AWS::AccountId}:table/
${tableName}",
        {
          "tableName": {
            "Ref": "TableName"
```

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

DynamoDBRestoreFromBackupPolicy

백업에서 테이블을 복원할 수 있는 권한을 부여합니다.

```
"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

DynamoDB 스트림과 레코드를 설명하고 읽을 수 있는 권한을 부여합니다.

```
"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

DynamoDB 테이블에 쓰기 전용 권한 부여

```
"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

Amazon EC2 이미지를 복사할 수 있는 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ec2:CopyImage"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:ec2:${AWS::Region}:${AWS::AccountId}:image/${imageId}",
        {
          "imageId": {
            "Ref": "ImageId"
          }
        }
      ]
    }
  }
]
```

EC2DescribePolicy

Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스를 설명할 수 있는 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeRegions",
      "ec2:DescribeInstances"
    ],
    "Resource": "*"
  }
```

```
]
```

EcsRunTaskPolicy

태스크 정의에 대해 새 태스크를 시작할 수 있는 권한을 부여합니다.

```
"Statement": [
  {
    "Action": [
      "ecs:RunTask"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:ecs:${AWS::Region}:${AWS::AccountId}:task-definition/
${taskDefinition}",
        {
          "taskDefinition": {
            "Ref": "TaskDefinition"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

EFSWriteAccessPolicy

쓰기 액세스 권한이 있는 Amazon EFS 파일 시스템을 마운트할 수 있는 권한을 부여합니다.

```
"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

Amazon Elastic Kubernetes Service(Amazon EKS) 클러스터를 설명하거나 나열할 수 있는 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "eks:DescribeCluster",
      "eks:ListClusters"
    ],
    "Resource": "*"
  }
]
```

ElasticMapReduceAddJobFlowStepsPolicy

실행 중인 클러스터에 새 단계를 추가할 수 있는 권한을 부여합니다.

```
"Statement": [
  {
    "Action": "elasticmapreduce:AddJobFlowSteps",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

ElasticMapReduceCancelStepsPolicy

실행 중인 클러스터에서 대기 중 단계를 취소할 수 있는 권한을 부여합니다.

```
"Statement": [
  {
    "Action": "elasticmapreduce:CancelSteps",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

ElasticMapReduceModifyInstanceFleetPolicy

클러스터 내 인스턴스 플릿의 세부 정보를 나열하고 용량을 수정할 수 있는 권한을 부여합니다.

```
"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

클러스터 내 인스턴스 그룹의 세부 정보를 나열하고 설정을 수정할 수 있는 권한을 부여합니다.

```
"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

클러스터에 대한 종료 보호를 설정할 수 있는 권한을 부여합니다.

```
"Statement": [
  {
    "Action": "elasticmapreduce:SetTerminationProtection",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

ElasticMapReduceTerminateJobFlowsPolicy

클러스터를 종료할 수 있는 권한을 부여합니다.

```
"Statement": [
  {
    "Action": "elasticmapreduce:TerminateJobFlows",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:elasticmapreduce:${AWS::Region}:
${AWS::AccountId}:cluster/${clusterId}",
        {
          "clusterId": {
            "Ref": "ClusterId"
          }
        }
      ]
```

```
    },
    "Effect": "Allow"
  }
]
```

ElasticsearchHttpPostPolicy

Amazon OpenSearch Service에 POST 및 PUT 권한을 부여합니다.

```
"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

Amazon EventBridge에 사용자 지정 이벤트를 보낼 수 있는 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": "events:PutEvents",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:events:${AWS::Region}:${AWS::AccountId}:event-bus/
${eventBusName}",
        {
```

```
          "eventBusName": {
            "Ref": "EventBusName"
          }
        }
      ]
    }
  }
]
```

FilterLogEventsPolicy

지정된 로그 그룹에서 CloudWatch Logs 이벤트를 필터링할 수 있는 권한을 부여합니다.

```
"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

Firehose 전송 스트림을 생성, 쓰기, 업데이트 및 삭제할 수 있는 권한을 부여합니다.

```
"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

Firehose 전송 스트림에 쓸 수 있는 권한을 부여합니다.

```
"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

Amazon Kinesis 스트림을 생성, 게시 및 삭제할 수 있는 권한을 부여합니다.

```
"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

Amazon Kinesis 스트림을 나열하고 읽을 수 있는 권한을 부여합니다.

```
"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

AWS Key Management Service(AWS KMS) 키로 복호화할 수 있는 권한을 부여합니다. 단, `keyId`는 키 별칭이 아니라 AWS KMS 키 ID여야 한다는 점에 유의하세요.

```
"Statement": [
  {
    "Action": "kms:Decrypt",
    "Effect": "Allow",
    "Resource": {
      "Fn::Sub": [
```

```
          "arn:${AWS::Partition}:kms:${AWS::Region}:${AWS::AccountId}:key/${keyId}",
          {
            "keyId": {
              "Ref": "KeyId"
            }
          }
        ]
      }
    }
]
```

KMSEncryptPolicy

AWS KMS 키로 암호화할 수 있는 권한을 부여합니다. keyId 는 키 별칭이 아니라 AWS KMS 키 ID여야 한다는 점에 유의하세요.

```
"Statement": [
  {
    "Action": "kms:Encrypt",
    "Effect": "Allow",
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:kms:${AWS::Region}:${AWS::AccountId}:key/${keyId}",
        {
          "keyId": {
            "Ref": "KeyId"
          }
        }
      ]
    }
  }
]
```

LambdaInvokePolicy

AWS Lambda 함수, 별칭 또는 버전을 간접 호출할 수 있는 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "lambda:InvokeFunction"
```

```
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:lambda:${AWS::Region}:${AWS::AccountId}:function:
${functionName}*",
        {
          "functionName": {
            "Ref": "FunctionName"
          }
        }
      ]
    }
  }
]
```

MobileAnalyticsWriteOnlyAccessPolicy

모든 애플리케이션 리소스에 이벤트 데이터를 넣을 수 있는 쓰기 전용 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "mobileanalytics:PutEvents"
    ],
    "Resource": "*"
  }
]
```

OrganizationsListAccountsPolicy

하위 계정 이름 및 ID를 나열할 수 있는 읽기 전용 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "organizations:ListAccounts"
    ],
    "Resource": "*"
  }
]
```

PinpointEndpointAccessPolicy

Amazon Pinpoint 애플리케이션의 엔드포인트를 가져오고 업데이트할 수 있는 권한을 부여합니다.

```
"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

Amazon Polly 어휘 리소스에 대한 전체 액세스 권한을 부여합니다.

```
"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

얼굴, 레이블, 텍스트를 감지할 수 있는 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:DetectFaces",
      "rekognition:DetectLabels",
      "rekognition:DetectModerationLabels",
      "rekognition:DetectText"
    ],
    "Resource": "*"
  }
]
```

RekognitionFacesManagementPolicy

Amazon Rekognition 컬렉션에서 얼굴을 추가, 삭제 및 검색할 수 있는 권한을 부여합니다.

```
"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

얼굴과 레이블을 비교하고 감지할 수 있는 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:CompareFaces",
      "rekognition:DetectFaces"
    ],
    "Resource": "*"
  }
]
```

RekognitionLabelsPolicy

객체 및 중재 레이블을 탐지할 수 있는 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "rekognition:DetectLabels",
      "rekognition:DetectModerationLabels"
    ],
    "Resource": "*"
  }
]
```

RekognitionNoDataAccessPolicy

얼굴과 레이블을 비교하고 감지할 수 있는 권한을 부여합니다.

```
"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

얼굴을 나열하고 검색할 수 있는 권한을 부여합니다.

```
"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

얼굴을 모으고 인덱싱할 수 있는 권한을 부여합니다.

```
"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

Route 53에서 리소스 레코드 세트를 변경할 수 있는 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "route53:ChangeResourceRecordSets"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:route53:::hostedzone/${HostedZoneId}",
        {
          "HostedZoneId": {
            "Ref": "HostedZoneId"
          }
        }
      ]
    }
  }
]
```

S3CrudPolicy

Amazon S3 버킷의 객체에 대해 작업을 수행할 수 있는 생성, 읽기, 업데이트 및 삭제 권한을 부여합니다.

```
"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

Amazon S3 버킷의 객체에 대해 작업을 수행할 수 있는 전체 액세스 권한을 부여합니다.

```
"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

Amazon Simple Storage Service(S3) 버킷에 있는 객체를 읽을 수 있는 읽기 전용 권한을 부여합니다.

```
"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

Amazon S3 버킷에 객체를 쓸 수 있는 쓰기 권한을 부여합니다.

```
"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

SageMaker AI에서 엔드포인트 구성을 생성할 수 있는 권한을 부여합니다.

```
"Statement": [
  {
```

```
    "Action": [
      "sagemaker:CreateEndpointConfig"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sagemaker:${AWS::Region}:${AWS::AccountId}:endpoint-
config/${endpointConfigName}",
        {
          "endpointConfigName": {
            "Ref": "EndpointConfigName"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

SageMakerCreateEndpointPolicy

SageMaker AI에서 엔드포인트를 생성할 수 있는 권한을 부여합니다.

```
"Statement": [
  {
    "Action": [
      "sagemaker:CreateEndpoint"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sagemaker:${AWS::Region}:${AWS::AccountId}:endpoint/
${endpointName}",
        {
          "endpointName": {
            "Ref": "EndpointName"
          }
        }
      ]
    },
    "Effect": "Allow"
  }
]
```

ServerlessRepoReadWriteAccessPolicy

AWS Serverless Application Repository(AWS SAM) 서비스에서 애플리케이션을 만들고 나열할 수 있는 권한을 부여합니다.

```
"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

Amazon SES 이메일, 템플릿 기반 이메일, 템플릿 형식의 대량 이메일을 보내고 ID를 확인할 수 있는 권한을 부여합니다.

> **Note**
>
> `ses:SendTemplatedEmail` 작업을 수행하려면 템플릿 ARN이 필요합니다. 대신 `SESBulkTemplatedCrudPolicy_v2`을 사용하세요.

```
"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

Amazon SES 이메일, 템플릿 기반 이메일, 템플릿 형식의 대량 이메일을 보내고 ID를 확인할 수 있는 권한을 부여합니다.

```
"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

이메일을 보내고 ID를 확인할 수 있는 권한을 부여합니다.

```
"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

Amazon SES 이메일 템플릿을 만들고, 가져오고, 나열하고, 업데이트하고, 삭제할 수 있는 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ses:CreateTemplate",
      "ses:GetTemplate",
      "ses:ListTemplates",
      "ses:UpdateTemplate",
      "ses:DeleteTemplate",
      "ses:TestRenderTemplate"
    ],
    "Resource": "*"
  }
]
```

SESSendBouncePolicy

Amazon Simple Email Service(Amazon SES) ID에 대한 SendBounce 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ses:SendBounce"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:ses:${AWS::Region}:${AWS::AccountId}:identity/
${identityName}",
```

```
        {
          "identityName": {
            "Ref": "IdentityName"
          }
        }
      ]
    }
  }
]
```

SNSCrudPolicy

Amazon SNS 주제를 만들고, 게시하고, 구독할 수 있는 권한을 부여합니다.

```
"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

Amazon Simple Notification Service(Amazon SNS) 주제에 이벤트 메시지 게시

```
"Statement": [
```

```
  {
    "Effect": "Allow",
    "Action": [
      "sns:Publish"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sns:${AWS::Region}:${AWS::AccountId}:${topicName}",
        {
          "topicName": {
            "Ref": "TopicName"
          }
        }
      ]
    }
  }
]
```

SQSPollerPolicy

Amazon Simple Queue Service(Amazon SQS) 대기열을 폴링할 수 있는 권한을 부여합니다.

```
"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

Amazon SQS 대기열에 메시지를 보낼 수 있는 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "sqs:SendMessage*"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:sqs:${AWS::Region}:${AWS::AccountId}:${queueName}",
        {
          "queueName": {
            "Ref": "QueueName"
          }
        }
      ]
    }
  }
]
```

SSMParameterReadPolicy

Amazon EC2 Systems Manager(SSM) 파라미터 스토어의 파라미터에 액세스하여 이 계정에 보안 암호를 로드할 수 있는 권한을 부여합니다. 파라미터 이름에 슬래시 접두사가 없을 때 사용합니다.

> **Note**
> 기본 키를 사용하지 않는 경우 `KMSDecryptPolicy` 정책도 필요합니다.

```
"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

Amazon EC2 Systems Manager(SSM) 파라미터 스토어의 파라미터에 액세스하여 이 계정에 보안 암호를 로드할 수 있는 권한을 부여합니다. 파라미터 이름에 슬래시 접두사가 있는 경우 사용합니다.

> **Note**
>
> 기본 키를 사용하지 않는 경우 `KMSDecryptPolicy` 정책도 필요합니다.

```
"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

Step Functions 상태 머신 실행을 시작하기 위해 권한을 제공합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "states:StartExecution"
    ],
    "Resource": {
      "Fn::Sub": [
        "arn:${AWS::Partition}:states:${AWS::Region}:${AWS::AccountId}:stateMachine:
${stateMachineName}",
        {
          "stateMachineName": {
            "Ref": "StateMachineName"
          }
        }
      ]
    }
  }
]
```

TextractDetectAnalyzePolicy

Amazon Textract를 사용하여 문서를 탐지하고 분석할 수 있는 액세스 권한을 제공합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "textract:DetectDocumentText",
      "textract:StartDocumentTextDetection",
      "textract:StartDocumentAnalysis",
      "textract:AnalyzeDocument"
    ],
    "Resource": "*"
  }
]
```

TextractGetResultPolicy

Amazon Textract에서 문서를 탐지하고 분석할 수 있는 액세스 권한을 제공합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "textract:GetDocumentTextDetection",
      "textract:GetDocumentAnalysis"
    ],
    "Resource": "*"
  }
]
```

TextractPolicy

Amazon Textract에 대한 모든 액세스 권한을 부여합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "textract:*"
    ],
```

```
    "Resource": "*"
  }
]
```

VPCAccessPolicy

탄력적 네트워크 인터페이스를 생성, 삭제, 설명 및 분리할 수 있는 액세스 권한을 제공합니다.

```
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "ec2:CreateNetworkInterface",
      "ec2:DeleteNetworkInterface",
      "ec2:DescribeNetworkInterfaces",
      "ec2:DetachNetworkInterface"
    ],
    "Resource": "*"
  }
]
```

CloudFormation 메커니즘을 사용하여 AWS SAM 권한 관리

AWS 리소스에 대한 액세스를 제어하기 위해 AWS Serverless Application Model(AWS SAM)은 CloudFormation과 동일한 메커니즘을 사용할 수 있습니다. 자세한 내용은 AWS CloudFormation 사용 설명서에서 [AWS Identity and Access Management를 사용한 액세스 제어](https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-iam-template.html) 섹션을 참조하세요.

사용자에게 서버리스 애플리케이션을 관리할 수 있는 권한을 부여하는 데는 세 가지 기본 옵션이 있습니다. 각 옵션은 사용자에게 다양한 수준의 액세스 제어를 제공합니다.

- 관리자 권한을 부여합니다.
- 필요한 AWS 관리형 정책을 연결합니다.
- 특정 AWS Identity and Access Management(IAM) 권한을 부여합니다.

선택한 옵션에 따라 사용자는 액세스 권한이 있는 AWS 리소스가 포함된 서버리스 애플리케이션만 관리할 수 있습니다.

다음 단원에서는 각 프로세스를 상세히 설명합니다.

관리자 권한 부여

사용자에게 관리자 권한을 부여하면 AWS 리소스 조합을 원하는 대로 포함하는 서버리스 애플리케이션을 관리할 수 있습니다. 이는 가장 간단한 방법이지만 사용자에게 가장 광범위한 권한 집합을 부여하므로, 사용자가 가장 큰 영향을 미치는 작업을 수행할 수 있습니다.

사용자에게 관리자 권한을 부여하는 방법에 대한 자세한 내용은 IAM 사용자 설명서의 첫 번째 IAM 사용자 및 관리자 그룹 생성을 참조하세요.

필요한 AWS 관리형 정책 연결

전체 관리자 권한을 부여하는 대신 AWS 관리형 정책을 사용하여 사용자에게 권한의 일부를 부여할 수 있습니다. 이 옵션을 사용하는 경우 AWS 관리형 정책 집합에 사용자가 관리하는 서버리스 애플리케이션에 필요한 모든 작업과 리소스가 포함되는지 확인하세요.

예를 들어, 다음 AWS 관리형 정책은 샘플 Hello World 애플리케이션을 배포하기에 충분합니다.

- AWSCloudFormationFullAccess
- IAMFullAccess
- AWSLambda_FullAccess
- AmazonAPIGatewayAdministrator
- AmazonS3FullAccess
- AmazonEC2ContainerRegistryFullAccess

정책 연결에 대한 자세한 내용은 IAM 사용자 설명서의 IAM 사용자의 권한 변경을 참조하세요.

특정 IAM 권한 부여

가장 세분화된 수준의 액세스 제어를 위해 정책 설명을 사용하여 사용자에게 특정 IAM 권한을 부여할 수 있습니다. 이 옵션을 사용하는 경우 사용자가 관리하는 서버리스 애플리케이션에 필요한 모든 작업과 리소스가 정책 설명에 포함되는지 확인하세요.

이 옵션의 모범 사례는 사용자에게 Lambda 실행 역할을 비롯한 역할 생성 권한을 거부하여 사용자가 자신에게 에스컬레이션된 권한을 부여할 수 없도록 하는 것입니다. 따라서 관리자는 먼저 사용자가 관리할 서버리스 애플리케이션에 지정될 Lambda 실행 역할을 생성해야 합니다. Lambda 실행 역할 생성에 대한 자세한 내용은 IAM 콘솔에서 실행 역할 생성 섹션을 참조하세요.

샘플 Hello World 애플리케이션의 경우 AWSLambdaBasicExecutionRole만 있으면 애플리케이션을 실행하기에 충분합니다. Lambda 실행 역할을 생성한 후 샘플 Hello World 애플리케이션의 AWS SAM 템플릿 파일을 수정하여 `AWS::Serverless::Function` 리소스에 다음 속성을 추가합니다.

```
Role: lambda-execution-role-arn
```

수정된 Hello World 애플리케이션이 적용되면 다음 정책 설명은 사용자에게 애플리케이션을 배포, 업데이트 및 삭제할 수 있는 충분한 권한을 부여합니다.

> **Note**
>
> 이 섹션의 예제 정책 설명은 샘플 Hello World 애플리케이션을 배포, 업데이트 및 삭제할 수 있는 충분한 권한을 부여합니다. 애플리케이션에 추가 리소스 유형을 추가하는 경우 다음을 포함하도록 정책 설명을 업데이트해야 합니다.
>
> 1. 애플리케이션이 서비스 작업을 호출할 수 있는 권한.
> 2. 서비스 주체(서비스 활동에 필요한 경우)
>
> 예를 들어 Step Functions 워크플로를 추가하는 경우, 여기에 나열된 작업에 대한 권한과 `states.amazonaws.com` 서비스 주체를 추가해야 할 수 있습니다.

IAM 정책에 대한 자세한 내용은 IAM 사용자 설명서의 IAM 정책 관리 섹션을 참조하세요.

AWS SAM 템플릿으로 API 액세스 제어

API Gateway API에 대한 액세스를 제어하면 서버리스 애플리케이션이 안전하게 보호되며 활성화한 권한 부여를 통해서만 액세스할 수 있습니다. API Gateway API에 액세스할 수 있는 사용자를 제어하기 위해 AWS SAM 템플릿 내에서 권한 부여를 활성화할 수 있습니다.

AWS SAM은 귀하의 API Gateway API에 대하여 액세스를 제어하기 위한 다수의 메커니즘을 지원합니다. 지원되는 메커니즘 세트는 `AWS::Serverless::HttpApi` 및 `AWS::Serverless::Api` 리소스 유형 간에 차이가 있습니다.

다음 표에는 각 리소스 유형이 지원하는 메커니즘이 요약되어 있습니다.

액세스를 제어하기 위한 메커니즘	AWS::Serverless::HttpApi	AWS::Serverless::Api
Lambda 권한 부여자	✓	✓
IAM 권한		✓

액세스를 제어하기 위한 메커니즘	AWS::Serverless::HttpApi	AWS::Serverless::Api
Amazon Cognito 사용자 풀	✓ *	✓
API 키		✓
리소스 정책		✓
OAuth 2.0/JWT 권한 부여자	✓	

* Amazon Cognito를 `AWS::Serverless::HttpApi`리소스 유형과 함께 JSON Web Token(JWT) 발급자로 사용할 수 있습니다.

- Lambda 권한 부여자 – Lambda 권한 부여자(이전에는 사용자 지정 권한 부여자로 칭함)는 API에 대한 액세스를 제어하기 위해 제공하는 Lambda 함수입니다. API가 호출되면 이 Lambda 함수는 클라이언트 애플리케이션이 제공하는 요청 컨텍스트 또는 권한 부여 토큰을 사용하여 호출됩니다. Lambda 함수는 호출자가 요청된 작업을 수행할 권한이 있는지 여부에 응답합니다.

  `AWS::Serverless::HttpApi` 및 `AWS::Serverless::Api` 리소스 유형 모두 Lambda 권한 부여자를 지원합니다.

  `AWS::Serverless::HttpApi`을 사용하는 Lambda 권한 부여자에 대한 자세한 내용은 API Gateway 개발자 안내서의 HTTP API를 위한 AWS Lambda권한 부여자 사용을 참조하세요. `AWS::Serverless::Api`을 사용하는 Lambda 권한 부여자에 대한 자세한 내용은 API Gateway 개발자 안내서의 API Gateway Lambda 권한 부여자 사용을 참조하세요.

  각 리소스 유형에 대한 Lambda 권한 부여자의 예는 AWS SAM의 Lambda 권한 부여자 예제 섹션을 참조하세요.

- IAM 권한 - AWS Identity and Access Management(IAM) 권한을 사용하여 누가 귀하의 API를 호출할 수 있는지 제어할 수 있습니다. 귀하의 API를 호출하는 사용자는 IAM 자격 증명으로 인증을 받아야 합니다. API 호출자를 나타내는 IAM 사용자에게 연결된 IAM 정책, 사용자를 포함하는 IAM 그룹 또는 사용자가 위임하는 IAM 역할이 있는 경우에만 API 호출이 성공합니다.

  `AWS::Serverless::Api` 리소스 유형만 IAM 권한을 지원합니다.

  자세한 내용은 API Gateway 개발자 가이드에서 IAM 권한 부여자를 사용하는 API에 대한 액세스 제어를 참조하세요. 예시는 AWS SAM의 IAM 권한 예제에서 확인하십시오.

- Amazon Cognito 사용자 풀 – Amazon Cognito 사용자 풀은 Amazon Cognito의 사용자 디렉터리입니다. API 클라이언트는 먼저 사용자를 사용자 풀에 로그인시키고 해당 사용자에 대한 자격 증명 또는 액세스 토큰을 얻어야 합니다. 그러면 클라이언트가 반환된 토큰 중 하나를 사용하여 귀하의 API를 호출합니다. API 호출은 필수 토큰이 유효한 경우에만 성공합니다.

  `AWS::Serverless::Api` 리소스 유형은 Amazon Cognito 사용자 풀을 지원합니다. `AWS::Serverless::HttpApi` 리소스 유형은 Amazon Cognito를 JWT 발급자로 사용할 수 있도록 지원합니다.

  자세한 내용은 API Gateway 개발자 가이드에서 Amazon Cognito 사용자 풀을 권한 부여자로 사용하여 REST API에 대한 액세스 제어를 참조하세요. 예시는 AWS SAM의 Amazon Cognito 사용자 풀 예제에서 확인하십시오.

- API 키는 앱 개발자 고객에게 귀하의 API에 대한 액세스 권한을 부여하기 위해 귀하가 배포하는 영숫자 문자열 값입니다.

  `AWS::Serverless::Api` 리소스 유형만 API 키를 지원합니다.

  자세한 내용을 알아보려면 API Gateway 개발자 안내서의 Amazon API Gateway에서 API 키 생성 및 사용 계획을 참조하십시오. API 키의 예는 AWS SAM의 API 키 예제 섹션을 참조하세요.

- 리소스 정책 – 리소스 정책은 API Gateway API에 연결할 수 있는 JSON 정책 문서입니다. 리소스 정책을 사용하여 지정된 보안 주체(보통 IAM 사용자 또는 역할)가 API를 호출할 수 있는지 여부를 제어합니다.

  `AWS::Serverless::Api` 리소스 유형만 API Gateway API에 대한 액세스를 제어하는 메커니즘으로 리소스 정책을 지원합니다.

  리소스 정책에 대한 자세한 내용은 API Gateway 개발자 안내서의 API Gateway 리소스 정책을 사용하는 액세스 제어를 참조하세요. 리소스 정책의 예는 AWS SAM의 리소스 정책 예제 섹션을 참조하세요.

- 귀하는 OAuth 2.0/JWT 권한 부여자 - OpenID Connect(OIDC) 및 OAuth 2.0 프레임워크의 일부로 JWT를 사용하여 API에 대한 액세스를 제어할 수 있습니다. API Gateway는 클라이언트가 API 요청과 함께 제출하는 JWT를 검증하고 토큰 검증 및 선택적으로 토큰의 범위를 기반으로 요청을 허용하거나 거부합니다.

  `AWS::Serverless::HttpApi` 리소스 유형만 OAuth 2.0/JWT 권한 부여자를 지원합니다.

자세한 내용은 API Gateway 개발자 가이드에서 JWT 권한 부여자를 사용하는 HTTP API에 대한 액세스 제어를 참조하십시오. 예시는 AWS SAM의 OAuth 2.0/JWT 권한 부여자 예제에서 확인하십시오.

액세스 제어를 위한 메커니즘의 선택

귀하의 API Gateway API에 대한 액세스를 제어하는 데 사용할 메커니즘은 몇 가지 요인에 따라 달라집니다. 예를 들어, 승인 또는 액세스 제어를 설정하지 않은 미개발 프로젝트가 있는 경우 Amazon Cognito 사용자 풀이 최선의 선택일 수 있습니다. 사용자 풀을 설정하면 인증과 액세스 제어도 자동으로 설정되기 때문입니다.

하지만 애플리케이션에 이미 인증이 설정되어 있는 경우에는 Lambda 권한 부여자를 사용하는 것이 가장 좋은 방법일 수 있습니다. 기존 인증 서비스를 호출하고 응답에 따라 정책 문서를 반환할 수 있기 때문입니다. 또한 애플리케이션에 사용자 풀이 지원하지 않는 사용자 지정 인증 또는 액세스 제어 로직이 필요한 경우 Lambda 권한 부여자가 최선의 옵션일 수 있습니다.

사용할 메커니즘을 선택했으면 예시선택된 메커니즘을 사용하도록 애플리케이션을 구성하기 위해 AWS SAM를 사용하는 방법에 대한 해당을 참조하세요

오류 응답 사용자 지정

AWS SAM을 사용하여 일부 API Gateway 오류 응답의 콘텐츠를 사용자 지정할 수 있습니다. `AWS::Serverless::Api` 리소스 유형만 사용자 지정된 API Gateway 응답을 지원합니다.

API Gateway 응답에 대한 자세한 내용은 API Gateway 개발자 안내서의 API Gateway에서의 게이트웨이 응답을 참조하세요. 사용자 지정 응답의 예는 AWS SAM의 사용자 지정 응답 예제 섹션을 참조하세요.

예시

- AWS SAM의 Lambda 권한 부여자 예제
- AWS SAM의 IAM 권한 예제
- AWS SAM의 Amazon Cognito 사용자 풀 예제
- AWS SAM의 API 키 예제
- AWS SAM의 리소스 정책 예제
- AWS SAM의 OAuth 2.0/JWT 권한 부여자 예제
- AWS SAM의 사용자 지정 응답 예제

AWS SAM의 Lambda 권한 부여자 예제

`AWS::Serverless::Api` 리소스 유형은 두 가지 유형의 Lambda 권한 부여자, 즉 `TOKEN` 권한 부여자와 `REQUEST` 권한 부여자를 지원합니다. `AWS::Serverless::HttpApi`리소스 유형은 `REQUEST` 권한 부여자만 지원합니다. 다음은 각 유형의 예입니다.

Lambda **TOKEN** 권한 부여자 예제(AWS::Serverless::Api)

귀하의 AWS SAM 템플릿 내에 Lambda `TOKEN`권한 부여자를 정의하여 귀하의 API에 대한 액세스를 제어할 수 있습니다. 이 작업을 수행하려면 ApiAuth 데이터 유형을 사용합니다.

다음은 Lambda `TOKEN`권한 부여자를 위한 예제 AWS SAM 템플릿 섹션입니다.

> **Note**
>
> 다음 예제에서는 SAM `FunctionRole`이 암시적으로 생성됩니다.

```
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
```

Lambda 권한 부여자에 대한 자세한 내용은 API Gateway 개발자 안내서의 API Gateway Lambda 권한 부여자 사용을 참조하세요.

Lambda **REQUEST** 권한 부여자 예제(AWS::Serverless::Api)

귀하의 AWS SAM 템플릿 내에 Lambda REQUEST권한 부여자를 정의하여 귀하의 API에 대한 액세스를 제어할 수 있습니다. 이 작업을 수행하려면 ApiAuth 데이터 유형을 사용합니다.

다음은 Lambda REQUEST권한 부여자를 위한 예제 AWS SAM 템플릿 섹션입니다.

```
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
```

Lambda 권한 부여자에 대한 자세한 내용은 API Gateway 개발자 안내서의 API Gateway Lambda 권한 부여자 사용을 참조하세요.

Lambda 권한 부여자 예제(AWS::Serverless::HttpApi)

귀하의 AWS SAM 템플릿 내에 Lambda 권한 부여자를 정의하여 귀하의 HTTP API에 대한 액세스를 제어할 수 있습니다. 이 작업을 수행하려면 HttpApiAuth 데이터 유형을 사용합니다.

다음은 Lambda 권한 부여자를 위한 예제 AWS SAM 템플릿 섹션입니다.

```
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
```

AWS SAM의 IAM 권한 예제

귀하의 AWS SAM 템플릿 내에서 IAM 권한을 정의하여 API에 대한 액세스를 제어할 수 있습니다. 이 작업을 수행하려면 ApiAuth 데이터 유형을 사용합니다.

다음은 IAM 권한을 위해 사용하는 예제 AWS SAM 템플릿입니다.

```
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}
```

IAM 권한에 대한 자세한 내용은 API Gateway 개발자 안내서의 API 호출을 위한 액세스 제어를 참조하세요.

AWS SAM의 Amazon Cognito 사용자 풀 예제

귀하의 AWS SAM 템플릿 내에 Amazon Cognito 사용자 풀을 정의하여 API에 대한 액세스를 제어할 수 있습니다. 이 작업을 수행하려면 ApiAuth 데이터 유형을 사용합니다.

다음은 사용자 풀의 AWS SAM 템플릿 섹션 예시입니다.

```
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
```

Amazon Cognito 사용자 풀에 대한 자세한 내용은 API Gateway 개발자 안내서에서 Amazon Cognito 사용자 풀을 권한 부여자로 사용하여 REST API에 대한 액세스 제어를 참조하세요.

AWS SAM의 API 키 예제

귀하의 AWS SAM 템플릿 내에서 API 키를 요구하여 귀하의 API에 대한 액세스를 제어할 수 있습니다. 이 작업을 수행하려면 ApiAuth 데이터 유형을 사용합니다.

다음은 API 키에 대한 예제 AWS SAM 템플릿 섹션입니다.

```
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
```

API 키에 대한 자세한 내용은 API Gateway 개발자 가이드의 API 키로 사용 계획 생성 및 사용을 참조하세요.

AWS SAM의 리소스 정책 예제

귀하의 AWS SAM 템플릿 내에 리소스 정책을 연결하여 API에 대한 액세스를 제어할 수 있습니다. 이 작업을 수행하려면 ApiAuth 데이터 유형을 사용합니다.

다음은 프라이빗 API의 예제 AWS SAM 템플릿입니다. 프라이빗 API에는 배포할 리소스 정책이 있어야 합니다.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Resources:
  MyPrivateApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: Prod
      EndpointConfiguration: PRIVATE  # Creates a private API. Resource policies are
 required for all private APIs.
      Auth:
        ResourcePolicy:
          CustomStatements:
            - Effect: 'Allow'
              Action: 'execute-api:Invoke'
              Resource: ['execute-api:/*/*/*']
              Principal: '*'
            - Effect: 'Deny'
              Action: 'execute-api:Invoke'
              Resource: ['execute-api:/*/*/*']
              Principal: '*'
```

```
  MyFunction:
    Type: 'AWS::Serverless::Function'
    Properties:
      InlineCode: |
        def handler(event, context):
          return {'body': 'Hello World!', 'statusCode': 200}
      Handler: index.handler
      Runtime: python3.10
      Events:
        AddItem:
          Type: Api
          Properties:
            RestApiId:
              Ref: MyPrivateApi
            Path: /
            Method: get
```

리소스 정책에 대한 자세한 내용은 API Gateway 개발자 안내서의 API Gateway 리소스 정책을 사용하는 액세스 제어를 참조하세요. 프라이빗 API에 대한 자세한 내용은 API Gateway 개발자 안내서의 Amazon API Gateway에서 프라이빗 API 생성을 참조하세요.

AWS SAM의 OAuth 2.0/JWT 권한 부여자 예제

JWT를 OpenID Connect (OIDC) 및 OAuth 2.0 프레임워크의 일부로 사용하여 API에 대한 액세스를 제어할 수 있습니다. 이 작업을 수행하려면 HttpApiAuth 데이터 유형을 사용합니다.

다음은 OAuth 2.0/JWT 권한 부여자를 위한 예제 AWS SAM 템플릿 섹션입니다.

```
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
```

OAuth 2.0/JWT 권한 부여자에 대한 자세한 내용은 API Gateway 개발자 가이드에서 JWT 권한 부여자를 사용하는 HTTP API에 대한 액세스 제어를 참조하세요.

AWS SAM의 사용자 지정 응답 예제

귀하의 AWS SAM 템플릿 내에 응답 헤더를 정의하여 일부 API Gateway 오류 응답을 사용자 지정할 수 있습니다. 이렇게 하려면 게이트웨이 응답 객체 데이터 유형을 사용합니다.

다음은 DEFAULT_5XX 오류에 대한 사용자 지정 응답을 생성하는 예제 AWS SAM 템플릿입니다.

```
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
```

API Gateway 응답에 대한 자세한 내용은 API Gateway 개발자 안내서의 API Gateway에서의 게이트웨이 응답을 참조하세요.

AWS SAM에서 Lambda 계층을 사용하여 효율성 향상

AWS SAM을 사용하면 서버리스 애플리케이션에 계층을 포함할 수 있습니다. AWS Lambda 계층을 사용하면 Lambda 함수에서 Lambda 계층으로 코드를 추출한 다음 여러 Lambda 함수에서 사용할 수 있습니다. 이렇게 하면 배포 패키지의 크기를 줄이고, 핵심 함수 로직을 종속성과 분리하며, 여러 함수에서 종속성을 공유할 수 있습니다. 계층에 대한 자세한 내용은 AWS Lambda 개발자 안내서의 Lambda 계층을 참조하세요.

이 주제는 다음과 관련된 정보를 제공합니다.

- 애플리케이션에 레이어 포함
- 레이어가 로컬에 캐싱되는 방법

사용자 지정 레이어를 구축하는 방법에 대한 자세한 정보는 에서 Lambda 계층 빌드 AWS SAM를 참조하세요.

애플리케이션에 레이어 포함

애플리케이션에 레이어를 포함하려면 [AWS::Serverless::Function](#) 리소스 유형의 `Layers` 속성을 사용합니다.

다음은 레이어를 포함하는 Lambda 함수가 포함된 예제 AWS SAM 템플릿입니다.

```
ServerlessFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: .
    Handler: my_handler
    Runtime: Python3.7
    Layers:
        - <LayerVersion ARN>
```

레이어가 로컬에 캐싱되는 방법

`sam local` 명령 중 하나를 사용하여 함수를 호출하면 함수의 레이어 패키지가 다운로드되어 로컬 호스트에 캐시됩니다.

다음 표는 각 운영 체제의 기본 캐시 디렉토리 위치를 나타냅니다.

OS	위치
Windows 7	`C:\Users\<user>\AppData\Roaming\AWS SAM`
Windows 8	`C:\Users\<user>\AppData\Roaming\AWS SAM`
Windows 10	`C:\Users\<user>\AppData\Roaming\AWS SAM`
macOS	`~/.aws-sam/layers-pkg`
Unix	`~/.aws-sam/layers-pkg`

패키지가 캐시되면 함수를 호출하는 데 사용되는 Docker 이미지에 레이어를 AWS SAM CLI 오버레이합니다. AWS SAM CLI는 그것이 구축한 이미지의 이름과 캐시에 보관된 LayerVersion을 생성합니다. 다음 단원들에서 스키마에 관한 더 자세한 내용을 확인할 수 있습니다.

오버레이된 레이어를 검사하려면 다음 명령을 실행하여 검사하려는 이미지에서 배쉬 세션을 시작합니다.

```
docker run -it --entrypoint=/bin/bash samcli/lambda:<Tag following the schema outlined
 in Docker Image Tag Schema> -i
```

레이어 캐싱 디렉터리 이름 스키마

귀하의 템플릿에 정의된 LayerVersionARN이 주어지면 AWS SAM CLI는 ARN에서 레이어 이름과 버전을 추출합니다. 그러면 레이어 콘텐츠를 명명된 `LayerName-Version-<first 10 characters of sha256 of ARN>`에 배치할 디렉터리가 생성됩니다.

예시

```
ARN = arn:aws:lambda:us-west-2:111111111111:layer:myLayer:1
Directory name = myLayer-1-926eeb5ff1
```

Docker 이미지 태그 스키마

고유 레이어 해시를 계산하려면 모든 고유 레이어 이름을 구분자 '-'로 결합하고 SHA256 해시를 가져온 다음 처음 10자를 취합니다.

예시

```
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
```

고유 이름은 레이어 캐싱 디렉터리 이름 스키마와 동일하게 계산됩니다.

```
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
```

고유한 레이어 해시를 계산하려면 모든 고유 레이어 이름을 구분자 '-'로 결합하고 sha256 해시를 사용하여 처음 25자를 취합니다.

```
myLayer-1-926eeb5ff1-mySecondLayer-1-6bc1022bdf = 2dd7ac5ffb30d515926aef
```

그런 다음 이 값을 함수의 런타임 및 아키텍처와 결합하고 구분자를 '-'로 지정합니다.

```
python3.7-x86_64-2dd7ac5ffb30d515926aefffd
```

AWS SAM에서 중첩 애플리케이션을 사용하여 코드 및 리소스 재사용

서버리스 애플리케이션에는 하나 혹은 다수의 중첩 애플리케이션이 포함될 수 있습니다. 중첩 애플리케이션은 더 큰 애플리케이션의 일부이며 독립형 아티팩트 또는 더 큰 애플리케이션의 구성 요소로 패키징하고 배포할 수 있습니다. 중첩 애플리케이션을 사용하면 자주 사용되는 코드를 고유한 애플리케이션으로 전환할 수 있으며, 이 애플리케이션을 더 큰 서버리스 애플리케이션이나 여러 서버리스 애플리케이션에서 재사용할 수 있습니다.

서버리스 아키텍처가 성장함에 따라 일반적으로 여러 애플리케이션 템플릿에 동일한 구성 요소가 정의되는 공통 패턴이 나타납니다. 중첩 애플리케이션을 사용하면 별도의 AWS SAM 템플릿에서 공통 코드, 기능, 리소스 및 구성을 재사용할 수 있으므로 단일 소스의 코드만 유지 관리하면 됩니다. 이를 통해 중복 코드와 구성을 줄일 수 있습니다. 또한 이 모듈식 접근 방식을 사용하면 개발을 간소화하고, 코드 구성을 개선하며, 서버리스 애플리케이션 간에서 원활하게 일관성을 유지할 수 있습니다. 중첩된 애플리케이션을 사용하면 애플리케이션 고유의 비즈니스 로직에 더 집중할 수 있습니다.

서버리스 애플리케이션에서 중첩된 애플리케이션을 정의하려면 [AWS::Serverless::Application](#) 리소스 유형을 사용하십시오.

다음 두 소스에서 중첩 애플리케이션을 정의할 수 있습니다.

- AWS Serverless Application Repository애플리케이션 - AWS Serverless Application Repository 내의 사용자 계정에서 사용할 수 있는 애플리케이션을 사용하여 중첩 애플리케이션을 정의할 수 있습니다. 이러한 애플리케이션은 계정의 비공개 애플리케이션, 귀하의 계정과 비공개로 공유되는 애플리케이션 또는 공개적으로 공유되는 AWS Serverless Application Repository 내의 애플리케이션일 수 있습니다. 다양한 배포 권한 수준에 대한 자세한 내용은 개발자 가이드의 애플리케이션 배포 권한 및 AWS Serverless Application Repository애플리케이션 공개를 참조하세요.
- 로컬 애플리케이션 - 로컬 파일 시스템에 저장된 애플리케이션을 사용하여 중첩된 애플리케이션을 정의할 수 있습니다.

서버리스 애플리케이션에서 AWS SAM을 사용하여 이러한 유형의 중첩 애플리케이션을 모두 정의하는 방법에 대한 자세한 내용은 다음 섹션을 참조하세요.

> **Note**
>
> 서버리스 애플리케이션에 중첩될 수 있는 최대 애플리케이션 수는 200개입니다.

중첩된 애플리케이션이 가질 수 있는 인자의 최대 수는 60개입니다.

AWS Serverless Application Repository에서 중첩된 애플리케이션 정의

AWS Serverless Application Repository에서 사용할 수 있는 애플리케이션으로 중첩 애플리케이션을 정의할 수 있습니다. AWS Serverless Application Repository을 사용하여 중첩된 애플리케이션이 포함된 애플리케이션을 저장하고 배포할 수도 있습니다. AWS Serverless Application Repository에서 중첩된 애플리케이션의 세부 정보를 검토하려면,AWS SDK, AWS CLI, 또는 Lambda 콘솔을 사용할 수 있습니다.

서버리스 애플리케이션의 AWS Serverless Application Repository 템플릿 AWS SAM 내에서 호스팅되는 애플리케이션을 정의하려면 모든 애플리케이션의 세부 정보 페이지에서 SAM 리소스로 AWS Serverless Application Repository복사 버튼을 사용하십시오. 이렇게 하려면 다음 단계를 따릅니다.

1. AWS Management 콘솔에 로그인되어 있는지 확인합니다.
2. AWS Serverless Application Repository개발자 가이드의 [애플리케이션 탐색, 검색 및 배포](#) 섹션에 있는 단계를 사용하여 AWS Serverless Application Repository 내에서 중첩하려는 애플리케이션을 찾으십시오.
3. SAM 리소스로 복사 버튼을 선택합니다. 현재 보고 있는 애플리케이션의 SAM 템플릿 섹션이 이제 클립보드에 있습니다.
4. 이 애플리케이션에 중첩하려는 애플리케이션의 SAM 템플릿 파일 `Resources:` 섹션에 SAM 템플릿 섹션을 붙여 넣습니다.

다음은 AWS Serverless Application Repository에서 호스팅되는 중첩 애플리케이션을 위한 예제 SAM 템플릿 섹션입니다.

```
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
```

필요한 인자 설정이 없는 경우 템플릿의 `Parameters:` 섹션을 생략할 수 있습니다.

> **Important**
>
> AWS Serverless Application Repository 내에서 호스팅된 중첩 애플리케이션이 포함된 애플리케이션은 중첩 애플리케이션의 공유 제한을 승계합니다.
> 예를 들어, 어떤 애플리케이션이 공개적으로 공유되지만 특히 애플리케이션을 생성한 AWS 계정과 비공개로만 공유되는 중첩 애플리케이션을 포함하고 있는 경우를 가정해 봅니다. 이 경우, 귀하의 AWS 계정에 중첩 애플리케이션을 배포할 권한이 없으면 귀하는 상위 애플리케이션을 배포할 수 없습니다. 애플리케이션 배포 권한에 대한 자세한 내용은 개발자 가이드의 애플리케이션 배포 권한 및 AWS Serverless Application Repository애플리케이션 공개를 참조하세요.

로컬 파일 시스템에서 중첩된 애플리케이션 정의

로컬 파일 시스템에 저장된 애플리케이션으로 중첩된 애플리케이션을 정의할 수 있습니다. 로컬 파일 시스템에 저장되어 있는 AWS SAM 템플릿 파일의 경로를 지정하면 됩니다.

다음은 중첩된 로컬 애플리케이션을 위한 예제 SAM 템플릿 섹션입니다.

```
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
```

인자 설정이 없는 경우 템플릿의 `Parameters:` 섹션을 생략할 수 있습니다.

중첩 애플리케이션 배포

AWS SAM CLI 명령 `sam deploy`을 사용하여 중첩 애플리케이션을 배포할 수 있습니다. 자세한 내용은 AWS SAM을 사용하여 애플리케이션 및 리소스 배포을 확인하십시오.

> **Note**
>
> 중첩 애플리케이션이 포함된 애플리케이션을 배포하는 경우 애플리케이션에 중첩 애플리케이션이 포함되어 있음을 알려야 합니다. `CAPABILITY_AUTO_EXPAND`를 CreateCloudFormationChangeSet API에 전달하거나 `aws serverlessrepo create-cloud-formation-change-set` AWS CLI 명령을 사용하여 이 작업을 수행할 수 있습니다. 중첩된 애플리케이션을 승인하는 방법에 대한 자세한 내용은 개발자 가이드의 애플리케이션 배포 시 AWS Serverless Application Repository IAM 역할, 리소스 정책 및 중첩된 애플리케이션 승인을 참조하세요.

AWS SAM에서 EventBridge 스케줄러를 사용하여 시간 기반 이벤트 관리

이 주제의 내용에서는 Amazon EventBridge Scheduler의 정의, AWS SAM이 제공하는 지원 내용, Scheduler 이벤트를 생성하는 방법 및 Scheduler 이벤트를 생성할 때 참조할 수 있는 예제에 대한 세부 정보를 제공합니다.

Amazon EventBridge Scheduler란 무엇인가요?

EventBridge 스케줄러를 사용하여 AWS SAM 템플릿에서 이벤트를 예약합니다. Amazon EventBridge Scheduler는 모든 AWS 서비스에서 수천만 개의 이벤트와 태스크를 생성, 시작 및 관리할 수 있는 일정 관리 서비스입니다. 이 서비스는 시간 관련 이벤트에 특히 유용합니다. 이를 사용하여 이벤트 및 반복적인 시간 기반 호출을 예약할 수 있습니다. 또한 일회성 이벤트와 시작 및 종료 시간이 있는 비율 및 시간 표현식을 지원합니다.

Amazon EventBridge Scheduler에 대한 자세한 내용은 EventBridge Scheduler 사용 설명서의 Amazon EventBridge Scheduler란 무엇인가요?를 참조하세요.

주제

- AWS SAM의 EventBridge Scheduler 지원
- AWS SAM에서 EventBridge 스케줄러 이벤트 생성
- 예시
- 자세히 알아보기

AWS SAM의 EventBridge Scheduler 지원

AWS Serverless Application Model(AWS SAM) 템플릿 사양은 AWS Lambda 및 AWS Step Functions에 관한 EventBridge 스케줄러에서 이벤트를 스케줄링하는 데 사용할 수 있는 간단하고 간단한 명령문 제공.

AWS SAM에서 EventBridge 스케줄러 이벤트 생성

`ScheduleV2` 속성을 AWS SAM 템플릿의 이벤트 유형으로 설정하여 EventBridge 스케줄러 이벤트를 정의합니다. 이 속성은 `AWS::Serverless::Function` 및 `AWS::Serverless::StateMachine` 리소스 유형을 지원합니다.

```
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 스케줄러 이벤트 스케줄링은 처리되지 않은 이벤트에 대한 DLQ(Dead Letter Queue)도 지원합니다. dead-letter 대기열에 대한 자세한 내용은 EventBridge 스케줄러 사용자 가이드의 [EventBridge 스케줄러용 DLQ 대기열 구성]을 참조하세요.

DLQ ARN이 지정되면 AWS SAM는 스케줄러 스케줄에서 DLQ로 메시지를 전송할 수 있는 권한을 구성합니다. DLQ ARN이 지정되지 않은 경우 AWS SAM은 DLQ 리소스를 생성합니다.

예시

AWS SAM을 사용하여 EventBridge 스케줄러 이벤트를 정의하는 기본 예제

```
Transform: AWS::Serverless-2016-10-31
Resources:
  MyLambdaFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: python3.8
      InlineCode: |
        def handler(event, context):
            print(event)
            return {'body': 'Hello World!', 'statusCode': 200}
      MemorySize: 128
      Events:
        Schedule:
          Type: ScheduleV2
          Properties:
            ScheduleExpression: rate(1 minute)
            Input: '{"hello": "simple"}'

  MySFNFunction:
    Type: AWS::Serverless::Function
    Properties:
      Handler: index.handler
      Runtime: python3.8
      InlineCode: |
        def handler(event, context):
            print(event)
            return {'body': 'Hello World!', 'statusCode': 200}
      MemorySize: 128

  StateMachine:
    Type: AWS::Serverless::StateMachine
    Properties:
      Type: STANDARD
      Definition:
        StartAt: MyLambdaState
        States:
          MyLambdaState:
            Type: Task
            Resource: !GetAtt MySFNFunction.Arn
```

```
            End: true
      Policies:
        - LambdaInvokePolicy:
            FunctionName: !Ref MySFNFunction
      Events:
        Schedule:
          Type: ScheduleV2
          Properties:
            ScheduleExpression: rate(1 minute)
            Input: '{"hello": "simple"}'
```

자세히 알아보기

ScheduleV2 EventBridge 스케줄러 속성을 정의하는 방법에 대한 자세한 내용은 다음을 참조하세요.

- ScheduleV2(AWS::Serverless::Function일 때)
- ScheduleV2(AWS::Serverless::StateMachine일 때)

AWS Step Functions를 사용하여 AWS SAM 리소스를 오케스트레이션합니다.

귀하는 AWS Step Functions을 사용하여 AWS Lambda함수 및 기타 AWS 리소스를 오케스트레이션함으로써 복잡하고 강력한 워크플로를 구성할 수 있습니다. Step Functions는 AWS Lambda 함수와 같은 AWS 리소스가 사용되는 시기와 조건을 애플리케이션에 알려줍니다. 이를 통해 복잡하지만 견고한 워크플로를 형성하는 프로세스가 간소화됩니다. AWS::Serverless::StateMachine을 사용하여 워크플로의 개별 단계를 정의하고 각 단계의 리소스를 연결한 다음 이러한 단계의 순서를 지정합니다. 필요한 경우 전환과 조건도 추가합니다. 이를 통해 복잡하지만 견고한 워크플로를 만드는 프로세스가 간소화됩니다.

> **Note**
>
> Step Functions 상태 기기가 포함된 AWS SAM 템플릿을 관리하려면 AWS SAM CLI의 버전 0.52.0 이상을 사용해야 합니다. 설치되어 있는 버전을 확인하려면 `sam --version` 명령을 실행하십시오.

Step Functions는 작업 및 상태 기기의 개념을 기반으로 합니다. 귀하는 JSON에 기반한 Amazon States Language을 사용하여 상태 기기를 정의합니다. Step Functions 콘솔에는 상태 기기의 구조가 그래픽으로 표시되므로 상태 기기의 논리를 시각적으로 확인하고 실행을 모니터링할 수 있습니다.

AWS Serverless Application Model(AWS SAM)에서의 Step Functions의 지원으로 귀하는 다음 작업을 수행할 수 있습니다.

- AWS SAM 템플릿 내에서 직접 또는 별도의 파일에서 상태 기기를 정의합니다.
- AWS SAM 정책 템플릿, 인라인 정책 또는 관리형 정책을 통해 상태 시스템 실행 역할을 생성합니다.
- 템플릿 AWS SAM 내 일정에 따라 API Gateway 또는 Amazon EventBridge 이벤트를 사용하여 또는 API를 직접 호출하여 스테이트 기기 실행을 촉발합니다.
- 일반적인 Step Functions 개발 패턴에는 사용 가능한 AWS SAM정책 템플릿을 사용합니다.

예시

AWS SAM 템플릿 파일의 다음 예제 스니펫은 정의 파일에서 Step Functions 상태 기기를 정의합니다. 참고로 `my_state_machine.asl.json` 파일은 Amazon States Language로 작성해야 합니다.

```
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
      ...
```

Step Functions 상태 기기가 포함된 샘플 AWS SAM 애플리케이션을 다운로드하려면 AWS Step Functions개발자 안내서에서 Step Functions 사용 상태 기기 만들기AWS SAM를 잠조하세요.

추가 정보

Step Functions 및 이를 AWS SAM과 함께 사용하는 것에 대해 자세히 알아보려면 다음을 잠조하세요.

- AWS Step Functions 작동 방식
- AWS Step Functions 및 AWS Serverless Application Model
- 사용 지침서: AWS SAM를 사용하여 Step Functions 사용 상태 기기 만들기
- AWS SAM 사양: AWS::Serverless::StateMachine

AWS SAM 애플리케이션에 대한 코드 서명 설정

신뢰할 수 있는 코드만 배포되게 하려면 AWS SAM을 사용하여 서버리스 애플리케이션에서 코드 서명을 활성화할 수 있습니다. 코드에 서명하면 서명 이후 코드가 변경되지 않았는지 확인할 수 있으며 신뢰할 수 있는 게시자의 서명된 코드 패키지만 Lambda 함수에서 실행되도록 할 수 있습니다. 이렇게 하면 조직이 배포 파이프라인에서 게이트키퍼 구성 요소를 구축해야 하는 부담을 덜 수 있습니다.

코드 서명에 대한 자세한 내용은 AWS Lambda 개발자 안내서의 Lambda 함수에 대한 코드 서명 구성을 참조하세요.

귀하의 서버리스 애플리케이션에 대한 코드 서명을 구성하려면 먼저 AWS Signer를 사용하여 서명 프로파일을 생성해야 합니다. 이 서명 프로필을 사용하여 다음 작업을 수행할 수 있습니다.

1. 코드 서명 구성 생성 - `AWS::Lambda::CodeSigningConfig`리소스를 선언하여 신뢰할 수 있는 게시자의 서명 프로필을 지정하고 유효성 검사를 위한 정책 조치를 설정합니다. 이 객체를 서버리스 함수와 동일한 AWS SAM 템플릿, 다른 AWS SAM템플릿 또는 CloudFormation 템플릿에서 선언할 수 있습니다. 그런 다음 `AWS::Lambda::CodeSigningConfig`리소스의 Amazon 리소스 이름(ARN) 으로 함수 `CodeSigningConfigArn`속성을 지정하여 서버리스 함수의 코드 서명을 활성화합니다.
2. 귀하의 코드 서명 – `sam package`또는 `sam deploy`명령을 `--signing-profiles`옵션과 함께 사용합니다.

> **Note**
>
> `sam package` 또는 `sam deploy` 명령으로 코드에 성공적으로 서명하려면 이러한 명령과 함께 사용하는 Amazon S3 버킷의 버전 관리를 활성화해야 합니다. 귀하가 AWS SAM이 자동으로 생성하는 Amazon S3 버킷을 사용하는 경우 버전 관리가 자동으로 활성화됩니다. Amazon S3 버킷 버전 관리에 대한 자세한 내용과 귀하가 제공하는 Amazon S3 버킷의 버전 관리를 활성화하기 위한 지침은 Amazon Simple Storage Service 사용자 가이드의 Amazon S3 버킷의 버전 관리 사용을 참조하세요.

서버리스 애플리케이션을 배포하면 Lambda는 귀하가 코드 서명을 활성화한 모든 함수에 대해 검증 검사를 수행합니다. 또한 Lambda는 해당 함수가 의존하는 모든 레이어에 대해 검증 검사를 수행합니다. Lambda의 검증 검사에 대한 자세한 내용은 AWS Lambda개발자 안내서의 서명 검증을 참조하세요.

예시

서명 프로필 생성

서명 프로필을 생성하려면 다음 명령을 실행합니다.

```
aws signer put-signing-profile --platform-id "AWSLambda-SHA384-ECDSA" --profile-
name MySigningProfile
```

이전 명령이 성공적인 경우, 서명 프로필의 ARN이 반환된 것을 볼 수 있습니다. 예:

```
{
    "arn": "arn:aws:signer:us-east-1:111122223333:/signing-profiles/MySigningProfile",
    "profileVersion": "SAMPLEverx",
    "profileVersionArn": "arn:aws:signer:us-east-1:111122223333:/signing-
profiles/MySigningProfile/SAMPLEverx"
}
```

`profileVersionArn` 필드에는 코드 서명 구성을 생성할 때 사용할 ARN이 포함됩니다.

코드 서명 구성 생성 및 함수에 대한 코드 서명 활성화

다음 예제 AWS SAM 템플릿은 AWS::Lambda::CodeSigningConfig리소스를 선언하고 Lambda 함수에 대한 코드 서명을 활성화합니다. 이 예제에는 신뢰할 수 있는 프로필이 하나 있으며 서명 검사에 실패하면 배포가 거부됩니다.

```
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"
```

코드 서명

애플리케이션을 패키징하거나 배포할 때 코드에 서명할 수 있습니다. 다음 예제 명령과 같이 `sam package` 또는 `sam deploy` 명령을 사용하여 `--signing-profiles` 옵션을 지정합니다.

귀하의 애플리케이션을 패키징할 때 함수 코드에 서명:

```
sam package --signing-profiles HelloWorld=MySigningProfile --s3-bucket amzn-s3-demo-bucket --output-template-file packaged.yaml
```

귀하의 애플리케이션을 패키징할 때 함수 코드와 함수가 의존하는 레이어에 모두 서명:

```
sam package --signing-profiles HelloWorld=MySigningProfile MyLayer=MySigningProfile --s3-bucket amzn-s3-demo-bucket --output-template-file packaged.yaml
```

함수 코드와 계층에 서명한 후 배포 수행:

```
sam deploy --signing-profiles HelloWorld=MySigningProfile MyLayer=MySigningProfile --s3-bucket amzn-s3-demo-bucket --template-file packaged.yaml --stack-name --region us-east-1 --capabilities CAPABILITY_IAM
```

> **Note**
>
> `sam package` 또는 `sam deploy` 명령으로 코드에 성공적으로 서명하려면 이러한 명령과 함께 사용하는 Amazon S3 버킷의 버전 관리를 활성화해야 합니다. 귀하가 AWS SAM이 자동으로 생성하는 Amazon S3 버킷을 사용하는 경우 버전 관리가 자동으로 활성화됩니다. Amazon S3 버킷 버전 관리에 대한 자세한 내용과 귀하가 제공하는 Amazon S3 버킷의 버전 관리를 활성화하기 위한 지침은 Amazon Simple Storage Service 사용자 가이드의 [Amazon S3 버킷의 버전 관리 사용을](#) 참조하세요.

`sam deploy --guided`로 서명 프로필 제공:

귀하가 코드 서명으로 구성된 서버리스 애플리케이션으로 `sam deploy --guided` 명령을 실행하면, AWS SAM는 귀하에게 코드 서명에 사용할 서명 프로필을 제공하라는 프롬프트를 표시합니다. `sam` deploy --guided 프롬프트에 대한 자세한 정보는 AWS SAMCLI 명령 참조의을 잠조하세요sam deploy

AWS SAM 템플릿 파일 검증

sam validate로 템플릿의 유효성을 검증합니다. 현재 이 명령은 제공된 템플릿이 유효한 JSON/YAML인지 검증합니다. 대부분의 AWS SAM CLI 명령과 마찬가지로 기본적으로 현재 작업 디렉터리에서 template.[yaml|yml] 파일을 찾습니다. 귀하는 -t 또는 --template 옵션을 사용하여 다른 템플릿 파일/위치를 지정할 수 있습니다.

예시

```
$ sam validate
<path-to-template>/template.yaml is a valid SAM Template
```

> **Note**
>
> sam validate 명령을 실행하려면 AWS 자격 증명을 구성해야 합니다. 자세한 내용은 AWS SAM CLI 구성을 참조하십시오

AWS SAM을 사용한 애플리케이션 빌드

AWS SAM 템플릿에 코드형 인프라(IaC)를 추가하면 sam build 명령을 사용하여 애플리케이션 빌드를 시작할 준비가 된 것입니다. 이 명령은 애플리케이션 프로젝트 디렉터리의 파일에서 빌드 아티팩트를 생성합니다. 즉, AWS SAM 템플릿 파일, 애플리케이션 코드, 적용 가능한 모든 언어별 파일 및 종속성을 생성합니다. 이러한 빌드 아티팩트는 로컬 테스트, AWS 클라우드 배포와 같은 애플리케이션 개발의 후속 단계에 사용할 수 있도록 서버리스 애플리케이션을 준비합니다. 테스트와 배포 모두 빌드 아티팩트를 입력으로 사용합니다.

sam build를 사용하여 전체 서버리스 애플리케이션을 빌드할 수 있습니다. 또한 특정 함수, 계층 또는 사용자 지정 런타임이 있는 빌드와 같은 사용자 지정 빌드를 생성할 수 있습니다. sam build를 사용하는 방법과 이유에 대해 자세히 알아보려면 이 섹션의 주제를 참조하세요. sam build 명령 사용에 대한 소개는 AWS SAM의 빌드 소개 단원을 참조하세요.

주제

- AWS SAM의 빌드 소개

- AWS SAM을 사용한 기본 빌드
- AWS SAM을 사용하여 빌드 사용자 지정

AWS SAM의 빌드 소개

AWS Serverless Application Model Command Line Interface(AWS SAM CLI) `sam build`명령을 사용하여 로컬 테스트 또는 AWS 클라우드 배포와 같은 개발 워크플로의 후속 단계를 위해 서버리스 애플리케이션을 준비할 수 있습니다. 이 명령은 `sam local`과 `sam deploy`에서 요구하는 형식과 위치로 애플리케이션을 구성하는 `.aws-sam` 디렉터리를 만듭니다.

- AWS SAM CLI에 대한 소개는 란 무엇입니까 AWS SAMCLI? 섹션을 참조하세요.
- `sam build` 명령 옵션 목록은 sam build 섹션을 참조하세요.
- 일반적인 개발 워크플로 중 `sam build`를 사용하는 예는 2단계: 귀하의 애플리케이션 빌드 섹션을 참조하세요.

> **Note**
>
> `sam build`를 사용하려면 개발 시스템에 있는 서버리스 애플리케이션의 기본 구성 요소부터 시작해야 합니다. 여기에는 AWS SAM 템플릿, AWS Lambda 함수 코드, 모든 언어별 파일 및 종속성이 포함됩니다. 자세한 내용은 에서 애플리케이션 생성 AWS SAM 섹션을 참조하세요.

주제

- sam build로 애플리케이션 빌드
- 로컬 테스트 및 배포
- 모범 사례
- sam build 옵션
- 문제 해결
- 예시
- 자세히 알아보기

sam build로 애플리케이션 빌드

`sam build`를 사용하기 전에 다음을 구성하는 것이 좋습니다.

1. Lambda 함수 및 계층 - `sam build` 명령은 Lambda 함수 및 계층을 구축할 수 있습니다. Lambda 계층 사용에 대해 자세히 알아보려면 에서 Lambda 계층 빌드 AWS SAM 섹션을 참조하세요.
2. Lambda 런타임 – 런타임은 간접적으로 호출될 때 실행 환경에서 함수를 실행하는 언어별 환경을 제공합니다. 네이티브 런타임과 사용자 지정 런타임을 구성할 수 있습니다.
   a. 네이티브 런타임 - 지원되는 Lambda 런타임에서 Lambda 함수를 작성하고 AWS 클라우드에서 네이티브 Lambda 런타임을 사용하도록 함수를 빌드합니다.
   b. 사용자 지정 런타임 - 모든 프로그래밍 언어를 사용하여 Lambda 함수를 작성하고, makefile 또는 esbuild와 같은 타사 빌더에 정의된 사용자 지정 프로세스를 사용하여 런타임을 빌드합니다. 자세한 내용은 AWS SAM에서 사용자 지정 런타임을 사용하여 Lambda 함수 빌드를 참조하세요.
3. Lambda 패키지 유형 - Lambda 함수는 다음과 같은 Lambda 배포 패키지 유형으로 패키징될 수 있습니다.
   a. .zip 파일 아카이브 – 애플리케이션 코드와 해당 종속 항목이 포함됩니다.
   b. 컨테이너 이미지 – 기본 운영 체제, 런타임, Lambda 익스텐션, 애플리케이션 코드 및 해당 종속 항목이 포함됩니다.

`sam init`를 사용하여 애플리케이션을 초기화할 때 이러한 애플리케이션 설정을 구성할 수 있습니다.

- `sam init`를 사용하는 방법에 대한 자세한 내용은 에서 애플리케이션 생성 AWS SAM 섹션을 참조하세요.
- 애플리케이션에서 이러한 설정을 구성하는 방법에 대해 자세히 알아보려면 AWS SAM을 사용한 기본 빌드 섹션을 참조하세요.

애플리케이션을 빌드하려면

1. 프로젝트의 루트에 대한 `cd`. AWS SAM 템플릿과 같은 위치입니다.

```
$ cd sam-app
```

2. 다음을 실행합니다.

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

> **Note**
>
> 일반적으로 사용되는 옵션은 `--use-container`입니다. 자세한 내용은 제공된 컨테이너 내에 Lambda 함수 빌드를 참조하세요.

다음은 AWS SAM CLI 출력의 예시입니다.

```
sam-app $ sam build
Starting Build use cache
Manifest file is changed (new hash: 3298f1304...d4d421) or dependency folder (.aws-
sam/deps/4d3dfad6-a267-47a6-a6cd-e07d6fae318c) is missing for (HelloWorldFunction),
 downloading dependencies and copying/building source
Building codeuri: /Users/.../sam-app/hello_world runtime: python3.12 metadata: {}
 architecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:CleanUp
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource
Running PythonPipBuilder:CopySource

Build Succeeded

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

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

3. AWS SAM CLI는 `.aws-sam` 빌드 디렉터리를 생성합니다. 다음은 예제입니다.

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

```
### build.toml
```

애플리케이션 구성 방식에 따라 AWS SAM CLI은 다음을 수행합니다.

1. `.aws-sam/build` 디렉터리에서 종속성을 다운로드, 설치 및 구성합니다.
2. Lambda 코드를 준비합니다. 여기에는 코드 컴파일, 실행 가능한 바이너리 생성, 컨테이너 이미지 구축이 포함될 수 있습니다.
3. 빌드 아티팩트를 `.aws-sam` 디렉터리에 복사합니다. 형식은 애플리케이션 패키지 유형에 따라 달라집니다.
   a. .zip 패키지 유형의 경우 로컬 테스트에 사용할 수 있도록 아티팩트가 아직 압축되지 않았습니다. AWS SAM CLI는 `sam deploy`를 사용할 때 애플리케이션을 압축합니다.
   b. 컨테이너 이미지 패키지 유형의 경우 컨테이너 이미지가 로컬에서 생성되고 `.aws-sam/build.toml` 파일에서 참조됩니다.
4. AWS SAM 템플릿을 `.aws-sam` 디렉터리에 복사하고 필요한 경우 새 파일 경로로 수정합니다.

`.aws-sam` 디렉터리의 빌드 아티팩트를 구성하는 주요 구성 요소는 다음과 같습니다.

- 빌드 디렉터리 - 서로 독립적으로 구성된 Lambda 함수 및 계층을 포함합니다. 그 결과 `.aws-sam/build` 디렉터리의 각 함수 또는 계층이 고유한 구조를 갖게 됩니다.
- AWS SAM 템플릿 - 빌드 프로세스 중 변경 내용을 기반으로 업데이트된 값으로 수정되었습니다.
- build.toml 파일 - AWS SAM CLI에서 사용하는 빌드 설정이 포함된 구성 파일입니다.

로컬 테스트 및 배포

`sam local`를 사용하여 로컬 테스트를 수행하거나 `sam deploy`를 사용하여 배포를 수행할 때 AWS SAM CLI는 다음을 수행합니다.

1. 먼저 `.aws-sam` 디렉터리가 존재하는지, 해당 디렉터리에 AWS SAM 템플릿이 있는지 확인합니다. 이러한 조건이 충족되면 AWS SAM CLI는 이 디렉터리를 애플리케이션의 루트 디렉터리로 간주합니다.
2. 이러한 조건이 충족되지 않는 경우 AWS SAM CLI는 AWS SAM 템플릿의 원래 위치를 애플리케이션의 루트 디렉터리로 간주합니다.

개발할 때 원본 애플리케이션 파일이 변경되면 로컬에서 테스트하기 전에 `sam build`를 실행하여 `.aws-sam` 디렉터리를 업데이트합니다.

모범 사례

- `.aws-sam/build` 디렉터리 아래의 코드는 수정하지 않습니다. 대신 프로젝트 폴더의 원본 소스 코드를 업데이트하고 `sam build`를 실행하여 `.aws-sam/build` 디렉터리를 업데이트합니다.
- 원본 파일을 수정할 때는 `sam build`를 실행하여 `.aws-sam/build` 디렉터리를 업데이트합니다.
- `sam local`을 개발하고 테스트할 때와 같이 AWS SAM CLI가 `.aws-sam` 디렉터리 대신 프로젝트의 원래 루트 디렉터리를 참조하도록 할 수도 있습니다. `.aws-sam` 디렉터리나 `.aws-sam` 디렉터리의 AWS SAM 템플릿을 삭제하여 AWS SAM CLI가 원래 프로젝트 디렉터리를 루트 프로젝트 디렉터리로 인식하도록 합니다. 준비가 되면 `sam build`를 다시 실행하여 `.aws-sam` 디렉터리를 생성합니다.
- `sam build`를 실행하면 매번 `.aws-sam/build` 디렉터리를 덮어쓰게 됩니다. `.aws-sam` 디렉터리는 그렇지 않습니다. 로그와 같은 파일을 저장하려면 파일을 덮어쓰지 않도록 `.aws-sam`에 저장하세요.

sam build 옵션

단일 리소스 빌드

해당 리소스만 빌드하려면 리소스의 논리적 ID를 제공합니다. 다음은 예제입니다.

```
$ sam build HelloWorldFunction
```

중첩된 애플리케이션 또는 스택의 리소스를 구축하려면 다음 형식 *<stack-logical-id>/<resource-logical-id>*을 사용하여 애플리케이션 또는 스택 논리 ID를 리소스 논리 ID와 함께 제공하세요.

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

제공된 컨테이너 내에 Lambda 함수 빌드

`--use-container` 옵션은 컨테이너 이미지를 다운로드하고 이를 사용하여 Lambda 함수를 빌드합니다. 그러면 로컬 컨테이너가 `.aws-sam/build.toml` 파일에서 참조됩니다.

이 옵션은 Docker가 설치되어 있어야 합니다. 지침은 [Docker 설치] 섹션을 참조하세요.

다음은 그러한 명령의 예입니다.

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

--build-image 옵션과 함께 사용할 컨테이너 이미지를 지정할 수 있습니다. 다음은 예제입니다.

```
$ sam build --use-container --build-image amazon/aws-sam-cli-build-image-nodejs20.x
```

단일 함수에 사용할 컨테이너 이미지를 지정하려면 함수 논리적 ID를 제공합니다. 다음은 예제입니다.

```
$ sam build --use-container --build-image Function1=amazon/aws-sam-cli-build-image-
python3.12
```

컨테이너로 전달할 환경 변수입니다.

빌드 컨테이너로 환경 변수를 전달하기 위해 --container-env-var를 사용합니다. 다음은 예제입니다.

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

파일에서 환경 변수를 전달하려면 --container-env-var-file 옵션을 사용합니다. 다음은 예제입니다.

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

env.json 파일의 예:

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

여러 함수를 포함하는 애플리케이션을 빠르게 빌드할 수 있습니다.

여러 함수가 있는 애플리케이션에서 sam build를 실행하는 경우 AWS SAM CLI는 각 함수를 한 번에 하나씩 빌드합니다. 빌드 프로세스의 속도를 높이려면 --parallel 옵션을 사용하세요. 이 옵션을 사용하면 모든 함수와 계층이 동시에 빌드됩니다.

다음은 그러한 명령의 예입니다.

```
$ sam build —-parallel
```

소스 폴더에서 프로젝트를 빌드하여 빌드 시간 단축

지원되는 런타임과 빌드 메서드의 경우 `--build-in-source` 옵션을 사용하여 소스 폴더에서 직접 프로젝트를 빌드할 수 있습니다. 기본적으로 AWS SAM CLI는 임시 디렉터리에 빌드하며, 여기에는 소스 코드와 프로젝트 파일을 복사하는 작업이 수반됩니다. `--build-in-source`를 사용하면 AWS SAM CLI가 소스 폴더에서 직접 빌드하므로 파일을 임시 디렉터리에 복사할 필요가 없어 빌드 프로세스 속도가 빨라집니다.

지원되는 런타임 및 빌드 메서드 목록은 `--build-in-source`를 참조하세요.

문제 해결

AWS SAM CLI 문제를 해결하려면 AWS SAMCLI 문제 해결 섹션을 참조하세요.

예시

네이티브 런타임과.zip 패키지 유형을 사용하는 애플리케이션 빌드

이 예에서는 자습서: AWS SAM에서 Hello World 애플리케이션 배포 섹션을 참조하세요.

네이티브 런타임과 이미지 패키지 유형을 사용하는 애플리케이션 구축

먼저 `sam init`를 실행하여 새 애플리케이션을 초기화합니다. 대화형 흐름 중에 `Image` 패키지 유형을 선택합니다. 다음은 예제입니다.

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

Choose an AWS Quick Start application template
        1 - Hello World Example
        2 - Multi-step workflow
        3 - Serverless API
        4 - Scheduled task
        5 - Standalone function
        6 - Data processing
```

```
        7 - Hello World Example With Powertools
        8 - Infrastructure event management
        9 - Serverless Connector Hello World Example
        10 - Multi-step workflow with Connectors
        11 - Lambda EFS example
        12 - DynamoDB Example
        13 - Machine Learning
Template: 1

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

Which runtime would you like to use?
        ...
        10 - java8
        11 - nodejs20.x
        12 - nodejs18.x
        13 - nodejs16.x
        ...
Runtime: 12

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

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

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

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

Project name [sam-app]: ENTER

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

    -----------------------
    Generating application:
    -----------------------
    Name: sam-app
```

```
    Base Image: amazon/nodejs18.x-base
    Architectures: x86_64
    Dependency Manager: npm
    Output Directory: .
    Configuration file: sam-app/samconfig.toml

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

...
```

AWS SAM CLI는 애플리케이션을 초기화하고 다음 프로젝트 디렉터리를 만듭니다.

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

다음으로 애플리케이션을 빌드하기 위해 `sam build`를 실행합니다.

```
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
```

컴파일된 프로그래밍 언어를 포함하는 애플리케이션 빌드

이 예제에서는 Go 런타임을 사용하여 Lambda 함수를 포함하는 애플리케이션을 빌드합니다.

먼저 `sam init`를 사용하여 새 애플리케이션을 초기화하고 Go를 사용하도록 애플리케이션을 구성합니다.

```
$ 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 SAM CLI는 애플리케이션을 초기화합니다. 다음은 애플리케이션 디렉터리 구조의 예제입니다.

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

이 애플리케이션의 요구 사항은 README.md 파일을 참조합니다.

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

다음으로 함수를 테스트하기 위해 `sam local invoke`를 실행합니다. 이 명령은 Go가 로컬 컴퓨터에 설치되지 않았으므로 오류가 발생합니다.

```
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"}%
```

다음으로 애플리케이션을 빌드하기 위해 `sam build`를 실행합니다. 로컬 컴퓨터에 Go가 설치되지 않아 오류가 발생했습니다.

```
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
```

함수를 제대로 빌드하도록 로컬 시스템을 구성할 수도 있지만, 대신 `--use-container` 옵션에 `sam build`를 사용합니다. AWS SAMCLI는 컨테이너 이미지를 다운로드하고, 네이티브 GoModulesBuilder를 사용하여 함수를 빌드하고, 결과 바이너리를 `.aws-sam/build/HelloWorldFunction` 디렉터리에 복사합니다.

```
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
```

다음은 `.aws-sam` 파일의 예입니다.

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

다음으로 `sam local invoke`를 실행합니다. 함수가 성공적으로 호출되었습니다.

```
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"}%
```

자세히 알아보기

`sam build` 명령 사용에 대한 자세한 내용은 다음을 참조하세요.

- AWS SAM 학습: sam build – YouTube의 서버리스 랜드 'AWS SAM 학습' 시리즈.
- AWS SAM 학습 | sam build | E3 – YouTube의 서버리스 랜드 'AWS SAM 학습' 시리즈.
- AWS SAM 빌드: 배포용 아티팩트를 제공하는 방법(SAM S2E8을 사용한 세션) - YouTube의 AWS SAM 시리즈를 사용한 세션.
- AWS SAM 사용자 지정 빌드: Makefiles를 사용하여 SAM(S2E9)에서 빌드를 사용자 지정하는 방법 - YouTube의 AWS SAM 시리즈를 사용한 세션.

AWS SAM을 사용한 기본 빌드

서버리스 애플리케이션을 구축하려면 `sam build` 명령을 사용합니다. 또한 이 명령은 애플리케이션 종속성의 구축 아티팩트를 수집하여 로컬 테스트, 패키징, 배포와 같은 다음 단계를 위해 적절한 형식과 위치에 배치합니다.

`requirements.txt`(Python) 또는 `package.json`(Node.js) 같은 매니페스트 파일에서 또는 함수 리소스의 `Layers` 속성을 사용하여 애플리케이션의 종속성을 지정합니다. `Layers` 속성에는 Lambda 함수가 의존하는 AWS Lambda 계층 리소스 목록이 포함되어 있습니다.

애플리케이션 구축 아티팩트의 형식은 각 함수의 `PackageType` 속성에 따라 달라집니다. 이 속성의 옵션은 다음과 같습니다.

- **`Zip`** – 애플리케이션 코드와 그 종속 항목을 포함하는 .zip 파일 아카이브입니다. 코드를 .zip 파일 아카이브로 패키징하는 경우 함수의 Lambda 런타임을 지정해야 합니다.
- **`Image`** - 애플리케이션 코드 및 그 종속 항목에 추가하여 기본 운영 체제, 런타임, 익스텐션 등을 포함하는 컨테이너 이미지입니다.

Lambda 패키지 유형에 대한 자세한 내용은 개발자 가이드의 AWS Lambda Lambda 배포 패키지를 참조하세요.

주제

- .zip 파일 아카이브 만들기
- 컨테이너 이미지 구축
- 컨테이너 환경 변수 파일
- 소스 폴더에서 프로젝트를 빌드하여 빌드 시간 단축
- 예시
- AWS SAM의 외부에서 함수 구축

.zip 파일 아카이브 만들기

서버리스 애플리케이션을 .zip 파일 아카이브로 구축하려면 서버리스 함수에 대해 `PackageType: Zip`을 선언합니다.

AWS SAM은 지정한 아키텍처에 맞게 애플리케이션을 구축합니다. 아키텍처를 지정하지 않는 경우 AWS SAM은 기본적으로 `x86_64`를 사용합니다.

Lambda 함수가 기본적으로 컴파일된 프로그램이 있는 패키지에 의존하는 경우 `--use-container` 플래그를 사용하십시오. 이 플래그는 Lambda 환경처럼 동작하는 컨테이너에서 함수를 로컬로 컴파일하므로 이를 AWS 클라우드에 배포할 때 올바른 형식을 갖게 됩니다.

`--use-container` 옵션을 사용하면 기본적으로 AWS SAM가 Amazon ECR Public에서 컨테이너 이미지를 추출합니다. 다른 리포지토리에서 컨테이너 이미지를 가져오거나 AWS SAM CLI의 특정 버전을 사용하려면, `--build-image` 옵션을 사용하고 대체 컨테이너 이미지의 URI를 제공할 수 있습니다. 다음은 AWS SAM CLI의 특정 버전에서 컨테이너 이미지를 사용하여 애플리케이션을 빌드하는 두 가지 예시 명령어입니다.

```
# Build a Node.js 20 application using a container image for a specific version of AWS
 SAM CLI (1.136.0)
sam build --use-container --build-image public.ecr.aws/sam/build-nodejs22.x:1.136.0

# Build a function resource using the Python 3.13 container image from a specific
 version of AWS SAM CLI (1.136.0)(
sam build --use-container --build-image Function1=public.ecr.aws/sam/build-
python3.13:1.136.0
```

.zip 파일 아카이브 애플리케이션을 구축하는 추가 예제는 이 주제 뒷부분의 예제 섹션을 참조하세요.

컨테이너 이미지 구축

서버리스 애플리케이션을 컨테이너 이미지로 구축하려면 서버리스 함수에 대해 `PackageType: Image`을 선언하십시오. 또한 다음 항목들을 사용하여 `Metadata` 리소스 속성을 선언해야 합니다.

`Dockerfile`

Lambda 함수와 연결되는 Dockerfile의 이름입니다.

`DockerContext`

Dockerfile의 위치입니다.

`DockerTag`

(선택 사항) 구축된 이미지에 적용할 태그.

`DockerBuildArgs`

구축에 대한 구축 인수입니다.

> **Important**
>
> AWS SAM CLI는 `DockerBuildArgs` 인수에 포함된 정보를 삭제하거나 난독화하지 않습니다. 이 섹션을 사용하여 암호나 보안 암호와 같은 민감한 정보를 저장하지 않는 것이 좋습니다.

다음은 예제 `Metadata` 리소스 속성 섹션입니다.

```
Metadata:
  Dockerfile: Dockerfile
  DockerContext: ./hello_world
  DockerTag: v1
```

`Image` 패키지 유형으로 구성된 샘플 애플리케이션을 다운로드하려면 [자습서: AWS SAM에서 Hello World 애플리케이션 배포](#) 섹션을 참조하세요. 설치할 패키지 유형을 묻는 메시지가 나타나면 `Image`을 선택합니다.

> **Note**
>
> Dockerfile에서 다중 아키텍처 기본 이미지를 지정하는 경우 AWS SAM은 호스트 머신의 아키텍처용 컨테이너 이미지를 구축합니다. 다른 아키텍처용으로 구축하려면 특정 대상 아키텍처를 사용하는 기본 이미지를 지정하십시오.

컨테이너 환경 변수 파일

구축 컨테이너의 환경 변수가 포함된 JSON 파일을 제공하려면 `--container-env-var-file` 인수를 `sam build` 명령과 함께 사용하십시오. 모든 서버리스 리소스에 적용되는 단일 환경 변수를 제공하거나 각 리소스에 대해 서로 다른 환경 변수를 제공할 수 있습니다.

형식

구축 컨테이너에 환경 변수를 전달하는 형식은 리소스에 제공하는 환경 변수의 개수에 따라 달라집니다.

모든 리소스에 단일 환경 변수를 제공하려면 다음과 같이 `Parameters` 객체를 지정하십시오.

```
{
  "Parameters": {
    "GITHUB_TOKEN": "TOKEN_GLOBAL"
  }
}
```

각 리소스에 서로 다른 환경 변수를 제공하려면 다음과 같이 각 리소스의 객체를 지정하십시오.

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

환경 변수를 파일로(예를 들면 `env.json`이라는 이름으로) 저장합니다. 다음 명령은 이 파일을 사용하여 환경 변수를 구축 컨테이너에 전달합니다.

```
sam build --use-container --container-env-var-file env.json
```

우선 순위

- 특정 리소스에 귀하가 제공하는 환경 변수는 모든 리소스의 단일 환경 변수보다 우선합니다.
- 명령 행에 제공하는 환경 변수는 한 파일 내 환경 변수보다 우선합니다.

소스 폴더에서 프로젝트를 빌드하여 빌드 시간 단축

지원되는 런타임과 빌드 메서드의 경우 `--build-in-source` 옵션을 사용하여 소스 폴더에서 직접 프로젝트를 빌드할 수 있습니다. 기본적으로 AWS SAM CLI는 임시 디렉터리에 빌드하며, 여기에는 소스 코드와 프로젝트 파일을 복사하는 작업이 수반됩니다. `--build-in-source`를 사용하면 AWS

SAM CLI가 소스 폴더에서 직접 빌드하므로 파일을 임시 디렉터리에 복사할 필요가 없어 빌드 프로세스 속도가 빨라집니다.

지원되는 런타임 및 빌드 메서드 목록은 `--build-in-source`를 참조하세요.

예시

예제 1: .zip 파일 아카이브

다음 `sam build` 명령은.zip 파일 아카이브를 구축합니다.

```
# Build all functions and layers, and their dependencies
sam build

# Run the build process inside a Docker container that functions like a Lambda
 environment
sam build --use-container

# Build a Node.js 20 application using a container image for a specific version of AWS
 SAM CLI (1.136.0)
sam build --use-container --build-image public.ecr.aws/sam/build-nodejs22.x:1.136.0

# Build a function resource using the Python 3.13 container image from a specific
 version of AWS SAM CLI (1.136.0)(
sam build --use-container --build-image Function1=public.ecr.aws/sam/build-
python3.13:1.136.0

# Build and run your functions locally
sam build && sam local invoke

# For more options
sam build --help
```

예제 2: 컨테이너 이미지

다음 AWS SAM 템플릿은 컨테이너 이미지로 구축됩니다.

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      PackageType: Image
```

```
      ImageConfig:
        Command: ["app.lambda_handler"]
    Metadata:
      Dockerfile: Dockerfile
      DockerContext: ./hello_world
      DockerTag: v1
```

다음은 Dockfile 파일의 예입니다.

```
FROM public.ecr.aws/lambda/python:3.12

COPY app.py requirements.txt ./

RUN python3.12 -m pip install -r requirements.txt

# Overwrite the command by providing a different command directly in the template.
CMD ["app.lambda_handler"]
```

예제 3: npm ci

Node.js 애플리케이션의 경우 종속물을 설치하기 위하여 npm `ci`대신 npm `install`를 사용할 수 있습니다. npm `ci`을 사용하려면 Lambda 함수의 `UseNpmCi: True` 리소스 속성 내 `BuildProperties` 이하에 `Metadata`를 지정하십시오. npm `ci`을 사용하려면 애플리케이션에 귀하의 Lambda 함수용 `package-lock.json`에 존재하는 `npm-shrinkwrap.json` 또는 `CodeUri` 파일이 있어야 합니다.

다음 예제는 npm `ci`의 실행 시 종속 항목을 설치하기 위해 `sam build`를 사용합니다.

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello-world/
      Handler: app.handler
      Runtime: nodejs20.x
      Architectures:
        - x86_64
      Events:
        HelloWorld:
          Type: Api
          Properties:
            Path: /hello
```

```
            Method: get
    Metadata:
      BuildProperties:
        UseNpmCi: True
```

Python 상위 패키지

Python 애플리케이션의 경우, 빌드 프로세스 중에 패키지 구조를 보존하여 절대 임포트를 활성화할 수 있습니다. 패키지 구조를 보존하려면, Lambda 함수의 `Metadata` 리소스 속성에서 `BuildProperties` 아래에 `parent_python_packages`을(를) 지정하세요.

다음 예시는 `sam build`을(를) 실행할 때 app 패키지 구조를 보존합니다.

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello-world/
      Handler: app.main.handler
      Runtime: python3.12
      Architectures:
        - x86_64
    Metadata:
      BuildProperties:
        parent_python_packages: app
```

이 구성을 사용하면, 코드가 `from .utils import logger`와(과) 같은 상대 임포트 대신 `from app.utils import logger`와(과) 같은 절대 임포트를 사용할 수 있습니다.

AWS SAM의 외부에서 함수 구축

기본적으로 sam build을 실행하면 AWS SAM가 모든 함수 리소스를 구축합니다. 그 밖에도 다음과 같은 옵션이 있습니다.

- 모든 함수 리소스를 AWS SAM외부에서 구축 - 모든 함수 리소스를 수동으로 구축하거나 다른 도구를 통해 구축하는 경우에는 sam build이 필요하지 않습니다. sam build을 건너뛰고 로컬 테스트 수행 또는 애플리케이션 배포와 같은 프로세스의 다음 단계로 이동할 수 있습니다.
- 일부 함수 리소스를 AWS SAM외부에서 구축 – 함수 리소스 중 일부는 AWS SAM으로 하여금 구축하게 하고 다른 함수 리소스는 AWS SAM의 외부에서 구축하려는 경우 AWS SAM의 템플릿에서 이를 지정할 수 있습니다.

일부 함수 리소스를 AWS SAM 외부에서 구축하기

AWS SAM을 사용할 때 sam build로 하여금 함수를 건너뛰도록 하려면 AWS SAM 템플릿에서 다음을 구성하십시오.

1. 함수에 `SkipBuild: True` 메타데이터 속성을 추가합니다.
2. 구축된 함수 리소스의 경로를 지정합니다.

다음은 `TestFunction`을 건너뛰도록 구성한 예제입니다. 구축된 리소스는 `built-resources/TestFunction.zip`에 있습니다.

```
TestFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: built-resources/TestFunction.zip
    Handler: TimeHandler::handleRequest
    Runtime: java11
  Metadata:
    SkipBuild: True
```

이제 sam build을 실행하면 AWS SAM가 다음과 같은 작업을 수행합니다.

1. AWS SAM이 `SkipBuild: True`로 구성된 함수를 건너뛰게 됩니다.
2. AWS SAM이 다른 모든 함수 리소스를 구축하고 이들을 `.aws-sam` 구축 디렉터리에 캐시합니다.
3. 건너뛰는 함수의 경우 `.aws-sam` 구축 디렉터리 내 이들의 템플릿이 구축된 함수 리소스의 지정된 경로를 참조하도록 자동으로 업데이트됩니다.

   다음은 `TestFunction` 구축 디렉터리에 있는 `.aws-sam`를 위한 캐시된 템플릿의 예입니다.

   ```
   TestFunction:
     Type: AWS::Serverless::Function
     Properties:
       CodeUri: ../../built-resources/TestFunction.zip
       Handler: TimeHandler::handleRequest
       Runtime: java11
     Metadata:
       SkipBuild: True
   ```

AWS SAM을 사용하여 빌드 사용자 지정

특정 Lambda 함수 또는 Lambda 계층을 포함하도록 빌드를 사용자 지정할 수 있습니다. 함수는 Lambda에서 코드를 실행하기 위해 간접 호출할 수 있는 리소스입니다. Lambda 계층을 사용하면 Lambda 함수에서 코드를 추출한 다음 여러 Lambda 함수에서 재사용할 수 있습니다. 공유 종속성 또는 리소스를 관리하는 복잡성 없이 개별 서버리스 함수를 개발하고 배포하는 데 집중하려는 경우 특정 Lambda 함수에 맞게 빌드를 사용자 지정할 수 있습니다. 또한 Lambda 계층을 빌드하도록 선택하여 배포 패키지의 크기를 줄이고, 핵심 함수 로직을 종속성과 분리하며, 여러 함수에서 종속성을 공유할 수 있습니다.

이 섹션의 주제에서는 AWS SAM을 사용하여 Lambda 함수를 빌드할 수 있는 여러 방법을 살펴봅니다. 여기에는 고객 런타임을 사용하여 Lambda 함수를 빌드하고 Lambda 계층을 빌드하는 것이 포함됩니다. 사용자 지정 런타임을 사용하면 AWS Lambda 개발자 안내서의 Lambda 런타임에 나열되지 않은 언어를 설치하고 사용할 수 있습니다. 이렇게 하면 서버리스 함수 및 애플리케이션을 실행할 수 있는 특수한 실행 환경을 생성할 수 있습니다. 전체 애플리케이션을 빌드하는 대신 Lambda 계층만 빌드함으로써 몇 가지 이점을 얻을 수 있습니다. 예를 들어 배포 패키지의 크기를 줄이고, 핵심 함수 로직을 종속성과 분리하며, 여러 함수에서 종속성을 공유할 수 있습니다.

자세한 내용은 AWS Lambda 개발자 안내서의 Lambda 개념을 참조하세요.

주제

- AWS SAM에서 esbuild를 사용하여 Node.js Lambda 함수 빌드
- AWS SAM에서 네이티브 AOT 컴파일로 .NET Lambda 함수 빌드
- AWS SAM에서 Cargo Lambda를 사용하여 Rust Lambda 함수 빌드
- AWS SAM에서 사용자 지정 런타임을 사용하여 Lambda 함수 빌드
- 에서 Lambda 계층 빌드 AWS SAM

AWS SAM에서 esbuild를 사용하여 Node.js Lambda 함수 빌드

Node.js AWS Lambda 함수를 빌드하고 패키징하려면 AWS SAM CLI를 esbuild JavaScript 번들러와 함께 사용할 수 있습니다. esbuild 번들러는 TypeScript로 작성하는 Lambda 함수를 지원합니다.

esbuild를 사용하여 Node.js Lambda 함수를 빌드하려면 `AWS::Serverless::Function` 리소스에 `Metadata` 객체를 추가하고 `BuildMethod`에 `esbuild`를 지정합니다. sam build 명령을 실행하면 AWS SAM은 esbuild를 사용하여 Lambda 함수 코드를 번들링합니다.

메타데이터 속성

`Metadata` 객체는 다음 속성을 지원합니다.

BuildMethod

애플리케이션의 URL 접두사를 지정합니다. 지원되는 유일한 값은 `esbuild`입니다.

BuildProperties

Lambda 함수 코드의 빌드 속성을 지정합니다.

`BuildProperties` 객체는 다음 속성을 지원합니다. 다른 모든 속성은 선택 사항입니다. 기본적으로 AWS SAM는 Lambda 함수 핸들러를 진입점에 사용합니다.

EntryPoint

애플리케이션의 진입점을 지정합니다.

외부

빌드에서 제외할 패키지 목록을 지정합니다. 자세한 내용은 esbuild 웹사이트의 외부 섹션을 참조하세요.

형식

애플리케이션에서 생성된 JavaScript 파일의 출력 형식을 지정합니다. 자세한 내용은 esbuild 웹 사이트의 형식을 참조하세요.

로더

지정된 파일 유형의 데이터를 로드하기 위한 구성 목록을 지정합니다.

MainFields

패키지를 확인할 때 가져오려고 시도할 `package.json` 필드를 지정합니다. 기본값은 `main,module`입니다.

Minify

번들링된 출력 코드를 축소할지 여부를 지정합니다. 기본값은 `true`입니다.

OutExtension

esbuild가 생성하는 파일의 파일 확장자를 사용자 지정합니다. 자세한 내용은 esbuild 웹사이트의 확장을 참조하세요.

Sourcemap

번들러가 소스 맵 파일을 생성할지 여부를 지정합니다. 기본값은 `false`입니다.

`true`로 설정되면 `NODE_OPTIONS: --enable-source-maps`는 Lambda 함수의 환경 변수에 추가되고, 소스 맵이 생성되어 함수에 포함됩니다.

또는 `NODE_OPTIONS: --enable-source-maps`가 함수의 환경 변수에 포함된 경우 `Sourcemap`가 자동으로 `true`로 설정됩니다.

충돌하는 경우 `Sourcemap: false`는 `NODE_OPTIONS: --enable-source-maps`보다 우선합니다.

> **Note**
> 기본적으로 Lambda는 저장된 모든 환경 변수를 AWS Key Management Service(AWS KMS)로 암호화합니다. 소스 맵을 사용할 때 배포가 성공하려면 함수의 실행 역할에 `kms:Encrypt` 작업을 수행할 수 있는 권한이 있어야 합니다.

SourcesContent

소스 맵 파일에 소스 코드를 포함할지 여부를 지정합니다. `Sourcemap`이 `'true'`로 설정되면 이 속성을 구성합니다.

- 모든 소스 코드를 포함하도록 `SourcesContent: 'true'`를 지정합니다.
- 모든 소스 코드를 제외하도록 `SourcesContent: 'false'`를 지정합니다. 이렇게 하면 소스 맵 파일 크기가 작아지므로 시작 시간을 줄여 프로덕션에 유용합니다. 하지만 소스 코드는 디버거에서 사용할 수 없습니다.

기본값은 `SourcesContent: true`입니다.

자세한 내용은 esbuild 웹 사이트의 [소스 콘텐츠](#)를 참조하세요.

대상

대상 ECMAScript 버전을 지정합니다. 기본값은 `es2020`입니다.

TypeScript Lambda 함수

다음 예제 AWS SAM 템플릿 스니펫은 esbuild를 사용하여 `hello-world/app.ts`의 TypeScript 코드에서 Node.js Lambda 함수를 생성합니다.

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello-world/
      Handler: app.handler
      Runtime: nodejs20.x
      Architectures:
        - x86_64
      Events:
        HelloWorld:
          Type: Api
          Properties:
            Path: /hello
            Method: get
      Environment:
        Variables:
          NODE_OPTIONS: --enable-source-maps
    Metadata:
      BuildMethod: esbuild
      BuildProperties:
        Format: esm
        Minify: false
        OutExtension:
          - .js=.mjs
        Target: "es2020"
        Sourcemap: true
        EntryPoints:
          - app.ts
        External:
          - "<package-to-exclude>"
```

AWS SAM에서 네이티브 AOT 컴파일로 .NET Lambda 함수 빌드

AWS Serverless Application Model(AWS SAM)을 사용하여 .NET 8 AWS Lambda 함수를 빌드하고 패키징하면 네이티브 어헤드 오브 타임(AOT) 컴파일을 활용하여 AWS Lambda 콜드 스타트 시간을 개선할 수 있습니다.

주제

- [.NET 8 네이티브 AOT 개요]
- [.NET 8 Lambda 함수와 함께 AWS SAM 사용]

- 사전 조건 설치
- AWS SAM 템플릿에서 .NET 8 Lambda 함수 정의
- AWS SAMCLI를 사용한 애플리케이션 구축
- 자세히 알아보기

.NET 8 네이티브 AOT 개요

전통적으로 .NET Lambda 함수의 콜드 스타트 시간은 사용자 경험, 시스템 지연 시간, 서버리스 애플리케이션 사용 비용에 영향을 미칩니다. .NET 네이티브 AOT 컴파일을 사용하면 Lambda 함수의 콜드 스타트 시간을 개선할 수 있습니다. .NET 8용 네이티브 AOT에 대한 자세한 내용은 Dotnet GitHub 리포지토리의 네이티브 AOT 사용을 참조하세요.

.NET 8 Lambda 함수와 함께 AWS SAM 사용

다음을 수행하여 .NET 8 Lambda 함수에서 AWS Serverless Application Model(AWS SAM)을 구성합니다.

- 귀하의 개발 기기에 사전 조건을 설치합니다.
- AWS SAM 템플릿에서 .NET 8 Lambda 함수를 정의합니다.
- AWS SAM CLI를 사용하여 애플리케이션을 구축하십시오.

사전 조건 설치

다음은 필수적 사전 조건입니다.

- 이 AWS SAM CLI
- .NET Core CLI
- Amazon.Lambda.Tools .NET Core Global Tool
- Docker

AWS SAM CLI 설치

1. 이미 AWS SAM CLI가 설치되어 있는지 확인하려면 다음을 실행합니다.

```
sam --version
```

2. AWS SAM CLI을 설치하려면 AWS SAM CLI 설치을 참조하세요.

3. AWS SAM CLI의 설치된 버전을 업그레이드하려면 AWS SAMCLI업그레이드 을 참조하세요.

.NET Core CLI 설치

1. .NET Core CLI를 다운로드하고 설치하려면 Microsoft 웹 사이트에서 .NET 다운로드를 참조하세요.
2. .NET Core CLI에 대한 자세한 내용은 AWS Lambda개발자 가이드의 .NET Core CLI를 참조하세요.

Amazon.Lambda.Tools .NET Core Global Tool 설치

1. 다음 명령을 실행합니다.

```
dotnet tool install -g Amazon.Lambda.Tools
```

2. 이 도구가 이미 설치되어 있으면 다음 명령을 사용하여 최신 버전인지 확인할 수 있습니다.

```
dotnet tool update -g Amazon.Lambda.Tools
```

3. Amazon.Lambda.Tools .NET Core Global에 대한 자세한 내용은 GitHub의 AWS Extensions for .NET CLI 리포지토리를 참조하세요.

Docker 설치

- 네이티브 AOT로 구축하려면 Docker이 설치되어야 합니다. 설치 지침은 AWS SAMCLI와 함께 사용할 Docker 설치 을 확인하십시오.

AWS SAM 템플릿에서 .NET 8 Lambda 함수 정의

AWS SAM 템플릿에서 .NET8 Lambda 함수를 정의하려면 다음을 수행합니다.

1. 선택한 시작 디렉터리에서 다음 명령을 실행합니다.

```
sam init
```

2. `AWS Quick Start Templates`을 선택하여 시작 템플릿을 선택합니다.
3. `Hello World Example` 템플릿을 선택합니다.
4. n을 입력하여 가장 많이 사용하는 런타임 및 패키지 유형을 사용하지 않도록 선택합니다.

5. 런타임에서 `dotnet8`을 선택합니다.
6. 패키지 유형에서 `Zip`을 선택합니다.
7. 스타터 템플릿에서 `Hello World Example using native AOT`를 선택합니다.

Docker 설치

- 네이티브 AOT로 구축하려면 Docker이 설치되어야 합니다. 설치 지침은 [AWS SAMCLI와 함께 사용할 Docker 설치](#) 을 확인하십시오.

```
Resources:
HelloWorldFunction:
  Type: AWS::Serverless::Function
  Properties:
    CodeUri: ./src/HelloWorldAot/
    Handler: bootstrap
    Runtime: dotnet8
    Architectures:
      - x86_64
    Events:
      HelloWorldAot:
        Type: Api
        Properties:
          Path: /hello
          Method: get
```

> **Note**
>
> Event의 `AWS::Serverless::Function` 속성이 `Api`(으)로 설정되었지만 `RestApiId` 속성이 지정되지 않은 경우 AWS SAM이(가) `AWS::ApiGateway::RestApi` CloudFormation 리소스를 생성합니다.

AWS SAMCLI를 사용한 애플리케이션 구축

프로젝트의 루트 디렉터리에서 `sam build` 명령을 실행하여 애플리케이션 구축을 시작합니다. .NET 8 프로젝트 파일에 `PublishAot` 속성을 정의한 경우 AWS SAM CLI가 네이티브 AOT 컴파일을 사용하여 빌드합니다. `PublishAot`속성에 대한 자세한 내용은 Microsoft의 [.NET 설명서](#)에서 네이티브 AOT 배포를 참조하세요.

함수를 구축하려면 AWS SAM CLI가 Amazon.Lambda.Tools .NET Core Global Tool을 사용하는 .NET Core CLI를 간접 호출합니다.

> **Note**
>
> 구축 시 `.sln` 파일이 프로젝트의 동일 또는 상위 디렉터리에 있는 경우 `.sln` 파일이 들어있는 디렉터리가 컨테이너에 마운트됩니다. `.sln` 파일을 찾을 수 없는 경우 프로젝트 폴더만 마운트됩니다. 따라서 다중 프로젝트 응용 프로그램을 구축하는 경우 `.sln` 파일이 속성에 있는지 확인하십시오.

자세히 알아보기

.NET 8 Lambda 함수 빌드에 대한 자세한 내용은 AWS Lambda용 .NET 8 런타임 소개를 참조하세요.

sam build 명령에 대한 참조는 sam build를 확인하세요.

AWS SAM에서 Cargo Lambda를 사용하여 Rust Lambda 함수 빌드

이 기능은 AWS SAM 평가판 릴리스에 있으므로 변경될 수 있습니다.

AWS Serverless Application Model 명령줄 인터페이스(AWS SAM AWS Lambda)를 Rust CLI 함수와 함께 사용합니다.

주제

- 사전 조건
- Rust Lambda 함수와 함께 사용하도록 AWS SAM 구성
- 예시

사전 조건

Rust 언어

Rust를 설치하려면 Rust 언어 웹 사이트의 Rust 설치 섹션을 참조하세요.

Cargo Lambda

AWS SAM CLI에는 Cargo의 하위 명령인 Cargo Lambda의 설치가 필요합니다. 설치에 대한 지침은 Cargo Lambda 설명서의 설치 섹션을 참조하세요.

Docker

Rust Lambda 함수를 빌드하고 테스트하려면 Docker가 필요합니다. 설치 지침은 [Docker 설치](#)을 확인하세요.

AWS SAM CLI 베타 기능에 대한 옵트인

이 기능은 미리 보기 버전이므로 다음 방법 중 하나를 사용하여 옵트인해야 합니다.

1. `SAM_CLI_BETA_RUST_CARGO_LAMBDA=1` 환경 변수를 사용합니다.
2. 다음을 `samconfig.toml` 파일에 추가합니다.

```
[default.build.parameters]
beta_features = true
[default.sync.parameters]
beta_features = true
```

3. 지원되는 AWS SAM CLI 명령을 사용할 때는 이 `--beta-features` 옵션을 사용합니다. 예:

```
$ sam build --beta-features
```

4. AWS SAM CLI에 옵트인하라는 메시지가 표시되면 y 옵션을 선택합니다. 다음은 예제입니다.

```
$ 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
```

Rust Lambda 함수와 함께 사용하도록 AWS SAM 구성

1단계: AWS SAM 템플릿 구성

다음과 같이 AWS SAM 템플릿을 구성합니다.

- 바이너리 - 선택 사항. 템플릿에 여러 Rust Lambda 함수가 포함된 경우를 지정합니다.
- BuildMethod – `rust-cargolambda`.
- CodeUri – `Cargo.toml` 파일에 대한 경로
- 핸들러 – `bootstrap`.
- 런타임 – `provided.al2`.

사용자 지정 런타임에 대해 자세히 알아보려면 AWS Lambda 개발자 안내서의 사용자 지정 AWS Lambda 런타임을 참조하세요.

구성된 AWS SAM 템플릿의 예는 다음과 같습니다.

```
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
...
```

2단계: Rust Lambda 함수와 함께 AWS SAMCLI 사용

CLI 템플릿으로 아무 AWS SAM AWS SAM 명령이나 사용할 수 있습니다. 자세한 내용은 AWS SAM CLI 섹션을 참조하세요.

예시

Hello World 예제

이 예제에서는 Rust를 런타임으로 사용하여 샘플 Hello World 애플리케이션을 빌드합니다.

먼저 `sam init`를 사용하여 새 서버리스 애플리케이션을 초기화합니다. 대화형 흐름 중에 Hello World 애플리케이션을 선택하고 Rust 런타임을 선택합니다.

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

Choose an AWS Quick Start application template
        1 - Hello World Example
```

```
        2 - Multi-step workflow
        3 - Serverless API
        ...
Template: 1

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

Which runtime would you like to use?
        1 - dotnet8
        2 - dotnet6
        3 - go (provided.al2)
        ...
        18 - python3.11
        19 - python3.10
        20 - ruby3.3
        21 - ruby3.2
        22 - rust (provided.al2)
        23 - rust (provided.al2023)
Runtime: 22

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
```

Hello World 애플리케이션의 구조는 다음과 같습니다.

```
hello-rust
### README.md
### events
#   ### event.json
### rust_app
#   ### Cargo.toml
#   ### src
#       ### main.rs
### samconfig.toml
### template.yaml
```

AWS SAM 템플릿에서 Rust 함수는 다음과 같이 정의됩니다.

```
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
```

다음으로 애플리케이션을 빌드하고 배포를 준비하기 위해 `sam build`를 실행합니다. AWS SAM CLI는 `.aws-sam` 디렉터리를 생성하고 여기에 빌드 아티팩트를 구성합니다. 함수는 Cargo Lambda를 사용하여 빌드되고 `.aws-sam/build/HelloWorldFunction/bootstrap`에 실행 가능한 바이너리로 저장됩니다.

> **Note**
>
> MacOS 에서 sam local invoke 명령을 실행하려는 경우 간접적으로 호출하기 전에 다른 함수를 빌드해야 합니다. 이렇게 하려면 다음 명령을 사용합니다.
>
> - SAM_BUILD_MODE=debug sam build
>
> 이 명령은 로컬 테스트를 수행하는 경우에만 필요합니다. 배포를 위해 빌드할 때는 권장되지 않습니다.

```
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
```

다음으로 `sam deploy --guided`를 사용하여 애플리케이션을 배포합니다.

```
hello-rust$ sam deploy --guided

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

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

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

        Looking for resources needed for deployment:

        ...

        Uploading to hello-rust/56ba6585d80577dd82a7eaaee5945c0b  817973 / 817973
 (100.00%)
```

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

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

        Uploading to hello-rust/a4fc54cb6ab75dd0129e4cdb564b5e89.template  1239 / 1239
 (100.00%)


Waiting for changeset to be created..

CloudFormation stack changeset
---------------------------------------------------------------------------------------------------------
Operation                    LogicalResourceId          ResourceType
 Replacement
---------------------------------------------------------------------------------------------------------
+ Add                        HelloWorldFunctionHelloW   AWS::Lambda::Permission    N/A

                             orldPermissionProd

...
---------------------------------------------------------------------------------------------------------

Changeset created successfully. arn:aws:cloudformation:us-
west-2:012345678910:changeSet/samcli-deploy1681427201/
f0ef1563-5ab6-4b07-9361-864ca3de6ad6


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

2023-04-13 13:07:17 - Waiting for stack create/update to complete
```

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

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

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

테스트를 위해 API 엔드포인트를 사용하여 Lambda 함수를 호출할 수 있습니다.

```
$ curl https://ggdxec9le9.execute-api.us-west-2.amazonaws.com/Prod/hello/
Hello World!%
```

함수를 로컬에서 테스트하려면 먼저 함수의 `Architectures` 속성이 로컬 시스템과 일치하는지 확인합니다.

```
...
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
...
```

이 예제에서는 아키텍처를 `x86_64`에서 `arm64`로 수정했으므로 빌드 아티팩트를 업데이트하기 위해 `sam build`를 실행합니다. 그런 다음 `sam local invoke`를 실행하여 함수를 로컬에서 호출합니다.

```
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
```

단일 Lambda 함수 프로젝트

다음은 Rust Lambda 함수 하나를 포함하는 서버리스 애플리케이션의 예입니다.

프로젝트 디렉터리 구조:

```
.
### Cargo.lock
### Cargo.toml
### src
#   ### main.rs
### template.yaml
```

AWS SAM 템플릿:

```
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
...
```

다중 Lambda 함수 프로젝트

다음은 여러 Rust Lambda 함수를 포함하는 서버리스 애플리케이션의 예입니다.

프로젝트 디렉터리 구조:

```
.
### Cargo.lock
### Cargo.toml
```

```
### src
#   ### function_a.rs
#   ### function_b.rs
### template.yaml
```

AWS SAM 템플릿:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  FunctionA:
    Type: AWS::Serverless::Function
    Metadata:
      BuildMethod: rust-cargolambda
      BuildProperties:
        Binary: function_a
    Properties:
      CodeUri: ./
      Handler: bootstrap
      Runtime: provided.al2
  FunctionB:
    Type: AWS::Serverless::Function
    Metadata:
      BuildMethod: rust-cargolambda
      BuildProperties:
        Binary: function_b
    Properties:
      CodeUri: ./
      Handler: bootstrap
      Runtime: provided.al2
```

`Cargo.toml` 파일:

```
[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"
```

AWS SAM에서 사용자 지정 런타임을 사용하여 Lambda 함수 빌드

`sam build` 명령을 사용하여 귀하의 Lambda 함수에 필요한 사용자 지정 런타임을 구축할 수 있습니다. Lamda 함수를 선언하여 사용자 지정 런타임을 사용하기 위해 해당 Lambda 함수에 대해 `Runtime: provided`을 지정하십시오.

사용자 지정 런타임을 구축하려면 `Metadata` 항목이 있는 `BuildMethod: makefile` 리소스 속성을 선언하십시오. 런타임용 빌드 명령이 포함된 양식의 빌드 대상을 `build-`*`function-logical-id`* 선언하는 사용자 지정 makefile을 제공하십시오. 귀하의 makefile은 필요한 경우 사용자 지정 런타임을 컴파일하고 워크플로의 후속 단계에 필요한 적절한 위치에 빌드 아티팩트를 복사합니다. makefile의 위치는 함수 리소스의 `CodeUri` 속성으로 지정되며 `Makefile`라는 이름을 지정해야 합니다.

예시

예제 1: Rust로 작성된 함수의 사용자 지정 런타임

> **Note**
>
> Cargo Lambda을 사용해 Lambda 함수를 구축하는 것이 좋습니다. 자세한 내용은 AWS SAM에서 Cargo Lambda를 사용하여 Rust Lambda 함수 빌드을 참조하십시오.

다음 AWS SAM 템플릿은 Rust로 작성된 Lambda 함수의 사용자 지정 런타임을 사용할 함수를 선언하고 `sam build` 빌드 대상에 대한 `build-HelloRustFunction` 명령을 실행하도록 지시합니다.

```
Resources:
  HelloRustFunction:
    Type: AWS::Serverless::Function
    Properties:
```

```
      FunctionName: HelloRust
      Handler: bootstrap.is.real.handler
      Runtime: provided
      MemorySize: 512
      CodeUri: .
    Metadata:
      BuildMethod: makefile
```

다음 makefile에는 빌드 대상과 실행될 명령이 포함되어 있습니다. 참고로 `CodeUri` 속성은 `.`로 설정되어 있으므로 makefile은 프로젝트 루트 디렉터리(즉, 애플리케이션의 AWS SAM 템플릿 파일과 동일한 디렉터리)에 있어야 합니다. 파일 이름은 `Makefile`이어야 합니다.

```
build-HelloRustFunction:
 cargo build --release --target x86_64-unknown-linux-musl
 cp ./target/x86_64-unknown-linux-musl/release/bootstrap $(ARTIFACTS_DIR)
```

`cargo build` 이전 명령어를 실행`makefile`하기 위한 개발 환경 설정의 자세한 내용은 블로그 게시물용 [Rust RuntimeAWS Lambda](https://aws.amazon.com/blogs/opensource/rust-runtime-for-aws-lambda/)을 참조하세요.

예제 2: Python3.12용 Makefile 빌더(번들 빌더 사용의 대안)

번들 빌더에 포함되지 않은 라이브러리 또는 모듈을 사용하고 싶을 수도 있습니다. 이 예제는 makefile 빌더가 있는 Python3.12 런타임용 AWS SAM 템플릿을 보여줍니다.

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: hello_world/
      Handler: app.lambda_handler
      Runtime: python3.12
    Metadata:
      BuildMethod: makefile
```

다음 makefile에는 빌드 대상과 실행될 명령이 포함되어 있습니다. 참고로 `CodeUri` 속성은 `hello_world`로 설정되었으므로 makefile은 `hello_world` 하위 디렉토리의 루트에 있어야 하고 파일 이름은 `Makefile`이어야 합니다.

```
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
```

에서 Lambda 계층 빌드 AWS SAM

AWS SAM 를 사용하여 사용자 지정 Lambda 계층을 빌드할 수 있습니다. Lambda 계층을 사용하면 Lambda 함수에서 코드를 추출한 다음 여러 Lambda 함수에서 재사용할 수 있습니다. 전체 애플리케이션을 빌드하는 대신 Lambda 계층만 빌드함으로써 몇 가지 이점을 얻을 수 있습니다. 예를 들어 배포 패키지의 크기를 줄이고, 핵심 함수 로직을 종속성과 분리하며, 여러 함수에서 종속성을 공유할 수 있습니다. 레이어에 대한 자세한 내용을 알아보려면 AWS 개발자 가이드 내 AWS Lambda Lambda 레이어를 확인하십시오.

에서 Lambda 계층을 빌드하는 방법 AWS SAM

> **Note**
>
> Lambda 계층을 빌드하려면 먼저 AWS SAM 템플릿에 Lambda 계층을 작성해야 합니다. 이 작업 수행과 관련된 자세한 내용과 예제는 AWS SAM에서 Lambda 계층을 사용하여 효율성 향상 섹션을 참조하세요.

사용자 지정 계층을 빌드하려면 AWS Serverless Application Model (AWS SAM) 템플릿 파일에 계층을 선언하고 `Metadata` 리소스 속성 섹션을 `BuildMethod` 항목과 함께 포함합니다. `BuildMethod`의 유효한 값은 AWS Lambda 런타임의 혹은 `makefile`의 식별자입니다. 레이어가 지원하는 명령어 세트 아키텍처를 지정하기 위해 `BuildArchitecture`을 포함하십시오. `BuildArchitecture`의 유효한 값은 Lambda 명령 세트 아키텍처입니다.

`makefile`을 지정하는 경우 사용자 지정 makefile을 제공하고 거기에서 레이어의 빌드 명령이 포함된 `build-`*`layer-logical-id`* 양식의 빌드 대상을 선언하십시오. makefile은 필요한 경우 레이어를 컴파일하고 워크플로의 후속 단계에 필요한 적절한 위치에 빌드 아티팩트를 복사합니다. makefile의 위치는 레이어 리소스의 `ContentUri` 속성으로 지정되며 `Makefile`라는 이름을 지정해야 합니다.

> **Note**
>
> 사용자 지정 계층을 생성할 때는 환경 변수에 AWS Lambda 따라 계층 코드를 찾습니다. Lambda 런타임에는 레이어 코드가 복사되어 들어가는 `/opt` 디렉토리의 경로가 포함됩니다. 사용자 지정 레이어 코드를 찾을 수 있도록 프로젝트의 빌드 아티팩트 폴더 구조가 런타임의 예상 폴더 구조와 일치해야 합니다.

예를 들어, Python의 경우 귀하의 코드를 python/ 하위 디렉터리에 배치할 수 있습니다. NodeJS의 경우 코드를 nodejs/node_modules/ 하위 디렉터리에 배치할 수 있습니다.

자세한 내용은 개발자 가이드의 레이어에 AWS Lambda 라이브러리 종속 항목 포함하기를 참조하세요.

다음은 예제 Metadata 리소스 속성 섹션입니다.

```
Metadata:
  BuildMethod: python3.12
  BuildArchitecture: arm64
```

Note

Metadata 리소스 속성 섹션을 포함하지 않으면 계층을 빌드 AWS SAM 하지 않습니다. 대신 그것은 레이어 리소스의 CodeUri 속성에 지정된 위치에서 빌드 아티팩트를 복사합니다. 자세한 내용은 리소스 유형의 AWS::Serverless::LayerVersionContentUri 속성을 참조하세요.

Metadata 리소스 속성 섹션을 포함할 때 sam build 명령을 사용하여 독립 객체 또는 AWS Lambda 함수의 종속성으로 계층을 빌드할 수 있습니다.

- 독립 객체로서. 예를 들어 전체 애플리케이션을 빌드하지 않아도 레이어에 대한 코드 변경을 로컬에서 테스트하는 경우에는 레이어 객체만 빌드하는 것이 좋습니다. 레이어를 독립적으로 구축하려면 sam build *layer-logical-id* 명령으로 레이어 리소스를 지정하십시오.
- Lambda 함수의 종속 항목으로서. 동일한 Layers 템플릿 파일에 있는 Lambda 함수의 AWS SAM 속성에 레이어의 논리적 ID를 포함시키면 레이어는 해당 Lambda 함수의 종속 항목이 됩니다. 해당 레이어에 Metadata 항목이 포함된 BuildMethod 리소스 속성 섹션도 포함된 경우 sam build 명령으로 전체 애플리케이션을 빌드하거나 sam build *function-logical-id* 명령으로 함수 리소스를 지정하여 레이어를 구축합니다.

예제

템플릿 예제 1: Python 3.12 런타임 환경에 대한 계층 구축

다음 예제 AWS SAM 템플릿은 Python 3.12 런타임 환경에 대한 계층을 빌드합니다.

```
Resources:
  MyLayer:
    Type: AWS::Serverless::LayerVersion
    Properties:
      ContentUri: my_layer
      CompatibleRuntimes:
        - python3.12
    Metadata:
      BuildMethod: python3.12   # Required to have AWS SAM build this layer
```

템플릿 예제 2: 사용자 지정 makefile을 사용하여 레이어 구축

다음 예제 AWS SAM 템플릿은 사용자 지정를 사용하여 계층`makefile`을 빌드합니다.

```
Resources:
  MyLayer:
    Type: AWS::Serverless::LayerVersion
    Properties:
      ContentUri: my_layer
      CompatibleRuntimes:
        - python3.12
    Metadata:
      BuildMethod: makefile
```

다음`makefile`은 빌드 타겟과 실행될 명령을 포함합니다. 참고로 `ContentUri` 속성은 `my_layer`로 설정되었으므로 makefile은 `my_layer` 하위 디렉토리의 루트에 있어야 하고 파일 이름은 `Makefile`이어야 합니다. 또한 빌드 아티팩트는 `python/` 하위 디렉터리로 복사되므로 AWS Lambda에서 계층 코드를 찾을 수 있습니다.

```
build-MyLayer:
  mkdir -p "$(ARTIFACTS_DIR)/python"
  cp *.py "$(ARTIFACTS_DIR)/python"
  python -m pip install -r requirements.txt -t "$(ARTIFACTS_DIR)/python"
```

> **Note**
>
> `makefile`이(가) 호출되면 적절한 대상이 트리거되고 아티팩트가 노출된 환경 변수 `$ARTIFACTS_DIR`에 복사되어야 합니다. 자세한 내용은 [GitHub의 aws-lambda-builders]를 참조하세요.

sam build 명령어 예시

다음 `sam build` 명령은 `Metadata` 리소스 속성 섹션을 포함하는 레이어를 구축합니다.

```
# 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
```

를 사용하여 서버리스 애플리케이션 테스트 AWS SAM

애플리케이션을 작성하고 빌드한 후에는 애플리케이션이 제대로 작동하는지 확인하기 위해 애플리케이션을 테스트해야 합니다. AWS SAM 명령줄 인터페이스(CLI)를 사용하면 AWS 클라우드에 업로드하기 전에 서버리스 애플리케이션을 로컬에서 테스트할 수 있습니다. 애플리케이션을 테스트하면 애플리케이션의 기능, 안정성 및 성능을 모두 확인하는 동시에 해결해야 할 문제(버그)를 식별할 수 있습니다.

이 섹션에서는 애플리케이션을 테스트할 때 참조할 수 있는 일반적인 관행에 대한 지침을 제공합니다. 이 섹션의 주제는 주로 AWS 클라우드에 배포하기 전에 수행할 수 있는 로컬 테스트에 중점을 둡니다. 배포 전 테스트는 문제를 사전에 식별하여 배포 문제와 관련된 불필요한 비용을 줄이는 데 도움을 줍니다. 이 섹션의 각 주제에서는 수행할 수 있는 테스트를 설명하고, yoxi include href? u 사용의 이점을 설명하고, 테스트 수행 방법을 보여주는 예제를 포함합니다. 애플리케이션을 테스트한 후에는 발견된 문제를 디버깅해야 합니다.

주제

- sam local 명령을 사용한 테스트 소개
- AWS SAM에서 Lambda 함수를 로컬로 호출
- AWS SAM에서 로컬로 API Gateway 실행
- sam remote test-event를 사용한 클라우드 테스트 소개
- sam remote invoke를 사용하여 클라우드에서 테스트 소개
- AWS SAM에서 로컬 통합 테스트 자동화
- AWS SAM에서 샘플 이벤트 페이로드 생성
- 내구성 함수 테스트 및 디버깅

sam local 명령을 사용한 테스트 소개

AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) `sam local` 명령을 사용하여 서버리스 애플리케이션을 로컬에서 테스트할 수 있습니다.

AWS SAM CLI에 대한 소개는 란 무엇입니까 AWS SAMCLI? 섹션을 참조하세요.

사전 조건

`sam local`를 사용하려면 다음을 완료하여 AWS SAM CLI를 설치합니다.

- AWS SAM 사전 조건.
- AWS SAM CLI 설치.

`sam local`를 사용하기 전에 다음 사항에 대한 기본적인 이해를 하는 것이 좋습니다.

- AWS SAM CLI 구성.
- 에서 애플리케이션 생성 AWS SAM.
- AWS SAM의 빌드 소개.
- AWS SAM 배포 소개.

sam local 명령 사용

`sam local` 명령을 해당 하위 명령과 함께 사용하여 애플리케이션에 대해 다양한 유형의 로컬 테스트를 수행할 수 있습니다.

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

각 하위 명령에 대해 자세히 알아보려면 다음을 참조하세요.

- sam local generate-event 소개 - 로컬 테스트용 AWS 서비스 이벤트를 생성합니다.
- sam local invoke 소개 - 로컬에서 AWS Lambda 함수의 일회성 호출을 시작합니다.
- sam local start-api 소개 - 로컬 HTTP 서버를 사용하여 Lambda 함수를 실행합니다.
- sam local start-lambda 소개 - AWS CLI 또는 SDK와 함께 사용하기 위해 로컬 HTTP 서버를 사용하여 Lambda 함수를 실행합니다.

sam local generate-event를 사용한 테스트 소개

AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) `sam local generate-event` 하위 명령을 사용하여 지원되는 AWS 서비스에 대한 이벤트 페이로드 샘플을 생성합니다. 그런 다음 이러한 이벤트를 수정하여 로컬 리소스로 전달해 테스트를 수행할 수 있습니다.

- AWS SAM CLI에 대한 소개는 란 무엇입니까 AWS SAMCLI? 섹션을 참조하세요.
- `sam local generate-event` 명령 옵션 목록은 sam local generate-event 섹션을 참조하세요.

이벤트는 AWS 서비스가 작업 또는 태스크를 수행할 때 생성되는 JSON 객체입니다. 이러한 이벤트에는 처리된 데이터나 이벤트의 타임스탬프와 같은 특정 정보가 포함됩니다. 대부분의 AWS 서비스는 이벤트를 생성하고 각 서비스의 이벤트는 해당 서비스에 맞게 고유한 형식으로 지정됩니다.

한 서비스에서 생성된 이벤트는 다른 서비스에 이벤트 소스로 전달됩니다. 예를 들어 Amazon Simple Storage Service(S3) 버킷에 배치된 항목은 이벤트를 생성할 수 있습니다. 그러면 이 이벤트를 AWS Lambda 함수의 이벤트 소스로 사용하여 데이터를 추가로 처리할 수 있습니다.

`sam local generate-event`를 사용하여 생성하는 이벤트는 AWS 서비스에서 생성한 실제 이벤트와 동일한 구조로 형식이 지정됩니다. 이러한 이벤트의 콘텐츠를 수정하고 이를 사용하여 애플리케이션의 리소스를 테스트할 수 있습니다.

사전 조건

`sam local generate-event`를 사용하려면 다음을 완료하여 AWS SAM CLI를 설치합니다.

- AWS SAM 사전 조건.
- AWS SAM CLI 설치.

`sam local generate-event`를 사용하기 전에 다음 사항에 대한 기본적인 이해를 하는 것이 좋습니다.

- AWS SAM CLI 구성.
- 에서 애플리케이션 생성 AWS SAM.
- AWS SAM의 빌드 소개.
- AWS SAM 배포 소개.

샘플 이벤트 생성

AWS SAM CLI `sam local generate-event` 하위 명령을 사용하여 지원되는 AWS 서비스에 대한 이벤트를 생성합니다.

지원되는 AWS 서비스의 목록을 보려면

1. 다음을 실행합니다.

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

2. 지원되는 AWS 서비스의 목록이 표시됩니다. 다음은 예제입니다.

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

로컬 이벤트를 생성하려면

1. `sam local generate-event`를 실행하고 지원되는 서비스 이름을 제공합니다. 그러면 생성할 수 있는 이벤트 유형 목록이 표시됩니다. 다음은 예제입니다.

```
$ sam local generate-event s3

Usage: sam local generate-event s3 [OPTIONS] COMMAND [ARGS]...

Options:
  -h, --help  Show this message and exit.

Commands:
  batch-invocation  Generates an Amazon S3 Batch Operations Invocation Event
  delete            Generates an Amazon S3 Delete Event
  put               Generates an Amazon S3 Put Event
```

2. 샘플 이벤트를 생성하려면 `sam local generate-event`를 실행하여 서비스와 이벤트 유형을 제공합니다.

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

다음은 예제입니다.

```
$ sam local generate-event s3 put
{
  "Records": [
    {
```

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

이 샘플 이벤트에는 자리 표시자 값이 포함되어 있습니다. 애플리케이션의 실제 리소스를 참조하거나 로컬 테스트에 도움이 되는 값을 참조하도록 이러한 값을 수정할 수 있습니다.

샘플 이벤트를 수정하려면

1. 명령 프롬프트에서 샘플 이벤트를 수정할 수 있습니다. 옵션을 보려면 다음을 실행합니다.

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

다음은 예제입니다.

```
$ sam local generate-event s3 put --help

Usage: sam local generate-event s3 put [OPTIONS]

Options:
  --region TEXT       Specify the region name you'd like, otherwise the
                      default = us-east-1
  --partition TEXT    Specify the partition name you'd like, otherwise the
                      default = aws
  --bucket TEXT       Specify the bucket name you'd like, otherwise the
                      default = example-bucket
  --key TEXT          Specify the key name you'd like, otherwise the default =
                      test/key
  --debug             Turn on debug logging to print debug message generated
                      by AWS SAM CLI and display timestamps.
  --config-file TEXT  Configuration file containing default parameter values.
                      [default: samconfig.toml]
  --config-env TEXT   Environment name specifying default parameter values in
                      the configuration file.  [default: default]
  -h, --help          Show this message and exit.
```

2. 명령 프롬프트에서 이러한 옵션 중 하나를 사용하여 샘플 이벤트 페이로드를 수정합니다. 다음은 예제입니다.

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

{
  "Records": [
    {
      "eventVersion": "2.0",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-1",
      "eventTime": "1970-01-01T00:00:00.000Z",
      "eventName": "ObjectCreated:Put",
```

```
        "userIdentity": {
          "principalId": "EXAMPLE"
        },
        "requestParameters": {
          "sourceIPAddress": "127.0.0.1"
        },
        "responseElements": {
          "x-amz-request-id": "EXAMPLE123456789",
          "x-amz-id-2": "EXAMPLE123/5678abcdefghijklambdaisawesome/
mnopqrstuvwxyzABCDEFGH"
        },
        "s3": {
          "s3SchemaVersion": "1.0",
          "configurationId": "testConfigRule",
          "bucket": {
            "name": "sam-s3-demo-bucket",
            "ownerIdentity": {
              "principalId": "EXAMPLE"
            },
            "arn": "arn:aws:s3:::sam-s3-demo-bucket"
          },
          "object": {
            "key": "test/key",
            "size": 1024,
            "eTag": "0123456789abcdef0123456789abcdef",
            "sequencer": "0A1B2C3D4E5F678901"
          }
        }
      }
    ]
}
```

로컬 테스트에 생성된 이벤트 사용

생성된 이벤트를 로컬에 저장하고 다른 `sam local` 하위 명령을 사용하여 테스트를 수행합니다.

생성된 이벤트를 로컬에 저장하려면

- 다음을 실행합니다.

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

다음은 프로젝트 events 폴더에 s3.json 파일로 저장되는 이벤트에 대한 예시입니다.

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

로컬 테스트에 생성된 이벤트를 사용하려면

- --event 옵션을 사용하여 다른 sam local 하위 명령과 함께 이벤트를 전달합니다.

  다음은 s3.json 이벤트를 사용하여 Lambda 함수를 로컬에서 호출하는 예제입니다.

```
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
```

자세히 알아보기

sam local generate-event 옵션 목록은 sam local generate-event 섹션을 참조하세요.

sam local 사용 데모는 로컬 개발용 AWS SAM 섹션을 참조하세요. YouTube의 SAM 시리즈를 사용한 서버리스 랜드 세션의 로컬 개발 환경의 AWS 클라우드 리소스를 테스트합니다.

sam local invoke를 사용한 테스트 소개

AWS Serverless Application Model 명령줄 인터페이스(AWS SAMCLI) sam local invoke 하위 명령을 사용하여 로컬에서 AWS Lambda 함수의 일회성 호출을 시작합니다.

- 에 대한 소개는 단원을 AWS SAMCLI참조하십시오. 란 무엇입니까 AWS SAMCLI?
- sam local invoke 명령 옵션 목록은 sam local invoke 섹션을 참조하세요.

- 일반적인 개발 워크플로 중 sam local invoke를 사용하는 예는 7단계: (선택 사항) 로컬에서 애플리케이션 테스트 섹션을 참조하세요.

> **Note**
>
> sam local invoke는 자동 체크포인트 및 재생 기능을 갖춘 내구성 있는 함수를 지원합니다. 내구성이 뛰어난 함수를 로컬에서 호출하면 실행 상태가 자동으로 관리됩니다.

사전 조건

sam local invoke를 사용하려면 다음을 완료하여 AWS SAM CLI를 설치합니다.

- AWS SAM 사전 조건.
- AWS SAM CLI 설치.

sam local invoke를 사용하기 전에 다음 사항에 대한 기본적인 이해를 하는 것이 좋습니다.

- AWS SAM CLI 구성.
- 에서 애플리케이션 생성 AWS SAM.
- AWS SAM의 빌드 소개.
- AWS SAM 배포 소개.

Lambda 함수를 호출합니다.

sam local invoke를 실행하면 AWS SAMCLI는 현재 작업 디렉터리가 프로젝트의 루트 디렉터리라고 가정합니다. AWS SAMCLI는 먼저 .aws-sam 하위 폴더에서 template.[yaml|yml] 파일을 찾습니다. 찾을 수 없는 경우 AWS SAMCLI는 현재 작업 디렉터리 내에서 template.[yaml|yml] 파일을 찾습니다.

Lambda 함수를 로컬에서 호출하려면

1. 프로젝트 디렉터리에서 다음 명령을 실행합니다.

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

2. 애플리케이션에 둘 이상의 함수가 포함된 경우 함수의 논리적 ID를 제공합니다. 다음은 예제입니다.

```
$ sam local invoke HelloWorldFunction
```

3. AWS SAMCLI는 Docker를 사용하여 로컬 컨테이너에 함수를 빌드합니다. 그런 다음 함수를 호출하고 함수의 응답을 출력합니다.

   다음은 예제입니다.

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

로그 관리

`sam local invoke`를 사용하면 Lambda 함수 런타임 출력(예: 로그)이 `stderr`로 출력되고, Lambda 함수 결과는 `stdout`로 출력됩니다.

다음은 기본 Lambda 함수 출력의 예제입니다.

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

이러한 표준 출력을 저장할 수 있습니다. 다음은 예제입니다.

```
$ 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
```

이러한 표준 출력을 사용하여 로컬 개발 프로세스를 더욱 자동화할 수 있습니다.

옵션

사용자 지정 이벤트를 전달하여 Lambda 함수 호출

Lambda 함수에 이벤트를 전달하려면 `--event` 옵션을 사용합니다. 다음은 예제입니다.

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

`sam local generate-event` 하위 명령으로 이벤트를 생성할 수 있습니다. 자세한 내용은 [sam local generate-event를 사용한 테스트 소개](#) 섹션을 참조하세요.

Lambda 함수를 호출할 때 환경 변수 전달

Lambda 함수가 환경 변수를 사용하는 경우 `--env-vars` 옵션을 사용하여 로컬 테스트 중에 환경 변수를 전달할 수 있습니다. 이는 클라우드에 이미 배포된 애플리케이션의 서비스를 사용하여 Lambda 함수를 로컬에서 테스트할 수 있는 좋은 방법입니다. 다음은 예제입니다.

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

템플릿 또는 함수 지정

AWS SAMCLI가 참조할 템플릿을 지정하려면 `--template` 옵션을 사용합니다. AWS SAMCLI는 해당 AWS SAM 템플릿과 해당 템플릿이 가리키는 리소스만 로드합니다.

중첩된 애플리케이션 또는 스택의 함수를 호출하려면 함수 논리적 ID와 함께 애플리케이션 또는 스택 논리 ID를 제공합니다. 다음은 예제입니다.

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

Terraform 프로젝트에서 Lambda 함수 테스트

`--hook-name` 옵션을 사용하여 Terraform 프로젝트에서 Lambda 함수를 로컬에서 테스트할 수 있습니다. 자세한 내용은 [로컬 디버깅 및 테스트에 Terraform과 함께 AWS SAMCLI 사용하기]를 참조하세요.

다음은 예제입니다.

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

모범 사례

애플리케이션에 실행 중인 `sam build`의 `.aws-sam` 디렉터리가 있는 경우 함수 코드를 업데이트할 때마다 `sam build`를 실행해야 합니다. 그런 다음 `sam local invoke`를 실행하여 업데이트된 함수 코드를 로컬에서 테스트합니다.

로컬 테스트는 클라우드에 배포하기 전에 빠르게 개발하고 테스트할 수 있는 훌륭한 솔루션입니다. 하지만 로컬 테스트는 클라우드의 리소스 간 권한 등 모든 것을 검증하지는 않습니다. 가능한 한 클라우드에서 애플리케이션을 테스트하세요. 클라우드 테스트 워크플로의 속도를 높이려면 [`sam sync`를 사용]하는 것이 좋습니다.

예제

Amazon API Gateway 샘플 이벤트를 생성하고 이를 사용하여 Lambda 함수를 로컬에서 호출합니다.

먼저 API Gateway HTTP API 이벤트 페이로드를 생성하여 `events` 폴더에 저장합니다.

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

다음으로 이벤트에서 파라미터 값을 반환하도록 Lambda 함수를 수정합니다.

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

다음으로 Lambda 함수를 로컬에서 호출하고 사용자 지정 이벤트를 제공합니다.

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

Lambda 함수를 로컬에서 호출할 때 환경 변수 전달

이 애플리케이션에는 Amazon DynamoDB 테이블 이름에 환경 변수를 사용하는 Lambda 함수가 있습니다. 다음은 AWS SAM 템플릿에 정의된 함수의 예입니다.

```
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
...
```

Lambda 함수를 클라우드의 DynamoDB 테이블과 상호 작용하도록 하면서 로컬에서 테스트하려고 합니다. 이를 위해 환경 변수 파일을 만들고 프로젝트의 루트 디렉터리에 `locals.json`로 저장합니다. 여기 SAMPLE_TABLE에 제공된 값은 클라우드의 DynamoDB 테이블을 참조합니다.

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

다음으로 `sam local invoke`를 실행하고, `--env-vars` 옵션과 함께 환경 변수로 전달합니다.

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

자세히 알아보기

`sam local invoke` 옵션 목록은 sam local invoke 섹션을 참조하세요.

`sam local` 사용 데모는 로컬 개발용AWS SAM 섹션을 참조하세요. 의 SAM 시리즈를 사용하여 서버리스 토지 세션의 로컬 개발 환경에서 AWS 클라우드 리소스를 테스트합니다. YouTube

sam local start-api를 사용한 테스트 소개

AWS Serverless Application Model 명령줄 인터페이스(AWS SAMCLI) `sam local start-api` 하위 명령을 사용하여 로컬에서 함수를 실행하고 로컬 HTTP 서버 호스트를 통해 테스트합니다 AWS

Lambda . 이 유형의 테스트는 Amazon API Gateway 엔드포인트에서 호출되는 Lambda 함수에 유용합니다.

- 에 대한 소개는 단원을 AWS SAMCLI참조하십시오. 란 무엇입니까 AWS SAMCLI?
- `sam local start-api` 명령 옵션 목록은 sam local start-api 섹션을 참조하세요.
- 일반적인 개발 워크플로 중 `sam local start-api`를 사용하는 예는 7단계: (선택 사항) 로컬에서 애플리케이션 테스트 섹션을 참조하세요.

> **Note**
>
> `sam local start-api`는 자동 체크포인트 및 재생 기능을 갖춘 내구성 있는 함수를 지원합니다. 내구성 함수는 로컬 API Gateway 엔드포인트와 원활하게 작동합니다.

사전 조건

`sam local start-api`를 사용하려면 다음을 완료하여 AWS SAM CLI를 설치합니다.

- AWS SAM 사전 조건.
- AWS SAM CLI 설치.

`sam local start-api`를 사용하기 전에 다음 사항에 대한 기본적인 이해를 하는 것이 좋습니다.

- AWS SAM CLI 구성.
- 에서 애플리케이션 생성 AWS SAM.
- AWS SAM의 빌드 소개.
- AWS SAM 배포 소개.

sam local start-api 사용하기

`sam local start-api`를 실행하면 AWS SAMCLI는 현재 작업 디렉터리가 프로젝트의 루트 디렉터리라고 가정합니다. AWS SAMCLI는 먼저 `.aws-sam` 하위 폴더에서 `template.[yaml|yml]` 파일을 찾습니다. 찾을 수 없는 경우 AWS SAMCLI는 현재 작업 디렉터리 내에서 `template.[yaml|yml]` 파일을 찾습니다.

로컬 HTTP 서버를 시작하려면

1. 프로젝트 디렉터리에서 다음 명령을 실행합니다.

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

2. AWS SAMCLI는 Lambda 함수를 로컬 Docker 컨테이너에 빌드합니다. 그런 다음 HTTP 서버 엔드포인트의 로컬 주소를 출력합니다. 다음은 예제입니다.

```
$ 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. 브라우저 또는 명령 프롬프트를 통해 Lambda 함수를 호출할 수 있습니다. 다음은 예제입니다.

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

4. Lambda 함수 코드를 변경할 때는 다음 사항을 고려하여 로컬 HTTP 서버를 새로 고칩니다.

   - 애플리케이션에 `.aws-sam` 디렉터리가 없고 함수가 해석된 언어를 사용하는 경우, AWS SAMCLI는 새 컨테이너를 생성하고 호스팅하여 함수를 자동으로 업데이트합니다.
   - 애플리케이션에 `.aws-sam` 디렉터리가 있는 경우 함수를 업데이트하려면 `sam build`를 실행해야 합니다. 그런 다음 다시 `sam local start-api`를 실행하여 함수를 호스팅합니다.

- 함수가 컴파일된 언어를 사용하거나 프로젝트에 복잡한 패키징 지원이 필요한 경우, 자체 빌드 솔루션을 실행하여 함수를 업데이트합니다. 그런 다음 다시 `sam local start-api`를 실행하여 함수를 호스팅합니다.

Lambda 권한 부여자를 사용하는 Lambda 함수

> **Note**
>
> AWS SAM CLI 버전 1.80.0의 새로운 기능 업그레이드하려면 [AWS SAMCLI업그레이드](#) 섹션을 참조하세요.

Lambda 권한 부여자를 사용하는 Lambda 함수의 경우 AWS SAMCLI는 Lambda 함수 엔드포인트를 호출하기 전에 Lambda 권한 부여자를 자동으로 호출합니다.

다음은 Lambda 권한 부여자를 사용하는 함수의 로컬 HTTP 서버를 시작하는 예제입니다.

```
$ 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
```

로컬 HTTP 서버를 통해 Lambda 함수 엔드포인트를 호출하면 AWS SAMCLI가 먼저 Lambda 권한 부여자를 호출합니다. 승인이 성공하면 AWS SAMCLI가 Lambda 함수 엔드포인트를 호출합니다. 다음은 예제입니다.

```
$ curl http://127.0.0.1:3000/authorized-request --header "header:my_token"
{"message": "from authorizer"}%

Invoking app.authorizer_handler (python3.8)
Local image is up-to-date
Using local image: public.ecr.aws/lambda/python:3.8-rapid-x86_64.

Mounting /Users/.../sam-app/... as /var/task:ro,delegated, inside runtime container
START RequestId: 38d3b472-a2c8-4ea6-9a77-9b386989bef0 Version: $LATEST
END RequestId: 38d3b472-a2c8-4ea6-9a77-9b386989bef0
REPORT RequestId: 38d3b472-a2c8-4ea6-9a77-9b386989bef0    Init Duration: 1.08 ms
 Duration: 628.26 msBilled Duration: 629 ms    Memory Size: 128 MB    Max Memory Used:
 128 MB
Invoking app.request_handler (python3.8)
Using local image: public.ecr.aws/lambda/python:3.8-rapid-x86_64.

Mounting /Users/.../sam-app/... as /var/task:ro,delegated, inside runtime container
START RequestId: fdc12255-79a3-4365-97e9-9459d06446ff Version: $LATEST
END RequestId: fdc12255-79a3-4365-97e9-9459d06446ff
REPORT RequestId: fdc12255-79a3-4365-97e9-9459d06446ff    Init Duration: 0.95 ms
 Duration: 659.13 msBilled Duration: 660 ms    Memory Size: 128 MB    Max Memory Used:
 128 MB
No Content-Type given. Defaulting to 'application/json'.
2023-04-17 15:03:03 127.0.0.1 - - [17/Apr/2023 15:03:03] "GET /authorized-request
 HTTP/1.1" 200 -
```

옵션

컨테이너를 지속적으로 재사용하여 로컬 함수 호출 속도 높이기

기본적으로 AWS SAMCLI는 함수가 로컬 HTTP 서버를 통해 호출될 때마다 새 컨테이너를 만듭니다. `--warm-containers` 옵션을 사용하면 함수 호출에 컨테이너를 자동으로 재사용할 수 있습니다. 이렇게 하면 AWS SAMCLI가 로컬 호출을 위해 Lambda 함수를 준비하는 데 걸리는 시간이 단축됩니다. `eager` 또는 `lazy` 인수를 제공하여 이 옵션을 추가로 사용자 지정할 수 있습니다.

- `eager` - 모든 함수의 컨테이너는 시작 시 로드되며 호출 이후에도 유지됩니다.
- `lazy` - 컨테이너는 각 함수를 처음 호출할 때만 로드됩니다. 그런 다음 추가 호출 시에도 계속 유지됩니다.

다음은 예제입니다.

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

`--warm-containers`를 사용하고 Lambda 함수 코드를 수정할 때:

- 애플리케이션에 `.aws-sam` 디렉터리가 있는 경우 `sam build`를 실행하여 애플리케이션 빌드 아티팩트의 함수 코드를 업데이트합니다.
- 코드 변경이 감지되면 AWS SAMCLI가 Lambda 함수 컨테이너를 자동으로 종료합니다.
- 함수를 다시 호출하면 AWS SAMCLI가 자동으로 새 컨테이너를 생성합니다.

Lambda 함수에 사용할 컨테이너 이미지 지정

기본적으로 AWS SAMCLI는 Amazon Elastic Container Registry(Amazon ECR)에서 Lambda 기본 이미지를 사용하여 로컬에서 함수를 호출합니다. `--invoke-image` 옵션을 사용하여 사용자 지정 컨테이너 이미지를 참조할 수 있습니다. 다음은 예제입니다.

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

사용자 지정 컨테이너 이미지와 함께 사용할 함수를 지정할 수 있습니다. 다음은 예제입니다.

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

로컬에서 테스트할 템플릿 지정

AWS SAMCLI가 참조할 템플릿을 지정하려면 `--template` 옵션을 사용합니다. AWS SAMCLI는 해당 AWS SAM 템플릿과 해당 템플릿이 가리키는 리소스만 로드합니다. 다음은 예제입니다.

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

Lambda 함수의 호스트 개발 환경 지정

기본적으로 `sam local start-api` 하위 명령은 IP 주소 `127.0.0.1`가 포함된 `localhost`를 사용하여 HTTP 서버를 생성합니다. 로컬 개발 환경이 로컬 컴퓨터와 격리되어 있는 경우 이러한 값을 사용자 지정할 수 있습니다.

`--container-host` 옵션을 사용하여 호스트를 지정합니다. 다음은 예제입니다.

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

`--container-host-interface` 옵션을 사용하여 컨테이너 포트가 바인딩해야 하는 호스트 네트워크의 IP 주소를 지정합니다. 다음은 예제입니다.

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

모범 사례

애플리케이션에 실행 중인 `sam build`의 `.aws-sam` 디렉터리가 있는 경우 함수 코드를 업데이트할 때마다 `sam build`를 실행해야 합니다. 그런 다음 `sam local start-api`를 실행하여 업데이트된 함수 코드를 로컬에서 테스트합니다.

로컬 테스트는 클라우드에 배포하기 전에 빠르게 개발하고 테스트할 수 있는 훌륭한 솔루션입니다. 하지만 로컬 테스트는 클라우드의 리소스 간 권한 등 모든 것을 검증하지는 않습니다. 가능한 한 클라우드에서 애플리케이션을 테스트하세요. 클라우드 테스트 워크플로의 속도를 높이려면 `sam sync`를 사용하는 것이 좋습니다.

자세히 알아보기

`sam local start-api` 옵션 목록은 sam local start-api 섹션을 참조하세요.

sam local start-lambda를 사용한 테스트 소개

AWS SAMCLI 하위 명령을 사용하여 AWS CLI 및 SDK`sam local start-lambda`를 통해 Lambda 함수를 호출합니다. SDKs 이 명령은 Lambda를 에뮬레이션하는 로컬 엔드포인트를 시작합니다.

- 에 대한 소개는 단원을 AWS SAMCLI참조하십시오. 란 무엇입니까 AWS SAMCLI?
- `sam local start-lambda` 명령 옵션 목록은 sam local start-lambda 섹션을 참조하세요.

> **Note**
>
> `sam local start-lambda`는 자동 체크포인트 및 재생 기능을 갖춘 내구성 있는 함수를 지원합니다. 내구성 함수는 로컬 Lambda 엔드포인트와 원활하게 작동합니다.

사전 조건

`sam local start-lambda`를 사용하려면 다음을 완료하여 AWS SAM CLI를 설치합니다.

- [AWS SAM 사전 조건](#).
- [AWS SAM CLI 설치](#).

`sam local start-lambda`를 사용하기 전에 다음 사항에 대한 기본적인 이해를 하는 것이 좋습니다.

- AWS SAM CLI 구성.
- 에서 애플리케이션 생성 AWS SAM.
- AWS SAM의 빌드 소개.
- AWS SAM 배포 소개.

sam local start-lambda 사용하기

`sam local start-lambda`를 실행하면 AWS SAMCLI는 현재 작업 디렉터리가 프로젝트의 루트 디렉터리라고 가정합니다. AWS SAMCLI는 먼저 `.aws-sam` 하위 폴더에서 `template.[yaml|yml]` 파일을 찾습니다. 찾을 수 없는 경우 AWS SAMCLI는 현재 작업 디렉터리 내에서 `template.[yaml|yml]` 파일을 찾습니다.

sam local start-lambda를 사용하려면

1. 프로젝트 디렉터리에서 다음 명령을 실행합니다.

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

2. AWS SAMCLI는 Lambda 함수를 로컬 Docker 컨테이너에 빌드합니다. 그런 다음 로컬 주소를 HTTP 서버 엔드포인트로 출력합니다. 다음은 예제입니다.

```
$ 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. AWS CLI 또는 SDKs를 사용하여 로컬에서 Lambda 함수를 호출합니다.

   다음은 AWS CLI사용을 보여 주는 예제입니다.

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

   StatusCode: 200
   (END)
   ```

   다음은 용를 AWS SDK 사용하는 예제입니다Python.

   ```
   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")
   ```

옵션

템플릿 지정의 경우 다음을 수행합니다.

AWS SAMCLI가 참조할 템플릿을 지정하려면 `--template` 옵션을 사용합니다. AWS SAMCLI는 해당 AWS SAM 템플릿과 해당 템플릿이 가리키는 리소스만 로드합니다. 다음은 예제입니다.

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

AWS SAM 템플릿에 대한 자세한 내용은 단원을 참조하십시오AWS SAM 템플릿 구조.

모범 사례

애플리케이션에 실행 중인 `sam build`의 `.aws-sam` 디렉터리가 있는 경우 함수 코드를 업데이트할 때마다 `sam build`를 실행해야 합니다. 그런 다음 `sam local start-lambda`를 실행하여 업데이트된 함수 코드를 로컬에서 테스트합니다.

로컬 테스트는 클라우드에 배포하기 전에 빠르게 개발하고 테스트할 수 있는 훌륭한 솔루션입니다. 하지만 로컬 테스트는 클라우드의 리소스 간 권한 등 모든 것을 검증하지는 않습니다. 가능한 한 클라우드에서 애플리케이션을 테스트하세요. 클라우드 테스트 워크플로의 속도를 높이려면 `sam sync`를 사용하는 것이 좋습니다.

자세히 알아보기

`sam local start-lambda` 옵션 목록은 sam local start-lambda 섹션을 참조하세요.

AWS SAM에서 Lambda 함수를 로컬로 호출

클라우드에서 테스트하거나 배포하기 전에 Lambda 함수를 로컬로 호출하면 다양한 이점이 있습니다. 이를 통해 함수의 로직을 더 빠르게 테스트할 수 있습니다. 먼저 로컬로 테스트하면 클라우드에서 테스트할 때 또는 배포 중에 문제가 발생할 가능성을 줄여 불필요한 비용을 방지할 수 있습니다. 또한 로컬 테스트를 사용하면 디버깅을 더 쉽게 수행할 수 있습니다.

sam local invoke 명령을 사용하고 함수의 논리적 ID와 이벤트 파일을 제공하여 Lambda 함수를 로컬에서 호출할 수 있습니다. sam local invoke는 이벤트로 `stdin`도 수락합니다. 자세한 내용은 AWS Lambda 개발자 안내서의 이벤트를 참조하세요. 다양한 AWS 서비스의 이벤트 메시지 형식에 대한 자세한 내용은 AWS Lambda 개발자 안내서의 다른 서비스와 함께 AWS Lambda 사용을 참조하세요.

> **Note**
>
> sam local invoke 명령은 AWS Command Line Interface(AWS CLI)명령 `aws lambda invoke`에 해당합니다. 두 명령 중 하나를 사용하여 Lambda 함수를 호출할 수 있습니다.

호출하려는 함수가 포함된 프로젝트 디렉터리에서 sam local invoke 명령을 실행해야 합니다.

예시:

```
# 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
```

환경 변수 파일

템플릿에 정의된 값을 재정의하는 환경 변수를 로컬에서 선언하려면 다음을 수행하십시오.

1. 재정의할 환경 변수가 포함된 JSON 파일을 만듭니다.
2. `--env-vars` 인수를 사용하여 템플릿에 정의된 값을 재정의합니다.

환경 변수 선언

모든 리소스에 전체적으로 적용되는 환경 변수를 선언하려면 다음과 같이 `Parameters` 객체를 지정합니다.

```
{
    "Parameters": {
        "TABLE_NAME": "localtable",
        "BUCKET_NAME": "amzn-s3-demo-bucket",
        "STAGE": "dev"
    }
}
```

각 리소스에 대해 서로 다른 환경 변수를 선언하려면 다음과 같이 각 리소스에 대하여 객체를 지정합니다.

```
{
    "MyFunction1": {
        "TABLE_NAME": "localtable",
        "BUCKET_NAME": "amzn-s3-demo-bucket",
    },
    "MyFunction2": {
        "TABLE_NAME": "localtable",
        "STAGE": "dev"
    }
```

```
}
```

각 리소스에 객체를 지정할 때는 우선 순위가 높은 순위에서 가장 낮은 순으로 나열된 다음 식별자를 사용할 수 있습니다.

1. `logical_id`
2. `function_id`
3. `function_name`
4. 완전 경로 식별자

단일 파일에서 환경 변수를 선언하는 위의 두 가지 방법을 모두 함께 사용할 수 있습니다. 이렇게 하면 특정 리소스에 대해 제공한 환경 변수가 글로벌 환경 변수보다 우선합니다.

귀하의 환경 변수를 JSON 파일(예:`env.json`)에 저장합니다.

환경 변수 값 재정의

환경 변수를 JSON 파일에 정의된 변수로 재정의하려면 invoke 또는 start-api 명령과 함께 `--env-vars` 인수를 사용합니다. 예:

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

계층

귀하의 애플리케이션에 레이어가 포함된 경우 로컬 호스트의 레이어 관련 문제를 디버깅하는 방법에 대한 자세한 내용은을 참조하세요AWS SAM에서 Lambda 계층을 사용하여 효율성 향상.

자세히 알아보기

함수를 로컬에서 호출하는 실제 예시는 전체 AWS SAM 워크샵의 모듈 2 - 로컬로 실행 섹션을 참조하세요.

AWS SAM에서 로컬로 API Gateway 실행

Amazon API Gateway를 로컬에서 실행하면 다양한 이점이 있습니다. 예를 들어, API Gateway를 로컬에서 실행하면 AWS 클라우드에 배포하기 전에 로컬에서 API 엔드포인트를 테스트할 수 있습니다. 먼

저 로컬에서 테스트하면 클라우드에서 수행하는 테스트 및 개발을 줄일 수 있으므로 비용을 절감할 수 있습니다. 또한 로컬에서 실행하면 디버깅이 더 쉬워집니다.

HTTP 요청/응답 기능을 테스트하는 데 사용할 수 있는 API Gateway의 로컬 인스턴스를 시작하려면 `sam local start-api` AWS SAM CLI 명령을 사용합니다. 이 기능에는 핫 리로딩 기능이 있어 함수를 빠르게 개발하고 반복할 수 있습니다.

> **Note**
>
> 핫 리로딩은 변경된 파일만 새로 고치고 애플리케이션 상태는 동일하게 유지되는 것을 말합니다. 반대로 라이브 재로드는 전체 애플리케이션을 새로 고치고 애플리케이션의 당시 상태가 손실되는 경우입니다.

`sam local start-api` 명령을 사용하는 방법은 sam local start-api를 사용한 테스트 소개 섹션을 참조하세요.

기본적으로 AWS SAM은 AWS Lambda 프록시 통합을 사용하며 `HttpApi`과 `Api`리소스 유형을 모두 지원합니다. `HttpApi` 리소스 유형의 프록시 통합에 대한 자세한 내용은 API Gateway 개발자 안내서의 HTTP API용 AWS Lambda 프록시 통합 작업을 참조하세요. 이러한 `Api` 리소스 유형으로 실행하는 프록시 통합에 대한 자세한 내용은 API Gateway 개발자 안내서의 API Gateway Lambda 프록시 통합을 참조하세요.

예시:

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

AWS SAM은 귀하의 AWS SAM템플릿 내에서 `HttpApi` 혹은 `Api` 이벤트 소스가 정의되도록 하는 모든 함수를 자동으로 찾습니다. 그런 다음 정의된 HTTP 경로에 함수를 마운트합니다.

다음 `Api` 예제에서 `Ratings` 함수는 귀하의 요청 `ratings.py:handler()` 시 `/ratings` 마운트됩니다. `GET`

```
Ratings:
  Type: AWS::Serverless::Function
  Properties:
    Handler: ratings.handler
    Runtime: python3.9
    Events:
```

```
      Api:
        Type: Api
        Properties:
          Path: /ratings
          Method: get
```

다음은 Api 응답의 예시입니다.

```
// 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"
    });
}
```

귀하의 함수 코드를 수정하는 경우 `sam local start-api`를 위한 `sam build` 명령을 실행하여 변경 내용을 감지하십시오.

환경 변수 파일

템플릿에 정의된 값을 재정의하는 환경 변수를 로컬에서 선언하려면 다음을 수행하십시오.

1. 재정의할 환경 변수가 포함된 JSON 파일을 만듭니다.
2. `--env-vars` 인수를 사용하여 템플릿에 정의된 값을 재정의합니다.

환경 변수 선언

모든 리소스에 전체적으로 적용되는 환경 변수를 선언하려면 다음과 같이 `Parameters` 객체를 지정합니다.

```
{
    "Parameters": {
        "TABLE_NAME": "localtable",
        "BUCKET_NAME": "amzn-s3-demo-bucket",
        "STAGE": "dev"
    }
}
```

각 리소스에 대해 서로 다른 환경 변수를 선언하려면 다음과 같이 각 리소스에 대하여 객체를 지정합니다.

```
{
    "MyFunction1": {
        "TABLE_NAME": "localtable",
        "BUCKET_NAME": "amzn-s3-demo-bucket",
    },
    "MyFunction2": {
        "TABLE_NAME": "localtable",
        "STAGE": "dev"
    }
}
```

각 리소스에 객체를 지정할 때는 우선 순위가 높은 순위에서 가장 낮은 순으로 나열된 다음 식별자를 사용할 수 있습니다.

1. `logical_id`
2. `function_id`
3. `function_name`
4. 완전 경로 식별자

단일 파일에서 환경 변수를 선언하는 위의 두 가지 방법을 모두 함께 사용할 수 있습니다. 이렇게 하면 특정 리소스에 대해 제공한 환경 변수가 글로벌 환경 변수보다 우선합니다.

귀하의 환경 변수를 JSON 파일(예:`env.json`)에 저장합니다.

환경 변수 값 재정의

환경 변수를 JSON 파일에 정의된 변수로 재정의하려면 invoke 또는 start-api 명령과 함께 `--env-vars` 인수를 사용합니다. 예:

```
$ sam local start-api --env-vars env.json
```

계층

귀하의 애플리케이션에 레이어가 포함된 경우 로컬 호스트의 레이어 관련 문제를 디버깅하는 방법에 대한 자세한 내용은을 참조하세요[AWS SAM에서 Lambda 계층을 사용하여 효율성 향상](#).

sam remote test-event를 사용한 클라우드 테스트 소개

AWS Serverless Application Model Command Line Interface(AWS SAM CLI) `sam remote test-event` 명령을 사용하여 AWS Lambda 함수의 공유 가능한 테스트 이벤트에 액세스하고 관리할 수 있습니다.

공유 가능한 테스트 이벤트에 대해 자세히 알아보려면 AWS Lambda 개발자 안내서의 공유 가능한 테스트 이벤트 섹션을 참조하세요.

주제

- sam remote test-event를 사용하도록 AWS SAMCLI를 설정합니다.
- sam remote test-event 명령 사용
- 공유 가능한 테스트 이벤트 사용
- 공유 가능한 테스트 이벤트 관리

사전 조건

`sam remote test-event`를 사용하려면 다음을 완료하여 AWS SAM CLI를 설치합니다.

- AWS SAM 사전 조건.
- AWS SAM CLI 설치.

이미 AWS SAM CLI를 설치한 경우 최신 버전의 AWS SAM CLI 버전으로 업그레이드하는 것이 좋습니다. 자세한 내용은 AWS SAMCLI업그레이드 섹션을 참조하세요.

`sam remote test-event`를 사용하기 전에 다음 사항에 대한 기본적인 이해를 하는 것이 좋습니다.

- AWS SAM CLI 구성.
- 에서 애플리케이션 생성 AWS SAM.
- AWS SAM의 빌드 소개.
- AWS SAM 배포 소개.
- sam sync를 사용하는 AWS 클라우드 동기화 소개.

sam remote test-event를 사용하도록 AWS SAMCLI를 설정합니다.

AWS SAM CLI `sam remote test-event` 명령을 사용하려면 다음 설정 단계를 완료합니다.

1. AWS 계정를 사용하도록 AWS SAM CLI를 구성하세요.- Lambda의 공유 가능한 테스트 이벤트는 동일한 AWS 계정 내에서 사용자가 액세스하고 관리할 수 있습니다. AWS 계정를 사용하도록 AWS SAM CLI를 구성하려면 AWS SAM CLI 구성 섹션을 참조하세요.
2. 공유 가능한 테스트 이벤트에 대한 권한을 구성합니다. - 공유 가능한 테스트 이벤트에 액세스하고 관리하려면 적절한 권한이 있어야 합니다. 자세히 알아보려면 AWS Lambda 개발자 안내서의 공유 가능한 테스트 이벤트 섹션을 참조하세요.

sam remote test-event 명령 사용

이 AWS SAM CLI `sam remote test-event` 명령은 공유 가능한 테스트 이벤트에 액세스하고 관리하는 데 사용할 수 있는 다음과 같은 하위 명령을 제공합니다.

- `delete` - Amazon EventBridge 스키마 레지스트리에서 공유 가능한 테스트 이벤트를 삭제합니다.
- `get` - EventBridge 스키마 레지스트리에서 공유 가능한 테스트 이벤트를 가져옵니다.
- `list` - EventBridge 스키마 레지스트리에서 함수에 대한 기존의 공유 가능한 테스트 이벤트를 나열합니다.
- `put` - 로컬 파일의 이벤트를 EventBridge 스키마 레지스트리에 저장합니다.

AWS SAM CLI를 사용하여 이러한 하위 명령을 나열하려면 다음을 실행합니다.

```
$ sam remote test-event --help
```

공유 가능한 테스트 이벤트 삭제

다음과 함께 `delete` 하위 명령을 사용하여 공유 가능한 테스트 이벤트를 삭제할 수 있습니다.

- 삭제할 공유 가능한 테스트 이벤트의 이름을 제공합니다.
- 이벤트와 연결된 Lambda 함수의 허용 가능한 ID를 제공합니다.
- Lambda 함수 논리적 ID를 제공하는 경우 Lambda 함수와 관련된 AWS CloudFormation 스택 이름도 제공해야 합니다.

다음은 예제입니다.

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

delete 하위 명령과 함께 사용할 옵션 목록은 sam remote test-event delete 섹션을 참조하세요. AWS SAM CLI에서 다음을 수행할 수도 있습니다.

```
$ sam remote test-event delete --help
```

공유 가능한 테스트 이벤트 생성

다음과 함께 get 하위 명령을 사용하여 EventBridge 스키마 레지스트리에서 공유 가능한 테스트 이벤트를 가져올 수 있습니다.

- 가져올 공유 가능한 테스트 이벤트의 이름을 제공합니다.
- 이벤트와 연결된 Lambda 함수의 허용 가능한 ID를 제공합니다.
- Lambda 함수 논리적 ID를 제공하는 경우 Lambda 함수와 관련된 AWS CloudFormation 스택 이름도 제공해야 합니다.

다음은 sam-app 스택의 HelloWorldFunction Lambda 함수와 연결된 demo-event라는 공유 가능한 테스트 이벤트를 가져오는 예제입니다. 이 명령은 이벤트를 콘솔에 인쇄합니다.

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

공유 가능한 테스트 이벤트를 가져와 로컬 컴퓨터에 저장하려면 --output-file 옵션을 사용하고 파일 경로와 이름을 제공합니다. 다음은 현재 작업 디렉터리에 demo-event를 demo-event.json로 저장하는 예제입니다.

```
$ sam remote test-event get HelloWorldFunction --stack-name sam-app --name demo-event
 --output-file demo-event.json
```

get 하위 명령과 함께 사용할 옵션 목록은 sam remote test-event get 섹션을 참조하세요. AWS SAM CLI에서 다음을 수행할 수도 있습니다.

```
$ sam remote test-event get --help
```

공유 가능한 테스트 이벤트 나열

스키마 레지스트리에서 특정 Lambda 함수에 대한 모든 공유 가능한 테스트 이벤트를 나열할 수 있습니다. 다음과 함께 list 하위 명령을 사용합니다.

- 이벤트와 연결된 Lambda 함수의 허용 가능한 ID를 제공합니다.
- Lambda 함수 논리적 ID를 제공하는 경우 Lambda 함수와 관련된 CloudFormation 스택 이름도 제공해야 합니다.

다음은 `sam-app` 스택의 `HelloWorldFunction` Lambda 함수와 관련된 모든 공유 가능한 테스트 이벤트 목록을 가져오는 예제입니다.

```
$ sam remote test-event list HelloWorldFunction --stack-name sam-app
```

`list` 하위 명령과 함께 사용할 옵션 목록은 sam remote test-event list 섹션을 참조하세요. AWS SAM CLI에서 다음을 수행할 수도 있습니다.

```
$ sam remote test-event list --help
```

공유 가능한 테스트 이벤트 저장

공유 가능한 테스트 이벤트를 EventBridge 스키마 레지스트리에 저장할 수 있습니다. 다음과 함께 `put` 하위 명령을 사용합니다.

- 공유 가능한 테스트 이벤트와 관련된 Lambda 함수의 허용 가능한 ID를 제공합니다.
- 공유 가능한 테스트 이벤트의 이름을 제공합니다.
- 업로드할 로컬 이벤트의 파일 경로와 이름을 제공합니다.

다음은 로컬 `demo-event.json` 이벤트를 `demo-event`로 저장하고 이를 `sam-app` 스택의 `HelloWorldFunction` Lambda 함수와 연결하는 예제입니다.

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

EventBridge 스키마 레지스트리에 이름이 같은 공유 가능한 테스트 이벤트가 있는 경우 AWS SAM CLI는 이를 덮어쓰지 않습니다. 덮어쓰려면 `--force` 옵션을 명령에 추가합니다.

`put` 하위 명령과 함께 사용할 옵션 목록은 sam remote test-event put 섹션을 참조하세요. AWS SAM CLI에서 다음을 수행할 수도 있습니다.

```
$ sam remote test-event put --help
```

공유 가능한 테스트 이벤트 사용

공유 가능한 테스트 이벤트를 사용하여 `sam remote invoke` 명령으로 AWS 클라우드에서 Lambda 함수를 테스트합니다. 자세한 내용은 [공유 가능한 테스트 이벤트를 클라우드의 Lambda 함수에 전달](#)를 참조하세요.

공유 가능한 테스트 이벤트 관리

이 주제에는 공유 가능한 테스트 이벤트를 관리하고 사용하는 방법에 대한 예제가 포함되어 있습니다.

공유 가능한 테스트 이벤트를 가져와 수정하고 사용하기

EventBridge 스키마 레지스트리에서 공유 가능한 테스트 이벤트를 가져오고, 로컬에서 수정하고, AWS 클라우드에서 Lambda 함수와 함께 로컬 테스트 이벤트를 사용할 수 있습니다. 다음은 예제입니다.

1. 공유 가능한 테스트 이벤트 검색 - `sam remote test-event get` 하위 명령을 사용하여 특정 Lambda 함수에 대한 공유 가능한 테스트 이벤트를 검색하고 로컬에 저장합니다.

```
$ sam remote test-event get HelloWorldFunction --stack-name sam-app --name demo-event
 --output-file demo-event.json
```

2. 공유 가능한 테스트 이벤트 수정 - 원하는 텍스트 편집기를 사용하여 공유 가능한 테스트 이벤트를 수정합니다.
3. 공유 가능한 테스트 이벤트 사용 - `sam remote invoke` 명령을 사용하고 이벤트의 파일 경로와 이름을 `--event-file`로 입력합니다.

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

공유 가능한 테스트 이벤트를 가져와 수정하고 업로드하고 사용하기

EventBridge 스키마 레지스트리에서 공유 가능한 테스트 이벤트를 가져와서 로컬에서 수정하고 업로드할 수 있습니다. 그런 다음 공유 가능한 테스트 이벤트를 AWS 클라우드의 Lambda 함수에 직접 전달할 수 있습니다. 다음은 예제입니다.

1. 공유 가능한 테스트 이벤트 검색 - `sam remote test-event get` 하위 명령을 사용하여 특정 Lambda 함수에 대한 공유 가능한 테스트 이벤트를 검색하고 로컬에 저장합니다.

```
$ sam remote test-event get HelloWorldFunction --stack-name sam-app --name demo-event
 --output-file demo-event.json
```

2. 공유 가능한 테스트 이벤트 수정 - 원하는 텍스트 편집기를 사용하여 공유 가능한 테스트 이벤트를 수정합니다.

3. 공유 가능한 테스트 이벤트 업로드 - `sam remote test-event put` 하위 명령을 사용하여 공유 가능한 테스트 이벤트를 EventBridge 스키마 레지스트리에 업로드하고 저장합니다. 이 예제에서는 `--force` 옵션을 사용하여 공유 가능 테스트의 이전 버전을 덮어씁니다.

```
$ sam remote test-event put HelloWorldFunction --stack-name sam-app --name demo-event
 --file demo-event.json --force
```

4. 공유 가능한 테스트 이벤트를 Lambda 함수로 전달 - `sam remote invoke` 명령을 사용하여 공유 가능한 테스트 이벤트를 AWS 클라우드의 Lambda 함수에 직접 전달합니다.

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

sam remote invoke를 사용하여 클라우드에서 테스트 소개

AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) `sam remote invoke` 명령을 사용하여 AWS 클라우드에서 지원되는 AWS 리소스와 상호 작용할 수 있습니다. 다음 리소스를 간접 호출하기 위해 `sam remote invoke`를 사용할 수 있습니다.

- Amazon Kinesis Data Streams - 데이터 레코드를 Kinesis Data Streams 애플리케이션으로 전송합니다.
- AWS Lambda - 이벤트를 간접 호출하고 Lambda 함수로 전달합니다.
- Amazon Simple Queue Service (Amazon SQS) – 메시지를 Amazon Simple Queue Service(Amazon SQS) 대기열에 전송합니다.
- AWS Step Functions – Step Functions 상태 머신을 간접 호출하여 실행을 시작합니다.

AWS SAM CLI에 대한 소개는 [란 무엇입니까 AWS SAMCLI?](#) 섹션을 참조하세요.

일반적인 개발 워크플로 중 `sam remote invoke`를 사용하는 예는 [5단계: AWS 클라우드에서 함수와 상호작용하십시오.](#) 섹션을 참조하세요.

주제

- sam remote invoke 명령 사용
- sam remote invoke 명령 옵션 사용
- 프로젝트 구성 파일을 구성합니다.
- 예시
- 관련 링크

사전 조건

`sam remote invoke`를 사용하려면 다음을 완료하여 AWS SAM CLI를 설치합니다.

- AWS SAM 사전 조건.
- AWS SAM CLI 설치.

최신 버전의 AWS SAM CLI로 업그레이드하는 것이 좋습니다. 자세한 내용은 AWS SAMCLI업그레이드 섹션을 참조하세요.

`sam remote invoke`를 사용하기 전에 다음 사항에 대한 기본적인 이해를 하는 것이 좋습니다.

- AWS SAM CLI 구성.
- 에서 애플리케이션 생성 AWS SAM.
- AWS SAM의 빌드 소개.
- AWS SAM 배포 소개.
- sam sync를 사용하는 AWS 클라우드 동기화 소개.

sam remote invoke 명령 사용

이 명령을 사용하기 전에 리소스를 AWS 클라우드에 배포해야 합니다.

다음 명령 구조를 사용하고 프로젝트의 루트 디렉터리에서 실행합니다.

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

> **Note**
>
> 이 페이지에는 명령 프롬프트에서 제공되는 옵션이 표시됩니다. 명령 프롬프트에서 옵션을 전달하는 대신 프로젝트의 구성 파일에서 옵션을 구성할 수도 있습니다. 자세한 내용은 프로젝트 설정 구성 섹션을 참조하세요.

`sam remote invoke` 인수 및 옵션에 대한 설명은 sam remote invoke 섹션을 참조하세요.

Kinesis Data Streams 사용

Kinesis Data Streams 애플리케이션에 데이터 레코드를 보낼 수 있습니다. AWS SAM CLI는 데이터 레코드를 전송하고 샤드 ID와 시퀀스 번호를 반환합니다. 다음은 예제입니다.

```
$ 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"
 }%
```

데이터 기록을 보내려면

1. 리소스 ID 값을 Kinesis Data Streams 애플리케이션의 인수로 제공합니다. 자세한 내용은 리소스 ID를 참조하세요.
2. 데이터 레코드를 이벤트로 제공하여 Kinesis Data Streams 애플리케이션에 전송합니다. `--event` 옵션을 사용하여 명령줄에서 이벤트를 제공하거나 `--event-file`을 사용하는 파일에서 이벤트를 제공할 수 있습니다. 이벤트를 제공하지 않으면 AWS SAM CLI는 빈 이벤트를 보냅니다.

Lambda 함수와 함께 사용

클라우드에서 Lambda 함수를 간접 호출하고 빈 이벤트를 전달하거나 명령줄 또는 파일에서 이벤트를 제공할 수 있습니다. AWS SAM CLI는 Lambda 함수를 간접 호출하고 해당 응답을 반환합니다. 다음은 예제입니다.

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

Lambda 함수를 간접 호출하기 위해

1. 리소스 ID 값을 Lambda 함수에 대한 인수로 제공합니다. 자세한 내용은 리소스 ID를 참조하세요.
2. Lambda 함수에 전송할 이벤트를 제공합니다. `--event` 옵션을 사용하여 명령줄에서 이벤트를 제공하거나 `--event-file`을 사용하는 파일에서 이벤트를 제공할 수 있습니다. 이벤트를 제공하지 않으면 AWS SAM CLI는 빈 이벤트를 보냅니다.

응답 스트리밍으로 구성된 Lambda 함수

이 `sam remote invoke` 명령은 응답을 스트리밍하도록 구성된 Lambda 함수를 지원합니다. AWS SAM 템플릿의 `FunctionUrlConfig` 속성을 사용하여 응답을 스트리밍하도록 Lambda 함수를 구성할 수 있습니다. `sam remote invoke`를 사용하면 AWS SAM CLI가 Lambda 구성을 자동으로 감지하고 응답 스트리밍을 통해 간접 호출합니다.

예시는 응답을 스트리밍하도록 Lambda 함수 구성 섹션을 참조하세요.

공유 가능한 테스트 이벤트를 클라우드의 Lambda 함수에 전달

공유 가능한 테스트 이벤트는 동일한 AWS 계정 계정의 다른 사용자와 공유할 수 있는 테스트 이벤트입니다. 자세히 알아보려면 AWS Lambda 개발자 안내서의 공유 가능한 테스트 이벤트 섹션을 참조하세요.

공유 가능한 테스트 이벤트 액세스 및 관리

AWS SAM CLI `sam remote test-event` 명령을 사용하여 공유 가능한 테스트 이벤트에 액세스하고 관리할 수 있습니다. 예를 들면, 다음을 수행하기 위해 `sam remote test-event`를 사용할 수 있습니다.

- Amazon EventBridge 스키마 레지스트리에서 공유 가능한 테스트 이벤트를 검색합니다.
- 공유 가능한 테스트 이벤트를 로컬에서 수정하고 EventBridge 스키마 레지스트리에 업로드합니다.
- EventBridge 스키마 레지스트리에서 공유 가능한 테스트 이벤트를 삭제합니다.

자세한 내용은 sam remote test-event를 사용한 클라우드 테스트 소개 섹션을 참조하세요.

공유 가능한 테스트 이벤트를 클라우드의 Lambda 함수에 전달

EventBridge 스키마 레지스트리에서 클라우드의 Lambda 함수로 공유 가능한 테스트 이벤트를 전달하려면 `--test-event-name` 옵션을 사용하고 공유 가능한 테스트 이벤트의 이름을 제공합니다. 다음은 예제입니다.

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

공유 가능한 테스트 이벤트를 로컬에 저장하는 경우 `--event-file` 옵션을 사용하여 로컬 테스트 이벤트의 파일 경로와 이름을 제공할 수 있습니다. 다음은 예제입니다.

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

Amazon SQS에서 사용

메시지를 Amazon SQS 대기열로 전송할 수 있습니다. AWS SAM CLI는 다음을 반환합니다.

- 메시지 ID
- 메시지 본문의 MD5
- 응답 메타데이터

다음은 예제입니다.

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

```
Sending message to SQS queue MySqsQueue

{
  "MD5OfMessageBody": "5d41402abc4b2a76b9719d911017c592",
  "MessageId": "05c7af65-9ae8-4014-ae28-809d6d8ec652"
}%
```

메시지를 전송하려면

1. Amazon SQS 대기열의 인수로 리소스 ID 값을 제공합니다. 자세한 내용은 리소스 ID를 참조하세요.
2. Amazon SQS 대기열로 전송할 이벤트를 제공합니다. `--event` 옵션을 사용하여 명령줄에서 이벤트를 제공하거나 `--event-file`을 사용하는 파일에서 이벤트를 제공할 수 있습니다. 이벤트를 제공하지 않으면 AWS SAM CLI는 빈 이벤트를 보냅니다.

Step Functions와 함께 사용

Step Functions 상태 시스템 실행을 시작합니다. AWS SAM CLI는 상태 머신 워크플로가 완료될 때까지 대기한 후, 실행의 마지막 단계에 대한 출력을 반환합니다. 다음은 예제입니다.

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

Invoking Step Function HelloWorldStateMachine

"Hello Developer World"%
```

상태 머신을 실행하려면

1. Step Functions 상태 시스템의 인수로 리소스 ID 값을 제공합니다. 자세한 내용은 리소스 ID를 참조하세요.
2. 상태 시스템에 전송할 이벤트를 제공합니다. `--event` 옵션을 사용하여 명령줄에서 이벤트를 제공하거나 `--event-file`을 사용하는 파일에서 이벤트를 제공할 수 있습니다. 이벤트를 제공하지 않으면 AWS SAM CLI는 빈 이벤트를 보냅니다.

sam remote invoke 명령 옵션 사용

이 섹션에서는 sam remote invoke 명령과 함께 사용할 수 있는 몇 가지 주요 옵션을 다룹니다. 옵션의 전체 목록은 sam remote invoke 섹션을 참조하세요.

리소스에 이벤트 전달

다음 옵션을 사용하여 클라우드의 리소스에 이벤트를 전달할 수 있습니다.

- --event - 명령줄에서 이벤트를 전달합니다.
- --event-file - 파일에서 이벤트를 전달합니다.

Lambda 예제

--event를 사용하여 명령줄에서 이벤트를 문자열 값으로 전달합니다.

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

--event-file를 사용하여 파일에서 이벤트를 전달하고 파일 경로를 제공합니다.

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

stdin를 사용하여 이벤트를 전달합니다.

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

AWS SAMCLI 응답 출력을 구성합니다.

`sam remote invoke`를 사용하여 지원되는 리소스를 간접 호출하면 AWS SAM CLI는 다음을 포함하는 응답을 반환합니다.

- 요청 메타데이터 - 요청과 관련된 메타데이터입니다. 여기에는 요청 ID 및 요청 시작 시간이 포함됩니다.
- 리소스 응답 - 클라우드에서 간접 호출된 후 리소스가 보내는 응답입니다.

`--output` 옵션을 사용하여 AWS SAM CLI 출력 응답을 구성할 수 있습니다. 다음과 같은 값을 사용할 수 있습니다.

- `json` - 메타데이터 및 리소스 응답이 JSON 구조로 반환됩니다. 응답에는 전체 SDK 출력이 포함됩니다.
- `text` - 메타데이터가 텍스트 구조로 반환됩니다. 리소스 응답은 리소스의 출력 형식으로 반환됩니다.

다음은 json 출력의 예제입니다.

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

Invoking Lambda Function HelloWorldFunction


{
  "ResponseMetadata": {
    "RequestId": "3bdf9a30-776d-4a90-94a6-4cccc0fc7b41",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "date": "Mon, 19 Jun 2023 17:15:46 GMT",
      "content-type": "application/json",
      "content-length": "57",
      "connection": "keep-alive",
      "x-amzn-requestid": "3bdf9a30-776d-4a90-94a6-4cccc0fc7b41",
      "x-amzn-remapped-content-length": "0",
      "x-amz-executed-version": "$LATEST",
      "x-amz-log-result":
 "U1RBUlQgUmVxdWVzdElkOiAzYmRmOWEzMC03NzZkLTRhOTAtOTRhNi00Y2NjYzBmYzdiNDEgVmVyc2lvbjogJExBVEVTV
      "x-amzn-trace-id":
 "root=1-64908d42-17dab270273fcc6b527dd6b8;sampled=0;lineage=2301f8dc:0"
    },
    "RetryAttempts": 0
  },
  "StatusCode": 200,
  "LogResult":
 "U1RBUlQgUmVxdWVzdElkOiAzYmRmOWEzMC03NzZkLTRhOTAtOTRhNi00Y2NjYzBmYzdiNDEgVmVyc2lvbjogJExBVEVTV
  "ExecutedVersion": "$LATEST",
  "Payload": "{\"statusCode\":200,\"body\":\"{\\\"message\\\":\\\"hello world\\\"}\"}"
}%
```

json 출력을 지정하면 전체 응답이 stdout에 반환됩니다. 다음은 예제입니다.

```
$ sam remote invoke --stack-name sam-app --output json 1> stdout.log

Invoking Lambda Function HelloWorldFunction



$ cat stdout.log

```

```
{
  "ResponseMetadata": {
    "RequestId": "d30d280f-8188-4372-bc94-ce0f1603b6bb",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "date": "Mon, 19 Jun 2023 17:35:56 GMT",
      "content-type": "application/json",
      "content-length": "57",
      "connection": "keep-alive",
      "x-amzn-requestid": "d30d280f-8188-4372-bc94-ce0f1603b6bb",
      "x-amzn-remapped-content-length": "0",
      "x-amz-executed-version": "$LATEST",
      "x-amz-log-result":
 "U1RBUlQgUmVxdWVzdElkOiBkMzBkMjgwZi04MTg4LTQzNzItYmM5NC1jZTBmMTYwM2I2YmIgVmVyc2lvbjogJExBVEVTV
      "x-amzn-trace-id":
 "root=1-649091fc-771473c7778689627a6122b7;sampled=0;lineage=2301f8dc:0"
    },
    "RetryAttempts": 0
  },
  "StatusCode": 200,
  "LogResult":
 "U1RBUlQgUmVxdWVzdElkOiBkMzBkMjgwZi04MTg4LTQzNzItYmM5NC1jZTBmMTYwM2I2YmIgVmVyc2lvbjogJExBVEVTV
  "ExecutedVersion": "$LATEST",
  "Payload": "{\"statusCode\":200,\"body\":\"{\\\"message\\\":\\\"hello world\\\"}\"}"
}%
```


다음은 `text` 출력의 예제입니다.

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

`text` 출력을 지정하면 Lambda 함수 런타임 출력(예: 로그)이 `stderr`로 반환됩니다. Lambda 함수 페이로드가 `stdout`로 반환됩니다. 다음은 예제입니다.

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

Boto3 파라미터를 사용자 지정합니다.

`sam remote invoke`의 경우 AWS SAM CLI는 Python(Boto3)용 AWS SDK를 활용하여 클라우드의 리소스와 상호 작용합니다. `--parameter` 옵션을 사용하여 Boto3 파라미터를 사용자 지정할 수 있습니다. 사용자 지정할 수 있는 지원되는 파라미터 목록은 `--parameter` 섹션을 참조하세요.

예시

Lambda 함수를 간접 호출하여 파라미터 값을 검증하고 권한을 확인합니다.

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

단일 명령으로 **`--parameter`** 옵션을 여러 번 사용하여 여러 파라미터를 제공합니다.

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

기타 옵션

sam remote invoke 옵션의 전체 목록은 sam remote invoke 섹션을 참조하세요.

프로젝트 구성 파일을 구성합니다.

구성 파일에서 sam remote invoke를 구성하려면 테이블에서 remote_invoke를 사용합니다. 다음은 sam remote invoke 명령의 기본값을 구성하는 samconfig.toml 파일의 예입니다.

```
...
version =0.1

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

예시

sam remote invoke 사용에 대한 기본 예제는AWS 컴퓨팅 블로그의 AWS SAM 원격과 함께 AWS Lambda 함수를 테스트 섹션을 참조하세요.

Kinesis Data Streams 예제

기본 예제

파일에서 Kinesis Data Streams 애플리케이션으로 데이터 레코드를 전송합니다. 리소스 ID에 대한 ARN을 제공하여 Kinesis Data Streams 애플리케이션을 식별할 수 있습니다.

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

명령줄에 제공된 이벤트를 Kinesis Data Streams 애플리케이션으로 전송합니다.

```
$ 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"
}%
```

Kinesis Data Streams 애플리케이션의 물리적 ID를 가져옵니다. 그런 다음 명령줄에서 이벤트를 제공합니다.

```
$ 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"
}%
```

명령줄에 JSON 문자열을 이벤트로 제공합니다.

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

Putting record to Kinesis data stream KinesisStream
```

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

Kinesis Data Streams 애플리케이션에 빈 이벤트를 전송합니다.

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

Putting record to Kinesis data stream KinesisStream

{
  "ShardId": "shardId-000000000000",
  "SequenceNumber": "49646251411914806775980904866469008589597168190416224258"
}%
```

AWS SAM CLI 응답을 JSON 형식으로 반환합니다.

```
$ sam remote invoke KinesisStream --stack-name kinesis-example --event '{"hello":
 "world"}' --output json

Putting record to Kinesis data stream KinesisStream

{
  "ShardId": "shardId-000000000000",
  "SequenceNumber": "49646251411914806775980905078409420803696667195489648642",
  "ResponseMetadata": {
    "RequestId": "ebbbd307-3e9f-4431-b67c-f0715e9e353e",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "x-amzn-requestid": "ebbbd307-3e9f-4431-b67c-f0715e9e353e",
      "x-amz-id-2": "Q3yBcgTwtPaQTV26IKclbECmZikUYOzKY+CzcxA84ZHgCkc5T2N/
ITWg6RPOQcWw8Gn0tNPcEJBEHyVVqboJAPgCritqsvCu",
      "date": "Thu, 09 Nov 2023 18:13:10 GMT",
      "content-type": "application/x-amz-json-1.1",
      "content-length": "110"
    },
    "RetryAttempts": 0
  }
}%
```

JSON 출력을 stdout으로 반환합니다.

```
$ 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": "49646251411914806775980906397778675950399883490067742 74",
  "ResponseMetadata": {
    "RequestId": "f4290006-d84b-b1cd-a9ee-28306eeb2939",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "x-amzn-requestid": "f4290006-d84b-b1cd-a9ee-28306eeb2939",
      "x-amz-id-2": "npCqz
+IBKpoL4sQ1ClbUmxuJlbeA24Fx1UgpIrS6mm2NoIeV2qdZSN5AhNurdssykXajBrXaC9anMhj2eG/h7Hnbf
+bPuotU",
      "date": "Thu, 09 Nov 2023 18:33:26 GMT",
      "content-type": "application/x-amz-json-1.1",
      "content-length": "110"
    },
    "RetryAttempts": 0
  }
}%
```

Lambda 예제

기본 예제

ARN을 리소스 ID로 제공하여 Lambda 함수를 간접 호출합니다.

```
$ sam remote invoke arn:aws:lambda:us-west-2:012345678910:function:sam-app-
HelloWorldFunction-ohRFEn2RuAvp
```

논리적 ID를 리소스 ID로 제공하여 Lambda 함수를 간접 호출합니다.

또한 --stack-name 옵션을 사용하여 CloudFormation 스택 이름을 제공해야 합니다. 다음은 예제입니다.

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

애플리케이션에 단일 Lambda 함수가 포함된 경우 해당 함수의 논리적 ID를 지정하지 않아도 됩니다. `--stack-name` 옵션만 제공할 수 있습니다. 다음은 예제입니다.

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

물리적 ID를 리소스 ID로 제공하여 Lambda 함수를 간접 호출합니다.

CloudFormation를 사용하여 배포하면 물리적 ID가 생성됩니다.

```
$ sam remote invoke sam-app-HelloWorldFunction-TZvxQRFNv0k4
```

하위 스택의 Lambda 함수를 간접 호출합니다.

이 예제에서 애플리케이션에는 다음과 같은 디렉터리 구조를 포함합니다.

```
lambda-example
### childstack
#   ### function
#   #   ### __init__.py
#   #   ### app.py
#   #   ### requirements.txt
#   ### template.yaml
### events
#   ### event.json
### samconfig.toml
### template.yaml
```

`childstack`에 대한 Lambda 함수를 간접 호출하기 위해 다음을 실행합니다.

```
$ 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\": {}}"}%
```

응답을 스트리밍하도록 Lambda 함수 구성

이 예제에서는 AWS SAM CLI를 사용하여 응답을 스트리밍하도록 구성된 Lambda 함수를 포함하는 새로운 서버리스 애플리케이션을 초기화합니다. 애플리케이션을 AWS 클라우드에 배포하고 `sam remote invoke`를 사용하여 클라우드의 함수와 상호 작용합니다.

먼저 `sam init` 명령을 실행하여 새 서버리스 애플리케이션을 생성합니다. Lambda 응답 스트리밍 빠른 시작 템플릿을 선택하고 애플리케이션 이름을 lambda-streaming-nodejs-app으로 지정합니다.

```
$ 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 SAM CLI는 다음과 같은 구조로 프로젝트를 생성합니다.

```
lambda-streaming-nodejs-app
 ### README.md
 ### __tests__
 #   ### unit
 #       ### index.test.js
 ### package.json
 ### samconfig.toml
 ### src
 #   ### index.js
 ### template.yaml
```

다음은 Lambda 함수 코드의 예제입니다.

```
exports.handler = awslambda.streamifyResponse(
   async (event, responseStream, context) => {
     const httpResponseMetadata = {
       statusCode: 200,
       headers: {
         "Content-Type": "text/html",
         "X-Custom-Header": "Example-Custom-Header"
       }
     };

     responseStream = awslambda.HttpResponseStream.from(responseStream,
 httpResponseMetadata);
     // It's recommended to use a `pipeline` over the `write` method for more complex
 use cases.
     // Learn more: https://docs.aws.amazon.com/lambda/latest/dg/configuration-
response-streaming.html
     responseStream.write("<html>");
     responseStream.write("<p>First write!</p>");

     responseStream.write("<h1>Streaming h1</h1>");
     await new Promise(r => setTimeout(r, 1000));
     responseStream.write("<h2>Streaming h2</h2>");
     await new Promise(r => setTimeout(r, 1000));
     responseStream.write("<h3>Streaming h3</h3>");
     await new Promise(r => setTimeout(r, 1000));

     // Long strings will be streamed
     const loremIpsum1 = "Lorem ipsum dolor sit amet, consectetur adipiscing elit.
 Quisque vitae mi tincidunt tellus ultricies dignissim id et diam. Morbi pharetra eu
 nisi et finibus. Vivamus diam nulla, vulputate et nisl cursus, pellentesque vehicula
 libero. Cras imperdiet lorem ante, non posuere dolor sollicitudin a. Vestibulum ipsum
 lacus, blandit nec augue id, lobortis dictum urna. Vestibulum ante ipsum primis in
 faucibus orci luctus et ultrices posuere cubilia curae; Morbi auctor orci eget tellus
 aliquam, non maximus massa porta. In diam ante, pulvinar aliquam nisl non, elementum
 hendrerit sapien. Vestibulum massa nunc, mattis non congue vitae, placerat in quam.
 Nam vulputate lectus metus, et dignissim erat varius a.";
     responseStream.write(`<p>${loremIpsum1}</p>`);
     await new Promise(r => setTimeout(r, 1000));

     responseStream.write("<p>DONE!</p>");
     responseStream.write("</html>");
     responseStream.end();
   }
```

```
);
```

다음은 `template.yaml` 파일의 예제입니다. Lambda 함수의 응답 스트리밍은 `FunctionUrlConfig` 속성을 사용하여 구성됩니다.

```
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
```

일반적으로 `sam build` 및 `sam deploy --guided`를 사용하여 프로덕션 애플리케이션을 빌드하고 배포할 수 있습니다. 이 예제에서는 개발 환경을 가정하고 `sam sync` 명령을 사용하여 애플리케이션을 빌드하고 배포해 보겠습니다.

> **Note**
>
> 이 `sam sync` 명령은 개발 환경에 권장됩니다. 자세한 내용은 [sam sync를 사용하는 AWS 클라우드 동기화 소개](#) 섹션을 참조하세요.

`sam sync`를 실행하기 전에 `samconfig.toml` 파일에 프로젝트가 올바르게 구성되어 있는지 확인합니다. 가장 중요한 것은 `stack_name` 및 `watch`의 값을 검증하는 것입니다. 구성 파일에 이러한 값을 지정하면 명령줄에서 값을 제공할 필요가 없습니다.

```
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"
```

다음으로 `sam sync`를 실행하여 애플리케이션을 빌드하고 배포합니다. `--watch` 옵션이 구성 파일에 구성되어 있으므로 AWS SAM CLI는 애플리케이션을 빌드하고, 애플리케이션을 배포하고, 변경 사항을 확인합니다.

```
$ sam sync
```

```
 The SAM CLI will use the AWS Lambda, Amazon API Gateway, and AWS StepFunctions APIs to 
 upload your code 
 without 

 performing a CloudFormation deployment. This will cause drift in your CloudFormation 
 stack. 
 **The sync command should only be used against a development stack**.



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

 Starting infra sync.

 Building codeuri: 

 /Users/.../lambda-streaming-nodejs-app/src runtime: nodejs18.x metadata: {} 
 architecture: x86_64 functions: StreamingFunction 
 package.json file not found. Continuing the build without dependencies. 

 Running NodejsNpmBuilder:CopySource


 Build Succeeded

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


         Deploying with following values
         ===============================
         Stack name                   : lambda-streaming-nodejs-app
         Region                       : us-west-2
         Disable rollback             : False
         Deployment s3 bucket         : aws-sam-cli-managed-default-samclisam-s3-demo-
bucket-1a4x26zbcdkqr
         Capabilities                 : ["CAPABILITY_NAMED_IAM", 
 "CAPABILITY_AUTO_EXPAND"]
         Parameter overrides          : {}
         Signing Profiles             : null
```

```
Initiating deployment
=====================


2023-06-20 12:11:16 - Waiting for stack create/update to complete

CloudFormation events from stack operations (refresh every 0.5 seconds)
-------------------------------------------------------------------------------------------------
ResourceStatus            ResourceType              LogicalResourceId
ResourceStatusReason
-------------------------------------------------------------------------------------------------
CREATE_IN_PROGRESS        AWS::CloudFormation::St   lambda-streaming-
Transformation
                          ack                       nodejs-app
succeeded
CREATE_IN_PROGRESS        AWS::IAM::Role            StreamingFunctionRole     -

CREATE_IN_PROGRESS        AWS::CloudFormation::St   AwsSamAutoDependencyLay   -

                          ack                       erNestedStack

CREATE_IN_PROGRESS        AWS::IAM::Role            StreamingFunctionRole     Resource
creation

Initiated
CREATE_IN_PROGRESS        AWS::CloudFormation::St   AwsSamAutoDependencyLay   Resource
creation
                          ack                       erNestedStack
Initiated
CREATE_COMPLETE           AWS::IAM::Role            StreamingFunctionRole     -

CREATE_COMPLETE           AWS::CloudFormation::St   AwsSamAutoDependencyLay   -

                          ack                       erNestedStack

CREATE_IN_PROGRESS        AWS::Lambda::Function     StreamingFunction         -

CREATE_IN_PROGRESS        AWS::Lambda::Function     StreamingFunction         Resource
creation

Initiated
CREATE_COMPLETE           AWS::Lambda::Function     StreamingFunction         -
```

```
CREATE_IN_PROGRESS        AWS::Lambda::Url          StreamingFunctionUrl      -

CREATE_IN_PROGRESS        AWS::Lambda::Url          StreamingFunctionUrl      Resource
creation

Initiated
CREATE_COMPLETE           AWS::Lambda::Url          StreamingFunctionUrl      -

CREATE_COMPLETE           AWS::CloudFormation::St   lambda-streaming-         -

                          ack                       nodejs-app

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

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

----------------------------------------------------------------------------------------
Key                 StreamingFunction

Description         Streaming Lambda Function ARN

Value               arn:aws:lambda:us-west-2:012345678910:function:lambda-streaming-
nodejs-app-
StreamingFunction-gUmhO833A0vZ


Key                 StreamingFunctionURL

Description         Streaming Lambda Function URL

Value               https://wxgkcc2dyntgtrwhf2dgdcvylu0rnnof.lambda-url.us-
west-2.on.aws/
----------------------------------------------------------------------------------------



Stack creation succeeded. Sync infra completed.



Infra sync completed.
```

이제 함수가 클라우드에 배포되었으므로 `sam remote invoke`를 사용하여 함수와 상호 작용할 수 있습니다. AWS SAM CLI는 함수가 응답 스트리밍을 위해 구성되었음을 자동으로 감지하고 즉시 함수의 스트리밍된 응답을 실시간으로 출력하기 시작합니다.

```
$ 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
```

함수 코드를 수정하면 AWS SAM CLI는 변경 사항을 즉시 감지하고 즉시 배포합니다. 다음은 함수 코드를 변경한 후의 AWS SAM CLI 출력의 예입니다.

```
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.
```

이제 다시 `sam remote invoke`를 사용하여 클라우드에서 함수와 상호 작용하고 변경 사항을 테스트할 수 있습니다.

SQS 예제

기본 예제

ARN을 리소스 ID로 제공하여 Amazon SQS 대기열을 간접 호출합니다.

```
$ sam remote invoke arn:aws:sqs:us-west-2:01234567890:sqs-example-4DonhBsjsW1b --
event '{"hello": "world"}' --output json

Sending message to SQS queue MySqsQueue

{
  "MD5OfMessageBody": "49dfdd54b01cbcd2d2ab5e9e5ee6b9b9",
  "MessageId": "4f464cdd-15ef-4b57-bd72-3ad225d80adc",
  "ResponseMetadata": {
    "RequestId": "95d39377-8323-5ef0-9223-ceb198bd09bd",
    "HTTPStatusCode": 200,
    "HTTPHeaders": {
      "x-amzn-requestid": "95d39377-8323-5ef0-9223-ceb198bd09bd",
      "date": "Wed, 08 Nov 2023 23:27:26 GMT",
      "content-type": "application/x-amz-json-1.0",
      "content-length": "106",
      "connection": "keep-alive"
    },
    "RetryAttempts": 0
  }
}%
```

Step Functions 예제

기본 예제

물리적 ID를 리소스 ID로 제공하여 상태 시스템을 간접 호출합니다.

먼저 `sam list resources`를 사용하여 물리적 ID를 가져옵니다.

```
$ 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"
  }
]
```

다음으로 물리적 ID를 리소스 ID로 사용하여 상태 시스템을 간접 호출합니다. 명령줄에서 `--event` 옵션을 사용하여 이벤트를 전달합니다.

```
$ 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"%
```

빈 이벤트를 전달하여 상태 시스템을 간접 호출합니다.

```
$ sam remote invoke HelloWorldStateMachine --stack-name state-machine-example

Invoking Step Function HelloWorldStateMachine

"Hello World"%
```

관련 링크

`sam remote invoke`과 관련되고 AWS SAM CLI를 사용하는 설명서는 다음을 참조하세요.

- sam remote invoke
- AWS SAMCLI 문제 해결

AWS SAM에서 로컬 통합 테스트 자동화

sam local invoke를 사용한 테스트 소개 섹션에 따라 코드를 수동으로 테스트할 수 있지만 AWS SAM에서는 자동 통합 테스트를 사용하여 코드를 테스트할 수도 있습니다. 통합 테스트를 사용하면 개발 주기 초기에 문제를 감지하고, 코드 품질을 개선하고, 시간을 절약하면서 비용을 절감할 수 있습니다.

AWS SAM에서 자동 통합 테스트를 작성하려면 AWS 클라우드에 배포하기 전에 먼저 로컬 Lambda 함수에 대해 테스트를 실행합니다. sam local start-lambda를 사용한 테스트 소개 명령은 Lambda 간접 호출 엔드포인트를 에뮬레이션하는 로컬 엔드포인트를 시작합니다. 자동화된 테스트에서 이를 간접 호출할 수 있습니다. 이 엔드포인트는 Lambda 간접 호출 엔드포인트를 에뮬레이션하므로 테스트를 한 번 작성한 다음 로컬 Lambda 함수 또는 배포된 Lambda 함수에 대해 수정 없이 테스트를 실행할 수 있습니다. CI/CD 파이프라인에 배포된 AWS SAM 스택에 대해 동일한 테스트를 실행할 수도 있습니다.

프로세스 방법은 다음과 같습니다.

1. 로컬 Lambda 엔드포인트를 시작합니다.

   귀하의 AWS SAM 템플릿이 포함된 디렉터리에서 다음 명령을 실행하여 로컬 Lambda 엔드포인트를 시작합니다.

   ```
   sam local start-lambda
   ```

   이 명령은 AWS Lambda를 에뮬레이션하는 `http://127.0.0.1:3001`에서 로컬 엔드포인트를 시작합니다. 이 로컬 Lambda 엔드포인트에 대해 자동 테스트를 실행할 수 있습니다. 귀하가 AWS CLI 또는 SDK를 사용하여 이 엔드포인트를 간접 호출하면, 그것은 요청에 지정된 Lambda 함수를 로컬에서 실행하고 응답을 반환합니다.

2. 로컬 Lambda 엔드포인트에 대해 통합 테스트를 실행합니다.

   통합 테스트에서 AWS SDK를 사용하여 테스트 데이터로 Lambda 함수를 간접 호출하고, 응답을 기다린 다음, 응답이 예상과 일치하는지 확인할 수 있습니다. 통합 테스트를 로컬에서 실행하려면

Lambda 간접 호출 API 직접 호출을 전송하여 이전 단계에서 시작한 로컬 Lambda 엔드포인트를 간접 호출하도록 AWS SDK를 구성해야 합니다.

다음은 Python 예제입니다(다른 언어용 AWS SDK도 비슷한 구성을 가짐).

```
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"
```

`running_locally`을 `False`로 설정함으로써 배포된 Lambda 함수를 테스트하기 위해 이 코드를 사용할 수 있습니다. 이렇게 하면 AWS 클라우드 내 AWS Lambda에 연결할 AWS SDK가 설정됩니다.

AWS SAM에서 샘플 이벤트 페이로드 생성

Lambda 함수를 테스트하기 위해 다른 AWS 서비스에 의해 트리거될 때 Lambda 함수가 수신할 데이터를 모방하는 샘플 이벤트 페이로드를 생성하고 사용자 지정할 수 있습니다. 여기에는 API Gateway, CloudFormation, Amazon S3 등과 같은 서비스가 포함됩니다.

샘플 이벤트 페이로드를 생성하면 라이브 환경에서 작업할 필요 없이 다양한 입력을 사용하여 Lambda 함수의 동작을 테스트할 수 있습니다. 또한 함수를 테스트하기 위해 AWS 서비스 이벤트 샘플을 수동으로 생성하는 것과 비교하여 이 접근 방식을 사용하면 시간을 절약할 수 있습니다.

샘플 이벤트 페이로드를 생성할 수 있는 전체 서비스 목록을 보려면 다음 명령을 사용합니다.

```
sam local generate-event --help
```

특정 서비스에 사용할 수 있는 옵션 목록을 보려면 다음 명령을 사용합니다.

```
sam local generate-event [SERVICE] --help
```

예시:

```
#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
```

내구성 함수 테스트 및 디버깅

내구성 있는 함수를 로컬에서 테스트하고 디버깅하는 작업은 체크포인트 및 재생을 자동으로 지원하는 일반 Lambda 함수와 비슷하게 작동합니다. 이 가이드에서는 일반적인 테스트 시나리오와 문제 해결 기법을 다룹니다.

로컬 테스트 워크플로

내구성이 뛰어난 함수를 로컬에서 테스트할 때 워크플로는 일반 Lambda 함수와 다릅니다.

내구성 있는 함수 테스트 워크플로

1. 애플리케이션을 빌드합니다.

```
$ sam build
```

2. 내구성 있는 함수를 호출합니다.

```
$ sam local invoke MyDurableFunction --durable-execution-name test
```

3. 필요한 경우 실행 기록을 확인합니다.

```
$ sam local execution history execution-id
```

일반적인 테스트 시나리오

체크포인트 동작 테스트

함수가 상태를 올바르게 체크포인트하는지 테스트하려면:

```
# Example Python durable function
def handler(event, context):
    # This will create a checkpoint
    context.wait(10)  # Wait 10 seconds

    # Function resumes here after wait
    return {"message": "Function resumed after wait"}
```

이 함수를 로컬에서 호출하면 대기 기간이 자동으로 처리됩니다.

콜백 시나리오 테스트

외부 콜백을 기다리는 함수의 경우:

1. 콜백을 기다리는 내구성 있는 함수 시작
2. 다른 터미널에서 콜백을 해결합니다.

```
$ sam local callback succeed callback-id
```

3. 함수 재개 실행 관찰

문제 해결

내구성 함수가 제대로 실행되지 않음

문제: 함수가 내구성 있는 함수로 동작하지 않습니다.

솔루션:

- SAM 템플릿에 `DurableConfig`가 설정되어 있는지 확인
- 함수 코드가 내구성 있는 함수 SDK 메서드(예: `context.wait()`)를 사용하는지 확인합니다.
- 지원되는 런타임(TypeScript, JavaScript, Python)을 사용하고 있는지 확인합니다.

실행 기록을 검색할 수 없음

문제: `local execution history` 명령이 결과를 반환하지 않습니다.

솔루션:

- 실행 ID가 올바른지 확인
- 함수가 한 번 이상 호출되었는지 확인

콜백 명령이 작동하지 않음

문제: 콜백 명령이 보류 중인 작업을 해결하지 못합니다.

솔루션:

- 콜백 ID가 올바른지 확인
- 함수가 실제로 콜백을 기다리고 있는지 확인합니다.
- 올바른 콜백 명령 구문을 사용하고 있는지 확인

디버깅 팁

- 실행 기록 사용 - 실행 기록을 검토하여 내구성 있는 함수의 흐름을 이해합니다.
- 점진적으로 테스트 - 복잡한 로직을 추가하기 전에 간단한 대기 작업으로 시작
- 상세 로깅 사용 - 세부 로깅을 활성화하여 실행 흐름 추적

자세히 알아보기

테스트 및 디버깅에 대한 자세한 내용은 다음을 참조하세요.

- sam local invoke를 사용한 테스트 소개 - 로컬 호출 설명서
- sam local execution history - 실행 기록

AWS SAM을 사용하여 서버리스 애플리케이션 디버깅

애플리케이션을 테스트한 후에는 발견된 문제를 디버깅해야 합니다. AWS SAM 명령줄 인터페이스 (CLI)를 사용하면 서버리스 애플리케이션을 AWS 클라우드에 업로드하기 전에 로컬에서 테스트하고 디버깅할 수 있습니다. 애플리케이션을 디버깅하여 애플리케이션의 문제 또는 오류를 식별하고 수정합니다.

AWS SAM을 사용하면 코드 줄이나 명령을 한 번에 하나씩 실행하는 방법인 단계별 디버깅을 수행할 수 있습니다. AWS SAM CLI 내에서 디버그 모드로 Lambda 함수를 로컬로 호출하면 디버거를 연결할 수 있습니다. 디버거를 사용하면 코드를 한 줄씩 단계별로 실행하고, 다양한 변수의 값을 확인하고, 다른 애플리케이션과 동일한 방식으로 문제를 해결할 수 있습니다. 애플리케이션 패키징 및 배포 단계를 진행하기 전에 애플리케이션이 예상대로 작동하는지 확인하고, 무엇이 잘못되었는지 디버깅하고, 문제를 해결할 수 있습니다.

> **Note**
>
> 애플리케이션에 하나 이상의 계층이 포함된 경우 로컬에서 애플리케이션을 실행하고 디버깅하면 계층 패키지가 다운로드되고 로컬 호스트에 캐시됩니다. 자세한 내용은 레이어가 로컬에 캐싱되는 방법 섹션을 참조하세요.

주제

- AWS SAM에서 로컬로 함수 디버깅
- AWS SAM에서 디버깅할 때 여러 런타임 인수 전달
- CloudFormation Linter를 사용하여 AWS SAM 애플리케이션을 검증합니다.

AWS SAM에서 로컬로 함수 디버깅

다양한 AWS SAM 도구 키트 및 AWS 디버거와 함께 사용하여 로컬에서 서버리스 애플리케이션을 테스트하고 디버깅할 수 있습니다. Lambda 함수의 단계별 디버깅을 사용하면 로컬 환경에서 코드 줄이나 명령을 한 번에 하나씩 실행하여 애플리케이션의 문제를 식별하고 수정할 수 있습니다.

로컬 단계별 디버깅을 수행하는 방법에는 중단점을 설정하고, 변수를 검사하고, 함수 코드를 한 번에 한 줄씩 실행하는 것이 포함됩니다. 로컬 단계별 디버깅을 사용하면 클라우드에서 발생할 수 있는 문제를 찾아 해결할 수 있어 피드백 루프가 강화됩니다.

AWS 도구 키트를 사용하여 디버깅할 수 있으며 AWS SAM을 디버깅 모드에서 실행할 수도 있습니다. 자세한 내용은 이 섹션의 주제를 참조하세요.

AWS 도구 키트 사용

AWS도구 키트는 중단점 설정, 변수 검사, 함수 코드 실행 등 여러 일반적인 디버깅 작업을 한 번에 한 줄씩 수행할 수 있는 기능을 제공하는 통합 개발 환경 (IDE) 플러그인입니다. AWS 이 도구 키트를 통해 AWS SAM를 사용하여 구축된 서버리스 애플리케이션을 더 쉽게 개발, 디버그 및 배포할 수 있습니다. IDE에 통합된 Lambda 함수를 구축, 테스트, 디버깅, 배포 및 간접 호출할 수 있는 환경을 제공합니다.

AWS과 함께 사용할 수 있는 AWS SAM 도구 키트에 대한 자세한 내용은 다음을 참조하세요.

- AWS Toolkit for Visual Studio Code
- AWS Cloud9
- AWS Toolkit for JetBrains

다양한 IDE 및 런타임 조합과 함께 작동하는 여러 AWS 도구 키트가 있습니다. 다음 표에는 AWS SAM 애플리케이션의 단계별 디버깅을 지원하는 일반적인 IDE/런타임 조합이 나와 있습니다.

IDE	런타임	AWS 도구 키트	단계별 디버깅 지침
Visual Studio Code	• Node.js • Python • .NET • Java • Go	AWS Toolkit for Visual Studio Code	사용자 가이드AWS 서버리스 애플리케이션 내 AWS Toolkit for Visual Studio Code를 이용한 작업
AWS Cloud9	• Node.js • Python	AWS Cloud9, AWS 도구 키트가 활성화된 상태[1]	사용자 가이드AWS 내에서 AWS 도구 키트를 사용하는 AWS Cloud9서버리스 애플리케이션으로 작업합니다.
WebStorm	Node.js	AWS Toolkit for JetBrains[2]	https://docs.aws.amazon.com/toolki

IDE	런타임	AWS 도구 키트	단계별 디버깅 지침
			t-for-jetbrains/latest/userguide/invoke-lambda.html내 로컬 함수 AWS Toolkit for JetBrains실행(간접적으로 호출) 또는 디버깅
PyCharm	Python	AWS Toolkit for JetBrains[2]	https://docs.aws.amazon.com/toolkit-for-jetbrains/latest/userguide/invoke-lambda.html내 로컬 함수 AWS Toolkit for JetBrains실행(간접적으로 호출) 또는 디버깅
Rider	.NET	AWS Toolkit for JetBrains[2]	https://docs.aws.amazon.com/toolkit-for-jetbrains/latest/userguide/invoke-lambda.html내 로컬 함수 AWS Toolkit for JetBrains실행(간접적으로 호출) 또는 디버깅

IDE	런타임	AWS 도구 키트	단계별 디버깅 지침
IntelliJ	Java	AWS Toolkit for JetBrains[2]	https://docs.aws.amazon.com/toolkit-for-jetbrains/latest/userguide/invoke-lambda.html내 로컬 함수 AWS Toolkit for JetBrains실행(간접적으로 호출) 또는 디버깅
GoLand	Go	AWS Toolkit for JetBrains[2]	https://docs.aws.amazon.com/toolkit-for-jetbrains/latest/userguide/invoke-lambda.html내 로컬 함수 AWS Toolkit for JetBrains실행(간접적으로 호출) 또는 디버깅

참고:

1. AWS Cloud9애플리케이션AWS SAM을 단계별로 디버깅하는 데 사용하려면 AWS 도구 키트을 활성화해야 합니다. 자세한 내용은 사용자 가이드AWS에서 도구 키트AWS Cloud9 활성화를 참조하세요.
2. AWS Toolkit for JetBrains단계별 디버그 AWS SAM 응용 프로그램을 사용하려면 먼저 내의 설치 AWS Toolkit for JetBrains에 있는 지침에 따라 AWS Toolkit for JetBrains 응용 프로그램을 설치하고 구성해야 합니다.

디버그 모드에서 AWS SAM을 로컬에서 실행

AWS 도구 키트과 통합하는 것 외에도 디버그 모드에서 AWS SAM를 실행하여 ptvsd 또는 delve와 같은 타사 디버거에 연결할 수도 있습니다.

디버그 모드에서 AWS SAM을 실행하려면 [sam local invoke](#) 또는 [sam local start-api](#) 명령을 `--debug-port` 또는 `-d` 옵션과 함께 사용하십시오.

예:

```
# 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**
>
> `sam local start-api`을 사용하는 경우 로컬 API 게이트웨이 인스턴스는 모든 Lambda 함수를 노출합니다. 하지만 단일 디버그 포트를 지정할 수 있으므로 한 번에 하나의 함수만 디버깅할 수 있습니다. AWS SAM CLI가 포트에 바인딩하기 전에 API를 직접 호출해야 디버거가 연결할 수 있습니다.

AWS SAM에서 디버깅할 때 여러 런타임 인수 전달

문제를 검사하고 변수를 보다 효과적으로 해결하기 위해 AWS SAM에 추가 런타임 인수를 전달하도록 선택할 수 있습니다. 이렇게 하면 디버깅 프로세스에 대한 제어 및 유연성이 향상되어 사용자 지정 런타임 구성 및 환경에서 더 쉽게 작업할 수 있습니다.

함수를 디버깅할 때 추가 런타임 인수를 전달하려면 환경 변수 `DEBUGGER_ARGS`를 사용합니다. 그러면 함수를 시작하는 데 AWS SAM CLI가 사용하는 실행 명령에 인수 문자열이 직접 전달됩니다.

예를 들어, Python 함수의 런타임에 IkPDB와 같은 디버거를 로드하려면 다음 항목들을 `DEBUGGER_ARGS: -m ikpdb --ikpdb-port=5858 --ikpdb-working-directory=/var/task/ --ikpdb-client-working-directory=/myApp --ikpdb-address=0.0.0.0`로서 전달할 수 있습니다. 이렇게 하면 지정한 다른 인수와 함께 IkPDB가 런타임에 로드됩니다.

이 경우 전체 AWS SAM CLI 명령은 다음과 같습니다.

```
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
```

모든 런타임의 함수에 디버거 인수를 전달할 수 있습니다.

CloudFormation Linter를 사용하여 AWS SAM 애플리케이션을 검증합니다.

AWS CloudFormation Linter(cfn-lint)는 CloudFormation 템플릿에 대한 세부 검증을 수행하는 데 사용할 수 있는 오픈 소스 도구입니다. Cfn-lint에는 CloudFormation 리소스 사양에 따라 안내되는 규칙이 포함되어 있습니다. cfn-lint를 사용하여 리소스를 해당 규칙과 비교하여 오류, 경고 또는 정보 제안에 대한 자세한 메시지를 받을 수 있습니다. 또는 자체 사용자 지정 규칙을 만들어 유효성을 검사할 수도 있습니다. cfn-lint에 대해 자세히 알아보려면 CloudFormation GitHub 리포지토리에서 cfn-lint를 참조합니다.

cfn-lint를 사용하면 sam validate를 --lint 옵션과 함께 실행하여 AWS SAM 명령줄 인터페이스(AWS SAM CLI)를 통해 AWS Serverless Application Model(AWS SAM) 템플릿의 유효성을 검사할 수 있습니다.

```
sam validate --lint
```

사용자 지정 규칙 생성 또는 유효성 검사 옵션 지정과 같은 cfn-lint 동작을 사용자 지정하려면 구성 파일을 정의하면 됩니다. 자세한 내용은 cfn-lint CloudFormation GitHub 리포지토리의 Config File 섹션을 참조하세요. sam validate --lint를 실행하면 구성 파일에 정의된 cfn-lint 동작이 적용됩니다.

예시

AWS SAM 템플릿에서 cfn-lint 유효성 검사 수행

```
sam validate --lint --template myTemplate.yaml
```

자세히 알아보기

sam validate 명령에 대한 자세한 내용은 sam validate 섹션을 참조하세요.

AWS SAM을 사용하여 애플리케이션 및 리소스 배포

애플리케이션을 배포하면 AWS 클라우드에서 AWS 리소스가 프로비저닝되고 구성되며, 애플리케이션이 클라우드에서 실행됩니다. AWS SAM은 기본 배포 메커니즘으로 CloudFormation을 사용합니다. AWS SAM은 sam build 명령을 실행할 때 생성한 빌드 아티팩트를 서버리스 애플리케이션을 배포하기 위한 표준 입력으로 사용합니다.

AWS SAM을 사용하면 서버리스 애플리케이션을 수동으로 배포하거나 배포를 자동화할 수 있습니다. 배포를 자동화하려면 원하는 지속적 통합 및 지속적 배포(CI/CD) 시스템이 포함된 AWS SAM 파이프라인을 사용합니다. 배포 파이프라인은 서버리스 애플리케이션의 새 버전을 릴리스하기 위해 수행되는 자동화된 단계의 시퀀스입니다.

이 섹션의 주제에서는 자동 및 수동 배포 모두에 대한 지침을 제공합니다. 애플리케이션을 수동으로 배포하려면 AWS SAM CLI 명령을 사용합니다. 배포를 자동화하려면 이 섹션의 주제를 참조하세요. 특히 파이프라인과 CI/CD 시스템을 사용하여 배포를 자동화하는 것과 관련된 심층적인 콘텐츠를 제공합니다. 여기에는 스타터 파이프라인 생성, 자동화 설정, 배포 문제 해결, OpenID Connect(OIDC) 사용자 인증 사용, 배포 시 로컬 파일 업로드 등이 포함됩니다.

주제

- AWS SAM 배포 소개
- AWS SAM에서 애플리케이션을 배포하는 옵션
- AWS SAM에서 CI/CD 시스템 및 파이프라인을 사용하여 배포
- sam sync를 사용하는 AWS 클라우드 동기화 소개

AWS SAM 배포 소개

AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) `sam deploy` 명령을 사용하여 서버리스 애플리케이션을 AWS 클라우드에 배포합니다.

- AWS SAM CLI에 대한 소개는 란 무엇입니까 AWS SAMCLI? 섹션을 참조하세요.
- `sam deploy` 명령 옵션 목록은 sam deploy 섹션을 참조하세요.
- 일반적인 개발 워크플로 중 `sam deploy`를 사용하는 예는 3단계: AWS 클라우드에 대한 애플리케이션 배포 섹션을 참조하세요.

주제

- 사전 조건
- sam deploy를 사용하여 애플리케이션 배포하기
- 모범 사례
- sam deploy 옵션
- 문제 해결
- 예시
- 자세히 알아보기

사전 조건

`sam deploy`를 사용하려면 다음을 완료하여 AWS SAM CLI를 설치합니다.

- AWS SAM 사전 조건.
- AWS SAM CLI 설치.

`sam deploy`를 사용하기 전에 다음 사항에 대한 기본적인 이해를 하는 것이 좋습니다.

- AWS SAM CLI 구성.
- 에서 애플리케이션 생성 AWS SAM.
- AWS SAM의 빌드 소개.

sam deploy를 사용하여 애플리케이션 배포하기

서버리스 애플리케이션을 처음 배포하는 경우 `--guided` 옵션을 사용합니다. AWS SAM CLI는 애플리케이션의 배포 설정을 구성하는 대화형 흐름을 안내합니다.

대화형 흐름을 사용하여 애플리케이션을 배포하려면

1. 프로젝트의 루트 디렉터리로 이동합니다. AWS SAM 템플릿과 같은 위치입니다.

```
$ cd sam-app
```

2. 다음 명령을 실행합니다.

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

3. 대화형 흐름 중에 AWS SAM CLI에 애플리케이션의 배포 설정을 구성할 수 있는 옵션을 묻는 메시지가 표시됩니다.

   대괄호([ ])는 기본값을 나타냅니다. 기본값을 선택하려면 답을 비워둡니다. 기본값은 다음 구성 파일에서 가져옵니다.

   - `~/.aws/config` - 일반 AWS 계정 설정.
   - `~/.aws/credentials` - AWS 계정 자격 증명.
   - *`<project>`*`/samconfig.toml` - 프로젝트의 구성 파일.

   AWS SAM CLI 프롬프트에 응답하여 값을 제공합니다. 예를 들어, 예(yes)는 **y**, 아니요(no)는 **n**, 또는 문자열 값을 입력할 수 있습니다.

   AWS SAM CLI는 프로젝트 `samconfig.toml` 파일에 응답을 기록합니다. 후속 배포의 경우 `sam deploy`를 사용하여 이러한 구성된 값을 사용해 배포할 수 있습니다. 이러한 값을 재구성하려면 다시 `sam deploy --guided`를 사용하거나 직접 구성 파일을 수정합니다.

   다음은 출력의 예제입니다.

```
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. 그런 다음 AWS SAM CLI는 애플리케이션을 AWS 클라우드에 배포합니다. 배포하는 동안 명령 프롬프트에 진행 상황이 표시됩니다. 배포의 주요 단계는 다음과 같습니다.

   - .zip 파일 아카이브로 패키징된 AWS Lambda 함수가 있는 애플리케이션의 경우, AWS SAM CLI는 Amazon Simple Storage Service(S3) 버킷에 패키지를 압축하고 업로드합니다. 필요한 경우 AWS SAM CLI가 새 버킷을 생성합니다.
   - Lambda 함수가 컨테이너 이미지로 패키징되는 애플리케이션의 경우, AWS SAM CLI는 Amazon Elastic Container Registry(Amazon ECR)에 이미지를 업로드합니다. 필요한 경우 AWS SAM CLI가 새 리포지토리를 생성합니다.
   - AWS SAM CLI는 AWS CloudFormation 변경 세트를 생성하여 애플리케이션을 스택으로 CloudFormation에 배포합니다.
   - AWS SAM CLI는 배포된 `CodeUri` 템플릿을 Lambda 함수의 새 AWS SAM 값으로 수정합니다.

   다음은 AWS SAM CLI 배포의 예제입니다.

```
        Looking for resources needed for deployment:

        Managed S3 bucket: aws-sam-cli-managed-default-samclisam-s3-demo-
bucket-1a4x26zbcdkqr
        A different default S3 bucket can be set in samconfig.toml and auto
 resolution of buckets turned off by setting resolve_s3=False

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

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

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

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

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

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


Waiting for changeset to be created..

CloudFormation stack changeset
---------------------------------------------------------------------------------------------
Operation                LogicalResourceId        ResourceType
 Replacement
---------------------------------------------------------------------------------------------
+ Add                    HelloWorldFunctionHell   AWS::Lambda::Permissio   N/A

                         oWorldPermissionProd     n

+ Add                    HelloWorldFunctionRole   AWS::IAM::Role           N/A

+ Add                    HelloWorldFunction       AWS::Lambda::Function    N/A

+ Add                    ServerlessRestApiDeplo   AWS::ApiGateway::Deplo   N/A

                         yment47fc2d5f9d          yment
```

```
+ Add                         ServerlessRestApiProdS   AWS::ApiGateway::Stage   N/A

                              tage

+ Add                         ServerlessRestApi        AWS::ApiGateway::RestA   N/A

                                                       pi

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


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


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

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

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

CREATE_IN_PROGRESS       AWS::IAM::Role           HelloWorldFunctionRole   Resource
 creation

 Initiated
CREATE_COMPLETE          AWS::IAM::Role           HelloWorldFunctionRole   -

CREATE_IN_PROGRESS       AWS::Lambda::Function    HelloWorldFunction       -

CREATE_IN_PROGRESS       AWS::Lambda::Function    HelloWorldFunction       Resource
 creation

 Initiated
CREATE_COMPLETE          AWS::Lambda::Function    HelloWorldFunction       -
```

```
CREATE_IN_PROGRESS          AWS::ApiGateway::RestA   ServerlessRestApi        -

                            pi

CREATE_IN_PROGRESS          AWS::ApiGateway::RestA   ServerlessRestApi        Resource
 creation
                            pi
 Initiated
CREATE_COMPLETE             AWS::ApiGateway::RestA   ServerlessRestApi        -

                            pi

CREATE_IN_PROGRESS          AWS::Lambda::Permissio   HelloWorldFunctionHell   -

                            n                        oWorldPermissionProd

CREATE_IN_PROGRESS          AWS::ApiGateway::Deplo   ServerlessRestApiDeplo   -

                            yment                    yment47fc2d5f9d

CREATE_IN_PROGRESS          AWS::Lambda::Permissio   HelloWorldFunctionHell   Resource
 creation
                            n                        oWorldPermissionProd
 Initiated
CREATE_IN_PROGRESS          AWS::ApiGateway::Deplo   ServerlessRestApiDeplo   Resource
 creation
                            yment                    yment47fc2d5f9d
 Initiated
CREATE_COMPLETE             AWS::ApiGateway::Deplo   ServerlessRestApiDeplo   -

                            yment                    yment47fc2d5f9d

CREATE_IN_PROGRESS          AWS::ApiGateway::Stage   ServerlessRestApiProdS   -

                                                     tage

CREATE_IN_PROGRESS          AWS::ApiGateway::Stage   ServerlessRestApiProdS   Resource
 creation
                                                     tage
 Initiated
CREATE_COMPLETE             AWS::ApiGateway::Stage   ServerlessRestApiProdS   -

                                                     tage
```

```
CREATE_COMPLETE          AWS::Lambda::Permissio   HelloWorldFunctionHell   -

                         n                        oWorldPermissionProd

CREATE_COMPLETE          AWS::CloudFormation::S   sam-app-zip              -

                         tack

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

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

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

Description            Implicit IAM Role created for Hello World function

Value                  arn:aws:iam::012345678910:role/sam-app-zip-

HelloWorldFunctionRole-11ZOGSCG28H0M


Key                    HelloWorldApi

Description            API Gateway endpoint URL for Prod stage for Hello World
 function
Value                  https://njzfhdmls0.execute-api.us-west-2.amazonaws.com/Prod/
hello/

Key                    HelloWorldFunction

Description            Hello World Lambda Function ARN

Value                  arn:aws:lambda:us-west-2:012345678910:function:sam-app-

HelloWorldFunction-XPqNX4TBu7qn

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



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

5. 배포한 애플리케이션을 보려면 다음 작업을 수행합니다.

   1. https://console.aws.amazon.com/cloudformation URL을 사용하여 CloudFormation 콘솔을 직접 엽니다.
   2. 스택을 선택합니다.
   3. 애플리케이션 이름으로 스택을 식별하고 선택합니다.

배포 전에 변경 사항을 확인합니다.

CloudFormation 변경 세트를 표시하고 배포하기 전에 확인을 요청하도록 AWS SAM CLI를 구성할 수 있습니다.

배포 전에 변경 내용을 확인하려면

1. `sam deploy --guided` 중에 **Y**를 입력하여 배포 전에 변경 사항을 확인합니다.

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

   또는 다음과 같이 `samconfig.toml` 파일을 수정할 수 있습니다.

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

2. 배포 중에, 배포 전에 변경 사항을 확인하라는 메시지가 AWS SAM CLI에 표시됩니다. 다음은 예제입니다.

```
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
```

배포 중에 추가 파라미터 지정

배포 시 구성할 추가 파라미터 값을 지정할 수 있습니다. AWS SAM 템플릿을 수정하고 배포 중에 파라미터 값을 구성하여 이 작업을 수행할 수 있습니다.

추가 파라미터를 지정하려면

1. AWS SAM 템플릿의 `Parameters` 섹션을 수정합니다. 다음은 예제입니다.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Globals:
...
Parameters:
  DomainName:
    Type: String
```

```
    Default: example
    Description: Domain name
```

2. `sam deploy --guided`을(를) 실행합니다. 다음은 출력의 예제입니다.

```
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
```

Lambda 함수에 대한 코드 서명

배포 시 Lambda 함수의 코드 서명을 구성할 수 있습니다. AWS SAM 템플릿을 수정하고 배포 중에 코드 서명을 구성하여 이 작업을 수행할 수 있습니다.

코드 서명을 구성하려면

1. AWS SAM 템플릿에 `CodeSigningConfigArn`을 지정합니다. 다음은 예제입니다.

```
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. `sam deploy --guided`을(를) 실행합니다. AWS SAM CLI에 코드 서명을 구성하라는 메시지가 표시됩니다. 다음은 출력의 예제입니다.

```
#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):
```

모범 사례

- `sam deploy`를 사용하면 AWS SAM CLI가 `.aws-sam` 디렉터리에 있는 애플리케이션의 빌드 아티팩트를 배포합니다. 애플리케이션의 원본 파일을 변경하는 경우 배포하기 전에 `sam build`를 실행하여 `.aws-sam`디렉터리를 업데이트합니다.
- 애플리케이션을 처음 배포하는 경우 `sam deploy --guided`를 사용하여 배포 설정을 구성합니다. 후속 배포의 경우 `sam deploy`를 사용하여 구성된 설정으로 배포할 수 있습니다.

sam deploy 옵션

다음은 `sam deploy`에 일반적으로 사용되는 옵션입니다. 콘텐츠 옵션 목록은 sam deploy 섹션을 참조하세요.

안내된 대화형 흐름을 사용하여 애플리케이션을 배포합니다.

`--guided` 옵션을 사용하여 대화형 흐름을 통해 애플리케이션의 배포 설정을 구성할 수 있습니다. 다음은 예제입니다.

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

애플리케이션의 배포 설정은 프로젝트 `samconfig.toml` 파일에 저장됩니다. 자세한 내용은 프로젝트 설정 구성 섹션을 참조하세요.

문제 해결

AWS SAM CLI 문제를 해결하려면 AWS SAMCLI 문제 해결 섹션을 참조하세요.

예시

.zip 파일 아카이브로 패키징된 Lambda 함수를 포함하는 Hello World 애플리케이션을 배포합니다.

예제는 Hello World 애플리케이션 자습서의 3단계: AWS 클라우드에 대한 애플리케이션 배포 섹션을 참조하세요.

컨테이너 이미지로 패키징된 Lambda 함수를 포함하는 Hello World 애플리케이션을 배포합니다.

먼저 `sam init`를 사용하여 Hello World 애플리케이션을 만듭니다. 대화형 흐름 중에 `Python3.9` 런타임과 `Image` 패키지 유형을 선택합니다.

```
$ 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
...
```

다음으로 프로젝트의 루트 디렉터리에 cd를 사용하여 `sam build`를 실행합니다. AWS SAM CLI는 Docker를 사용하여 로컬에서 Lambda 함수를 빌드합니다.

```
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
```

다음으로 `sam deploy --guided`를 실행하여 애플리케이션을 배포합니다. AWS SAM CLI는 배포 설정을 구성하는 방법을 안내합니다. 그런 다음 AWS SAM CLI는 애플리케이션을 AWS 클라우드에 배포합니다.

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

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

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

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

        Looking for resources needed for deployment:

        Managed S3 bucket: aws-sam-cli-managed-default-samclisam-s3-demo-
bucket-1a4x26zbcdkqr
        A different default S3 bucket can be set in samconfig.toml and auto resolution
 of buckets turned off by setting resolve_s3=False

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

        Saved arguments to config file
```

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

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

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

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

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

```
Waiting for changeset to be created..

CloudFormation stack changeset
-------------------------------------------------------------------------------------------------
Operation                LogicalResourceId        ResourceType             Replacement

-------------------------------------------------------------------------------------------------
+ Add                    HelloWorldFunctionHell   AWS::Lambda::Permissio   N/A

                         oWorldPermissionProd     n

+ Add                    HelloWorldFunctionRole   AWS::IAM::Role           N/A

+ Add                    HelloWorldFunction       AWS::Lambda::Function    N/A

+ Add                    ServerlessRestApiDeplo   AWS::ApiGateway::Deplo   N/A

                         yment47fc2d5f9d          yment

+ Add                    ServerlessRestApiProdS   AWS::ApiGateway::Stage   N/A

                         tage

+ Add                    ServerlessRestApi        AWS::ApiGateway::RestA   N/A

                                                  pi

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


Changeset created successfully. arn:aws:cloudformation:us-
west-2:012345678910:changeSet/samcli-deploy1680634124/0ffd4faf-2e2b-487e-
b9e0-9116e8299ac4


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

2023-04-04 08:49:15 - Waiting for stack create/update to complete

CloudFormation events from stack operations (refresh every 5.0 seconds)
-------------------------------------------------------------------------------------------------
```

```
ResourceStatus            ResourceType              LogicalResourceId
 ResourceStatusReason
------------------------------------------------------------------------------------------------------
CREATE_IN_PROGRESS        AWS::CloudFormation::S    sam-app                   User
 Initiated
                          tack

CREATE_IN_PROGRESS        AWS::IAM::Role            HelloWorldFunctionRole    -

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

CREATE_COMPLETE           AWS::IAM::Role            HelloWorldFunctionRole    -

CREATE_IN_PROGRESS        AWS::Lambda::Function     HelloWorldFunction        -

CREATE_IN_PROGRESS        AWS::Lambda::Function     HelloWorldFunction        Resource
 creation
                                                                              Initiated

CREATE_COMPLETE           AWS::Lambda::Function     HelloWorldFunction        -

CREATE_IN_PROGRESS        AWS::ApiGateway::RestA    ServerlessRestApi         -

                          pi

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

CREATE_COMPLETE           AWS::ApiGateway::RestA    ServerlessRestApi         -

                          pi

CREATE_IN_PROGRESS        AWS::Lambda::Permissio    HelloWorldFunctionHell    -

                          n                         oWorldPermissionProd

CREATE_IN_PROGRESS        AWS::ApiGateway::Deplo    ServerlessRestApiDeplo    -

                          yment                     yment47fc2d5f9d
```

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

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

CREATE_COMPLETE           AWS::ApiGateway::Deplo   ServerlessRestApiDeplo   -

                          yment                    yment47fc2d5f9d

CREATE_IN_PROGRESS        AWS::ApiGateway::Stage   ServerlessRestApiProdS   -

                                                   tage

CREATE_IN_PROGRESS        AWS::ApiGateway::Stage   ServerlessRestApiProdS   Resource
 creation
                                                   tage                     Initiated

CREATE_COMPLETE           AWS::ApiGateway::Stage   ServerlessRestApiProdS   -

                                                   tage

CREATE_COMPLETE           AWS::Lambda::Permissio   HelloWorldFunctionHell   -

                          n                        oWorldPermissionProd

CREATE_COMPLETE           AWS::CloudFormation::S   sam-app                  -

                          tack

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

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

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

Description           Implicit IAM Role created for Hello World function
```

```
Value               arn:aws:iam::012345678910:role/sam-app-HelloWorldFunctionRole-
JFML1JOKHJ71

Key                 HelloWorldApi

Description         API Gateway endpoint URL for Prod stage for Hello World function

Value               https://endlwiqqod.execute-api.us-west-2.amazonaws.com/Prod/hello/


Key                 HelloWorldFunction

Description         Hello World Lambda Function ARN

Value               arn:aws:lambda:us-west-2:012345678910:function:sam-app-
HelloWorldFunction-
kyg6Y2iNRUPg

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


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

자세히 알아보기

AWS SAM CLI `sam deploy` 명령 사용에 대한 자세한 내용은 다음을 참조하세요.

- 전체 AWS SAM 워크숍: 모듈 3 - 수동으로 배포 – AWS SAM CLI를 사용하여 서버리스 애플리케이션을 빌드, 패키징 및 배포하는 방법을 알아보세요.

AWS SAM에서 애플리케이션을 배포하는 옵션

AWS SAM에서는 애플리케이션을 수동으로 배포하고 배포를 자동화할 수 있습니다. AWS SAM CLI를 사용하여 애플리케이션을 수동으로 배포합니다. 배포를 자동화하려면 파이프라인과 지속적 통합 및 지속적 배포(CI/CD) 시스템을 사용합니다. 이 섹션의 주제에서는 두 접근 방식 모두에 대한 정보를 제공합니다.

주제

- AWS SAM CLI를 사용하여 수동으로 배포하는 방법
- CI/CD 시스템 및 파이프라인을 사용하여 배포

- 점진적 배포
- AWS SAM CLI를 이용한 배포 문제 해결
- 자세히 알아보기

AWS SAM CLI를 사용하여 수동으로 배포하는 방법

로컬에서 서버리스 애플리케이션을 개발하고 테스트한 후 sam deploy 명령을 사용하여 애플리케이션을 배포할 수 있습니다.

AWS SAM으로부터 프롬프트와 함께 배포 과정을 안내받으려면 --guided 플래그를 지정합니다. 이 플래그를 지정하면 sam deploy 명령은 애플리케이션 아티팩트를 압축하여 Amazon Simple Storage Service(Amazon S3)(.zip 파일 아카이브용) 또는 Amazon Elastic Container Registry(Amazon ECR)(컨테이너 이미지용)에 업로드합니다. 이 명령은 귀하의 애플리케이션을 AWS클라우드에 배포합니다.

예시:

```
# Deploy an application using prompts:
sam deploy --guided
```

CI/CD 시스템 및 파이프라인을 사용하여 배포

AWS SAM에서는 파이프라인과 지속적 통합 및 지속적 배포(CI/CD) 시스템을 사용하여 배포를 자동화하도록 지원합니다. AWS SAM을 사용하면 파이프라인을 생성하고 서버리스 애플리케이션을 위한 CI/CD 작업을 간소화할 수 있습니다. 여러 CI/CD 시스템이 AWS SAM 빌드 컨테이너 이미지를 지원합니다. 또한 AWS SAM은 AWS의 배포 모범 사례를 캡슐화하는 여러 CI/CD 시스템을 위한 기본 파이프라인 템플릿 세트를 제공합니다.

자세한 내용은 AWS SAM에서 CI/CD 시스템 및 파이프라인을 사용하여 배포 섹션을 참조하십시오.

점진적 배포

AWS SAM 애플리케이션을 한 번에 모두 배포하지 않고 점진적으로 배포하려는 경우 귀하는 AWS CodeDeploy가 제공하는 배포 구성을 지정할 수 있습니다. 자세한 정보는 AWS CodeDeploy 사용자 가이드의 CodeDeploy에서 배포 구성으로 작업을 참조하세요.

점진적으로 배포되도록 귀하의 AWS SAM 애플리케이션을 구성하는 방법에 대한 자세한 내용은을 참조하세요AWS SAM에서 서버리스 애플리케이션의 점진적 배포.

AWS SAM CLI를 이용한 배포 문제 해결

AWS SAM CLI오류: "보안 제약 조건이 충족되지 않았음"

sam deploy --guided 실행 중에 질문 `HelloWorldFunction may not have authorization defined, Is this okay? [y/N]`을 묻는 메시지가 표시됩니다. 이 프롬프트에 **N**(기본 응답)으로 응답하면 다음 오류가 표시됩니다.

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

귀하가 배포하려는 애플리케이션에 승인 없이 Amazon API Gateway API가 구성되어 있을 수 있다는 메시지를 프롬프트가 표시합니다. 이 프롬프트에 **N**으로 응답하는 것은 괜찮지 않다고 말하는 것입니다.

이 문제를 해결할 수 있도록 다음 옵션이 제공됩니다.

- 권한 부여를 통해 애플리케이션을 구성하세요. 권한 부여 구성에 대한 자세한 내용은을 참조하세요AWS SAM 템플릿으로 API 액세스 제어
- 이 질문에 **Y**로 응답함으로써, 권한 없이 구성된 API Gateway API가 있는 애플리케이션을 배포해도 괜찮다는 의사를 표시하십시오.

자세히 알아보기

서버리스 애플리케이션 배포의 실제 사례는 전체 AWS SAM워크숍을 출처로 하는 다음 사항들을 참조하세요.

- 모듈 3 - 수동 배포 - AWS SAM CLI을 사용하여 서버리스 애플리케이션을 빌드, 패키징 및 배포하는 방법을 알아봅니다.
- 모듈 4 - CI/CD - 지속적 통합 및 전달 (CI/CD) 파이프라인을 만들어 빌드, 패키지, 배포 단계를 자동화하는 방법을 알아봅니다.

AWS SAM에서 CI/CD 시스템 및 파이프라인을 사용하여 배포

AWS SAM은 조직이 선호하는 CI/CD 시스템을 위한 파이프라인을 만들 수 있도록 지원하여 배포 빈도 가속화, 변경 리드 타임 단축, 배포 오류 감소와 같은 CI/CD의 이점을 최소한의 노력으로 실현할 수 있습니다.

AWS SAM은 컨테이너 이미지 구축을 통해 서버리스 애플리케이션의 CI/CD 작업을 간소화합니다. AWS SAM이 제공하는 이미지에는 지원되는 여러 AWS Lambda런타임을 위한 AWS SAM CLI 및 빌드 도구가 포함되어 있습니다. 따라서 AWS SAM CLI를 사용하여 서버리스 애플리케이션을 더 쉽게 구축하고 패키징할 수 있습니다. 또한 이러한 이미지를 사용하면 작업팀들이 CI/CD 시스템용 이미지를 직접 만들고 관리해야 할 필요성이 줄어듭니다. AWS SAM 빌드 컨테이너 이미지에 대한 자세한 내용은을 참조하세요[의 이미지 리포지토리 AWS SAM](#).

여러 CI/CD 시스템이 AWS SAM 빌드 컨테이너 이미지를 지원합니다. 사용해야 하는 CI/CD 시스템은 여러 요인에 따라 달라집니다. 여기에는 애플리케이션이 단일 런타임을 사용하는지 다중 런타임을 사용하는지, 컨테이너 이미지 내에 애플리케이션을 구축할지 아니면 가상 머신(VM) 또는 베어메탈 호스트와 같은 호스트 머신에 직접 빌드할지 여부가 포함됩니다.

AWS SAM은 또한 AWS의 배포 모범 사례를 캡슐화하는 여러 CI/CD 시스템을 위한 기본 파이프라인 템플릿 세트를 제공합니다. 이러한 기본 파이프라인 템플릿은 표준 JSON/YAML 파이프라인 구성 형식을 사용하며, 내장된 모범 사례는 다중 계정 및 다중 지역 배포를 수행하고 파이프라인이 인프라를 의도하지 않게 변경할 수 없도록 확인하는 데 도움이 됩니다.

서버리스 애플리케이션을 AWS SAM 배포하는 데 사용할 수 있는 두 가지 기본 옵션이 있습니다. 1) AWS SAM CLI 명령을 사용하도록 기존 파이프라인 구성을 수정하는 것 혹은 2) 귀하 자신의 애플리케이션의 시작점으로 사용할 수 있는 예제 CI/CD 파이프라인 구성을 생성하는 것입니다.

주제

- 파이프라인이란 무엇입니까?
- 배포 시 AWS SAM이 로컬 파일을 업로드하는 방법
- AWS SAM에서 스타터 CI/CD 파이프라인 생성
- AWS SAM에서 스타터 파이프라인을 사용자 지정하는 방법
- AWS SAM 애플리케이션 배포 자동화
- AWS SAM 파이프라인과 함께 OIDC 인증을 사용하는 방법

파이프라인이란 무엇입니까?

파이프라인은 애플리케이션의 새 버전을 릴리스하기 위해 수행되는 자동화된 단계의 시퀀스입니다. AWS SAM에서는 AWS CodePipeline, Jenkins, GitLab CI/CD 및 GitHub Action을 비롯한 여러 일반 CI/CD 시스템을 사용하여 애플리케이션을 배포할 수 있습니다.

파이프라인 템플릿에는 다중 계정 및 다중 리전 배포에 도움이 되는 AWS 배포 모범 사례가 포함되어 있습니다. 개발 및 프로덕션과 같은 AWS 환경은 일반적으로 서로 다른 AWS 계정에 존재합니다. 이를

통해 개발 팀은 의도하지 않은 인프라 변경을 방지하면서 안전한 배포 파이프라인을 구성할 수 있습니다.

고유한 사용자 지정 파이프라인 템플릿을 제공하여 전체 개발 팀에서 파이프라인을 표준화할 수도 있습니다.

배포 시 AWS SAM이 로컬 파일을 업로드하는 방법

AWS 클라우드에 애플리케이션을 배포할 때 AWS CloudFormation에서 먼저 로컬 파일을 Amazon Simple Storage Service(Amazon S3)와 같은 액세스 가능한 AWS 서비스에 업로드해야 합니다. 여기에는 AWS SAM 템플릿이 참조하는 로컬 파일이 포함됩니다. 이 요구 사항을 충족하기 위해 `sam deploy` 또는 `sam package` 명령을 사용할 경우 AWS SAM CLI는 다음을 수행합니다.

1. 로컬 파일을 액세스 가능한 AWS 서비스에 자동으로 업로드합니다.
2. 새 파일 경로를 참조하도록 애플리케이션 템플릿을 자동으로 업데이트합니다.

주제

- 데모: AWS SAMCLI를 사용하여 Lambda 함수 코드를 업로드합니다.
- 지원되는 사용 사례
- 자세히 알아보기

데모: AWS SAMCLI를 사용하여 Lambda 함수 코드를 업로드합니다.

이 데모에서는 Lambda 함수에 .zip 패키지 유형을 사용하여 샘플 Hello World 애플리케이션을 초기화합니다. AWS SAM CLI를 사용하여 Lambda 함수 코드를 Amazon S3에 자동으로 업로드하고 애플리케이션 템플릿에서 새 경로를 참조합니다.

먼저 Hello World 애플리케이션을 초기화하기 위해 `sam init`를 실행합니다.

```
$ 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

...
```

Lambda 함수 코드는 프로젝트의 `hello_world` 하위 디렉터리에 구성되어 있습니다.

```
demo
### README.md
### hello_world
#   ### __init__.py
#   ### app.py
#   ### requirements.txt
### template.yaml
### tests
```

AWS SAM 템플릿 내에서 `CodeUri` 속성을 사용하여 Lambda 함수 코드의 로컬 경로를 참조합니다.

```
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
      ...
```

다음으로 애플리케이션을 빌드하고 배포를 준비하기 위해 `sam build`를 실행합니다.

```
$ 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
...
```

다음으로 `sam deploy --guided`를 실행하여 애플리케이션을 배포합니다.

```
$ sam deploy --guided

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

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

        Setting default arguments for 'sam deploy'
        =========================================
```

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

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

File with same data already exists at demo/da3c598813f1c2151579b73ad788cac8, skipping
 upload

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

Initiating deployment
=====================
...
```

```
Waiting for changeset to be created..
CloudFormation stack changeset
-------------------------------------------------------------------------------------------------
Operation                 LogicalResourceId         ResourceType              Replacement


-------------------------------------------------------------------------------------------------
+ Add                     HelloWorldFunctionHell    AWS::Lambda::Permissio    N/A

                          oWorldPermissionProd      n

+ Add                     HelloWorldFunctionRole    AWS::IAM::Role            N/A

...
-------------------------------------------------------------------------------------------------

Changeset created successfully. arn:aws:cloudformation:us-
west-2:012345678910:changeSet/samcli-deploy1680906292/1164338d-72e7-4593-a372-
f2b3e67f542f

2023-04-07 12:24:58 - Waiting for stack create/update to complete

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

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

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


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

Description          Implicit IAM Role created for Hello World function

```

```
Value                arn:aws:iam::012345678910:role/demo-HelloWorldFunctionRole-
VQ4CU7UY7S2K

Key                  HelloWorldApi

Description          API Gateway endpoint URL for Prod stage for Hello World function

Value                https://satnon55e9.execute-api.us-west-2.amazonaws.com/Prod/hello/


Key                  HelloWorldFunction

Description          Hello World Lambda Function ARN

Value                arn:aws:lambda:us-west-2:012345678910:function:demo-

HelloWorldFunction-G14inKTmSQvK

-------------------------------------------------------------------------------------------------
Successfully created/updated stack - demo in us-west-2
```

배포 중에 AWS SAM CLI는 Lambda 함수 코드를 Amazon S3에 자동으로 업로드하고 템플릿을 업데이트합니다. CloudFormation 콘솔의 수정된 템플릿은 Amazon S3 버킷 경로를 반영합니다.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function
    Properties:
      CodeUri: s3://aws-sam-cli-managed-default-samclisam-s3-demo-bucket-1a4x26zbcdkqr/
demo/da3c598813f1c2151579b73ad788cac8
      Handler: app.lambda_handler
      ...
```

지원되는 사용 사례

AWS SAM CLI는 다양한 파일 유형, CloudFormation리소스 유형 및 CloudFormation 매크로에 대해 이 프로세스를 자동으로 진행할 수 있습니다.

파일 유형

애플리케이션 파일 및 Docker 이미지가 지원됩니다.

CloudFormation리소스 유형

다음은 지원되는 리소스 유형 및 해당 속성의 목록입니다.

리소스	속성
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

리소스	속성
`AWS::Glue::Job`	`Command.ScriptLocation`
`AWS::Lambda::Function`	`Code` `Code.ImageUri`
`AWS::Lambda::LayerVersion`	`Content`
`AWS::Serverless::Api`	`DefinitionUri`
`AWS::Serverless::Function`	`CodeUri` `ImageUri`

`AWS::Serverless::GraphQLApi`	`SchemaUri`
	`Function.CodeUri`
	`Resolver.CodeUri`

`AWS::Serverless::HttpApi`	`DefinitionUri`
`AWS::Serverless::LayerVersion`	`ContentUri`
`AWS::Serverless::StateMachine`	`DefinitionUri`
`AWS::StepFunctions::StateMachine`	`DefinitionS3Location`

CloudFormation 매크로

`AWS::Include` 변환 매크로를 사용하여 참조된 파일이 지원됩니다.

자세히 알아보기

`AWS::Include` 변환에 대해 자세히 알아보려면 AWS CloudFormation 사용 설명서의 AWS::Include 변환 섹션을 참조하세요.

AWS SAM 템플릿에서 `AWS::Include` 변환을 사용하는 예를 보려면 Serverless Land의 API Gateway HTTP API to SQS 패턴을 참조하세요.

AWS SAM에서 스타터 CI/CD 파이프라인 생성

배포를 자동화할 준비가 되었으면 AWS SAM의 스타터 파이프라인 템플릿 중 하나를 사용하여 사용하려는 CI/CD 시스템에 대한 배포 파이프라인을 생성할 수 있습니다. 서버리스 애플리케이션의 배포를 자동화하려면 배포 파이프라인을 구성하고 사용해야 합니다. 스타터 파이프라인 템플릿은 서버리스 애플리케이션에 필요한 배포 파이프라인을 빠르게 설정할 수 있도록 사전 구성되어 있습니다.

스타터 파이프라인 템플릿이 있으면 sam pipeline init 명령을 사용하여 몇 분 안에 파이프라인을 생성할 수 있습니다.

스타터 파이프라인 템플릿은 CI/CD 시스템의 친숙한 JSON/YAML 구문을 사용하며, 여러 계정 및 리전의 아티팩트 관리, 애플리케이션 배포에 필요한 최소한의 권한 사용 등의 모범 사례를 통합합니다. 현재 AWS SAM CLI는 AWS CodePipeline, Jenkins, GitLab CI/CD, GitHub Actions, Bitbucket Pipeline에 대한 스타터 CI/CD 파이프라인 구성 생성을 지원합니다.

스타터 파이프라인 구성을 생성하기 위해 수행해야 하는 고급 수준은 태스크는 다음과 같습니다.

1. 인프라 리소스 생성 - 파이프라인에는 특정 AWS 리소스(예: 필요한 권한이 있는 IAM 사용자 및 역할, Amazon S3 버킷, 선택적으로 Amazon ECR 리포지토리)가 필요합니다.
2. Git 리포지토리를 CI/CD 시스템에 연결 - CI/CD 시스템은 파이프라인 실행을 트리거할 Git 리포지토리를 알아야 합니다. 사용 중인 Git 리포지토리와 CI/CD 시스템의 조합에 따라 이 단계가 필요하지 않을 수도 있습니다.
3. 파이프라인 구성 생성 - 이 단계는 두 개의 배포 단계를 포함하는 스타터 파이프라인 구성을 생성합니다.
4. 파이프라인 구성을 Git 리포지토리에 커밋 - 이 단계는 CI/CD 시스템이 파이프라인 구성을 인식하도록 하는데 필요하며, 변경 사항이 커밋될 때 실행됩니다.

스타터 파이프라인 구성을 생성하여 Git 리포지토리에 커밋하면 누군가 해당 리포지토리에 코드 변경을 커밋할 때마다 파이프라인이 트리거되어 자동으로 실행됩니다.

이러한 단계의 순서와 각 단계의 세부 사항은 CI/CD 시스템에 따라 다릅니다.

- AWS CodePipeline을 사용 중인 경우 AWS SAM에서 AWS CodePipeline을 위한 스타터 파이프라인 생성 섹션을 참조하세요.

- Jenkins, GitLab CI/CD, GitHub Actions 또는 Bitbucket Pipeline을 사용하는 경우 AWS SAM을 사용하여 Jenkins, GitLab CI/CD, GitHub Action, Bitbucket Pipeline을 위한 스타터 파이프라인 생성 섹션을 참조하세요.

AWS SAM에서 AWS CodePipeline을 위한 스타터 파이프라인 생성

AWS CodePipeline에 대한 스타터 파이프라인 구성을 생성하려면 다음 태스크를 순서대로 수행합니다.

1. 인프라 리소스
2. 파이프라인 구성 생성
3. 파이프라인 구성을 Git에 커밋
4. Git 리포지토리를 CI/CD 시스템에 연결

> **Note**
>
> 다음 절차에서는 두 개의 AWS SAM CLI 명령, `sam pipeline bootstrap` 및 `sam pipeline init`을 사용합니다. 두 개의 명령이 있는 이유는 관리자(즉, IAM 사용자 및 역할과 같은 인프라AWS 리소스 설정 권한이 필요한 사용자)가 개발자보다 더 많은 권한(즉, 개별 파이프라인을 설정할 수 있는 권한만 필요하지만 필수 인프라 AWS 리소스는 필요하지 않은 사용자)을 처리하는 사용 사례를 처리하기 위한 것입니다.

1단계: 인프라 리소스 생성

AWS SAM을 사용하는 파이프라인에는 필요한 권한이 있는 IAM 사용자 및 역할, Amazon S3 버킷, 선택적으로 Amazon ECR 리포지토리와 같은 특정 AWS 리소스가 필요합니다. 파이프라인의 각 배포 단계를 위한 인프라 리소스 세트가 있어야 합니다.

다음 명령을 실행하여 이 설정을 편집하거나 추가할 수 있습니다.

```
sam pipeline bootstrap
```

> **Note**
>
> 파이프라인의 각 배포 단계에 대해 이전 명령어를 실행합니다.

2단계: 파이프라인 구성 생성

파이프라인 구성을 생성하려면 다음 명령을 실행합니다.

```
sam pipeline init
```

3단계: 파이프라인 구성을 Git 리포지토리에 커밋

이 단계는 CI/CD 시스템이 파이프라인 구성을 인식하도록 할 때 필요하며, 변경 사항이 커밋될 때 실행됩니다.

4단계: Git 리포지토리를 CI/CD 시스템에 연결

AWS CodePipeline의 경우 이제 다음 명령을 실행하여 연결을 생성할 수 있습니다.

```
sam deploy -t codepipeline.yaml --stack-name <pipeline-stack-name> --
capabilities=CAPABILITY_IAM --region <region-X>
```

GitHub 또는 Bitbucket을 사용하는 경우, 이전에 sam deploy 명령을 실행한 후 개발자 도구 콘솔 사용 안내서의 [보류 중인 연결 업데이트] 주제에 있는 연결을 완료하려면 섹션의 단계를 따라 연결을 완료합니다. 또한 sam deploy 명령의 출력에서 `CodeStarConnectionArn` 복사본을 저장합니다. 이 사본은 AWS CodePipeline을 `main`이 아닌 다른 브랜치와 함께 사용하려는 경우 필요합니다.

기타 브랜치 구성

기본적으로 AWS CodePipeline은 AWS SAM과 함께 `main` 브랜치를 사용합니다. `main` 이외의 브랜치를 사용하려면 sam deploy 명령을 다시 실행해야 합니다. 사용 중인 Git 리포지토리에 따라 `CodeStarConnectionArn`를 제공해야 합니다.

```
# 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>"
```

자세히 알아보기

CI/CD 파이프라인 설정에 대한 실제 예제는 전체 AWS SAM 워크샵의 [AWS CodePipeline을 사용한 CI/CD] 섹션을 참조하세요.

AWS SAM을 사용하여 Jenkins, GitLab CI/CD, GitHub Action, Bitbucket Pipeline을 위한 스타터 파이프라인 생성

Jenkins, GitLab CI/CD, GitHub Actions 또는 Bitbucket Pipeline에 대한 스타터 파이프라인 구성을 생성하려면 다음 태스크를 순서대로 수행합니다.

1. 인프라 리소스
2. Git 리포지토리를 CI/CD 시스템에 연결
3. 보안 인증 정보 객체 생성
4. 파이프라인 구성 생성
5. 파이프라인 구성을 Git 리포지토리에 커밋

> **Note**
>
> 다음 절차에서는 두 개의 AWS SAM CLI 명령, `sam pipeline bootstrap` 및 `sam pipeline init`을 사용합니다. 두 개의 명령이 있는 이유는 관리자(즉, IAM 사용자 및 역할과 같은 인프라AWS 리소스 설정 권한이 필요한 사용자)가 개발자보다 더 많은 권한(즉, 개별 파이프라인을 설정할 수 있는 권한만 필요하지만 필수 인프라 AWS 리소스는 필요하지 않은 사용자)을 처리하는 사용 사례를 처리하기 위한 것입니다.

1단계: 인프라 리소스 생성

AWS SAM을 사용하는 파이프라인에는 필요한 권한이 있는 IAM 사용자 및 역할, Amazon S3 버킷, 선택적으로 Amazon ECR 리포지토리와 같은 특정 AWS 리소스가 필요합니다. 파이프라인의 각 배포 단계를 위한 인프라 리소스 세트가 있어야 합니다.

다음 명령을 실행하여 이 설정을 편집하거나 추가할 수 있습니다.

```
sam pipeline bootstrap
```

> **Note**
>
> 파이프라인의 각 배포 단계에 대해 이전 명령어를 실행합니다.

파이프라인의 각 배포 단계에서 파이프라인 사용자의 AWS 보안 인증 정보(키 ID 및 보안 암호 키)을 캡처해야 합니다. 후속 단계에 필요하기 때문입니다.

2단계: Git 리포지토리를 CI/CD 시스템에 연결

CI/CD 시스템이 빌드 및 배포를 위해 애플리케이션 소스 코드에 액세스할 수 있으려면 Git 리포지토리를 CI/CD 시스템에 연결해야 합니다.

> **Note**
>
> 다음 중 하나를 사용하는 경우 연결은 자동으로 수행되므로 이 단계를 건너뛸 수 있습니다.
>
> 1. GitHub 리포지토리를 사용한 GitHub Action
> 2. GitLab 리포지토리를 사용한 GitLab CI/CD
> 3. Bitbucket 리포지토리가 있는 Bitbucket Pipeline

Git 리포지토리를 CI/CD 시스템에 연결하려면 다음 중 하나를 수행합니다.

- Jenkins를 사용하는 경우 Jenkins 설명서의 '브랜치 소스에 추가'를 참조하세요.
- GitLab이 아닌 GitLab CI/CD와 Git 리포지토리를 사용하고 있다면 GitLab 설명서에서 '외부 리포지토리에 연결'을 참조하세요.

3단계: 보안 인증 정보 객체 생성

각 CI/CD 시스템은 CI/CD 시스템이 Git 리포지토리에 액세스하는 데 필요한 보안 인증 정보를 관리하는 고유한 방법을 가집니다.

필요한 보안 인증 정보 객체를 생성하려면 다음 중 하나를 수행합니다.

- Jenkins를 사용하는 경우 키 ID와 보안 암호 키를 모두 저장하는 단일 '보안 인증 정보'을 만듭니다. AWS SAM과 Jenkins Pipeline를 빌드 블로그, Jenkins 구성 섹션의 지침을 따르세요. 다음 단계에서 '보안 인증 id'가 필요합니다.
- GitLab CI/CD를 사용하는 경우 키 ID와 보안 암호 키 각각에 하나씩, 두 개의 '보호되는 변수'를 만듭니다. GitLab 설명서의 지침을 따르세요. 다음 단계를 위해 두 개의 '변수 키'가 필요합니다.
- GitHub Actions를 사용하는 경우 키와 보안 암호 키 각각에 하나씩, 두 개의 '암호화된 보안 암호'를 만듭니다. GitHub 설명서의 지침을 따르세요. 다음 단계를 수행하려면 두 개의 '보안 암호 이름'이 필요합니다.

- Bitbucket Pipeline을 사용하는 경우, 키 ID와 보안 암호 키 각각에 하나씩, 두 개의 '보안 변수'를 만듭니다. 변수 및 보안 암호의 지침을 따르세요. 다음 단계를 수행하려면 두 개의 '보안 암호 이름'이 필요합니다.

4단계: 파이프라인 구성 생성

파이프라인 구성을 생성하려면 다음 명령을 실행합니다. 이전 단계에서 생성한 보안 인증 정보 객체를 입력해야 합니다.

```
sam pipeline init
```

5단계: 파이프라인 구성을 Git 리포지토리에 커밋

이 단계는 CI/CD 시스템이 파이프라인 구성을 인식하도록 할 때 필요하며, 변경 사항이 커밋될 때 실행됩니다.

자세히 알아보기

GitHub Actions을 사용하여 CI/CD 파이프라인을 설정하는 실제 예제는 전체 AWS SAM 워크샵의 GitHub을 사용한 CI/CD 섹션을 참조하세요.

AWS SAM에서 스타터 파이프라인을 사용자 지정하는 방법

CI/CD 관리자는 조직의 개발자가 파이프라인 구성을 생성하는 데 사용할 수 있는 스타터 파이프라인 템플릿과 관련 안내 프롬프트를 사용자 지정할 수 있습니다.

AWS SAM CLI는 스타터 템플릿을 생성할 때 Cookiecutter 템플릿을 사용합니다. 쿠키 커터 템플릿에 대한 자세한 내용은 Cookiecutter 섹션을 참조하세요.

`sam pipeline init` 명령을 사용하여 파이프라인 구성을 생성할 때 AWS SAM CLI가 사용자에게 표시하는 프롬프트를 사용자 지정할 수도 있습니다. 사용자 프롬프트를 사용자 지정하려면 다음을 수행합니다.

1. **`questions.json` 파일 만들기** - `questions.json` 파일은 프로젝트 저장소의 루트에 있어야 합니다. `cookiecutter.json` 파일과 동일한 디렉터리에 있습니다. `questions.json` 파일의 스키마를 보려면 questions.json.schema를 참조합니다. 예제 `questions.json` 파일을 보려면 questions.json을 참조합니다.
2. 질문 키와 쿠키 커터 이름 매핑 - `questions.json`파일의 각 객체에는 쿠키 커터 템플릿의 이름과 일치하는 키가 필요합니다. 이 키 매칭은 AWS SAM CLI가 사용자 프롬프트 응답을 쿠키 커터 템플릿에 매핑하는 방식입니다. 이 키 매칭의 예를 보려면 이 주제의 예제 파일 섹션을 참조하세요.

3. **`metadata.json`** 파일 만들기 - 파이프라인이 `metadata.json` 파일에 포함할 단계 수를 선언합니다. 단계의 수는 `sam pipeline init` 명령에 정보를 표시할 단계 수를 지정하거나, `--bootstrap` 옵션의 경우 인프라 리소스를 생성할 단계의 수를 지정합니다. 2단계로 구성된 파이프라인을 선언하는 예제 `metadata.json` 파일을 보려면 metadata.json을 참조하세요.

예제 프로젝트

다음은 Cookiecutter 템플릿, `questions.json` 파일, `metadata.json` 파일을 포함하는 예제 프로젝트입니다.

- Jenkins 예제: 2단계 Jenkins 파이프라인 템플릿
- CodePipeline 예제: 2단계 CodePipeline 파이프라인 템플릿

예제 파일

다음 파일 세트는 `questions.json` 파일의 질문이 Cookiecutter 템플릿 파일의 항목과 어떻게 연관되는지를 보여줍니다. 참고로 이 예제는 전체 파일이 아닌 파일 스니펫입니다. 전체 파일의 예제를 보려면 이 주제의 앞부분에 나오는 예제 프로젝트 섹션을 참조하세요.

예 **`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"
  }, {
    ...
```

예 **`cookiecutter.json`**:

```
{
  "outputDir": "aws-sam-pipeline",
  "pipeline_user_jenkins_credential_id": "",
  "sam_template": "",
    ...
```

예 **Jenkinsfile**:

```
pipeline {
  agent any
  environment {
    PIPELINE_USER_CREDENTIAL_ID =
 '{{cookiecutter.pipeline_user_jenkins_credential_id}}'
    SAM_TEMPLATE = '{{cookiecutter.sam_template}}'
    ...
```

AWS SAM 애플리케이션 배포 자동화

AWS SAM에서 AWS SAM 애플리케이션 배포를 자동화하는 방법은 사용 중인 CI/CD 시스템에 따라 달라집니다. 이 때문에 이 섹션의 예제에서는 AWS SAM 빌드 컨테이너 이미지에서 서버리스 애플리케이션 빌드를 자동화하도록 다양한 CI/CD 시스템을 구성하는 방법을 보여줍니다. 이러한 빌드 컨테이너 이미지를 사용하면 AWS SAM CLI를 사용하여 서버리스 애플리케이션을 쉽게 빌드하고 패키징할 수 있습니다.

AWS SAM를 사용하여 서버리스 애플리케이션을 배포하는 기존 CI/CD 파이프라인의 절차는 사용 중인 CI/CD 시스템에 따라 약간 다릅니다.

다음 항목에서는 AWS SAM빌드 컨테이너 이미지 내에 서버리스 애플리케이션을 빌드하도록 CI/CD 시스템을 구성하는 예제를 제공합니다.

주제

- AWS SAM에서 AWS CodePipeline을 사용하여 배포
- AWS SAM에서 Bitbucket Pipeline을 사용하여 배포
- AWS SAM에서 Jenkins를 사용하여 배포
- AWS SAM에서 GitLab CI/CD를 사용하여 배포
- AWS SAM에서 GitHub Action을 사용하여 배포

AWS SAM에서 AWS CodePipeline을 사용하여 배포

AWS SAM 애플리케이션 빌드와 배포를 자동화하도록 AWS CodePipeline 파이프라인을 구성하려면 CloudFormation 템플릿과 `buildspec.yml` 파일에 다음을 수행하는 줄이 포함되어야 합니다.

1. 사용 가능한 이미지에서 필요한 런타임이 포함된 빌드 컨테이너 이미지를 참조합니다. 다음 예제에서는 `public.ecr.aws/sam/build-nodejs20.x` 빌드 컨테이너 이미지를 사용합니다.
2. 필요한 AWS SAM 명령줄 인터페이스(CLI) 명령을 실행하도록 파이프라인 단계를 구성합니다. 다음 예제에서는 sam build 및 sam deploy(필수 옵션 포함) 두 AWS SAM CLI 명령을 실행합니다.

이 예제에서는 귀하가 `runtime: nodejs20.x`로 귀하의 AWS SAM 템플릿 파일 내 모든 함수와 레이어를 선언했다고 가정합니다.

CloudFormation 템플릿 코드 조각:

```
  CodeBuildProject:
    Type: AWS::CodeBuild::Project
    Properties:
      Environment:
        ComputeType: BUILD_GENERAL1_SMALL
        Image: public.ecr.aws/sam/build-nodejs20.x
        Type: LINUX_CONTAINER
      ...
```

`buildspec.yml` 코드 조각:

```
version: 0.2
phases:
  build:
    commands:
      - sam build
      - sam deploy --no-confirm-changeset --no-fail-on-empty-changeset
```

다양한 런타임에 사용할 수 있는 Amazon Elastic Container Registry(Amazon ECR)빌드 컨테이너 이미지 목록은 의 이미지 리포지토리 AWS SAM 섹션을 참조하세요.

AWS SAM에서 Bitbucket Pipeline을 사용하여 배포

AWS SAM 애플리케이션 빌드와 배포를 자동화하도록 Bitbucket Pipeline을 구성하려면 `bitbucket-pipelines.yml` 파일에 다음을 수행하는 줄이 포함되어야 합니다.

1. 사용 가능한 이미지에서 필요한 런타임이 포함된 빌드 컨테이너 이미지를 참조합니다. 다음 예제에서는 `public.ecr.aws/sam/build-nodejs20.x` 빌드 컨테이너 이미지를 사용합니다.
2. 필요한 AWS SAM 명령줄 인터페이스(CLI) 명령을 실행하도록 파이프라인 단계를 구성합니다. 다음 예제에서는 sam build 및 sam deploy(필수 옵션 포함) 두 AWS SAM CLI 명령을 실행합니다.

이 예제에서는 귀하가 `runtime: nodejs20.x`로 귀하의 AWS SAM 템플릿 파일 내 모든 함수와 레이어를 선언했다고 가정합니다.

```
image: public.ecr.aws/sam/build-nodejs20.x

pipelines:
  branches:
    main: # branch name
      - step:
          name: Build and Package
          script:
            - sam build
            - sam deploy --no-confirm-changeset --no-fail-on-empty-changeset
```

다양한 런타임에 사용할 수 있는 Amazon Elastic Container Registry(Amazon ECR)빌드 컨테이너 이미지 목록은 의 이미지 리포지토리 AWS SAM 섹션을 참조하세요.

AWS SAM에서 Jenkins를 사용하여 배포

귀하의 AWS SAM 애플리케이션 빌드와 배포를 자동화하도록 Jenkins 파이프라인을 구성하려면, 귀하의 `Jenkinsfile`이 다음을 수행하는 라인들을 포함해야 합니다.

1. 사용 가능한 이미지에서 필요한 런타임이 포함된 빌드 컨테이너 이미지를 참조합니다. 다음 예제에서는 `public.ecr.aws/sam/build-nodejs20.x` 빌드 컨테이너 이미지를 사용합니다.
2. 필요한 AWS SAM 명령줄 인터페이스(CLI) 명령을 실행하도록 파이프라인 단계를 구성합니다. 다음 예제에서는 sam build 및 sam deploy(필수 옵션 포함) 두 AWS SAM CLI 명령을 실행합니다.

이 예제에서는 귀하가 `runtime: nodejs20.x`로 귀하의 AWS SAM 템플릿 파일 내 모든 함수와 레이어를 선언했다고 가정합니다.

```
pipeline {
    agent { docker { image 'public.ecr.aws/sam/build-nodejs20.x' } }
    stages {
```

```
        stage('build') {
            steps {
                sh 'sam build'
                sh 'sam deploy --no-confirm-changeset --no-fail-on-empty-changeset'
            }
        }
    }
}
```

다양한 런타임에 사용할 수 있는 Amazon Elastic Container Registry(Amazon ECR)빌드 컨테이너 이미지 목록은 의 이미지 리포지토리 AWS SAM 섹션을 참조하세요.

AWS SAM에서 GitLab CI/CD를 사용하여 배포

AWS SAM 애플리케이션의 빌드와 배포를 자동화하도록 GitLab 파이프라인을 구성하려면 귀하의 `gitlab-ci.yml`파일에 다음을 수행하는 라인들이 포함되어야 합니다.

1. 사용 가능한 이미지에서 필요한 런타임이 포함된 빌드 컨테이너 이미지를 참조합니다. 다음 예제에서는 `public.ecr.aws/sam/build-nodejs20.x` 빌드 컨테이너 이미지를 사용합니다.
2. 필요한 AWS SAM 명령줄 인터페이스(CLI) 명령을 실행하도록 파이프라인 단계를 구성합니다. 다음 예제에서는 sam build 및 sam deploy(필수 옵션 포함) 두 AWS SAM CLI 명령을 실행합니다.

이 예제에서는 귀하가 `runtime: nodejs20.x`로 귀하의 AWS SAM 템플릿 파일 내 모든 함수와 레이어를 선언했다고 가정합니다.

```
image: public.ecr.aws/sam/build-nodejs20.x
deploy:
  script:
    - sam build
    - sam deploy --no-confirm-changeset --no-fail-on-empty-changeset
```

다양한 런타임에 사용할 수 있는 Amazon Elastic Container Registry(Amazon ECR)빌드 컨테이너 이미지 목록은 의 이미지 리포지토리 AWS SAM 섹션을 참조하세요.

AWS SAM에서 GitHub Action을 사용하여 배포

AWS SAM 애플리케이션 빌드와 배포를 자동화하도록 GitHub 파이프라인을 구성하려면 먼저 호스트에 AWS SAM 명령줄 인터페이스(CLI)를 설치해야 합니다. GitHub 워크플로의 GitHub Actions을 사용하면 이 설정에 도움이 될 수 있습니다.

다음 예제 GitHub 워크플로는 일련의 GitHub 작업을 사용하여 Ubuntu 호스트를 설정한 다음, AWS SAM CLI 명령을 실행하여 AWS SAM 애플리케이션을 빌드하고 배포합니다.

```
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
```

다양한 런타임에 사용할 수 있는 Amazon Elastic Container Registry(Amazon ECR)빌드 컨테이너 이미지 목록은 의 이미지 리포지토리 AWS SAM 섹션을 참조하세요.

AWS SAM 파이프라인과 함께 OIDC 인증을 사용하는 방법

AWS Serverless Application Model (AWS SAM) 은 Bitbucket, GitHub Actions에 대한 OpenID Connect (OIDC) 사용자 인증, GitLab의 지속적 통합 및 전달 (CI/CD) 플랫폼을 지원합니다. 이 지원을 통해 모든 플랫폼에서 인증된 CI/CD 사용자 계정을 사용하여 서버리스 애플리케이션 파이프라인을 관리할 수 있습니다. 그렇지 않으면 AWS SAM 파이프라인에 대한 액세스를 제어하기 위해 여러 AWS Identity and Access Management(IAM)사용자를 생성하고 관리해야 합니다.

AWS SAM 파이프라인으로 OIDC 설정

`sam pipeline bootstrap` 구성 프로세스 중에 다음을 수행하여 AWS SAM 파이프라인으로 OIDC를 설정합니다.

1. ID 제공자를 선택하라는 메시지가 표시되면 OIDC를 선택합니다.
2. 다음으로 지원되는 OIDC 공급자를 선택합니다.
3. **`https://`**로 시작하는 OIDC 공급자 URL을 입력합니다.

> **Note**
>
> AWS SAM는 `AWS::IAM::OIDCProvider` 리소스 유형을 생성할 때 이 URL을 참조합니다.

4. 그런 다음 프롬프트에 따라 선택한 플랫폼에 액세스하는 데 필요한 CI/CD 플랫폼 정보를 입력합니다. 이러한 세부 정보는 플랫폼마다 다르며 다음을 포함할 수 있습니다.
   - OIDC 클라이언트 ID.
   - 코드 리포지토리 이름 또는 범용 고유 식별자(UUID).
   - 리포지토리와 연관된 그룹 또는 조직 이름.
   - 코드 리포지토리가 속한 GitHub 조직.
   - GitHub 리포지토리 이름.
   - 배포가 이루어지는 브랜치.
5. AWS SAM에는 입력한 OIDC 구성의 요약이 표시됩니다. 편집하려면 설정 번호를 입력하거나 Enter를 눌러 계속 진행합니다.
6. 입력한 OIDC 연결을 지원하는 데 필요한 리소스 생성을 확인하라는 메시지가 표시되면 Y를 눌러 계속 진행합니다.

AWS SAM는 파이프라인 실행 역할을 맡는 제공된 구성으로 `AWS::IAM::OIDCProvider` AWS CloudFormation 리소스를 생성합니다. 이 CloudFormation 리소스 유형에 대해 자세히 알아보려면 AWS CloudFormation 사용 설명서의 AWS::IAM::OIDCProvider를 참조하세요.

> **Note**
>
> ID 제공업체(idP) 리소스가 이미 AWS 계정에 있는 경우, AWS SAM는 새 리소스를 만드는 대신 해당 리소스를 참조합니다.

예시

다음은 AWS SAM 파이프라인을 사용하여 OIDC를 설정하는 예제입니다.

```
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]:
```

자세히 알아보기

RDS와 AWS SAM를 함께 사용하는 방법에 대한 자세한 내용은 sam pipeline bootstrap 섹션을 참조하세요.

sam sync를 사용하는 AWS 클라우드 동기화 소개

AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) `sam sync` 명령은 로컬 애플리케이션 변경 내용을 AWS 클라우드에 빠르게 동기화하는 옵션을 제공합니다. 다음 작업을 수행하기 위해 애플리케이션을 개발할 때 `sam sync`를 사용합니다.

1. 로컬 변경 사항을 자동으로 감지하여 AWS 클라우드에 동기화합니다.
2. AWS 클라우드에 동기화할 로컬 변경 사항을 사용자 지정합니다.
3. 테스트 및 검증을 위해 클라우드에서 애플리케이션을 준비합니다.

`sam sync`를 사용하면 테스트 및 검증을 위해 로컬 변경 사항을 클라우드에 동기화하는 데 걸리는 시간을 단축하는 빠른 개발 워크플로를 만들 수 있습니다.

> **Note**
>
> 이 `sam sync` 명령은 개발 환경에 권장됩니다. 프로덕션 환경에서는 `sam deploy`를 사용하여 지속적 통합 및 제공(CI/CD) 파이프라인을 사용하거나 구성하는 것이 좋습니다. 자세한 내용은 AWS SAM을 사용하여 애플리케이션 및 리소스 배포 섹션을 참조하세요.

`sam sync` 명령은 AWS SAM Accelerate의 일부입니다. AWS SAM Accelerate는 AWS 클라우드에서 서버리스 애플리케이션을 개발하고 테스트하는 속도를 높이는 데 사용할 수 있는 도구를 제공합니다.

주제

- 로컬 변경 사항을 자동으로 감지하고 AWS 클라우드에 동기화
- AWS 클라우드에 동기화할 로컬 변경 사항 사용자 지정
- 테스트 및 검증을 위해 클라우드에서 애플리케이션 준비하기
- sam sync 명령의 옵션
- 문제 해결

- 예시
- 자세히 알아보기

로컬 변경 사항을 자동으로 감지하고 AWS 클라우드에 동기화

`--watch` 옵션으로 `sam sync`를 실행하여 애플리케이션을 AWS 클라우드에 동기화하기 시작합니다. 는 다음 동작을 수행합니다.

1. 애플리케이션 빌드 - 이 프로세스는 `sam build` 명령을 사용하는 것과 비슷합니다.
2. 애플리케이션 배포 - AWS SAM CLI는 기본 설정을 사용하여 애플리케이션을 AWS CloudFormation에 배포합니다. 기본값은 다음과 같습니다.
   a. `.aws` 사용자 폴더에 있는 AWS 보안 인증 정보 및 일반 구성 설정
   b. 애플리케이션의 `samconfig.toml` 파일에 있는 애플리케이션 배포 설정.

   기본값을 찾을 수 없는 경우 AWS SAM CLI는 이를 알리고 동기화 프로세스를 종료합니다.
3. 로컬 변경 사항 확인 - AWS SAM CLI는 계속 실행되면 애플리케이션의 로컬 변경 사항을 감시합니다. `--watch` 옵션이 제공하는 것은 다음과 같습니다.

   이 옵션은 기본적으로 활성화되어 있습니다. 기본값은 애플리케이션 `samconfig.toml` 파일을 참조하세요. 다음은 예제 파일입니다.

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

4. 로컬 변경 사항을 AWS 클라우드에 동기화 - 로컬에서 변경을 수행하면 AWS SAM CLI는 가능한 가장 빠른 방법을 통해 해당 변경 사항을 감지하여 AWS 클라우드에 동기화합니다. 변경 유형에 따라 다음과 같은 상황이 발생할 수 있습니다.
   a. 업데이트된 리소스가 AWS 서비스 API를 지원하는 경우 AWS SAM CLI에서는 이를 사용하여 변경 내용을 배포합니다. 그러면 빠르게 동기화되어 AWS 클라우드에서 리소스가 업데이트됩니다.
   b. 업데이트된 리소스가 AWS 서비스 API를 지원하지 않는 경우 AWS SAM CLI에서 CloudFormation 배포를 수행합니다. 이렇게 하면 AWS 클라우드의 전체 애플리케이션이 업데이트됩니다. 빠르지는 않지만 배포를 수동으로 시작하지 않아도 됩니다.

`sam sync` 명령은 AWS 클라우드에서 애플리케이션을 자동으로 업데이트하므로 개발 환경에서만 사용하는 것이 좋습니다. `sam sync`를 실행하면 다음을 확인하라는 메시지가 표시됩니다.

```
**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
```

AWS 클라우드에 동기화할 로컬 변경 사항 사용자 지정

AWS 클라우드에 동기화할 로컬 변경 사항을 사용자 지정하는 옵션을 제공합니다. 이를 통해 테스트 및 검증을 위해 클라우드에서 로컬 변경 사항을 확인하는 데 걸리는 시간을 단축할 수 있습니다.

예를 들어 AWS Lambda 함수 코드와 같은 코드 변경 사항만 동기화하는 `--code` 옵션을 제공합니다. 개발 중에 특히 Lambda 코드에 초점을 맞추면 테스트 및 검증을 위해 변경 사항을 클라우드로 빠르게 가져올 수 있습니다. 다음은 예제입니다.

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

특정 Lambda 함수 또는 계층의 코드 변경 사항만 동기화하려면 `--resource-id` 옵션을 사용합니다. 다음은 예제입니다.

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

테스트 및 검증을 위해 클라우드에서 애플리케이션 준비하기

이 `sam sync` 명령은 AWS 클라우드에서 애플리케이션을 업데이트할 수 있는 가장 빠른 방법을 자동으로 찾습니다. 이를 통해 개발 및 클라우드 테스트 워크플로의 속도를 높일 수 있습니다. AWS 서비스 API를 활용하면 지원되는 리소스를 신속하게 개발, 동기화 및 테스트할 수 있습니다. 실제 예제는 전체 AWS SAM 워크샵의 [모듈 6 - AWS SAM Accelerate] 섹션을 참조하세요.

sam sync 명령의 옵션

다음은 `sam sync` 명령을 수정하는 데 사용할 수 있는 몇 가지 주요 옵션입니다. 콘텐츠 옵션 목록은 [sam sync] 섹션을 참조하세요.

일회성 CloudFormation 배포 수행

`--no-watch` 옵션을 사용하여 자동 동기화를 끕니다. 다음은 예제입니다.

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

AWS SAM CLI는 일회성 CloudFormation 배포를 수행합니다. 이 명령은 `sam build` 및 `sam deploy` 명령으로 수행된 작업을 그룹화합니다.

초기 CloudFormation 배포는 건너뜁니다.

CloudFormation이 실행될 때마다 `sam sync` 배포가 필요한지 여부를 사용자 지정할 수 있습니다.

- `sam sync`를 실행할 때마다 CloudFormation 배포가 필요한 `--no-skip-deploy-sync`를 제공합니다. 이렇게 하면 로컬 인프라가 CloudFormation에 동기화되어 드리프트를 방지합니다. 이 옵션을 사용하면 개발 및 테스트 워크플로에 더 많은 시간을 할애할 수 있습니다.
- CloudFormation 배포를 선택 사항으로 지정하는 `--skip-deploy-sync`를 제공합니다. AWS SAM CLI는 로컬AWS SAM 템플릿을 배포된 CloudFormation 템플릿과 비교하여 변경 사항이 감지되지 않으면 초기 CloudFormation 배포를 건너뜁니다. CloudFormation 배포를 건너뛰면 로컬 변경 사항을 AWS 클라우드에 동기화할 때 시간을 절약할 수 있습니다.

  변경 사항이 감지되지 않는 경우 AWS SAM CLI는 다음 시나리오에서 계속 CloudFormation 배포를 수행합니다.

  - 마지막 CloudFormation 배포 이후 7일 이상이 지난 경우
  - Lambda 함수 코드 변경이 대량으로 감지되는 경우 애플리케이션을 업데이트하는 가장 빠른 방법은 CloudFormation 배포입니다.

다음은 예제입니다.

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

중첩된 스택에서 리소스 동기화

중첩된 스택에서 리소스를 동기화하려면

1. `--stack-name`를 사용하여 루트 스택을 제공합니다.
2. *`nestedStackId/resourceId`* 형식을 사용하여 중첩된 스택의 리소스를 식별합니다.
3. `--resource-id`를 사용하여 중첩된 스택의 리소스를 제공합니다.

다음은 예제입니다.

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

중첩된 애플리케이션 생성에 대한 자세한 내용은 AWS SAM에서 중첩 애플리케이션을 사용하여 코드 및 리소스 재사용 섹션을 참조하세요.

업데이트할 특정 CloudFormation 스택을 지정합니다.

업데이트할 특정 CloudFormation 스택을 지정하려면 `--stack-name` 옵션을 제공합니다. 다음은 예제입니다.

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

소스 폴더에서 프로젝트를 빌드하여 빌드 시간 단축

지원되는 런타임과 빌드 메서드의 경우 `--build-in-source` 옵션을 사용하여 소스 폴더에서 직접 프로젝트를 빌드할 수 있습니다. 기본적으로 AWS SAM CLI는 임시 디렉터리에 빌드하며, 여기에는 소스 코드와 프로젝트 파일을 복사하는 작업이 수반됩니다. `--build-in-source`를 사용하면 AWS SAM CLI가 소스 폴더에서 직접 빌드하므로 파일을 임시 디렉터리에 복사할 필요가 없어 빌드 프로세스 속도가 빨라집니다.

지원되는 런타임 및 빌드 메서드 목록은 `--build-in-source`를 참조하세요.

동기화를 시작하지 않는 파일 및 폴더 지정

`--watch-exclude` 옵션을 사용하여 업데이트 시 동기화를 시작하지 않을 파일이나 폴더를 지정할 수 있습니다. 이 옵션에 대한 자세한 내용은 `--watch-exclude` 섹션을 참조하세요.

다음은 `HelloWorldFunction` 함수와 관련된 `package-lock.json` 파일을 제외한 예시입니다.

```
$ sam sync --watch --watch-exclude HelloWorldFunction=package-lock.json
```

이 명령을 실행하면 AWS SAM CLI에서 동기화 프로세스를 시작합니다. 다음 내용이 포함됩니다:

- `sam build`를 실행하여 함수를 빌드하고 애플리케이션 배포를 준비합니다.
- `sam deploy`를 실행하여 애플리케이션을 배포합니다.
- 애플리케이션 변경을 확인하세요.

package-lock.json 파일을 수정해도 AWS SAM CLI는 동기화를 시작되지 않습니다. 다른 파일이 업데이트되면 AWS SAM CLI는 동기화를 시작하며, 이 동기화에는 package-lock.json 파일이 포함됩니다.

다음은 하위 스택의 Lambda 함수를 지정하는 예제입니다.

```
$ sam sync --watch --watch-exclude ChildStackA/MyFunction=database.sqlite3
```

문제 해결

AWS SAM CLI 문제를 해결하려면 [AWS SAMCLI 문제 해결](#) 섹션을 참조하세요.

예시

sam sync를 사용하여 Hello World 애플리케이션 업데이트

이 예제에서는 먼저 샘플 Hello World 애플리케이션을 초기화합니다. 애플리케이션 공유에 대한 자세한 내용은 [자습서: AWS SAM에서 Hello World 애플리케이션 배포](#) 섹션을 참조하세요.

sam sync를 실행하면 빌드 및 배포 프로세스가 시작됩니다.

```
$ sam sync

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

Confirm that you are synchronizing a development stack.

Enter Y to proceed with the command, or enter N to cancel:
 [Y/n]:
Queued infra sync. Waiting for in progress code syncs to complete...
Starting infra sync.
Manifest file is changed (new hash: 3298f13049d19cffaa37ca931dd4d421) or dependency
 folder (.aws-sam/deps/0663e6fe-a888-4efb-b908-e2344261e9c7) is missing for
 (HelloWorldFunction), downloading dependencies and copying/building source
Building codeuri: /Users/.../Demo/sync/sam-app/hello_world runtime: python3.9 metadata:
 {} architecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:CleanUp
Running PythonPipBuilder:ResolveDependencies
```

```
Running PythonPipBuilder:CopySource

Build Succeeded

Successfully packaged artifacts and wrote output template to file /var/
folders/45/5ct135bx3fn2551_ptl5g6_80000gr/T/tmpx_5t4u3f.
Execute the following command to deploy the packaged template
sam deploy --template-file /var/folders/45/5ct135bx3fn2551_ptl5g6_80000gr/T/tmpx_5t4u3f
 --stack-name <YOUR STACK NAME>


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

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



2023-03-17 11:17:19 - Waiting for stack create/update to complete

CloudFormation events from stack operations (refresh every 0.5 seconds)
-------------------------------------------------------------------------------------------------
ResourceStatus                    ResourceType
 LogicalResourceId                 ResourceStatusReason
-------------------------------------------------------------------------------------------------
CREATE_IN_PROGRESS                AWS::CloudFormation::Stack          sam-app
                     Transformation succeeded
CREATE_IN_PROGRESS                AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt   -
                                                                      ack
CREATE_IN_PROGRESS                AWS::IAM::Role
 HelloWorldFunctionRole            -
CREATE_IN_PROGRESS                AWS::IAM::Role
 HelloWorldFunctionRole            Resource creation Initiated
CREATE_IN_PROGRESS                AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt   Resource creation Initiated
```

```
                                                                        ack
CREATE_COMPLETE                    AWS::IAM::Role
 HelloWorldFunctionRole             -
CREATE_COMPLETE                    AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt  -
                                                                        ack
CREATE_IN_PROGRESS                 AWS::Lambda::Function
 HelloWorldFunction                 -
CREATE_IN_PROGRESS                 AWS::Lambda::Function
 HelloWorldFunction                 Resource creation Initiated
CREATE_COMPLETE                    AWS::Lambda::Function
 HelloWorldFunction                 -
CREATE_IN_PROGRESS                 AWS::ApiGateway::RestApi
 ServerlessRestApi                  -
CREATE_IN_PROGRESS                 AWS::ApiGateway::RestApi
 ServerlessRestApi                  Resource creation Initiated
CREATE_COMPLETE                    AWS::ApiGateway::RestApi
 ServerlessRestApi                  -
CREATE_IN_PROGRESS                 AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment47fc2d  -
                                                                        5f9d
CREATE_IN_PROGRESS                 AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi  -
                                                                        ssionProd
CREATE_IN_PROGRESS                 AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi  Resource creation Initiated
                                                                        ssionProd
CREATE_IN_PROGRESS                 AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment47fc2d  Resource creation Initiated
                                                                        5f9d
CREATE_COMPLETE                    AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment47fc2d  -
                                                                        5f9d
CREATE_IN_PROGRESS                 AWS::ApiGateway::Stage
 ServerlessRestApiProdStage         -
CREATE_IN_PROGRESS                 AWS::ApiGateway::Stage
 ServerlessRestApiProdStage         Resource creation Initiated
CREATE_COMPLETE                    AWS::ApiGateway::Stage
 ServerlessRestApiProdStage         -
CREATE_COMPLETE                    AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi  -
                                                                        ssionProd
CREATE_COMPLETE                    AWS::CloudFormation::Stack           sam-app
                      -
```

```
-------------------------------------------------------------------------------------------------

CloudFormation outputs from deployed stack
-------------------------------------------------------------------------------------------------
Outputs
-------------------------------------------------------------------------------------------------
Key                 HelloWorldFunctionIamRole
Description         Implicit IAM Role created for Hello World function
Value               arn:aws:iam::012345678910:role/sam-app-HelloWorldFunctionRole-
BUFVMO2PJIYF

Key                 HelloWorldApi
Description         API Gateway endpoint URL for Prod stage for Hello World function
Value               https://pcrx5gdaof.execute-api.us-west-2.amazonaws.com/Prod/hello/

Key                 HelloWorldFunction
Description         Hello World Lambda Function ARN
Value               arn:aws:lambda:us-west-2:012345678910:function:sam-app-
HelloWorldFunction-2PlN6TPTQoco
-------------------------------------------------------------------------------------------------
Stack creation succeeded. Sync infra completed.

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

배포가 완료되면 `HelloWorldFunction` 코드를 수정합니다. AWS SAM CLI는 이러한 변경 사항을 감지하고 애플리케이션을 AWS 클라우드에 동기화합니다. AWS Lambda가 AWS 서비스 API를 지원하므로 빠른 동기화가 수행됩니다.

```
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.
```

다음으로 애플리케이션의 AWS SAM 템플릿에서 API 엔드포인트를 수정합니다. `/hello`를 `/helloworld`로 변경합니다.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
...
```

```
Resources:
  HelloWorldFunction:
    ...
    Properties:
      ...
      Events:
        HelloWorld:
          Type: Api
          Properties:
            Path: /helloworld
            Method: get
```

Amazon API Gateway 리소스는 AWS 서비스 API를 지원하지 않으므로 AWS SAM CLI는 자동으로 CloudFormation 배포를 수행합니다. 다음은 출력의 예제입니다.

```
Queued infra sync. Waiting for in progress code syncs to complete...
Starting infra sync.
Manifest is not changed for (HelloWorldFunction), running incremental build
Building codeuri: /Users/.../Demo/sync/sam-app/hello_world runtime: python3.9 metadata:
 {} architecture: x86_64 functions: HelloWorldFunction
Running PythonPipBuilder:CopySource

Build Succeeded

Successfully packaged artifacts and wrote output template to file /var/
folders/45/5ct135bx3fn2551_ptl5g6_80000gr/T/tmpuabo0jb9.
Execute the following command to deploy the packaged template
sam deploy --template-file /var/folders/45/5ct135bx3fn2551_ptl5g6_80000gr/T/tmpuabo0jb9
 --stack-name <YOUR STACK NAME>


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

Initiating deployment
```

```
=====================



2023-03-17 14:41:18 - Waiting for stack create/update to complete

CloudFormation events from stack operations (refresh every 0.5 seconds)
---------------------------------------------------------------------------------------------------------
ResourceStatus                    ResourceType
 LogicalResourceId                 ResourceStatusReason
---------------------------------------------------------------------------------------------------------
UPDATE_IN_PROGRESS                AWS::CloudFormation::Stack          sam-app
                     Transformation succeeded
UPDATE_IN_PROGRESS                AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt   -
                                                                      ack
UPDATE_COMPLETE                   AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt   -
                                                                      ack
UPDATE_IN_PROGRESS                AWS::ApiGateway::RestApi
 ServerlessRestApi                 -
UPDATE_COMPLETE                   AWS::ApiGateway::RestApi
 ServerlessRestApi                 -
CREATE_IN_PROGRESS                AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment8cf30e   -
                                                                      d3cd
UPDATE_IN_PROGRESS                AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi   Requested update requires the
                                                                      ssionProd
                     creation of a new physical

                     resource; hence creating one.
UPDATE_IN_PROGRESS                AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi   Resource creation Initiated
                                                                      ssionProd
CREATE_IN_PROGRESS                AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment8cf30e   Resource creation Initiated
                                                                      d3cd
CREATE_COMPLETE                   AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment8cf30e   -
                                                                      d3cd
UPDATE_IN_PROGRESS                AWS::ApiGateway::Stage
 ServerlessRestApiProdStage        -
UPDATE_COMPLETE                   AWS::ApiGateway::Stage
 ServerlessRestApiProdStage        -
```

```
UPDATE_COMPLETE                    AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi   -
                                                                    ssionProd
UPDATE_COMPLETE_CLEANUP_IN_PROGRE   AWS::CloudFormation::Stack          sam-app
                     -
SS
DELETE_IN_PROGRESS                 AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi   -
                                                                    ssionProd
DELETE_IN_PROGRESS                 AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment47fc2d   -
                                                                    5f9d
DELETE_COMPLETE                    AWS::ApiGateway::Deployment
 ServerlessRestApiDeployment47fc2d   -
                                                                    5f9d
UPDATE_COMPLETE                    AWS::CloudFormation::Stack
 AwsSamAutoDependencyLayerNestedSt   -
                                                                    ack
DELETE_COMPLETE                    AWS::Lambda::Permission
 HelloWorldFunctionHelloWorldPermi   -
                                                                    ssionProd
UPDATE_COMPLETE                    AWS::CloudFormation::Stack          sam-app
                     -
-----------------------------------------------------------------------------------------------------

CloudFormation outputs from deployed stack
-----------------------------------------------------------------------------------------------------
Outputs
-----------------------------------------------------------------------------------------------------
Key                 HelloWorldFunctionIamRole
Description         Implicit IAM Role created for Hello World function
Value               arn:aws:iam::012345678910:role/sam-app-HelloWorldFunctionRole-
BUFVMO2PJIYF

Key                 HelloWorldApi
Description         API Gateway endpoint URL for Prod stage for Hello World function
Value               https://pcrx5gdaof.execute-api.us-west-2.amazonaws.com/Prod/hello/

Key                 HelloWorldFunction
Description         Hello World Lambda Function ARN
Value               arn:aws:lambda:us-west-2:012345678910:function:sam-app-
HelloWorldFunction-2PlN6TPTQoco
-----------------------------------------------------------------------------------------------------
```

```
Stack update succeeded. Sync infra completed.

Infra sync completed.
```

자세히 알아보기

모든 `sam sync` 옵션에 대한 설명은 sam sync 섹션을 참조하세요.

AWS SAM을 사용하여 서버리스 애플리케이션 모니터링

서버리스 애플리케이션을 배포한 후 이를 모니터링하여 운영에 필요한 통찰력을 얻고 문제 해결에 도움이 될 수 있는 이상 징후를 감지할 수 있습니다. 이 섹션에서는 서버리스 애플리케이션 모니터링에 대한 세부 정보를 제공합니다. 여기에는 Amazon CloudWatch가 이상 징후를 감지할 때 사용자에게 알리도록 구성하는 방법에 대한 정보가 포함되어 있습니다. 또한 오류 강조 표시와 로그 보기, 필터링, 가져오기 및 조정을 위한 팁을 포함하여 로그 작업에 대한 정보도 제공합니다.

주제

- CloudWatch Application Insights를 사용하여 AWS SAM 서버리스 애플리케이션 모니터링
- AWS SAM의 로그 작업

CloudWatch Application Insights를 사용하여 AWS SAM 서버리스 애플리케이션 모니터링

Amazon CloudWatch Application Insights는 귀하의 애플리케이션의 AWS리소스를 모니터링하여 잠재적 문제를 식별하는 데 도움이 됩니다. AWS리소스 데이터를 분석하여 문제의 징후를 찾아내고 자동화된 대시보드를 구축하여 문제를 시각화할 수 있습니다. 귀하의 AWS Serverless Application Model(AWS SAM) 애플리케이션과 함께 사용하도록 CloudWatch Application Insights를 구성할 수 있습니다. CloudWatch Application Insights에 대한 자세한 내용은 Amazon CloudWatch 사용자 가이드의 Amazon CloudWatch Application Insights를 잠조하세요.

주제

- AWS SAM를 사용하여 CloudWatch Application Insights 구성하기
- 다음 단계

AWS SAM를 사용하여 CloudWatch Application Insights 구성하기

AWS SAM명령줄 인터페이스(AWS SAM CLI)를 통해, 또는 귀하의 AWS SAM템플릿을 통해 AWS SAM 애플리케이션에 대해 CloudWatch Application Insights를 구성합니다.

AWS SAMCLI을 통해 구성하십시오.

sam init을 사용하여 애플리케이션을 초기화할 때는 대화형 흐름을 통해 또는 --application-insights 옵션을 사용하여 CloudWatch Application Insights를 활성화하십시오.

AWS SAM CLI 대화형 흐름을 통해 CloudWatch Application Insights를 활성화하려면 프롬프트가 표시될 때 **y**을 입력합니다.

```
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]:
```

--application-insights 옵션을 사용하여 CloudWatch Application Insights를 활성화하려면 다음과 같이 하십시오.

```
sam init --application-insights
```

sam init 명령의 사용에 대한 자세한 내용은 sam init를 참조하세요.

AWS SAM 템플릿을 통해 구성하십시오.

귀하의 AWS SAM 템플릿에 `AWS::ResourceGroups::Group` 및 `AWS::ApplicationInsights::Application`리소스를 정의하여 CloudWatch Application Insights를 활성화합니다.

```
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` – 많은 리소스에 대한 작업을 한 번에 관리하고 자동화하기 위해 귀하의 AWS 리소스를 구성하는 그룹을 만듭니다. 여기에서 CloudWatch Application Insights와 함께 사용할 리소스 그룹을 생성합니다. 이러한 리소스 유형에 대한 자세한 내용은 AWS CloudFormation 사용자 가이드의 [`AWS::ResourceGroups::Group`]를 잠조하세요.
- `AWS::ApplicationInsights::Application` – 리소스 그룹에 대하여 CloudWatch Application Insights를 구성합니다. 이러한 리소스 유형에 대한 자세한 내용은 AWS CloudFormation 사용자 가이드의 [`AWS::ApplicationInsights::Application`]를 잠조하세요.

두 리소스 모두 애플리케이션 배포 시 자동으로 CloudFormation에 전달됩니다. AWS SAM 템플릿의 CloudFormation 명령문을 사용하여 CloudWatch Application Insights를 추가로 구성할 수 있습니다. 자세한 내용은 Amazon CloudWatch 사용자 가이드의 [CloudFormation템플릿 사용]을 잠조하세요.

sam init --application-insights 명령을 사용하면 귀하의 AWS SAM템플릿에서 이 두 리소스가 모두 자동으로 생성됩니다. 다음은 생성된 템플릿의 예입니다.

```
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
```

다음 단계

CloudWatch Application Insights를 구성한 후에는 귀하의 애플리케이션을 구축하는데 sam build을 사용하고 애플리케이션을 배포하는 데 sam deploy을 사용합니다. 모든 CloudWatch Application Insights 지원 리소스는 모니터링을 위해 구성됩니다.

- 지원되는 리소스 목록은 Amazon CloudWatch 사용자 가이드에서 지원되는 로그 및 지표를 참조하세요.
- CloudWatch Application Insights에 액세스하는 방법을 알아보려면 Amazon CloudWatch 사용자 가이드의 CloudWatch Application Insights에 대한 액세스를 참조하세요.

AWS SAM의 로그 작업

문제 해결을 단순화하기 위해 AWS SAM CLI에는 `sam logs`이라는 명령이 있습니다. 이 명령을 사용하면 명령줄에서 Lambda 함수로 생성된 로그를 가져올 수 있습니다.

> **Note**
>
> 이 `sam logs` 명령은 귀하가 배포에 이용한 AWS SAM 함수뿐 아니라 모든 AWS Lambda 함수에 사용할 수 있습니다.

CloudFormation 스택으로 로그 가져오기

함수가 CloudFormation 스택의 일부인 경우 함수의 논리적 ID를 사용하여 로그를 가져올 수 있습니다.

```
sam logs -n HelloWorldFunction --stack-name mystack
```

Lambda 함수 이름으로 로그 가져오기

또는 함수 이름을 사용하여 로그를 가져올 수 있습니다.

```
sam logs -n mystack-HelloWorldFunction-1FJ8PD
```

테일링 로그

새 로그가 도착할 때까지 기다렸다가 도착하는 대로 확인할 수 있는 `--tail` 옵션을 추가합니다. 이는 배포 중이나 프로덕션 문제를 해결할 때 유용합니다.

```
sam logs -n HelloWorldFunction --stack-name mystack --tail
```

특정 시간 범위의 로그 보기

`-s` 및 `-e` 옵션을 사용하여 특정 시간 범위의 로그를 볼 수 있습니다.

```
sam logs -n HelloWorldFunction --stack-name mystack -s '10min ago' -e '2min ago'
```

로그 필터링

`--filter` 옵션을 사용하여 로그 이벤트에서 용어, 구문 또는 값과 일치하는 로그를 빠르게 찾을 수 있습니다.

```
sam logs -n HelloWorldFunction --stack-name mystack --filter "error"
```

출력 결과에서 AWS SAM CLI는 "오류"라는 단어가 나오는 모든 부분에 밑줄을 표시하므로 로그 출력 결과에서 필터 키워드를 쉽게 찾을 수 있습니다.

오류 강조 표시

Lambda 함수가 충돌하거나 제한 시간이 초과되면 AWS SAM CLI는 시간 종료 메시지를 빨간색으로 강조 표시합니다. 이를 통해 방대한 로그 출력 결과 스트림 내에서 제한 시간이 초과된 특정 실행을 쉽게 찾을 수 있습니다.

JSON 스타일리시 인쇄

귀하의 로그 메시지가 JSON 문자열을 인쇄하는 경우 AWS SAM CLI은 JSON을 자동으로 스타일리시하게 인쇄하여 귀하가 JSON을 시각적으로 분석하고 이해하는 데 도움을 줍니다.

AWS SAM 참조

이 섹션에는 AWS SAM 참조 자료가 포함되어 있습니다. 여기에는 AWS SAM CLI 명령에 대한 참조 정보와 같은 AWS SAM CLI 참조 자료와 구성, 버전 제어, 문제 해결 정보와 같은 추가 AWS SAM CLI 정보가 포함됩니다. 또한 이 섹션에는 커넥터, 이미지 리포지토리 및 배포에 대한 참조 정보와 같은 AWS SAM 사양 및 AWS SAM 템플릿에 대한 참조 정보가 포함되어 있습니다.

AWS SAM 사양 및 AWS SAM 템플릿

이 AWS SAM 사양은 Apache 2.0 라이선스에 따른 오픈 소스 사양입니다. AWS SAM 사양의 현재 버전은 AWS SAM 템플릿에서 사용할 수 있습니다. AWS SAM 사양은 서버리스 애플리케이션의 함수, 이벤트, API, 구성 및 권한을 정의하는 데 사용하는 간소화된 약식 구문과 함께 제공됩니다.

sam init 명령을 실행할 때 생성되는 폴더 및 파일인 AWS SAM 애플리케이션 프로젝트 디렉터리를 통해 AWS SAM 사양과 상호 작용합니다. 이 디렉터리에는 AWS 리소스를 정의하는 중요한 파일인 AWS SAM 템플릿이 포함되어 있습니다. AWS SAM 템플릿은 AWS CloudFormation 템플릿의 확장입니다. CloudFormation 템플릿에 대한 전체 참조는 AWS CloudFormation 사용 설명서의 템플릿 참조를 참조하세요.

AWS SAMCLI 명령 참조

AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI)는 AWS SAM 템플릿과 지원되는 타사 통합과 함께 사용하여 서버리스 애플리케이션을 빌드하고 실행할 수 있는 명령줄 도구입니다.

AWS SAM CLI 명령을 사용하여 서버리스 애플리케이션을 개발, 테스트하고 AWS 클라우드에 배포할 수 있습니다. 다음은 몇 가지 AWS SAM CLI 명령의 예입니다.

- `sam init` - 처음 AWS SAM CLI를 사용하는 경우, 파라미터 없이 `sam init` 명령을 실행하여 Hello World 애플리케이션을 만들 수 있습니다. 이 명령은 선택한 언어로 사전 구성된 AWS SAM 템플릿과 예제 애플리케이션 코드를 생성합니다.
- `sam local invoke` 및 `sam local start-api` - 이 명령을 사용하여 애플리케이션 코드를 로컬에서 테스트한 다음 애플리케이션 코드를 AWS 클라우드에 배포합니다.
- `sam logs` - 이 명령을 사용하여 Lambda 함수가 생성하는 로그를 가져올 수 있습니다. 이렇게 하면 애플리케이션을 AWS 클라우드에 배포한 후 애플리케이션을 테스트하고 디버깅하는 데 도움이 될 수 있습니다.

- `sam package` - 이 명령을 사용하여 애플리케이션 코드와 종속성을 배포 패키지로 번들링할 수 있습니다. AWS 클라우드에 애플리케이션을 업로드하려면 배포 패키지가 필요합니다.
- `sam deploy` - 이 명령을 사용하여 서버리스 애플리케이션을 AWS 클라우드에 배포합니다. AWS 리소스를 생성하고 AWS SAM 템플릿에 정의된 권한 및 기타 구성을 설정합니다.

AWS SAM CLI 설치 지침은 [AWS SAM CLI 설치](#) 섹션을 참조하세요.

AWS SAM 정책 템플릿

AWS SAM을 사용하면 정책 템플릿 목록에서 선택하여 AWS Lambda 함수의 권한 범위를 애플리케이션에서 사용하는 리소스로 지정할 수 있습니다. 사용 가능한 정책 템플릿 목록은 [정책 템플릿 테이블](#) 섹션을 참조하세요. 정책 템플릿 및 AWS SAM에 대한 일반 정보는 [AWS SAM 정책 템플릿](#) 섹션을 참조하세요.

주제

- AWS SAM 템플릿
- AWS SAM CLI 명령 참조
- AWS SAMCLI구성 파일
- AWS SAM 커넥터 참조
- AWS SAM 정책 템플릿
- 의 이미지 리포지토리 AWS SAM
- AWS SAM CLI 내 텔레메트리
- AWS SAM 템플릿에서 리소스 액세스 설정 및 관리

AWS SAM CLI 명령 참조

이 섹션에는 AWS SAM CLI 명령에 대한 참조 정보가 포함되어 있습니다. 여기에는 사용법, 각 명령어에 사용할 수 있는 다양한 옵션의 포괄적인 목록 및 추가 정보가 포함되어 있습니다. 해당하는 경우 추가 정보에는 인수, 환경 변수, 이벤트 등과 같은 세부 정보가 포함됩니다. 자세한 내용은 각 명령을 참조하세요. 설치에 대한 지침은 단원을 AWS SAMCLI참조하십시오AWS SAM CLI 설치.

주제

- sam build
- sam delete
- sam deploy
- sam init
- sam list
- sam local callback
- sam local execution
- sam local generate-event
- sam local invoke
- sam local start-api
- sam local start-lambda
- sam logs
- sam package
- sam pipeline bootstrap
- sam pipeline init
- sam publish
- sam remote callback
- sam remote execution
- sam remote invoke
- sam remote test-event
- sam sync
- sam traces
- sam validate

sam build

이 페이지는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) `sam build` 명령에 대한 참조 정보를 제공합니다.

- AWS SAM CLI에 대한 소개는 란 무엇입니까 AWS SAMCLI? 섹션을 참조하세요.

- AWS SAM CLI `sam build` 명령 사용에 대한 가이드는 AWS SAM의 빌드 소개 섹션을 참조하세요.

이 `sam build` 명령은 로컬 테스트 또는 AWS 클라우드에 대한 배포와 같은 개발자 워크플로의 후속 단계를 위해 응용 프로그램을 준비합니다.

사용법

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

인수

리소스 ID

선택 사항입니다. AWS SAM에게 AWS SAM 템플릿에 선언된 단일 리소스를 빌드하도록 지시합니다. 지정된 리소스의 빌드 아티팩트는 워크플로의 후속 명령(예: `sam package` 및 `sam deploy`)에 사용할 수 있는 유일한 아티팩트입니다.

옵션

`--base-dir, -s DIRECTORY`

이 디렉터리에 대해 함수 또는 계층의 소스 코드에 대한 상대 경로를 확인합니다. 소스 코드 폴더의 상대 경로를 확인하는 방법을 변경하려면 이 옵션을 사용하십시오. 기본적으로 상대 경로는 AWS SAM 템플릿 위치를 기준으로 확인됩니다.

이 옵션은 구축 중인 루트 애플리케이션이나 스택의 리소스 외에도 중첩된 애플리케이션 또는 스택에도 적용됩니다.

이 옵션은 다음 리소스 유형 및 속성에 적용됩니다.

- 리소스 유형: `AWS::Serverless::Function` 속성: `CodeUri`
- 리소스 유형: `AWS::Serverless::Function` 리소스 속성: `Metadata` 항목: `DockerContext`
- 리소스 유형: `AWS::Serverless::LayerVersion` 속성: `ContentUri`
- 리소스 유형: `AWS::Lambda::Function` 속성: `Code`
- 리소스 유형: `AWS::Lambda::LayerVersion` 속성: `Content`

`--beta-features | --no-beta-features`

베타 기능을 허용 또는 거부합니다.

`--build-dir, -b` *DIRECTORY*

구축된 아티팩트가 저장되는 디렉터리의 경로입니다. 이 옵션을 사용하면 이 디렉터리와 모든 내용이 제거됩니다.

`--build-image` *TEXT*

구축을 위해 가져오려는 컨테이너 이미지의 URI입니다. 기본 사항으로 AWS SAM은 Amazon ECR Public로부터 컨테이너 이미지를 가져옵니다. 다른 위치에서 이미지를 가져오려면 이 옵션을 사용합니다.

이 옵션은 여러 번 지정할 수 있습니다. 이 옵션의 각 인스턴스는 문자열 또는 키-값 쌍을 사용할 수 있습니다. 문자열을 지정하는 경우 이는 애플리케이션의 모든 리소스에 사용할 컨테이너 이미지의 URI입니다. 예를 들어 `sam build --use-container --build-image amazon/aws-sam-cli-build-image-python3.8`입니다. 키-값 쌍을 지정하는 경우 키는 리소스 이름이고 값은 해당 리소스에 사용할 컨테이너 이미지의 URI입니다. 예: `sam build --use-container --build-image Function1=amazon/aws-sam-cli-build-image-python3.8`. 키-값 쌍을 사용하면 리소스마다 다른 컨테이너 이미지를 지정할 수 있습니다.

이 옵션은 `--use-container` 옵션이 지정된 경우에만 적용되며, 그렇지 않으면 오류가 발생합니다.

`--build-in-source | --no-build-in-source`

소스 폴더에서 프로젝트를 직접 빌드하기 위한 `--build-in-source`를 제공합니다.

`--build-in-source` 옵션은 다음과 같은 런타임과 빌드 메서드를 지원합니다.

- 런타임 - [`sam init --runtime`] 옵션에서 지원하는 모든 Node.js 런타임.
- 빌드 메서드 - `Makefile`, `esbuild`.

`--build-in-source` 옵션은 다음 옵션과 호환되지 않습니다.

- `--hook-name`
- `--use-container`

기본값: `--no-build-in-source`

`--cached | --no-cached`

캐시된 빌드를 활성화 또는 비활성화합니다. 이 옵션을 사용하면 이전 빌드에서 변경되지 않은 빌드 아티팩트를 재사용할 수 있습니다. AWS SAM은 프로젝트 디렉터리에서 파일을 변경했는지 여

부를 평가합니다. 기본적으로 빌드는 캐시되지 않습니다. 이 `--no-cached` 옵션을 간접 호출하면 samconfig.toml의 `cached = true` 설정을 재정의합니다.

> **Note**
>
> 특정 버전을 제공하지 않은 경우 AWS SAM은 프로젝트가 의존하는 타사 모듈을 변경했는지를 평가하지 않습니다. 예를 들어 Python 함수에 항목 `requirements.txt`가 있는 `requests=1.x` 파일이 포함되어 있고 최신 요청 모듈 버전이 `1.1`에서 `1.2`로 변경되면 AWS SAM는 캐시되지 않은 빌드를 실행할 때까지 최신 버전을 가져오지 않습니다.

`--cache-dir`

`--cached`이 지정된 경우 캐시 아티팩트가 저장되는 디렉터리입니다. 기본 캐시 디렉터리는 `.aws-sam/cache`입니다.

`--config-env` *`TEXT`*

사용할 구성 파일의 기본 매개변수 값을 지정하는 환경 이름입니다. 기본값은 "기본값"입니다. 구성 파일에 대한 자세한 내용은 AWS SAMCLI구성 파일 섹션을 참조하세요.

`--config-file` *`PATH`*

사용할 기본 매개변수 값이 들어 있는 구성 파일의 경로 및 파일 이름입니다. 기본값은 프로젝트 디렉터리의 루트에 있는 "`samconfig.toml`"입니다. 구성 파일에 대한 자세한 내용은 AWS SAMCLI 구성 파일 섹션을 참조하세요.

`--container-env-var, -e` *`TEXT`*

빌드 컨테이너에 전달할 환경 변수입니다. 이 옵션은 여러 번 지정할 수 있습니다. 이 옵션의 각 인스턴스는 키-값 쌍을 사용합니다. 여기서 키는 리소스 및 환경 변수이고 값은 환경 변수의 값입니다. 예를 들어 `--container-env-var Function1.GITHUB_TOKEN=TOKEN1 --container-env-var Function2.GITHUB_TOKEN=TOKEN2`입니다.

이 옵션은 `--use-container` 옵션이 지정된 경우에만 적용되며, 그렇지 않으면 오류가 발생합니다.

`--container-env-var-file, -ef` *`PATH`*

컨테이너 환경 변수 값이 포함된 JSON 파일의 경로 및 파일 이름입니다. 컨테이너 환경 변수 구성에 대한 자세한 내용은 컨테이너 환경 변수 파일 섹션을 참조하세요.

이 옵션은 `--use-container` 옵션이 지정된 경우에만 적용되며, 그렇지 않으면 오류가 발생합니다.

`--debug`

디버그 로깅을 켜서 AWS SAM CLI가 생성한 디버그 메시지를 인쇄하고 타임스탬프를 표시합니다.

`--docker-network` *`TEXT`*

Lambda Docker 컨테이너가 연결되어야 하는 기존 Docker 네트워크의 이름 또는 ID와 기본 브리지 네트워크를 지정합니다. 지정되지 않으면 Lambda 컨테이너는 기본 브리지 Docker 네트워크에만 연결됩니다.

`--exclude, -x`

`sam build`에서 제외할 리소스의 이름입니다. 예를 들어 템플릿에 `Function1`, `Function2`, `Function3`이 포함되어 있고 `sam build --exclude Function2`를 실행하면, `Function1` 및 `Function3`만 구축됩니다.

`--help`

이 메시지를 표시한 후 종료합니다.

`--hook-name` *`TEXT`*

AWS SAM CLI 기능을 확장하는 데 사용되는 후크의 이름입니다.

허용되는 값: `terraform`.

`--manifest , -m` *`PATH`*

기본값 대신 사용할 사용자 지정 종속성 매니페스트 파일(예: package.json) 의 경로입니다.

`--mount-symlinks`

AWS SAM CLI이(가) 빌드하거나 간접 호출할 파일에 존재하는 심볼릭 링크를 항상 마운트하도록 보장합니다. 이는 최상위 디렉터리(즉, 함수의 루트에 직접 있는 심볼릭 링크)에만 적용됩니다. 기본적으로, NodeJS에서 `node_modules`을(를) 위해 `build-in-source`을(를) 사용하는 데 필요한 경우를 제외하고 심볼릭 링크는 마운트되지 않습니다.

`--no-use-container`

IDE 툴킷이 기본 동작을 설정하는 데 사용할 수 있는 옵션입니다. `sam build --no-use-container`을(를) 사용하여 Docker 컨테이너 대신 로컬 시스템에서 빌드를 실행할 수도 있습니다.

`--parallel`

병렬 빌드를 활성화합니다. 이 옵션을 사용하여 AWS SAM 템플릿의 함수와 레이어를 병렬로 구축할 수 있습니다. 기본적으로 함수와 계층은 순서대로 작성됩니다.

`--parameter-overrides`

(선택 사항) 키-값 쌍으로 인코딩된 CloudFormation 매개 변수 변경을 포함하는 문자열. AWS Command Line Interface(AWS CLI)과 같은 형식을 사용합니다. 예를 들어 '`ParameterKey=KeyPairName, ParameterValue=MyKey ParameterKey=InstanceType, ParameterValue=t1.micro`'입니다. 이 옵션은 `--hook-name`과 호환되지 않습니다.

`--profile` *TEXT*

AWS 보안 인증 정보를 가져오는 보안 인증 파일의 특정 프로필입니다.

`--region` *TEXT*

배포할 AWS 리전입니다. 예를 들어 us-east-1입니다.

`--save-params`

명령줄에서 AWS SAM 구성 파일에 제공한 매개변수를 저장합니다.

`--skip-prepare-infra`

인프라를 변경하지 않은 경우 준비 단계를 건너뜁니다. `--hook-name` 옵션과 함께 사용합니다.

`--skip-pull-image`

명령이 Lambda 런타임에 대한 최신 Docker 이미지를 가져오는 단계를 건너뛸지를 지정합니다.

`--template-file, --template, -t` *PATH*

AWS SAM 템플릿 파일 `[default: template.[yaml|yml]]`의 경로와 파일 이름입니다. 이 옵션은 `--hook-name`과 호환되지 않습니다.

`--terraform-project-root-path`

Terraform 구성 파일 또는 함수 소스 코드가 들어 있는 최상위 디렉터리의 상대적 또는 절대적 경로입니다. 이러한 파일이 Terraform 루트 모듈이 들어 있는 디렉터리 외부에 있는 경우, 이 옵션을 사용하여 절대적 또는 상대적 경로를 지정하십시오. 이 옵션을 사용하려면 `--hook-name`을 `terraform`로 설정해야 합니다.

`--use-container, -u`

함수가 네이티브 컴파일된 종속 항목을 포함하는 패키지를 사용하는 경우 이 옵션을 사용하여 Lambda와 유사한 Docker 컨테이너 내에 함수를 구축합니다.

예시

`sam build` 하위 명령 사용에 대한 자세한 예제와 심층적인 단계별 설명은 AWS SAM의 빌드 소개 섹션을 참조하세요.

sam delete

이 페이지는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) `sam delete` 명령에 대한 참조 정보를 제공합니다.

AWS SAM CLI에 대한 소개는 란 무엇입니까 AWS SAMCLI? 섹션을 참조하세요.

이 `sam delete` 명령은 AWS SAM 스택, Amazon S3와 Amazon ECR에 패키징되어 배포된 아티팩트, CloudFormation 템플릿 파일을 삭제하여 AWS SAM 애플리케이션을 삭제합니다.

이 명령은 또한 Amazon ECR 컴패니언 스택이 배포되어 있는지 확인하고, 배포되어 있다면 사용자에게 해당 스택 및 Amazon ECR 리포지토리를 삭제할지 묻는 메시지를 표시합니다. `--no-prompts`이 지정되면 컴패니언 스택과 Amazon ECR 리포지토리가 기본적으로 삭제됩니다.

사용법

```
$ sam delete <options>
```

옵션

`--config-env TEXT`

사용할 구성 파일의 기본 매개변수 값을 지정하는 환경 이름입니다. 기본값은 `default`입니다. 구성 파일에 대한 자세한 내용은 AWS SAMCLI구성 파일 섹션을 참조하세요.

`--config-file PATH`

사용할 기본 매개변수 값이 들어 있는 구성 파일의 경로 및 파일 이름입니다. 기본값은 프로젝트 디렉터리의 루트 내 `samconfig.toml`입니다. 구성 파일에 대한 자세한 내용은 AWS SAMCLI구성 파일 섹션을 참조하세요.

`--debug`

디버그 로깅을 켜서 AWS SAM CLI가 생성한 디버그 메시지를 인쇄한 다음, 타임스탬프를 표시합니다.

`--help`

이 메시지를 표시한 후 종료합니다.

`--no-prompts`

비대화형 모드에서 AWS SAM이 작동하도록 이 옵션을 지정하십시오. 스택 이름은 `--stack-name` 옵션과 함께 제공되거나 구성 `toml` 파일에 제공되어야 합니다.

`--profile` *TEXT*

AWS 보안 인증 정보를 가져오는 보안 인증 파일의 특정 프로필입니다.

`--region` *TEXT*

배포할 AWS 지역입니다. 예를 들어 us-east-1입니다.

`--s3-bucket`

삭제할 Amazon S3 버킷의 경로입니다.

`--s3-prefix`

삭제할 Amazon S3 버킷의 접두사입니다.

`--save-params`

명령줄에서 AWS SAM 구성 파일에 제공한 매개변수를 저장합니다.

`--stack-name` *TEXT*

삭제하고자 하는 CloudFormation 스택의 이름입니다.

예시

다음 명령은 `MY-STACK` 스택을 삭제합니다.

```
$ sam delete --stack-name MY-STACK
```

다음 명령은 `MY-STACK` 스택 및 `sam-s3-demo-bucket` S3 버킷을 삭제합니다.

```
$ sam delete \
    --stack-name MyStack \
    --s3-bucket MySAMBucket
```

sam deploy

이 페이지는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) `sam deploy` 명령에 대한 참조 정보를 제공합니다.

- AWS SAM CLI에 대한 소개는 란 무엇입니까 AWS SAMCLI? 섹션을 참조하세요.
- AWS SAM CLI `sam deploy` 명령 사용에 대한 가이드는 AWS SAM 배포 소개 섹션을 참조하세요.

`sam deploy` 명령은 AWS 클라우드을 사용하여 애플리케이션을 AWS CloudFormation에 배포합니다.

사용법

```
$ <environment variables> sam deploy <options>
```

환경 변수

`SAM_CLI_POLL_DELAY`

셸의 `SAM_CLI_POLL_DELAY` 환경 변수를 초 값으로 설정하여 AWS SAM CLI가 CloudFormation 스택 상태를 확인하는 빈도를 구성합니다. 이 스택 상태는 CloudFormation에서 제한을 확인할 때 유용합니다. 이 env 변수는 `sam deploy` 실행 중에 수행되는 `describe_stack` API 호출을 폴링하는 데 사용됩니다.

다음은 이 변수의 예입니다.

```
$ SAM_CLI_POLL_DELAY=5 sam deploy
```

옵션

`--capabilities` *LIST*

CloudFormation이 특정 스택을 생성할 수 있도록 지정해야 하는 기능 목록입니다. 일부 스택 세트 템플릿에는 예를 들어 새로운 AWS 계정(IAM 사용자)를 생성하는 방식으로 AWS Identity and Access Management에 대한 권한에 영향을 미칠 수 있는 리소스가 포함될 수 있습니다. 그러한 스택에서는 이 옵션을 지정하여 스택의 기능을 명확히 확인해야 합니다. 유일하게 유효한 값은 `CAPABILITY_IAM`과 `CAPABILITY_NAMED_IAM`입니다. IAM 리소스가 있는 경우 둘 중 어느 기능이든 지정할 수 있습니다. 사용자 정의 이름을 갖는 IAM 리소스가 있

는 경우 CAPABILITY_NAMED_IAM을 지정해야 합니다. 이 옵션을 지정하지 않으면 작업에서 InsufficientCapabilities 오류를 반환합니다.

중첩된 애플리케이션이 포함된 애플리케이션을 배포할 때는 CAPABILITY_AUTO_EXPAND를 사용하여 애플리케이션에 중첩 애플리케이션이 포함되어 있음을 알려야 합니다. 자세한 내용은 [중첩 애플리케이션 배포]을 참조하십시오

`--config-env` *TEXT*

사용할 구성 파일의 기본 매개변수 값을 지정하는 환경 이름입니다. 기본값은 default입니다. 구성 파일에 대한 자세한 내용은 [AWS SAMCLI구성 파일] 섹션을 참조하세요.

`--config-file` *PATH*

사용할 기본 매개변수 값이 들어 있는 구성 파일의 경로 및 파일 이름입니다. 기본값은 프로젝트 디렉터리의 루트 내 samconfig.toml입니다. 구성 파일에 대한 자세한 내용은 [AWS SAMCLI구성 파일] 섹션을 참조하세요.

`--confirm-changeset | --no-confirm-changeset`

AWS SAM CLI가 계산된 변경 세트를 배포할지를 확인해달라는 프롬프트입니다.

`--debug`

디버그 로깅을 켜면 AWS SAM CLI가 생성한 디버그 메시지를 인쇄하고 타임스탬프를 표시합니다.

`--disable-rollback | --no-disable-rollback`

배포 중에 오류가 발생하는 경우 CloudFormation 스택을 롤백할지를 지정합니다. 기본적으로 배포 중에 오류가 발생하면 CloudFormation 스택은 마지막 안정 상태로 롤백됩니다. --disable-rollback을 지정했는데 배포 중에 오류가 발생하면 오류가 발생하기 전에 생성되거나 업데이트된 리소스는 롤백되지 않습니다.

`--fail-on-empty-changeset | --no-fail-on-empty-changeset`

스택에 변경 사항이 없는 경우 0이 아닌 종료 코드를 반환할지 여부를 지정하십시오. 기본 동작은 0이 아닌 종료 코드를 반환하는 것입니다.

`--force-upload`

Amazon S3 버킷의 기존 아티팩트와 일치하더라도 아티팩트를 업로드하려면 이 옵션을 지정하십시오. 일치하는 아티팩트는 덮어씁니다.

`--guided, -g`

이 옵션을 지정하면 AWS SAM CLI에서 메시지를 사용하여 배포 과정을 안내합니다.

`--help`

이 메시지를 표시한 후 종료합니다.

`--image-repositories` *TEXT*

Amazon ECR 리포지토리 URI에 대한 함수의 매핑. 논리적 ID의 함수 참조. 다음은 그 예제입니다.

```
$ sam deploy --image-repositories Function1=123456789012.dkr.ecr.us-east-1.amazonaws.com/my-repo
```

이 옵션은 하나의 명령에서 여러 번 지정할 수 있습니다.

`--image-repository` *TEXT*

이 명령이 함수 이미지를 업로드하는 Amazon ECR 리포지토리의 이름입니다. 이 옵션은 `Image` 패키지 유형으로 선언된 함수에 필요합니다.

`--kms-key-id` *TEXT*

Amazon S3 버킷에 저장된 아티팩트를 암호화하는 데 사용되는 AWS Key Management Service(AWS KMS) 키의 ID입니다. 이 옵션을 지정하지 않을 경우 AWS SAM은 Amazon S3 관리형 암호화 키를 사용합니다.

`--metadata`

템플릿에서 참조되는 모든 아티팩트에 첨부할 메타데이터 맵입니다.

`--no-execute-changeset`

변경 세트를 적용할지를 나타냅니다. 변경 세트를 적용하기 전에 스택 변경 내용을 보려면 이 옵션을 지정하십시오. 이 명령은 CloudFormation 변경 세트를 만든 다음 변경 세트를 적용하지 않고 종료합니다. 변경 세트를 적용하려면 이 옵션 없이 동일한 명령을 실행합니다.

`--no-progressbar`

Amazon S3에 아티팩트를 업로드할 때 진행률 표시줄을 표시하지 마십시오.

`--notification-arns` *LIST*

CloudFormation이 스택과 열결하는 Amazon Simple Notification Service(Amazon SNS) 주제 ARN의 목록.

`--on-failure [ROLLBACK | DELETE | DO_NOTHING]`

스택을 생성하지 못할 때 취할 조치를 지정합니다.

다음과 같은 옵션을 사용할 수 있습니다.

- `ROLLBACK` – 스택을 이전에 알려진 정상 상태로 롤백합니다.
- `DELETE` – 스택이 존재하는 경우 이전에 알려진 정상 상태로 롤백합니다. 그렇지 않으면 스택을 삭제합니다.
- `DO_NOTHING` – 스택을 롤백하거나 삭제하지 않습니다. 효과는 `--disable-rollback`과 동일합니다.

기본값은 `ROLLBACK`입니다.

> **Note**
>
> `--disable-rollback` 옵션 또는 `--on-failure` 옵션을 지정할 수 있지만 둘 다 모두 지정할 수는 없습니다.

`--parameter-overrides` *LIST*

키-값 페어로 인코딩된 CloudFormation 매개변수를 포함하는 스트링이 우선합니다. 각 재정의는 `ParameterKey=name,ParameterValue=value` 형식을 사용합니다. 여러 재정의를 공백으로 구분합니다. 다음은 두 가지 예제입니다.

```
$ sam deploy --parameter-overrides ParameterKey=value1,ParameterValue=value2
```

```
$ sam deploy --parameter-overrides ParameterKey=value1,ParameterValue=value2
 ParameterKey=hello,ParameterValue=world ParameterKey=apple,ParameterValue=banana
```

`--profile` *TEXT*

AWS 보안 인증 정보를 가져오는 보안 인증 파일의 특정 프로필입니다.

`--region` *TEXT*

배포할 AWS 리전입니다. 예를 들어 us-east-1입니다.

`--resolve-image-repos`

가이드 없는 배포를 위한 패키징 및 배포에 사용할 Amazon ECR 리포지토리를 자동으로 생성합니다. 이 옵션은 `PackageType: Image`이 지정된 함수 및 계층에만 적용됩니다. `--guided` 옵션을 지정하는 경우, AWS SAM CLI은 `--resolve-image-repos`를 무시합니다.

> **Note**
>
> AWS SAM이 이 옵션을 사용하여 함수 또는 레이어에 대한 Amazon ECR 리포지토리를 자동으로 생성하고 나중에 AWS SAM 템플릿에서 해당 함수 또는 레이어를 삭제하면 해당 Amazon ECR 리포지토리가 자동으로 삭제됩니다.

`--resolve-s3`

가이드 없는 배포를 위한 패키징 및 배포에 사용할 Amazon S3 버킷을 자동으로 생성합니다. `--guided` 옵션을 지정하는 경우 AWS SAM CLI는 `--resolve-s3`을 무시합니다. `--s3-bucket` 및 `--resolve-s3` 옵션을 모두 지정하면 오류가 발생합니다.

`--role-arn` *TEXT*

변경 세트를 적용할 때 CloudFormation이 맡은 IAM 역할의 Amazon 리소스 이름(ARN).

`--s3-bucket` *TEXT*

이 명령으로 CloudFormation 템플릿을 업로드하는 Amazon S3 버킷의 이름입니다. 템플릿이 51,200바이트를 초과하는 경우 `--s3-bucket` 옵션 또는 `--resolve-s3` 옵션 중 하나가 필요합니다. `--s3-bucket` 및 `--resolve-s3` 옵션을 모두 지정하면 오류가 발생합니다.

`--s3-prefix` *TEXT*

접두사는 Amazon S3 버킷에 업로드되는 아티팩트의 이름에 추가됩니다. 접두사 이름은 Amazon S3 버킷의 경로 이름(폴더 이름)입니다.

`--save-params`

명령줄에서 AWS SAM 구성 파일에 제공한 매개변수를 저장합니다.

`--signing-profiles` *LIST*

배포 패키지에 서명하는 데 사용할 서명 프로필 목록입니다. 이 옵션은 키-값 쌍의 목록을 사용합니다. 여기서 키는 서명할 함수 또는 계층의 이름이고 값은 서명 프로필이며 선택적 프로필 소유자는 `:`로 제한됩니다. 예를 들어 `FunctionNameToSign=SigningProfileName1 LayerNameToSign=SigningProfileName2:SigningProfileOwner`입니다.

`--stack-name` *TEXT*

(필수) 배포할 CloudFormation 스택의 이름입니다. 기존 스택을 지정하시면 명령이 해당 스택을 업데이트합니다. 새 스택을 지정하시면 명령이 해당 스택을 생성합니다.

`--tags LIST`

생성되거나 업데이트된 스택과 연결할 태그 목록입니다. 또한 CloudFormation은 이러한 태그를 지원하는 스택의 리소스로 전파합니다.

`--template-file, --template, -t PATH`

AWS SAM 템플릿이 위치한 경로 및 파일 이름입니다.

> **Note**
> 이 옵션을 지정하는 경우 AWS SAM은 템플릿과 템플릿이 가리키는 로컬 리소스만에만 배포합니다.

`--use-json`

CloudFormation 템플릿의 JSON을 출력합니다. 기본 출력 결과는 YAML입니다.

예시

`sam deploy` 하위 명령 사용에 대한 자세한 예제와 심층적인 단계별 설명은 AWS SAM 배포 소개 섹션을 참조하세요.

sam init

이 페이지에서는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAMCLI) `sam init` 명령에 대한 참조 정보를 제공합니다.

- 에 대한 소개는 단원을 AWS SAMCLI참조하십시오. 란 무엇입니까 AWS SAMCLI?
- `sam init` 명령 사용에 대한 설명서는 섹션을 참조하세요 AWS SAMCLI에서 애플리케이션 생성 AWS SAM.

이 `sam init` 명령은 새 서버리스 애플리케이션을 초기화하는 옵션을 제공합니다.

사용법

```
$ sam init <options>
```

옵션

`--app-template` *TEXT*

사용하고자 하는 관리 애플리케이션 템플릿의 식별자입니다. 확실하지 않은 경우 대화형 워크플로를 위해 옵션 없이 `sam init`을 직접 호출합니다.

`--no-interactive`이 지정되고 `--location`가 제공되지 않은 경우, 이 매개변수는 필수입니다.

이 매개변수는 AWS SAM CLI 버전 0.30.0 이상에서만 사용할 수 있습니다. 이 매개변수를 이전 버전과 함께 지정하면 오류가 발생합니다.

`--application-insights | --no-application-insights`

Amazon CloudWatch Application Insights 모니터링을 활성화하십시오. 자세한 내용은 [CloudWatch Application Insights를 사용하여 AWS SAM 서버리스 애플리케이션 모니터링](#)을 참조하십시오.

기본 옵션은 `--no-application-insights`입니다.

`--architecture, -a` *[x86_64 | arm64]*

애플리케이션의 Lambda 함수에 대한 명령어 세트 아키텍처입니다. `x86_64` 또는 `arm64` 중 하나를 지정하십시오.

`--base-image` *[amazon/dotnet8-base | amazon/dotnet6-base | amazon/java25-base | amazon/java21-base | amazon/java17-base | amazon/java11-base | amazon/nodejs24.x-base | amazon/nodejs22.x-base | amazon/nodejs20.x-base | amazon/nodejs18.x-base | amazon/nodejs16.x-base | amazon/python3.14-base | amazon/python3.13-base | amazon/python3.12-base | amazon/python3.11-base | amazon/python3.10-base | amazon/python3.9-base | amazon/python3.8-base | amazon/ruby3.4-base | amazon/ruby3.3-base | amazon/ruby3.2-base]*

애플리케이션의 기본 이미지입니다. 이 옵션은 패키지 유형이 `Image`인 경우에만 적용됩니다.

`--no-interactive`이 지정되고, `--package-type`가 `Image`으로 지정되고, `--location`가 지정되지 않은 경우, 이 매개변수는 필수입니다.

`--config-env` *TEXT*

사용할 구성 파일의 기본 매개변수 값을 지정하는 환경 이름입니다. 기본값은 “기본값”입니다. 구성 파일에 대한 자세한 내용은 [AWS SAMCLI구성 파일](#) 섹션을 참조하세요.

`--config-file` *PATH*

사용할 기본 매개변수 값이 들어 있는 구성 파일의 경로 및 파일 이름입니다. 기본값은 프로젝트 디렉터리의 루트에 있는 “samconfig.toml”입니다. 구성 파일에 대한 자세한 내용은 AWS SAMCLI구성 파일 섹션을 참조하세요.

`--debug`

디버그 로깅을 켜서 AWS SAMCLI가 생성한 디버그 메시지를 인쇄하고 타임스탬프를 표시합니다.

`--dependency-manager, -d` *[gradle | mod | maven | bundler | npm | cli-package | pip]*

Lambda 런타임의 종속물 관리자.

`--extra-content`

템플릿의 `cookiecutter.json` 구성의 모든 사용자 지정 매개변수를 재정의합니다(예: `{"customParam1": "customValue1", "customParam2":"customValue2"}`).

`--help, -h`

이 메시지를 표시한 후 종료합니다.

`--location, -l` *TEXT*

템플릿 또는 애플리케이션 위치(Git, Mercurial, HTTP/HTTPS, .zip 파일, 경로)입니다.

`--no-interactive`이 지정되고 `--runtime`, `--name` 및 `--app-template`가 제공되지 않은 경우, 이 매개변수는 필수입니다.

Git 리포지토리의 경우 리포지토리의 루트 위치를 사용해야 합니다.

로컬 경로의 경우 템플릿은. zip 파일 또는 Cookiecutter 형식이어야 합니다.

`--name, -n` *TEXT*

디렉터리로 생성할 프로젝트의 이름입니다.

`--no-interactive`이 지정되고 `--location`가 제공되지 않은 경우, 이 매개변수는 필수입니다.

`--no-input`

Cookiecutter 프롬프트를 비활성화하고 템플릿 구성에 정의된 vcfdefault 값을 승인합니다.

`--no-interactive`

초기 매개변수에 대한 대화식 프롬프트를 비활성화하고 필수 값이 누락되면 실패합니다.

`--output-dir, -o` *PATH*

초기화된 응용 프로그램이 출력되는 위치입니다.

`--package-type` *[Zip | Image]*

예제 애플리케이션의 패키지 유형입니다. `Zip`이 .zip 파일 아카이브를 만들고 `Image`가 컨테이너 이미지를 만듭니다.

`--runtime, -r` *[dotnet8 | dotnet6 | java25 | java21 | java17 | java11 | nodejs24.x | nodejs22.x | nodejs20.x | nodejs18.x | nodejs16.x | python3.14 | python3.13 | python3.12 | python3.11 | python3.10 | python3.9 | python3.8 | ruby3.4 | ruby3.3 | ruby3.2]*

애플리케이션의 Lambda 런타임입니다. 이 옵션은 패키지 유형이 `Zip`인 경우에만 적용됩니다.

`--no-interactive`이 지정되고, `--package-type`가 `Zip`으로 지정되고, `--location`가 지정되지 않은 경우, 이 매개변수는 필수입니다.

`--save-params`

명령줄에 제공한 파라미터를 AWS SAM 구성 파일에 저장합니다.

`--tracing | --no-tracing`

Lambda 함수에 대한 AWS X-Ray 추적을 활성화합니다.

예제

`sam init` 하위 명령 사용에 대한 자세한 예제와 심층적인 단계별 설명은 에서 애플리케이션 생성 AWS SAM 섹션을 참조하세요.

sam list

이 페이지는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) `sam list` 명령에 대한 참조 정보를 제공합니다.

AWS SAM CLI에 대한 소개는 란 무엇입니까 AWS SAMCLI? 섹션을 참조하세요.

이 `sam list` 명령은 귀하의 서버리스 애플리케이션의 리소스 및 그러한 서버리스 애플리케이션의 상태에 대한 중요한 정보를 출력합니다. 배포 전후에 sam list을 사용하여 로컬 및 클라우드 개발을 지원할 수 있습니다.

사용법

```
$ sam list <options> <subcommand>
```

옵션

`--help, -h`

이 메시지를 표시한 후 종료합니다.

하위 명령

`endpoints`

귀하의 CloudFormation 스택의 클라우드 및 로컬 엔드포인트 목록을 표시합니다. 자세한 내용은 sam list endpoints을 참조하십시오

`resources`

배포 시 AWS CloudFormation에 생성된 AWS Serverless Application Model(AWS SAM) 템플릿의 리소스를 표시합니다. 자세한 내용은 sam list resources을 참조하십시오

`stack-outputs`

AWS SAM 또는 CloudFormation 템플릿의 귀하의 CloudFormation 스택의 출력을 표시합니다. 자세한 내용은 sam list stack-outputs을 참조하십시오.

sam list endpoints

이 페이지는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) `sam list endpoints` 하위 명령에 대한 참조 정보를 제공합니다.

AWS SAM CLI에 대한 소개는 란 무엇입니까 AWS SAMCLI? 섹션을 참조하세요.

`sam list endpoints` 하위 명령은 CloudFormation 스택의 클라우드 및 로컬 엔드포인트 목록을 표시합니다. sam local 및 sam sync명령을 통해 이러한 리소스와 상호 작용할 수 있습니다.

AWS Lambda과 Amazon API Gateway 리소스 유형은 이 명령에서 지원됩니다.

> **Note**
>
> Amazon API Gateway 리소스에 맞게 구성된 경우 사용자 지정 도메인이 지원됩니다. 이 명령은 기본 엔드포인트 대신 사용자 지정 도메인을 출력합니다.

사용법

```
$ sam list endpoints <options>
```

옵션

`--config-env TEXT`

사용할 구성 파일의 기본 매개변수 값을 지정하는 환경 이름입니다.

기본 값: `default`

구성 파일에 대한 자세한 내용은 AWS SAMCLI구성 파일 섹션을 참조하세요.

`--config-file TEXT`

사용할 기본 매개변수 값이 들어 있는 구성 파일의 경로 및 파일 이름입니다.

기본값: 현재 작업 디렉터리 내 `samconfig.toml`입니다.

구성 파일에 대한 자세한 내용은 AWS SAMCLI구성 파일 섹션을 참조하세요.

`--debug`

디버그 로깅을 켜면 AWS SAM CLI에서 생성한 디버그 메시지를 타임스탬프와 함께 인쇄할 수 있습니다.

`--help, -h`

이 메시지를 표시한 후 종료합니다.

`--output [json|table]`

결과 출력 형식을 지정합니다.

기본 값: `table`

`--profile TEXT`

보안 인증 정보 파일에서 특정 프로필을 선택하여 AWS 보안 인증을 가져옵니다.

`--region` *TEXT*

서비스 AWS 지역을 설정합니다. 예를 들어 `us-east-1`입니다.

`--save-params`

명령줄에서 AWS SAM 구성 파일에 제공한 매개변수를 저장합니다.

`--stack-name` *TEXT*

배포된 CloudFormation 스택의 이름입니다. 스택 이름은 애플리케이션의 `samconfig.toml` 파일 또는 지정된 구성 파일에서 찾을 수 있습니다.

이 옵션을 지정하지 않으면 템플릿에 정의된 로컬 리소스가 표시됩니다.

`--template-file, --template, -t` *PATH*

AWS SAM 템플릿 파일.

기본 값: `template.[yaml|yml|json]`

예시

이름이 CloudFormation로 지정된 `test-stack`스택의 배포된 리소스 엔드포인트 출력을 json 형식으로 표시합니다.

```
$ 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

이 페이지는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) `sam list resources` 하위 명령에 대한 참조 정보를 제공합니다.

AWS SAM CLI에 대한 소개는 [란 무엇입니까 AWS SAMCLI?](#) 섹션을 참조하세요.

`sam list resources` 하위 명령은 배포 시 AWS Serverless Application Model 변환으로 AWS SAM 내에 생성된 AWS CloudFormation(AWS SAM) 템플릿 내 리소스를 표시합니다.

배포 전에 sam list resources 템플릿과 함께 AWS SAM을 사용하면 생성될 리소스를 확인할 수 있습니다. 배포된 리소스가 포함된 통합 목록을 보려면 CloudFormation 스택 이름을 제공하십시오.

> **Note**
>
> AWS SAM 템플릿에서 리소스 목록을 생성기 위해 템플릿의 로컬 변환이 수행됩니다. 특정 지역 내와 같은 조건에 따라 배포될 리소스가 이 목록에 포함됩니다.

사용법

```
$ sam list resources <options>
```

옵션

`--config-env` *TEXT*

사용할 구성 파일의 기본 매개변수 값을 지정하는 환경 이름입니다.

기본 값: `default`

구성 파일에 대한 자세한 내용은 [AWS SAMCLI구성 파일](#) 섹션을 참조하세요.

`--config-file` *TEXT*

사용할 기본 매개변수 값이 들어 있는 구성 파일의 경로 및 파일 이름입니다.

기본값: 현재 작업 디렉터리 내 `samconfig.toml`입니다.

구성 파일에 대한 자세한 내용은 AWS SAMCLI구성 파일 섹션을 참조하세요.

`--debug`

디버그 로깅을 켜면 AWS SAM CLI에서 생성한 디버그 메시지를 타임스탬프와 함께 인쇄할 수 있습니다.

`--help, -h`

이 메시지를 표시한 후 종료합니다.

`--output [json|table]`

결과 출력 형식을 지정합니다.

기본 값: `table`

`--profile` *`TEXT`*

보안 인증 정보 파일에서 특정 프로필을 선택하여 AWS 보안 인증을 가져옵니다.

`--region` *`TEXT`*

서비스 AWS 지역을 설정합니다. 예를 들어 `us-east-1`입니다.

`--save-params`

명령줄에서 AWS SAM 구성 파일에 제공한 매개변수를 저장합니다.

`--stack-name` *`TEXT`*

배포된 CloudFormation 스택의 이름입니다. 스택 이름은 애플리케이션의 `samconfig.toml` 파일 또는 지정된 구성 파일에서 찾을 수 있습니다.

제공된 경우 템플릿의 리소스 논리 ID가 CloudFormation 내 상응하는 물리적 ID에 매핑됩니다. 물리적 ID에 대한 자세한 내용은 사용자 가이드의 AWS CloudFormation리소스 필드를 참조하세요.

이 옵션을 지정하지 않으면 템플릿에 정의된 로컬 리소스가 표시됩니다.

`--template-file, --template, -t` *`PATH`*

AWS SAM 템플릿 파일.

기본 값: `template.[yaml|yml|json]`

예시

AWS SAM 템플릿의 로컬 리소스와 CloudFormation로 명명된 `test-stack` 스택의 배포된 리소스의 출력 결과를 테이블 형식으로 표시합니다. 로컬 템플릿과 동일한 디렉터리에서 실행합니다.

```
$ sam list resources --stack-name test-stack --output table

------------------------------------------------------------------------------------------------------
Logical ID                                                  Physical ID
------------------------------------------------------------------------------------------------------
HelloWorldFunction                                          sam-app-test-list-
HelloWorldFunction-H85Y7yIV7ZLq
HelloWorldFunctionHelloWorldPermissionProd                  sam-app-test-list-

 HelloWorldFunctionHelloWorldPermissionProd-1QH7CPOCBL2IK
HelloWorldFunctionRole                                      sam-app-test-list-
HelloWorldFunctionRole-SRJDMJ6F7F41
ServerlessRestApi                                           uj80uoe2o2
ServerlessRestApiDeployment47fc2d5f9d                       pncw5f
ServerlessRestApiProdStage                                  Prod
ServerlessRestApiDeploymentf5716dc08b                       -
------------------------------------------------------------------------------------------------------
```

sam list stack-outputs

이 페이지는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) `sam list stack-outputs` 하위 명령에 대한 참조 정보를 제공합니다.

AWS SAM CLI에 대한 소개는 [란 무엇입니까 AWS SAMCLI?](#) 섹션을 참조하세요.

`sam list stack-outputs`하위 명령은 AWS Serverless Application Model(AWS SAM) 또는 CloudFormation 템플릿의 AWS CloudFormation 스택 출력을 표시합니다. `Outputs`에 관한 자세한 내용은 AWS CloudFormation 사용자 가이드의 [출력](#)을 참조하세요.

사용법

```
$ sam list stack-outputs <options>
```

옵션

`--config-env TEXT`

사용할 구성 파일의 기본 매개변수 값을 지정하는 환경 이름입니다.

기본 값: `default`

구성 파일에 대한 자세한 내용은 AWS SAMCLI구성 파일 섹션을 참조하세요.

`--config-file` *TEXT*

사용할 기본 매개변수 값이 들어 있는 구성 파일의 경로 및 파일 이름입니다.

기본값: 현재 작업 디렉터리 내 `samconfig.toml`입니다.

구성 파일에 대한 자세한 내용은 AWS SAMCLI구성 파일 섹션을 참조하세요.

`--debug`

디버그 로깅을 켜면 AWS SAM CLI에서 생성한 디버그 메시지를 타임스탬프와 함께 인쇄할 수 있습니다.

`--help, -h`

이 메시지를 표시한 후 종료합니다.

`--output [json|table]`

결과 출력 형식을 지정합니다.

기본 값: `table`

`--profile` *TEXT*

보안 인증 정보 파일에서 특정 프로필을 선택하여 AWS 보안 인증을 가져옵니다.

`--region` *TEXT*

서비스 AWS 지역을 설정합니다. 예를 들어 `us-east-1`입니다.

`--save-params`

명령줄에서 AWS SAM 구성 파일에 제공한 매개변수를 저장합니다.

`--stack-name` *TEXT*

배포된 CloudFormation 스택의 이름입니다. 스택 이름은 애플리케이션의 `samconfig.toml` 파일 또는 지정된 구성 파일에서 찾을 수 있습니다.

이 옵션은 필수입니다.

예시

`test-stack`로 이름이 지정된 귀하의 CloudFormation 스택의 리소스 출력을 표 형식으로 표시합니다.

```
$ sam list stack-outputs --stack-name test-stack --output table

------------------------------------------------------------------------------------------------------
OutputKey                               OutputValue
 Description
------------------------------------------------------------------------------------------------------
HelloWorldFunctionIamRole               arn:aws:iam::account-number:role/sam-
 Implicit IAM Role created for Hello
                                        app-test-list-HelloWorldFunctionRole-    World
 function
                                        SRJDMJ6F7F41
HelloWorldApi                           https://uj80uoe2o2.execute-api.us-       API
 Gateway endpoint URL for Prod
                                        east-1.amazonaws.com/Prod/hello/         stage
 for Hello World function
HelloWorldFunction                      arn:aws:lambda:us-                       Hello
 World Lambda Function ARN
                                        east-1:account-number:function:sam-app-
                                        test-list-
                                        HelloWorldFunction-H85Y7yIV7ZLq
------------------------------------------------------------------------------------------------------
```

sam local callback

이 페이지에서는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAMCLI) `sam local callback` 명령에 대한 참조 정보를 제공합니다.

에 대한 소개는 단원을 AWS SAMCLI참조하십시오. [란 무엇입니까 AWS SAMCLI?]

`sam local callback` 명령을 사용하면 개발 및 테스트 중에 로컬 내구성 함수 실행으로 콜백을 보낼 수 있습니다.

사용법

```
$ sam local callback <subcommand> <options>
```

옵션

`--help, -h`

이 메시지를 표시한 후 종료합니다.

하위 명령

`succeed`

내구성 있는 함수 실행으로 성공 콜백을 보냅니다. 자세한 내용은 sam local callback succeed 단원을 참조하십시오.

`fail`

내구성 있는 함수 실행으로 장애 콜백을 전송합니다. 자세한 내용은 sam local callback fail 단원을 참조하십시오.

`heartbeat`

하트비트 콜백을 내구성 있는 함수 실행으로 전송합니다. 자세한 내용은 sam local callback heartbeat 단원을 참조하십시오.

sam local callback succeed

내구성 있는 함수 실행으로 성공 콜백을 보냅니다.

사용법

```
$ sam local callback succeed CALLBACK_ID [OPTIONS]
```

필수 인수

`CALLBACK_ID`

성공 응답을 보낼 콜백의 고유 식별자입니다.

옵션

`-r, --result TEXT`

성공 결과 페이로드를 문자열로 표시합니다.

`--region TEXT`

서비스의 AWS 리전을 설정합니다(예: `us-east-1`).

`--profile TEXT`

자격 증명 파일에서 특정 프로필을 선택하여 AWS 자격 증명을 가져옵니다.

`--config-env TEXT`

구성 파일에서 기본 파라미터 값을 지정하는 환경 이름입니다. 기본값: `default`

`--config-file TEXT`

기본 파라미터 값이 포함된 구성 파일입니다. 기본값: `samconfig.toml`

`--save-params`

명령줄을 통해 제공된 파라미터를 구성 파일에 저장합니다.

`--beta-features / --no-beta-features`

베타 기능을 활성화/비활성화합니다.

`--debug`

디버그 로깅을 켜서 AWS SAM CLI에서 생성된 디버그 메시지를 인쇄하고 타임스탬프를 표시합니다.

`-h, --help`

이 메시지를 표시한 후 종료합니다.

예제

결과 없이 성공 콜백 전송:

```
$ sam local callback succeed my-callback-id
```

결과와 함께 성공 콜백 전송:

```
$ sam local callback succeed my-callback-id --result 'Task completed successfully'
```

짧은 옵션으로 성공 콜백 전송:

```
$ sam local callback succeed my-callback-id -r 'Success result'
```

sam local callback fail

내구성 있는 함수 실행으로 장애 콜백을 전송합니다.

사용법

```
$ sam local callback fail CALLBACK_ID [OPTIONS]
```

필수 인수

`CALLBACK_ID`

실패 응답을 보낼 콜백의 고유 식별자입니다.

옵션

`--error-data TEXT`

추가 오류 데이터.

`--stack-trace TEXT`

트레이스 항목을 스택합니다. 이 옵션은 여러 번 지정할 수 있습니다.

`--error-type TEXT`

오류의 유형입니다.

`--error-message TEXT`

자세한 오류 메시지입니다.

`--region TEXT`

서비스의 AWS 리전을 설정합니다(예: `us-east-1`).

`--profile TEXT`

자격 증명 파일에서 특정 프로필을 선택하여 AWS 자격 증명을 가져옵니다.

`--config-env TEXT`

구성 파일에서 기본 파라미터 값을 지정하는 환경 이름입니다. 기본값: `default`

`--config-file TEXT`

기본 파라미터 값이 포함된 구성 파일입니다. 기본값: `samconfig.toml`

`--save-params`

명령줄을 통해 제공된 파라미터를 구성 파일에 저장합니다.

`--beta-features / --no-beta-features`

베타 기능을 활성화/비활성화합니다.

`--debug`

디버그 로깅을 켜서 AWS SAM CLI에서 생성된 디버그 메시지를 인쇄하고 타임스탬프를 표시합니다.

`-h, --help`

이 메시지를 표시한 후 종료합니다.

예제

파라미터 없이 실패 콜백 전송:

```
$ sam local callback fail my-callback-id
```

오류 메시지와 함께 실패 콜백 전송:

```
$ sam local callback fail my-callback-id --error-message 'Task failed'
```

모든 파라미터와 함께 실패 콜백 전송:

```
$ sam local callback fail my-callback-id --error-message 'Task failed' --error-type
 'ValidationError' --stack-trace 'at line 42' --error-data '{"code": 500}'
```

sam local callback heartbeat

하트비트 콜백을 내구성 있는 함수 실행으로 전송합니다.

사용법

```
$ sam local callback heartbeat CALLBACK_ID [OPTIONS]
```

필수 인수

`CALLBACK_ID`

하트비트를 보낼 콜백의 고유 식별자입니다.

옵션

`--region TEXT`

서비스의 AWS 리전을 설정합니다(예: `us-east-1`).

`--profile TEXT`

자격 증명 파일에서 특정 프로필을 선택하여 AWS 자격 증명을 가져옵니다.

`--config-env TEXT`

구성 파일에서 기본 파라미터 값을 지정하는 환경 이름입니다. 기본값: `default`

`--config-file TEXT`

기본 파라미터 값이 포함된 구성 파일입니다. 기본값: `samconfig.toml`

`--save-params`

명령줄을 통해 제공된 파라미터를 구성 파일에 저장합니다.

`--beta-features / --no-beta-features`

베타 기능을 활성화/비활성화합니다.

`--debug`

디버그 로깅을 켜서 AWS SAM CLI에서 생성된 디버그 메시지를 인쇄하고 타임스탬프를 표시합니다.

`-h, --help`

이 메시지를 표시한 후 종료합니다.

예제

하트비트 콜백 전송:

```
$ sam local callback heartbeat my-callback-id
```

sam local execution

이 페이지에서는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAMCLI) `sam local execution` 명령에 대한 참조 정보를 제공합니다.

에 대한 소개는 단원을 AWS SAMCLI참조하십시오. 란 무엇입니까 AWS SAMCLI?

`sam local execution` 명령을 사용하면 개발 및 테스트 중에 로컬 내구성 함수 실행을 관리하고 검사할 수 있습니다.

> **Note**
>
> 이러한 명령은 AWS 자격 증명에 액세스할 필요가 없을 수 있습니다.

사용법

```
$ sam local execution <subcommand> <options>
```

옵션

`--help, -h`

이 메시지를 표시한 후 종료합니다.

하위 명령

`get`

내구성 있는 함수 실행에 대한 세부 정보를 가져옵니다. 자세한 내용은 sam local execution get 단원을 참조하십시오.

`history`

내구성 있는 함수 실행의 실행 기록을 가져옵니다. 자세한 내용은 sam local execution history 단원을 참조하십시오.

`stop`

내구성 있는 함수 실행을 중지합니다. 자세한 내용은 sam local execution stop 단원을 참조하십시오.

sam local execution get

특정 내구성 함수 실행의 세부 정보를 검색합니다.

> **Note**
>
> 이 명령은 AWS 자격 증명에 대한 액세스가 필요하지 않을 수 있습니다.

사용법

```
$ sam local execution get DURABLE_EXECUTION_ARN [OPTIONS]
```

필수 인수

`DURABLE_EXECUTION_ARN`

세부 정보를 검색할 내구성 함수 실행의 Amazon 리소스 이름(ARN)입니다.

옵션

`--format [summary|json]`

출력 형식. 기본값: `summary`

`--region TEXT`

서비스의 AWS 리전을 설정합니다(예: `us-east-1`).

`--profile TEXT`

자격 증명 파일에서 특정 프로필을 선택하여 AWS 자격 증명을 가져옵니다.

`--config-env TEXT`

구성 파일에서 기본 파라미터 값을 지정하는 환경 이름입니다. 기본값: `default`

`--config-file TEXT`

기본 파라미터 값이 포함된 구성 파일입니다. 기본값: `samconfig.toml`

`--save-params`

명령줄을 통해 제공된 파라미터를 구성 파일에 저장합니다.

`--beta-features / --no-beta-features`

베타 기능을 활성화/비활성화합니다.

`--debug`

디버그 로깅을 켜서 AWS SAM CLI에서 생성된 디버그 메시지를 인쇄하고 타임스탬프를 표시합니다.

`-h, --help`

이 메시지를 표시한 후 종료합니다.

예제

기본 요약 형식으로 실행 세부 정보를 가져옵니다.

```
$ sam local execution get arn:aws:lambda:us-east-1:123456789012:function:MyFunction:
$LATEST/durable-execution/my-execution-name/my-execution-id
```

JSON 형식으로 실행 세부 정보 가져오기:

```
$ sam local execution get arn:aws:lambda:us-east-1:123456789012:function:MyFunction:
$LATEST/durable-execution/my-execution-name/my-execution-id --format json
```

sam local execution history

내구성 있는 함수 실행의 실행 기록을 가져옵니다.

사용법

```
$ sam local execution history EXECUTION_ID [OPTIONS]
```

필수 인수

`EXECUTION_ID`

기록을 검색할 내구성 있는 함수 실행의 고유 식별자입니다.

옵션

`--format [table|json]`

출력 형식. 기본값: `table`

`--region TEXT`

서비스의 AWS 리전을 설정합니다(예: `us-east-1`).

`--profile TEXT`

자격 증명 파일에서 특정 프로필을 선택하여 AWS 자격 증명을 가져옵니다.

`--config-env TEXT`

구성 파일에서 기본 파라미터 값을 지정하는 환경 이름입니다. 기본값: `default`

`--config-file TEXT`

기본 파라미터 값이 포함된 구성 파일입니다. 기본값: `samconfig.toml`

`--save-params`

명령줄을 통해 제공된 파라미터를 구성 파일에 저장합니다.

`--beta-features / --no-beta-features`

베타 기능을 활성화/비활성화합니다.

`--debug`

디버그 로깅을 켜서 AWS SAM CLI에서 생성된 디버그 메시지를 인쇄하고 타임스탬프를 표시합니다.

`-h, --help`

이 메시지를 표시한 후 종료합니다.

예제

테이블 형식으로 실행 기록을 가져옵니다.

```
$ sam local execution history my-execution-id
```

JSON 형식으로 실행 기록 가져오기:

```
$ sam local execution history my-execution-id --format json
```

sam local execution stop

실행 중인 내구성 함수 실행을 중지합니다.

> **Note**
> 이 명령은 AWS 자격 증명에 대한 액세스가 필요하지 않을 수 있습니다.

사용법

```
$ sam local execution stop DURABLE_EXECUTION_ARN [OPTIONS]
```

필수 인수

`DURABLE_EXECUTION_ARN`

중지할 내구성 함수 실행의 Amazon 리소스 이름(ARN)입니다.

옵션

`--error-message TEXT`

중지된 실행과 연결할 오류 메시지입니다.

`--error-type TEXT`

중지된 실행과 연결할 오류 유형입니다.

`--error-data TEXT`

중지된 실행과 연결할 오류 데이터입니다.

`--stack-trace TEXT`

트레이스 항목을 스택합니다. 이 옵션은 여러 번 지정할 수 있습니다.

`--region TEXT`

서비스의 AWS 리전을 설정합니다(예: `us-east-1`).

`--profile TEXT`

자격 증명 파일에서 특정 프로필을 선택하여 AWS 자격 증명을 가져옵니다.

`--config-env TEXT`

구성 파일에서 기본 파라미터 값을 지정하는 환경 이름입니다. 기본값: `default`

`--config-file TEXT`

기본 파라미터 값이 포함된 구성 파일입니다. 기본값: `samconfig.toml`

`--save-params`

명령줄을 통해 제공된 파라미터를 구성 파일에 저장합니다.

`--beta-features / --no-beta-features`

베타 기능을 활성화/비활성화합니다.

`--debug`

디버그 로깅을 켜서 AWS SAM CLI에서 생성된 디버그 메시지를 인쇄하고 타임스탬프를 표시합니다.

`-h, --help`

이 메시지를 표시한 후 종료합니다.

예제

오류 세부 정보 없이 실행 중지:

```
$ sam local execution stop arn:aws:lambda:us-east-1:123456789012:function:MyFunction:
$LATEST/durable-execution/my-execution-name/my-execution-id
```

오류 메시지 및 유형으로 실행 중지:

```
$ sam local execution stop arn:aws:lambda:us-east-1:123456789012:function:MyFunction:
$LATEST/durable-execution/my-execution-name/my-execution-id --error-message "Execution
 cancelled" --error-type "UserCancellation"
```

전체 오류 세부 정보 및 스택 추적으로 실행을 중지합니다.

```
$ sam local execution stop arn:aws:lambda:us-east-1:123456789012:function:MyFunction:
$LATEST/durable-execution/my-execution-name/my-execution-id --error-message "Task
 failed" --error-type "TaskFailure" --error-data '{"reason":"timeout"}' --stack-trace
 "at function1()" --stack-trace "at function2()"
```

sam local generate-event

이 페이지는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) `sam local generate-event` 하위 명령에 대한 참조 정보를 제공합니다.

- AWS SAM CLI에 대한 소개는 [란 무엇입니까 AWS SAMCLI?] 섹션을 참조하세요.
- AWS SAM CLI `sam local generate-event` 명령 사용에 대한 가이드는 [sam local generate-event를 사용한 테스트 소개] 섹션을 참조하세요.

`sam local generate-event` 하위 명령은 지원되는 AWS 서비스을 위한 이벤트 페이로드 샘플을 생성합니다.

사용법

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

옵션

`--config-env TEXT`

사용할 구성 파일의 기본 매개변수 값을 지정하는 환경 이름입니다. 기본값은 "기본값"입니다. 구성 파일에 대한 자세한 내용은 [AWS SAMCLI구성 파일] 섹션을 참조하세요.

`--config-file PATH`

사용할 기본 매개변수 값이 들어 있는 구성 파일의 경로 및 파일 이름입니다. 기본값은 프로젝트 디렉터리의 루트 내 `samconfig.toml`입니다. 구성 파일에 대한 자세한 내용은 [AWS SAMCLI구성 파일] 섹션을 참조하세요.

`--help`

이 메시지를 표시한 후 종료합니다.

Service

지원되는 서비스 목록을 보려면 다음을 실행합니다.

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

Event

각 서비스에 대해 생성할 수 있는 지원되는 이벤트 목록을 보려면 다음을 실행하십시오.

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

이벤트 옵션

수정할 수 있는 지원되는 이벤트 옵션 목록을 보려면 다음을 실행하십시오.

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

예시

`sam local generate-event` 하위 명령 사용에 대한 예제는 샘플 이벤트 생성 섹션을 참조하세요.

sam local invoke

이 페이지에서는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAMCLI) `sam local invoke` 하위 명령에 대한 참조 정보를 제공합니다.

- 에 대한 소개는 단원을 AWS SAMCLI참조하십시오. 란 무엇입니까 AWS SAMCLI?
- `sam local invoke` 하위 명령 사용에 대한 설명서는 섹션을 참조하세요 AWS SAMCLIsam local invoke를 사용한 테스트 소개.

`sam local invoke` 하위 명령은 로컬에서 AWS Lambda 함수의 일회성 호출을 시작합니다.

> **Note**
>
> 내구성이 뛰어난 함수의 경우는 자동 체크포인트 및 재생을 통해 상태 저장 실행을 `sam local invoke` 지원합니다. 컨테이너는 상태 지속성 및 재개를 처리하기 위해 내구성 함수 실행 중에 계속 실행됩니다.

사용법

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

> **Note**
>
> AWS SAM 템플릿에 정의된 함수가 두 개 이상 있는 경우 호출하려는 함수 논리적 ID를 제공합니다.

인수

리소스 ID

간접 호출을 위한 Lambda 함수의 ID입니다.

이 인수는 선택 사항입니다. 애플리케이션에 단일 Lambda 함수가 포함된 경우 AWS SAM CLI는 이를 호출합니다. 애플리케이션에 여러 함수가 포함된 경우 간접 호출할 함수의 ID를 제공하십시오.

유효한 값: 리소스의 논리적 ID 또는 리소스 ARN입니다.

옵션

`--add-host` *`LIST`*

호스트 이름과 IP 주소 매핑을 Docker 컨테이너의 호스트 파일에 전달합니다. 이 파라미터는 여러 번 전달할 수 있습니다.

Example

예시: `--add-host` *`example.com:127.0.0.1`*

`--beta-features | --no-beta-features`

베타 기능을 허용 또는 거부합니다.

`--config-env` *`TEXT`*

사용할 구성 파일의 기본 매개변수 값을 지정하는 환경 이름입니다. 기본값은 “기본값”입니다. 구성 파일에 대한 자세한 내용은 [AWS SAMCLI구성 파일](#) 섹션을 참조하세요.

`--config-file` *`PATH`*

사용할 기본 매개변수 값이 들어 있는 구성 파일의 경로 및 파일 이름입니다. 기본값은 프로젝트 디렉터리의 루트에 있는 “`samconfig.toml`“입니다. 구성 파일에 대한 자세한 내용은 [AWS SAMCLI 구성 파일](#) 섹션을 참조하세요.

`--container-env-vars`

(선택 사항) 로컬에서 디버깅할 때 Lambda 함수 이미지 컨테이너에 환경 변수를 전달하십시오.

`--container-host` *TEXT*

로컬로 에뮬레이션된 Lambda 컨테이너의 호스트입니다. 기본값은 `localhost`입니다. macOS의 Docker 컨테이너에서 AWS SAMCLI를 실행하려는 경우 `host.docker.internal`을 지정할 수 있습니다. 다른 호스트에서 컨테이너를 실행하려면 원격 호스트의 IP 주소를 지정할 AWS SAMCLI 수 있습니다.

`--container-host-interface` *TEXT*

컨테이너 포트가 바인딩해야 하는 호스트 네트워크 인터페이스의 IP 주소입니다. 기본값은 `127.0.0.1`입니다. 모든 인터페이스에 바인딩하는 데 `0.0.0.0`을 사용합니다.

`--debug`

디버그 로깅을 켜서 AWS SAMCLI가 생성한 디버그 메시지를 인쇄하고 타임스탬프를 표시합니다.

`--debug-args` *TEXT*

디버거에 전달할 추가 인수입니다.

`--debug-port, -d` *TEXT*

지정되면 Lambda 함수 컨테이너를 디버그 모드에서 시작하고 이 포트를 로컬 호스트에 노출합니다.

`--debugger-path` *TEXT*

Lambda 컨테이너에 마운트된 디버거의 호스트 경로입니다.

`--docker-network` *TEXT*

Lambda Docker 컨테이너가 연결해야 하는 기존 Docker 네트워크의 이름 또는 ID와 기본 브리지 네트워크입니다. 이것이 지정되지 않으면 Lambda 컨테이너는 기본 브리지 Docker 네트워크에만 연결됩니다.

`--docker-volume-basedir, -v` *TEXT*

AWS SAM 파일이 있는 기본 디렉터리의 위치입니다. Docker가 원격 시스템에서 실행 중인 경우 Docker 시스템에 AWS SAM 파일이 있는 경로를 탑재하고 원격 시스템과 일치하도록이 값을 수정해야 합니다.

`--durable-execution-name` *TEXT*

내구성 실행 이름(내구성 함수에만 해당)

`--env-vars, -n PATH`

Lambda 함수의 환경 변수의 값을 포함하는 JSON 파일입니다. 환경 변수 파일에 대한 자세한 내용은 환경 변수 파일 섹션을 참조하세요.

`--event, -e PATH`

간접 호출 시 Lambda 함수로 전달되는 이벤트 데이터를 포함하는 JSON 파일입니다. 이 옵션을 지정하지 않으면 이벤트가 가정되지 않습니다. `stdin`에서 JSON을 입력하려면 '-' 값을 전달해야 합니다. 다양한 AWS 서비스의 이벤트 메시지 형식에 대한 자세한 내용은 AWS Lambda 개발자 안내서의 다른 서비스 작업을 참조하세요.

`--force-image-build`

레이어가 있는 Lambda 함수를 간접 호출하는 데 사용된 이미지를 AWS SAMCLI가 다시 구축해야 하는지를 지정합니다.

`--help`

이 메시지를 표시한 후 종료합니다.

`--hook-name TEXT`

AWS SAMCLI 기능을 확장하는 데 사용되는 후크의 이름입니다.

허용되는 값: `terraform`.

`--invoke-image TEXT`

로컬 함수 간접 호출에 사용하려는 컨테이너 이미지의 URI입니다. 기본적으로, AWS SAM 은 Amazon ECR 퍼블릭으로부터 컨테이너 이미지를 가져옵니다(의 이미지 리포지토리 AWS SAM에 나열됨). 다른 위치에서 이미지를 가져오려면 이 옵션을 사용합니다.

예를 들어 `sam local invoke MyFunction --invoke-image amazon/aws-sam-cli-emulation-image-python3.8`입니다.

`--layer-cache-basedir DIRECTORY`

템플릿에서 사용하는 레이어가 다운로드되는 기본 디렉토리의 위치를 지정합니다.

`--log-file, -l TEXT`

런타임 로그를 전송할 로그 파일입니다.

`--mount-symlinks`

AWS SAM CLI이(가) 빌드하거나 호출할 파일에 존재하는 심볼릭 링크를 항상 마운트하도록 보장합니다. 이는 최상위 디렉터리(즉, 함수의 루트에 직접 있는 심볼릭 링크)에만 적용됩니다. 기본적

으로, NodeJS에서 `node_modules`을(를) 위해 `build-in-source`을(를) 사용하는 데 필요한 경우를 제외하고 심볼릭 링크는 마운트되지 않습니다.

`--no-event`

빈 이벤트와 함께 함수를 간접 호출합니다.

`--no-memory-limit`

AWS SAM 템플릿에 메모리가 구성된 경우에도 로컬 호출 중에 컨테이너의 메모리 제한을 제거합니다.

`--parameter-overrides`

CloudFormation 파라미터가 포함된 문자열은 키-값 페어로 인코딩됩니다. AWS Command Line Interface ()와 동일한 형식을 사용합니다AWS CLI. AWS SAM CLI 형식은 명시적 키 및 값 키워드이며 각 재정의는 공백으로 구분됩니다. 다음은 두 가지 예제입니다.

- `--parameter-overrides ParameterKey=hello,ParameterValue=world`
- `--parameter-overrides ParameterKey=hello,ParameterValue=world ParameterKey=example1,ParameterValue=example2 ParameterKey=apple,ParameterValue=banana`

`--profile` *TEXT*

자격 증명을 가져오는 AWS 자격 증명 파일의 특정 프로필입니다.

`--region` *TEXT*

배포할 AWS 리전입니다. 예를 들어 us-east-1입니다.

`--runtime` *TEXT*

지정된 런타임을 사용하여 Lambda 함수를 로컬에서 간접 호출합니다. 이는 `template.yml` 파일에 정의된 런타임을 재정의합니다. 또한 원래 함수 구성을 수정하지 않고 다른 런타임으로 Lambda 함수를 테스트할 수 있습니다.

`--save-params`

명령줄에 제공한 파라미터를 AWS SAM 구성 파일에 저장합니다.

`--shutdown`

종료 동작의 확장 처리를 테스트하기 위하여 간접 호출 완료 이후라도 종료 이벤트를 에뮬레이션합니다.

`--skip-prepare-infra`

인프라를 변경하지 않은 경우 준비 단계를 건너뜁니다. `--hook-name` 옵션과 함께 사용합니다.

`--skip-pull-image`

기본 사항으로서, AWS SAMCLI 는 Lambda의 최신 원격 런타임 환경을 확인하고 로컬 이미지를 자동으로 업데이트하여 동기화를 유지합니다.

Lambda 런타임 환경의 최신 Docker 이미지를 가져오지 않으려면 이 옵션을 지정하십시오.

`--template, -t` *`PATH`*

AWS SAM 템플릿 파일입니다.

이 옵션은 `--hook-name`과 호환되지 않습니다.

> **Note**
>
> 이 옵션을 지정하면는 템플릿과 템플릿이 가리키는 로컬 리소스만 AWS SAM 로드합니다.

`--tenant-id` *`TEXT`*

다중 테넌트 Lambda 함수의 테넌트 ID입니다. 서로 다른 테넌트 간의 컴퓨팅 격리를 보장하는 데 사용됩니다. 테넌트 격리 모드로 구성된 함수를 호출할 때 필요합니다.

`--terraform-plan-file`

Terraform AWS SAM를 CLI와 사용할 때 로컬 Terraform Cloud 계획 파일의 상대적 또는 절대적 경로입니다. 이 옵션을 사용하려면 `--hook-name`을 `terraform`로 설정해야 합니다.

예제

다음 예제에서는 `s3.json` 이벤트를 통해 로컬에서 Lambda 함수를 간접 호출하여 로컬 테스트를 위해 생성된 이벤트를 사용합니다.

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

다음 예시는 Python 3.11 런타임을 사용하여 `HelloWorldFunction` 함수를 테스트합니다.

```
$ sam local invoke --runtime python3.11 HelloWorldFunction
```

다음 예제에서는 내구성 `HelloWorldFunction` 있는 실행 이름으로 함수를 테스트합니다.

```
$ sam local invoke HelloWorldFunction --durable-execution-name my-execution
```

sam local start-api

이 페이지는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) `sam local start-api` 하위 명령에 대한 참조 정보를 제공합니다.

- AWS SAM CLI에 대한 소개는 [란 무엇입니까 AWS SAMCLI?] 섹션을 참조하세요.
- AWS SAM CLI `sam local start-api` 하위 명령 사용에 대한 설명서는 [sam local start-api를 사용한 테스트 소개] 섹션을 참조하세요.

`sam local start-api` 하위 명령은 AWS Lambda 함수를 로컬에서 실행하여 로컬 HTTP 서버 호스트를 통해 테스트합니다.

사용법

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

옵션

`--add-host` *LIST*

호스트 이름과 IP 주소 매핑을 Docker 컨테이너의 호스트 파일에 전달합니다. 이 파라미터는 여러 번 전달할 수 있습니다.

Example

예시: `--add-host` *`example.com:127.0.0.1`*

`--beta-features | --no-beta-features`

베타 기능을 허용 또는 거부합니다.

`--config-env` *TEXT*

사용할 구성 파일의 기본 매개변수 값을 지정하는 환경 이름입니다. 기본값은 “기본값”입니다. 구성 파일에 대한 자세한 내용은 [AWS SAMCLI구성 파일] 섹션을 참조하세요.

`--config-file` *PATH*

사용할 기본 매개변수 값이 들어 있는 구성 파일의 경로 및 파일 이름입니다. 기본값은 프로젝트 디렉터리의 루트에 있는 “samconfig.toml”입니다. 구성 파일에 대한 자세한 내용은 AWS SAMCLI구성 파일 섹션을 참조하세요.

`--container-env-vars`

선택 사항입니다. 로컬 디버깅 시 이미지 컨테이너에 환경 변수를 전달합니다.

`--container-host` *TEXT*

로컬로 에뮬레이션된 Lambda 컨테이너의 호스트입니다. 기본값은 `localhost`입니다. macOS의 Docker 컨테이너에서 AWS SAM CLI를 실행하려는 경우 `host.docker.internal`을 지정할 수 있습니다. AWS SAM CLI가 아닌 다른 호스트에서 컨테이너를 실행하려는 경우 원격 호스트의 IP 주소를 지정할 수 있습니다.

`--container-host-interface` *TEXT*

컨테이너 포트가 바인딩해야 하는 호스트 네트워크 인터페이스의 IP 주소입니다. 기본값은 `127.0.0.1`입니다. 모든 인터페이스에 바인딩하는 데 `0.0.0.0`을 사용합니다.

`--debug`

디버그 로깅을 켜서 AWS SAM CLI 에 의해 생성된 디버그 메시지를 인쇄하고 타임스탬프를 표시합니다.

`--debug-args` *TEXT*

디버거에 전달할 추가 인수입니다.

`--debug-function`

선택 사항입니다. `--warm-containers`이 지정된 시점에 디버그 옵션을 적용할 Lambda 함수를 지정합니다. 이 매개변수는`--debug-port`, `--debugger-path` 및 `--debug-args`에 적용됩니다.

`--debug-port, -d` *TEXT*

지정되면 Lambda 함수 컨테이너를 디버그 모드에서 시작하고 이 포트를 로컬 호스트에 노출합니다.

`--debugger-path` *TEXT*

Lambda 컨테이너에 마운트할 디버거의 호스트 경로입니다.

`--docker-network` *TEXT*

Lambda Docker 컨테이너가 연결해야 하는 기존 Docker 네트워크의 이름 또는 ID와 기본 브리지 네트워크입니다. 이것을 지정하지 않으면 Lambda 컨테이너는 기본 브리지 Docker 네트워크에만 연결됩니다.

`--docker-volume-basedir, -v` *TEXT*

AWS SAM 파일이 있는 기본 디렉터리의 위치입니다. Docker가 원격 기기에서 실행 중인 경우 AWS SAM 파일이 있는 경로를 Docker 기기에 마운트하고 이 값을 원격 기기에 맞게 수정해야 합니다.

`--env-vars, -n` *PATH*

Lambda 함수의 환경 변수의 값을 포함하는 JSON 파일입니다.

`--force-image-build`

레이어를 사용하여 함수를 호출하는 데 사용된 이미지를 AWS SAM CLI가 다시 구축해야 하는지 지정합니다.

`--help`

이 메시지를 표시한 후 종료합니다.

`--hook-name` *TEXT*

AWS SAM CLI 기능을 확장하는 데 사용되는 후크의 이름입니다.

허용되는 값: `terraform`입니다.

`--host` *TEXT*

바인딩할 로컬 호스트 이름 또는 IP 주소(기본값: '127.0.0.1')입니다.

`--invoke-image` *TEXT*

Lambda 함수에 사용하려는 컨테이너 이미지의 URI입니다. 기본 사항으로 AWS SAM은 Amazon ECR Public로부터 컨테이너 이미지를 가져옵니다. 다른 위치에서 이미지를 가져오려면 이 옵션을 사용합니다.

이 옵션은 여러 번 지정할 수 있습니다. 이 옵션의 각 인스턴스는 문자열 또는 키-값 쌍을 사용할 수 있습니다. 문자열을 지정하는 경우 이는 애플리케이션의 모든 함수에 사용할 컨테이너 이미지의 URI입니다. 예를 들어 `sam local start-api --invoke-image public.ecr.aws/sam/emu-python3.8`입니다. 키-값 쌍을 지정하는 경우 키는 리소스 이름이고 값은 해당 리소스에 사용할 컨테이너 이미지의 URI입니다. 예: `sam local start-api --invoke-image` public.ecr.aws/sam/emu-python3.8 --invoke-image Function1=amazon/aws-sam-cli-emulation-image-python3.8 . 키-값 쌍을 사용하면 리소스마다 다른 컨테이너 이미지를 지정할 수 있습니다.

`--layer-cache-basedir` *DIRECTORY*

템플릿에서 사용하는 레이어가 다운로드되는 위치를 기준으로 지정합니다.

`--log-file, -l` *TEXT*

런타임 로그를 전송할 로그 파일입니다.

`--no-memory-limit`

AWS SAM 템플릿에서 메모리가 구성된 경우에도 로컬 간접 호출 중 컨테이너의 메모리 제한을 제거합니다.

`--parameter-overrides`

키-값 페어로 인코딩된 CloudFormation 매개변수를 포함하는 스트링이 우선합니다. AWS Command Line Interface(AWS CLI)와 같은 형식을 사용합니다. AWS SAM CLI 형식은 명시적 키 및 값 키워드이며 각 재정의는 공백으로 구분됩니다. 다음은 두 가지 예제입니다.

- `--parameter-overrides ParameterKey=hello,ParameterValue=world`
- `--parameter-overrides ParameterKey=hello,ParameterValue=world ParameterKey=example1,ParameterValue=example2 ParameterKey=apple,ParameterValue=banana`

`--port, -p` *INTEGER*

수신할 로컬 포트 번호(기본값: '3000')입니다.

`--profile` *TEXT*

AWS 보안 인증 정보를 가져오는 보안 인증 파일의 특정 프로필입니다.

`--region` *TEXT*

배포할 AWS 지역입니다. 예를 들어 us-east-1입니다.

`--save-params`

명령줄에서 AWS SAM 구성 파일에 제공한 매개변수를 저장합니다.

`--shutdown`

종료 동작의 확장 처리를 테스트하기 위하여 간접 호출 완료 이후라도 종료 이벤트를 에뮬레이션합니다.

`--skip-prepare-infra`

인프라를 변경하지 않은 경우 준비 단계를 건너뜁니다. `--hook-name` 옵션과 함께 사용합니다.

`--skip-pull-image`

CLI가 Lambda 런타임에 대한 최신 도커 이미지를 가져오는 단계를 건너뛸지를 지정합니다.

`--ssl-cert-file` *PATH*

SSL 인증서 파일 경로입니다(기본값: 없음). 이 옵션을 사용할 경우 `--ssl-key-file` 옵션도 사용해야 합니다.

`--ssl-key-file` *PATH*

SSL 키 파일 경로입니다(기본값: 없음). 이 옵션을 사용할 경우 `--ssl-cert-file` 옵션도 사용해야 합니다.

`--static-dir, -s` *TEXT*

이 디렉터리에 있는 모든 정적 자산 (예: CSS/자바스크립트/HTML) 파일은 `/`에 표시됩니다.

`--template, -t` *PATH*

AWS SAM 템플릿 파일입니다.

> **Note**
>
> 이 옵션을 지정하는 경우 AWS SAM은 템플릿과 템플릿이 가리키는 로컬 리소스만 로드합니다.

`--terraform-plan-file`

Terraform AWS SAM를 CLI와 사용할 때 로컬 Terraform Cloud 계획 파일의 상대적 또는 절대적 경로입니다. 이 옵션을 사용하려면 `--hook-name`을 `terraform`로 설정해야 합니다.

`--warm-containers` *[EAGER | LAZY]*

선택 사항입니다. AWS SAM CLI가 각 함수의 컨테이너를 관리하는 방법을 지정합니다.

사용 가능한 옵션은 다음 두 가지입니다.

EAGER: 모든 함수의 컨테이너는 시작 시 로드되며 호출과 호출 사이에도 유지됩니다.

LAZY: 컨테이너는 각 함수를 처음 간접 호출할 때만 로드됩니다. 이러한 컨테이너들은 추가 호출 시에도 계속 유지됩니다.

예시

다음 예제에서는 API를 통해 애플리케이션을 테스트할 수 있도록 로컬 서버를 시작합니다. 이 명령이 작동하려면 애플리케이션이 설치되어 있고 Docker가 실행되고 있어야 합니다.

```
$ sam local start-api --port 3000
```

sam local start-lambda

이 페이지는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) `sam local start-lambda` 하위 명령에 대한 참조 정보를 제공합니다.

- AWS SAM CLI에 대한 소개는 [란 무엇입니까 AWS SAMCLI?] 섹션을 참조하세요.
- AWS SAM CLI `sam local start-lambda` 하위 명령 사용에 대한 설명서는 [sam local start-lambda를 사용한 테스트 소개] 섹션을 참조하세요.

`sam local start-lambda` 하위 명령은 에뮬레이션할 로컬 엔드포인트를 시작합니다AWS Lambda.

사용법

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

옵션

`--add-host LIST`

호스트 이름과 IP 주소 매핑을 Docker 컨테이너의 호스트 파일에 전달합니다. 이 파라미터는 여러 번 전달할 수 있습니다.

Example

예시: `--add-host example.com:127.0.0.1`

`--beta-features | --no-beta-features`

베타 기능을 허용 또는 거부합니다.

`--config-env TEXT`

사용할 구성 파일의 기본 매개변수 값을 지정하는 환경 이름입니다. 기본값은 “기본값”입니다. 구성 파일에 대한 자세한 내용은 AWS SAMCLI구성 파일 섹션을 참조하세요.

`--config-file PATH`

사용할 기본 매개변수 값이 들어 있는 구성 파일의 경로 및 파일 이름입니다. 기본값은 프로젝트 디렉터리의 루트에 있는 “samconfig.toml”입니다. 구성 파일에 대한 자세한 내용은 AWS SAMCLI구성 파일 섹션을 참조하세요.

`--container-env-vars`

선택 사항입니다. 로컬 디버깅 시 이미지 컨테이너에 환경 변수를 전달합니다.

`--container-host TEXT`

로컬로 에뮬레이션된 Lambda 컨테이너의 호스트입니다. 기본값은 `localhost`입니다. macOS의 Docker 컨테이너에서 AWS SAM CLI를 실행하려는 경우 `host.docker.internal`을 지정할 수 있습니다. AWS SAM CLI가 아닌 다른 호스트에서 컨테이너를 실행하려는 경우 원격 호스트의 IP 주소를 지정할 수 있습니다.

`--container-host-interface TEXT`

컨테이너 포트가 바인딩해야 하는 호스트 네트워크 인터페이스의 IP 주소입니다. 기본값은 `127.0.0.1`입니다. 모든 인터페이스에 바인딩하는 데 `0.0.0.0`을 사용합니다.

`--debug`

디버그 로깅을 켜서 AWS SAM CLI 에 의해 생성된 디버그 메시지를 인쇄하고 타임스탬프를 표시합니다.

`--debug-args TEXT`

디버거에 전달할 추가 인수입니다.

`--debug-function`

선택 사항입니다. `--warm-containers`이 지정된 시점에 디버그 옵션을 적용할 Lambda 함수를 지정합니다. 이 매개변수는`--debug-port`, `--debugger-path` 및 `--debug-args`에 적용됩니다.

`--debug-port, -d TEXT`

지정되면 Lambda 함수 컨테이너를 디버그 모드에서 시작하고 이 포트를 로컬 호스트에 노출합니다.

`--debugger-path` *TEXT*

Lambda 컨테이너에 마운트된 디버거의 호스트 경로입니다.

`--docker-network` *TEXT*

Lambda Docker 컨테이너가 연결해야 하는 기존 Docker 네트워크의 이름 또는 ID와 기본 브리지 네트워크입니다. 이것이 지정되면 Lambda 컨테이너는 기본 브리지 Docker 네트워크에만 연결됩니다.

`--docker-volume-basedir, -v` *TEXT*

AWS SAM 파일이 있는 기본 디렉터리의 위치입니다. Docker가 원격 기기에서 실행 중인 경우 AWS SAM 파일이 있는 경로를 Docker 기기에 마운트하고 이 값을 원격 기기에 맞게 수정해야 합니다.

`--env-vars, -n` *PATH*

Lambda 함수의 환경 변수의 값을 포함하는 JSON 파일입니다.

`--force-image-build`

CLI이 레이어를 사용하여 함수를 호출하는 데 사용된 이미지를 다시 구축해야 하는지를 지정합니다.

`--help`

이 메시지를 표시한 후 종료합니다.

`--hook-name` *TEXT*

AWS SAM CLI 기능을 확장하는 데 사용되는 후크의 이름입니다.

허용되는 값: `terraform`입니다.

`--host` *TEXT*

바인딩할 로컬 호스트 이름 또는 IP 주소(기본값: '127.0.0.1')입니다.

`--invoke-image` *TEXT*

로컬 함수 간접 호출에 사용하려는 컨테이너 이미지의 URI입니다. 기본 사항으로 AWS SAM은 Amazon ECR Public로부터 컨테이너 이미지를 가져옵니다. 다른 위치에서 이미지를 가져오려면 이 옵션을 사용합니다.

예를 들어 `sam local start-lambda MyFunction --invoke-image amazon/aws-sam-cli-emulation-image-python3.8`입니다.

`--layer-cache-basedir DIRECTORY`

템플릿에서 사용하는 레이어가 다운로드되는 위치를 기준으로 지정합니다.

`--log-file, -l TEXT`

런타임 로그를 전송할 로그 파일입니다.

`--no-memory-limit`

AWS SAM 템플릿에서 메모리가 구성된 경우에도 로컬 간접 호출 중 컨테이너의 메모리 제한을 제거합니다.

`--parameter-overrides`

키-값 페어로 인코딩된 CloudFormation 매개변수를 포함하는 스트링이 우선합니다. AWS Command Line Interface(AWS CLI)와 같은 형식을 사용합니다. AWS SAM CLI 형식은 명시적 키 및 값 키워드이며 각 재정의는 공백으로 구분됩니다. 다음은 두 가지 예제입니다.

- `--parameter-overrides ParameterKey=hello,ParameterValue=world`
- `--parameter-overrides ParameterKey=hello,ParameterValue=world ParameterKey=example1,ParameterValue=example2 ParameterKey=apple,ParameterValue=banana`

`--port, -p INTEGER`

수신할 로컬 포트 번호(기본값: '3001')입니다.

`--profile TEXT`

AWS 보안 인증 정보를 가져오는 보안 인증 파일의 특정 프로필입니다.

`--region TEXT`

배포할 AWS 지역입니다. 예를 들어 us-east-1입니다.

`--save-params`

명령줄에서 AWS SAM 구성 파일에 제공한 매개변수를 저장합니다.

`--shutdown`

종료 동작의 확장 처리를 테스트하기 위하여 간접 호출 완료 이후라도 종료 이벤트를 에뮬레이션합니다.

`--skip-prepare-infra`

인프라를 변경하지 않은 경우 준비 단계를 건너뜁니다. `--hook-name` 옵션과 함께 사용합니다.

`--skip-pull-image`

CLI 이 Lambda 런타임에 대한 최신 도커 이미지를 가져오는 단계를 건너뛸지를 지정합니다.

`--template, -t` *PATH*

AWS SAM 템플릿 파일입니다.

> **Note**
>
> 이 옵션을 지정하는 경우 AWS SAM은 템플릿과 템플릿이 가리키는 로컬 리소스만 로드합니다. 이 옵션은 `--hook-name`과 호환되지 않습니다.

`--terraform-plan-file`

Terraform AWS SAM를 CLI와 사용할 때 로컬 Terraform Cloud 계획 파일의 상대적 또는 절대적 경로입니다. 이 옵션을 사용하려면 `--hook-name`을 `terraform`로 설정해야 합니다.

`--warm-containers` *[EAGER | LAZY]*

선택 사항입니다. AWS SAM CLI가 각 함수의 컨테이너를 관리하는 방법을 지정합니다.

사용 가능한 옵션은 다음 두 가지입니다.

- `EAGER`: 모든 함수의 컨테이너는 시작 시 로드되며 호출과 호출 사이에도 유지됩니다.
- `LAZY`: 컨테이너는 각 함수를 처음 간접 호출할 때만 로드됩니다. 이러한 컨테이너들은 추가 호출 시에도 계속 유지됩니다.

예시

`sam local start-lambda` 하위 명령 사용에 대한 자세한 예제와 심층적인 단계별 설명은 sam local start-lambda를 사용한 테스트 소개 섹션을 참조하세요.

sam logs

이 페이지는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) `sam logs` 명령에 대한 참조 정보를 제공합니다.

AWS SAM CLI에 대한 소개는 란 무엇입니까 AWS SAMCLI? 섹션을 참조하세요.

이 `sam logs` 명령은 AWS Lambda 함수에서 생성된 로그를 가져옵니다.

사용법

```
$ sam logs <options>
```

옵션

`--config-env TEXT`

사용할 구성 파일의 기본 매개변수 값을 지정하는 환경 이름입니다. 기본값은 "기본값"입니다. 구성 파일에 대한 자세한 내용은 AWS SAMCLI구성 파일 섹션을 참조하세요.

`--config-file PATH`

사용할 기본 매개변수 값이 들어 있는 구성 파일의 경로 및 파일 이름입니다. 기본값은 프로젝트 디렉터리의 루트에 있는 "samconfig.toml"입니다. 구성 파일에 대한 자세한 내용은 AWS SAMCLI구성 파일 섹션을 참조하세요.

`--cw-log-group LIST`

지정한 CloudWatch Logs 로그 그룹의 로그를 포함합니다. 이 옵션을 `name`과 함께 지정하는 경우, AWS SAM는 지정된 로그 그룹의 로그와 명명된 리소스의 로그도 포함됩니다.

`--debug`

디버그 로깅을 켜서 AWS SAM CLI 에 의해 생성된 디버그 메시지를 인쇄하고 타임스탬프를 표시합니다.

`---end-time, e TEXT`

이 시간까지의 로그를 가져옵니다. 시간은 '5분 전', '내일'과 같은 상댓값이거나 '2018-01-01 10:10:10'과 같은 형식이 지정된 타임스탬프일 수 있습니다.

`--filter TEXT`

표현식을 지정하여 로그 이벤트에서 단어, 구문 또는 값과 일치하는 로그를 빠르게 찾을 수 있습니다. 이는 간단한 키워드 (예: "error") 이거나 Amazon CloudWatch Logs에서 지원하는 패턴일 수 있습니다. 명령문에 대한 내용은 Amazon CloudWatch Logs 설명서를 참조하세요.

`--help`

이 메시지를 표시한 후 종료합니다.

`--include-traces`

로그 출력에 X-Ray 트레이스를 포함합니다.

`--name, -n` *TEXT*

로그를 가져올 리소스의 이름입니다. 이 리소스가 CloudFormation 스택의 일부인 경우 이는 CloudFormation/AWS SAM 템플릿에 있는 함수 리소스의 논리적 ID일 수 있습니다. 인자를 반복함으로써 여러 이름을 제공할 수 있습니다. 리소스가 중첩된 스택에 있는 경우 이름 앞에 중첩된 스택 이름을 추가하여 해당 리소스에서 로그를 가져올 수 있습니다 (NestedStackLogicalID/ResourceLogicalID). 리소스 이름을 지정하지 않으면 지정된 스택을 스캔하고 지원되는 모든 리소스에 대한 로그 정보를 가져옵니다. 이 옵션을 지정하지 않으면 AWS SAM은 지정한 스택에서 모든 리소스에 대한 로그를 가져옵니다. 다음과 같은 리소스 유형이 지원됩니다.

- `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*

로그에 대한 출력 형식을 지정합니다. 형식이 지정된 로그를 인쇄하려면 `text`을 지정합니다. 로그를 JSON으로 인쇄하려면 `json`을 지정합니다.

`--profile` *TEXT*

AWS 보안 인증 정보를 가져오는 보안 인증 파일의 특정 프로필입니다.

`--region` *TEXT*

배포할 AWS 지역입니다. 예를 들어 us-east-1입니다.

`--save-params`

명령줄에서 AWS SAM 구성 파일에 제공한 매개변수를 저장합니다.

`--stack-name` *TEXT*

리소스가 속한 CloudFormation 스택의 이름입니다.

`--start-time, -s` *TEXT*

이 시점부터 로그를 가져옵니다. 시간은 '5분 전', '어제'와 같은 상댓값이거나 '2018-01-01 10:10:10'과 같은 형식이 지정된 타임스탬프일 수 있습니다. 기본값은 '10분 전'입니다.

`--tail, -t`

로그 출력을 추적합니다. 이렇게 하면 종료 시간 인수를 무시하고 로그가 제공되는 대로 계속 가져옵니다.

예시

귀하의 함수가 CloudFormation 스택의 일부인 경우 스택 이름을 지정할 때 함수의 논리적 ID를 사용하여 로그를 가져올 수 있습니다.

```
$ sam logs -n HelloWorldFunction --stack-name myStack
```

-s(--start-time) 및 -e(--endtime) 옵션을 사용하여 특정 시간 범위의 로그를 볼 수 있습니다.

```
$ sam logs -n HelloWorldFunction --stack-name myStack -s '10min ago' -e '2min ago'
```

새 로그가 도착할 때까지 기다렸다가 로그가 도착하는 대로 확인하는 `--tail` 옵션을 추가할 수도 있습니다.

```
$ sam logs -n HelloWorldFunction --stack-name myStack --tail
```

`--filter` 옵션을 사용하면 로그 이벤트의 용어, 구문 또는 값과 일치하는 로그를 빠르게 찾을 수 있습니다.

```
$ sam logs -n HelloWorldFunction --stack-name myStack --filter "error"
```

하위 스택에 있는 리소스의 로그를 볼 수 있습니다.

```
$ sam logs --stack-name myStack -n childStack/HelloWorldFunction
```

귀하의 애플리케이션에서 지원되는 모든 리소스의 테일 로그입니다.

```
$ sam logs --stack-name sam-app --tail
```

애플리케이션의 특정 Lambda 함수 및 API 게이트웨이 API에 대한 로그를 가져옵니다.

```
$ sam logs --stack-name sam-app --name HelloWorldFunction --name HelloWorldRestApi
```

애플리케이션에서 지원되는 모든 리소스에 대한 로그를 가져오고, 추가로 지정된 로그 그룹에서도 로그를 가져옵니다.

```
$ sam logs --cw-log-group /aws/lambda/myfunction-123 --cw-log-group /aws/lambda/
myfunction-456
```

sam package

AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI)는 AWS SAM 애플리케이션을 패키징합니다.

이 명령은 코드 및 종속 항목 `.zip` 파일을 생성하고, 해당 파일을 Amazon Simple Storage Service(Amazon S3)에 업로드합니다. AWS SAM은 Amazon S3에 저장된 모든 파일의 암호화를 가능하게 합니다. 그런 다음 로컬 아티팩트에 대한 참조를 명령이 아티팩트를 업로드한 Amazon S3 위치로 변경하여 AWS SAM 템플릿의 사본을 반환합니다.

이 명령을 사용할 때 기본적으로 AWS SAM CLI는 현재 작업 디렉터리를 프로젝트의 루트 디렉터리라고 가정합니다. AWS SAM CLI는 우선 sam build 명령을 사용하여 구축되고 `.aws-sam` 하위 폴더에 있으며 이름이 `template.yaml`로 지정된 템플릿 파일을 찾으려고 합니다. 그런 다음 AWS SAM CLI는 현재 작업 디렉터리에서 `template.yaml` 혹은 `template.yml`라는 이름의 템플릿 파일을 찾으려고 합니다. `--template` 옵션을 지정하면 AWS SAM CLI의 기본 동작이 대체되고 해당 템플릿과 해당 AWS SAM 템플릿이 가리키는 로컬 리소스만 패키징됩니다.

> **Note**
>
> 이제 sam deploy이 `sam package` 기능을 묵시적으로 수행합니다. sam deploy 명령을 사용하여 애플리케이션을 패키징하고 배포할 수 있습니다.

사용법

```
$ sam package <arguments> <options>
```

인수

리소스 ID

패키징할 Lambda 함수의 ID입니다.

이 인수는 선택 사항입니다. 애플리케이션에 단일 Lambda 함수가 포함된 경우 AWS SAM CLI는 이를 패키징합니다. 애플리케이션에 여러 함수가 포함된 경우 함수 ID를 제공하여 단일 함수를 패키징하십시오.

유효한 값: 리소스의 논리적 ID 또는 리소스 ARN입니다.

옵션

`--config-env TEXT`

사용할 구성 파일의 기본 매개변수 값을 지정하는 환경 이름입니다. 기본값은 “기본값”입니다. 구성 파일에 대한 자세한 내용은 AWS SAMCLI구성 파일 섹션을 참조하세요.

`--config-file PATH`

사용할 기본 매개변수 값이 들어 있는 구성 파일의 경로 및 파일 이름입니다. 기본값은 프로젝트 디렉터리의 루트에 있는 “samconfig.toml”입니다. 구성 파일에 대한 자세한 내용은 AWS SAMCLI구성 파일 섹션을 참조하세요.

`--debug`

디버그 로깅을 켜서 AWS SAM CLI 에 의해 생성된 디버그 메시지를 인쇄하고 타임스탬프를 표시합니다.

`--force-upload`

Amazon S3 버킷의 기존 파일을 재정의합니다. Amazon S3 버킷의 기존 아티팩트와 일치하더라도 아티팩트를 업로드하려면 이 플래그를 지정하십시오.

`--help`

이 메시지를 표시한 후 종료합니다.

`--image-repository TEXT`

이 명령으로 함수 이미지를 업로드하는 Amazon Elastic Container Registry (Amazon ECR) 리포지토리의 URI. `Image` 패키지 유형으로 선언된 함수에 필요합니다.

`--kms-key-id TEXT`

Amazon S3 버킷에 저장된 아티팩트를 암호화하는 데 사용되는 AWS Key Management Service(AWS KMS) 키의 ID입니다. 이 옵션을 지정하지 않을 경우 AWS SAM은 Amazon S3 관리형 암호화 키를 사용합니다.

`--metadata`

(선택 사항) 템플릿에서 참조되는 모든 아티팩트에 첨부할 메타데이터 맵입니다.

`--no-progressbar`

Amazon S3에 아티팩트를 업로드할 때 진행률 표시줄을 표시하지 마십시오.

`--output-template-file` *PATH*

명령이 패키지 템플릿을 작성하는 파일의 경로입니다. 경로를 지정하지 않으면 이 명령은 템플릿을 표준 출력에 기록합니다.

`--profile` *TEXT*

AWS 보안 인증 정보를 가져오는 보안 인증 파일의 특정 프로필입니다.

`--region` *TEXT*

배포할 AWS 지역입니다. 예를 들어 us-east-1입니다.

`--resolve-s3`

패키징에 사용할 Amazon S3 버킷을 자동으로 생성합니다. `--s3-bucket` 및 `--resolve-s3` 옵션을 모두 지정하면 오류가 발생합니다.

`--s3-bucket` *TEXT*

이 명령으로 아티팩트를 업로드하는 Amazon S3 버킷의 이름입니다. 아티팩트가 51,200바이트를 초과하는 경우 `--s3-bucket` 또는 `--resolve-s3` 옵션이 필요합니다. `--s3-bucket` 및 `--resolve-s3` 옵션을 모두 지정하면 오류가 발생합니다.

`--s3-prefix` *TEXT*

Amazon S3 버킷에 업로드되는 객체 이름에 부가된 접두사입니다. 접두사 이름은 Amazon S3 버킷의 경로 이름(폴더 이름)입니다. 이는 `Zip` 패키지 유형으로 선언된 함수에만 적용됩니다.

`--save-params`

명령줄에서 AWS SAM 구성 파일에 제공한 매개변수를 저장합니다.

`--signing-profiles` *LIST*

(선택 사항) 배포 패키지에 서명하는 데 사용할 서명 프로필 목록입니다. 이 매개변수는 키-값 쌍의 목록을 사용합니다. 여기서 키는 서명할 함수 또는 레이어의 이름이고 값은 서명 프로필이며 선택적 프로필 소유자는 `:`로 한

정됩니다. 예를 들어 `FunctionNameToSign=SigningProfileName1 LayerNameToSign=SigningProfileName2:SigningProfileOwner`입니다.

`--template-file, --template, -t PATH`

AWS SAM 템플릿이 위치한 경로 및 파일 이름입니다.

> **Note**
>
> 이 옵션을 지정하는 경우 AWS SAM은 템플릿과 템플릿이 가리키는 로컬 리소스만 패키징합니다.

`--use-json`

CloudFormation 템플릿의 JSON을 출력합니다. 기본 사항으로 YAML이 사용됩니다.

예시

다음 예제에서는 Lambda 함수 및 CodeDeploy 애플리케이션을 위한 아티팩트를 생성하고 패키징합니다. 아티팩트는 Amazon S3 버킷에 업로드됩니다. 명령의 출력은 `package.yml`이라는 새 파일입니다.

```
$ sam package \
  --template-file template.yml \
  --output-template-file package.yml \
  --s3-bucket amzn-s3-demo-bucket
```

sam pipeline bootstrap

이 페이지는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) `sam local pipeline bootstrap` 하위 명령에 대한 참조 정보를 제공합니다.

AWS SAM CLI에 대한 소개는 [란 무엇입니까 AWS SAMCLI?](#) 섹션을 참조하세요.

`sam pipeline bootstrap` 하위 명령은 CI/CD 시스템에 연결하는 데 필요한 AWS 인프라 리소스를 생성합니다. sam pipeline init 명령을 실행하기 전에 파이프라인에서의 각 배포 단계에 대해 이 조치를 실행해야 합니다.

이 하위 명령은 다음과 같은 AWS 인프라 리소스를 설정합니다.

- 다음을 통해 파이프라인 권한을 구성하는 옵션:
  - CI/CD 시스템과 공유할 액세스 키 ID 및 액세스 키 보안 인증 정보를 가진 파이프라인 IAM 사용자입니다.

    > **Note**
    >
    > 정기적으로 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용자 가이드의 장기 보안 인증 정보가 필요한 사용 사례의 경우 정기적으로 액세스 키 교체를 참조하세요.

  - OIDC를 통해 CI/CD 플랫폼을 지원합니다. AWS SAM 파이프라인이 있는 OIDC 사용법에 대한 소개는 AWS SAM 파이프라인과 함께 OIDC 인증을 사용하는 방법 섹션을 참조하세요.
- CloudFormation가 CloudFormation 애플리케이션을 배포하기 위해 맡은 AWS SAM 실행 IAM 역할입니다.
- AWS SAM 아티팩트를 보관하는 Amazon S3 버킷입니다.
- 선택적으로 컨테이너 이미지 Lambda 배포 패키지를 보관할 Amazon ECR 이미지 리포지토리 (`Image` 패키지 유형의 리소스가 있는 경우)입니다.

사용법

```
$ sam pipeline bootstrap <options>
```

옵션

`--bitbucket-repo-uuid` *TEXT*

Bitbucket 리포지토리의 UUID입니다. 이 옵션은 Bitbucket OIDC를 권한에 따라 사용하는 경우에만 해당합니다.

> **Note**
>
> 이 값은 https://bitbucket.org/*workspace*/*repository*/admin/addon/admin/pipelines/openid-connect에서 찾을 수 있습니다

`--bucket` *TEXT*

AWS SAM 아티팩트를 보관하는 Amazon S3 버킷의 ARN입니다.

`--cicd-provider` *TEXT*

AWS SAM 파이프라인용 CI/CD 플랫폼입니다.

`--cloudformation-execution-role` *TEXT*

애플리케이션 스택을 배포할 때 CloudFormation이 떠맡는 IAM 역할의 ARN입니다. 자신의 역할을 사용하는 경우에만 입력합니다. 그렇지 않으면 명령으로 새 역할이 생성됩니다.

`--config-env` *TEXT*

사용할 구성 파일의 기본 매개변수 값을 지정하는 환경 이름입니다. 기본값은 **`default`**입니다. 구성 파일에 대한 자세한 내용은 AWS SAMCLI구성 파일 섹션을 참조하세요.

`--config-file` *PATH*

사용할 기본 매개변수 값이 들어있는 구성 파일의 경로 및 파일 이름입니다. 기본값은 프로젝트 디렉터리의 루트 내 `samconfig.toml`입니다. 구성 파일에 대한 자세한 내용은 AWS SAMCLI구성 파일 섹션을 참조하세요.

`--confirm-changeset | --no-confirm-changeset`

리소스 배포를 확인하라는 메시지를 표시합니다.

`--create-image-repository | --no-create-image-repository`

Amazon ECR 이미지 리포지토리가 제공되지 않은 경우 이를 생성할지 여부를 지정하십시오. Amazon ECR 리포지토리는 Lambda 함수 또는 패키지 유형이 `Image`인 레이어의 컨테이너 이미지를 보관합니다. 기본값은 `--no-create-image-repository`입니다.

`--debug`

디버그 로깅을 켜서 AWS SAM CLI가 생성한 디버그 메시지를 인쇄하고 타임스탬프를 표시합니다.

`--deployment-branch` *TEXT*

배포가 시작될 브랜치의 이름입니다. 이 옵션은 권한에 GitHub Actions OIDC를 사용하는 경우에만 해당합니다.

`--github-org` *TEXT*

리포지토리가 속한 GitHub 조직입니다. 조직이 없는 경우 리포지토리 소유자의 사용자 이름을 입력합니다. 이 옵션은 권한에 GitHub Actions OIDC를 사용하는 경우에만 해당합니다.

`--github-repo` *TEXT*

배포가 시작될 GitHub 리포지토리의 이름입니다. 이 옵션은 권한에 GitHub Actions OIDC를 사용하는 경우에만 해당합니다.

`--gitlab-group` *TEXT*

리포지토리가 속한 GitLab 그룹입니다. 이 옵션은 권한에 GitLab OIDC를 사용하는 경우에만 해당합니다.

`--gitlab-project` *TEXT*

GitLab 프로젝트 이름입니다. 이 옵션은 권한에 GitLab OIDC를 사용하는 경우에만 해당합니다.

`--help, -h`

이 메시지를 표시한 후 종료합니다.

`--image-repository` *TEXT*

Lambda 함수 또는 패키지 유형이 `Image`인 레이어의 컨테이너 이미지를 보관하는 Amazon ECR 이미지 리포지토리의 ARN입니다. 제공된 경우 `--create-image-repository` 옵션은 무시됩니다. 제공되지 않은 경우 `--create-image-repository`이 지정되면 명령이 하나를 생성합니다.

`--interactive | --no-interactive`

부트스트랩 매개변수에 대한 대화식 프롬프트를 비활성화하고 필수 매개변수가 누락되면 실패합니다. 기본값은 `--interactive`입니다. 이 명령에 관하여 `--stage`이 유일한 필수 매개변수입니다.

> **Note**
>
> `--no-interactive`이 `--use-oidc-provider`와 함께 지정된 경우, OIDC 공급자의 모든 필수 매개변수가 포함되어야 합니다.

`--oidc-client-id` *TEXT*

OIDC 제공업체와 함께 사용하도록 구성된 클라이언트 ID입니다.

`--oidc-provider` *[github-actions | gitlab | bitbucket-pipelines]*

OIDC 권한에 사용될 CI/CD 제공업체의 명칭입니다. GitLab, GitHub, Bitbucket이 지원됩니다.

`--oidc-provider-url` *TEXT*

OIDC ID 제공업체의 URL입니다. 값은 **`https://`**로 시작해야 합니다.

`--permissions-provider` *[oidc | iam]*

파이프라인 실행 역할을 맡을 권한 제공업체를 선택합니다. 기본값은 **`iam`**입니다.

`--pipeline-execution-role` *TEXT*

파이프라인 사용자가 이 단계에서 운영하도록 위임하는 IAM 역할의 ARN입니다. 자신의 역할을 사용하는 경우에만 입력합니다. 지정하지 않은 경우 이 명령은 새 역할을 생성합니다.

`--pipeline-user` *TEXT*

액세스 키 ID와 비밀 액세스 키를 CI/CD 시스템과 공유하는 IAM 사용자의 Amazon 리소스 이름(ARN)입니다. 이 IAM 사용자에게 해당 AWS 계정에 액세스할 수 있는 권한을 부여하는 데 사용됩니다. 제공하지 않는 경우 명령은 액세스 키 ID 및 보안 액세스 키 보안 인증 정보와 함께 IAM 사용자를 생성합니다.

`--profile` *TEXT*

AWS 보안 인증 정보를 가져오는 보안 인증 파일의 특정 프로필입니다.

`--region` *TEXT*

배포할 AWS 지역입니다. 예를 들어 `us-east-1`입니다.

`--save-params`

명령줄에서 AWS SAM 구성 파일에 제공한 매개변수를 저장합니다.

`--stage` *TEXT*

해당하는 배포 단계의 이름입니다. 생성된 AWS 인프라 리소스의 접미사로 사용됩니다.

문제 해결

오류: 필수 매개변수 누락

`--no-interactive`이 `--use-oidc-provider`와 함께 지정되었지만 필수 매개변수가 제공되지 않은 경우 이 오류 메시지가 누락된 매개변수에 대한 설명과 함께 표시됩니다.

예시

다음 예제에서는 CI/CD 시스템을 생성하는 데 필요한 AWS 리소스를 생성한 다음 디버그 로깅을 활성화하고 AWS SAM CLI에서 생성된 디버그 메시지를 출력합니다. 또한 `s3.json` 이벤트를 통해 로컬에서 Lambda 함수를 호출하여 로컬 테스트를 위해 생성된 이벤트를 사용합니다.

```
$ sam pipeline bootstrap --debug
```

sam pipeline init

이 페이지는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) `sam pipeline init` 하위 명령에 대한 참조 정보를 제공합니다.

AWS SAM CLI에 대한 소개는 [란 무엇입니까 AWS SAMCLI?](#) 섹션을 참조하세요.

`sam pipeline init`하위 명령은 귀하의 CI/CD 시스템이 AWS SAM를 사용하여 서버리스 애플리케이션을 배포하는 데 사용할 수 있는 파이프라인 구성 파일을 생성합니다.

sam pipeline init을 사용하기 전에 귀하의 파이프라인의 각 단계에 필요한 리소스를 부트스트랩해야 합니다. sam pipeline init --bootstrap를 실행하여 설정 및 구성 파일 생성 프로세스를 안내받거나 이전에 sam pipeline bootstrap 명령으로 만든 리소스를 참조하여 이 작업을 수행할 수 있습니다.

사용법

```
$ sam pipeline init <options>
```

옵션

`--bootstrap`

필요한 AWS 인프라 리소스를 만드는 과정을 사용자에게 안내하는 대화형 모드를 활성화합니다.

`--config-env` *TEXT*

사용할 구성 파일의 기본 매개변수 값을 지정하는 환경 이름입니다. 기본값은 `default`입니다. 구성 파일에 대한 자세한 내용은 [AWS SAMCLI구성 파일](#) 섹션을 참조하세요.

`--config-file` *TEXT*

사용할 기본 매개변수 값이 들어 있는 구성 파일의 경로 및 파일 이름입니다. 기본값은 프로젝트 디렉터리의 루트에 있는 `samconfig.toml`입니다. 구성 파일에 대한 자세한 내용은 [AWS SAMCLI구성 파일](#) 섹션을 참조하세요.

`--debug`

디버그 로깅을 켜서 AWS SAM CLI가 생성한 디버그 메시지를 인쇄하고 타임스탬프를 표시합니다.

`--help, -h`

이 메시지를 표시한 후 종료합니다.

`--save-params`

명령줄에서 AWS SAM 구성 파일에 제공한 매개변수를 저장합니다.

예시

다음 예제에서는 `--bootstrap` 옵션을 사용하여 필요한 AWS 인프라 리소스 생성을 안내하는 대화형 모드를 시작하는 방법을 보여줍니다.

```
$ sam pipeline init --bootstrap
```

sam publish

이 페이지는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) `sam publish` 명령에 대한 참조 정보를 제공합니다.

AWS SAM CLI에 대한 소개는 [란 무엇입니까 AWS SAMCLI?](#) 섹션을 참조하세요.

이 `sam publish` 명령은 AWS SAM 어플리케이션을 AWS Serverless Application Repository에 게시합니다. 이 명령은 패키징된 AWS SAM 템플릿을 사용하여 지정된 AWS 지역에 어플리케이션을 게시합니다.

이 `sam publish` 명령은 AWS SAM 템플릿에 게시에 필요한 어플리케이션 메타데이터가 포함된 `Metadata` 섹션이 포함될 것으로 예상합니다. `Metadata` 단원에서 `LicenseUrl` 및 `ReadmeUrl` 속성은 로컬 파일이 아니라 Amazon Simple Storage Service(S3) 버킷을 참조해야 합니다. AWS SAM 템플릿의 `Metadata` 단원에 대한 보다 상세한 내용은 [AWS SAMCLI를 사용하여 애플리케이션 게시](#) 섹션을 참조하세요.

기본 사항으로서, `sam publish`은 어플리케이션은 사적으로 생성합니다. 다른 AWS 계정에게 귀하의 어플리케이션을 열람 및 배포할 수 있도록 허용하려면 먼저 이를 공유해야 합니다. 어플리케이션 공유에 대한 자세한 내용은 AWS Serverless Application Repository개발자 안내서의 [AWS Serverless Application Repository리소스 기반 정책 예제](#)를 참조하세요.

> **Note**
>
> 현재 `sam publish`는 로컬로 지정된 중첩된 어플리케이션을 게시하는 것을 지원하지 않습니다. 응용 프로그램에 중첩된 어플리케이션이 포함된 경우 상위 어플리케이션을 게시하기 전에 해당 응용 프로그램을 AWS Serverless Application Repository에 별도로 게시해야 합니다.

사용법

```
$ sam publish <options>
```

옵션

`--config-env TEXT`

사용할 구성 파일의 기본 매개변수 값을 지정하는 환경 이름입니다. 기본값은 "기본값"입니다. 구성 파일에 대한 자세한 내용은 AWS SAMCLI구성 파일 섹션을 참조하세요.

`--config-file PATH`

사용할 기본 매개변수 값이 들어 있는 구성 파일의 경로 및 파일 이름입니다. 기본값은 프로젝트 디렉터리의 루트에 있는 "`samconfig.toml`"입니다. 구성 파일에 대한 자세한 내용은 AWS SAMCLI 구성 파일 섹션을 참조하세요.

`--debug`

디버그 로깅을 켜서 AWS SAM CLI가 생성한 디버그 메시지를 인쇄하고 타임스탬프를 표시합니다.

`--help`

이 메시지를 표시한 후 종료합니다.

`--profile TEXT`

AWS 보안 인증 정보를 가져오는 보안 인증 파일의 특정 프로필입니다.

`--region TEXT`

배포할 AWS 지역입니다. 예를 들어 us-east-1입니다.

`--save-params`

명령줄에서 AWS SAM 구성 파일에 제공한 매개변수를 저장합니다.

`--semantic-version TEXT`

(선택 사항) 이 옵션을 사용하면 템플릿 파일 `Metadata` 섹션의 `SemanticVersion` 속성을 재정의하는 귀하의 어플리케이션 시맨틱 버전을 제공할 수 있습니다. 시맨틱 버전 관리에 대한 자세한 내용은 시맨틱 버전 관리 사양을 참조하세요.

`--template, -t PATH`

AWS SAM템플릿 파일`[default: template.[yaml|yml]]`의 경로.

예시

어플리케이션의 게시:

```
$ sam publish --template packaged.yaml --region us-east-1
```

sam remote callback

이 페이지에서는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAMCLI) `sam remote callback` 명령에 대한 참조 정보를 제공합니다.

에 대한 소개는 단원을 AWS SAMCLI참조하십시오. [란 무엇입니까 AWS SAMCLI?]

`sam remote callback` 명령을 사용하면 AWS Lambda에서 실행되는 내구성이 뛰어난 원격 함수 실행에 콜백을 보낼 수 있습니다.

> **Note**
> 이러한 명령은 AWS 자격 증명에 대한 액세스 권한이 필요합니다.

사용법

```
$ sam remote callback <subcommand> <options>
```

옵션

`--help, -h`

이 메시지를 표시한 후 종료합니다.

하위 명령

`succeed`

원격 내구성 함수 실행에 콜백 성공을 전송합니다. 자세한 내용은 [sam remote callback succeed] 단원을 참조하십시오.

`fail`

원격 내구성 함수 실행에 콜백 실패를 보냅니다. 자세한 내용은 sam remote callback fail 단원을 참조하십시오.

`heartbeat`

원격 내구성 함수 실행으로 콜백 하트비트를 전송합니다. 자세한 내용은 sam remote callback heartbeat 단원을 참조하십시오.

sam remote callback succeed

원격 내구성 함수 실행에 콜백 성공을 전송합니다.

> **Note**
> 이 명령은 AWS 자격 증명에 대한 액세스 권한이 필요합니다.

사용법

```
$ sam remote callback succeed CALLBACK_ID [OPTIONS]
```

필수 인수

`CALLBACK_ID`

성공 응답을 보낼 콜백의 고유 식별자입니다.

옵션

`-r, --result TEXT`

성공 결과 문자열 페이로드입니다.

`--region TEXT`

서비스의 AWS 리전을 설정합니다(예: `us-east-1`).

`--profile TEXT`

자격 증명 파일에서 특정 프로필을 선택하여 AWS 자격 증명을 가져옵니다.

`--config-env TEXT`

구성 파일에서 기본 파라미터 값을 지정하는 환경 이름입니다. 기본값: `default`

`--config-file TEXT`

기본 파라미터 값이 포함된 구성 파일입니다. 기본값: `samconfig.toml`

`--save-params`

명령줄을 통해 제공된 파라미터를 구성 파일에 저장합니다.

`--beta-features / --no-beta-features`

베타 기능을 활성화/비활성화합니다.

`--debug`

디버그 로깅을 켜서 AWS SAM CLI에서 생성된 디버그 메시지를 인쇄하고 타임스탬프를 표시합니다.

`-h, --help`

이 메시지를 표시한 후 종료합니다.

예제

결과 없이 성공 콜백 전송:

```
$ sam remote callback succeed my-callback-id
```

결과와 함께 성공 콜백 전송:

```
$ sam remote callback succeed my-callback-id --result 'Task completed successfully'
```

sam remote callback fail

원격 내구성 함수 실행에 콜백 실패를 보냅니다.

> **Note**
>
> 이 명령은 AWS 자격 증명에 대한 액세스 권한이 필요합니다.

사용법

```
$ sam remote callback fail CALLBACK_ID [OPTIONS]
```

필수 인수

`CALLBACK_ID`

실패 응답을 보낼 콜백의 고유 식별자입니다.

옵션

`--error-data TEXT`

추가 오류 데이터.

`--stack-trace TEXT`

트레이스 항목을 스택합니다. 이 옵션은 여러 번 지정할 수 있습니다.

`--error-type TEXT`

오류의 유형입니다.

`--error-message TEXT`

자세한 오류 메시지입니다.

`--region TEXT`

서비스의 AWS 리전을 설정합니다(예: `us-east-1`).

`--profile TEXT`

자격 증명 파일에서 특정 프로필을 선택하여 AWS 자격 증명을 가져옵니다.

`--config-env TEXT`

구성 파일에서 기본 파라미터 값을 지정하는 환경 이름입니다. 기본값: `default`

`--config-file TEXT`

기본 파라미터 값이 포함된 구성 파일입니다. 기본값: `samconfig.toml`

`--save-params`

명령줄을 통해 제공된 파라미터를 구성 파일에 저장합니다.

`--beta-features / --no-beta-features`

베타 기능을 활성화/비활성화합니다.

`--debug`

디버그 로깅을 켜서 AWS SAM CLI에서 생성된 디버그 메시지를 인쇄하고 타임스탬프를 표시합니다.

`-h, --help`

이 메시지를 표시한 후 종료합니다.

예제

파라미터 없이 실패 콜백 전송:

```
$ sam remote callback fail my-callback-id
```

오류 메시지와 함께 실패 콜백 전송:

```
$ sam remote callback fail my-callback-id --error-message 'Task failed'
```

모든 파라미터와 함께 실패 콜백 전송:

```
$ sam remote callback fail my-callback-id --error-message 'Task failed' --error-type
 'ValidationError' --stack-trace 'at line 42' --error-data '{"code": 500}'
```

sam remote callback heartbeat

원격 내구성 함수 실행으로 콜백 하트비트를 전송합니다.

> **Note**
> 이 명령은 AWS 자격 증명에 대한 액세스 권한이 필요합니다.

사용법

```
$ sam remote callback heartbeat CALLBACK_ID [OPTIONS]
```

필수 인수

`CALLBACK_ID`

하트비트를 보낼 콜백의 고유 식별자입니다.

옵션

`--region TEXT`

서비스의 AWS 리전을 설정합니다(예: `us-east-1`).

`--profile TEXT`

자격 증명 파일에서 특정 프로필을 선택하여 AWS 자격 증명을 가져옵니다.

`--config-env TEXT`

구성 파일에서 기본 파라미터 값을 지정하는 환경 이름입니다. 기본값: `default`

`--config-file TEXT`

기본 파라미터 값이 포함된 구성 파일입니다. 기본값: `samconfig.toml`

`--save-params`

명령줄을 통해 제공된 파라미터를 구성 파일에 저장합니다.

`--beta-features / --no-beta-features`

베타 기능을 활성화/비활성화합니다.

`--debug`

디버그 로깅을 켜서 AWS SAM CLI에서 생성된 디버그 메시지를 인쇄하고 타임스탬프를 표시합니다.

`-h, --help`

이 메시지를 표시한 후 종료합니다.

예제

하트비트 콜백 전송:

```
$ sam remote callback heartbeat my-callback-id
```

sam remote execution

이 페이지에서는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAMCLI) `sam remote execution` 명령에 대한 참조 정보를 제공합니다.

에 대한 소개는 단원을 AWS SAMCLI참조하십시오. [란 무엇입니까 AWS SAMCLI?]

`sam remote execution` 명령을 사용하면 AWS Lambda에서 실행되는 내구성이 뛰어난 원격 함수 실행을 관리하고 검사할 수 있습니다.

> **Note**
> 이러한 명령은 AWS 자격 증명에 대한 액세스 권한이 필요합니다.

사용법

```
$ sam remote execution <subcommand> <options>
```

옵션

`--help, -h`

이 메시지를 표시한 후 종료합니다.

하위 명령

`get`

내구성 있는 실행에 대한 세부 정보를 가져옵니다. 자세한 내용은 [sam remote execution get] 단원을 참조하십시오.

`history`

내구성 있는 함수 실행의 실행 기록을 가져옵니다. 자세한 내용은 [sam remote execution history] 단원을 참조하십시오.

`stop`

내구성 있는 함수 실행을 중지합니다. 자세한 내용은 [sam remote execution stop] 단원을 참조하십시오.

sam remote execution get

AWS Lambda에서 특정 내구성 함수 실행의 세부 정보를 검색합니다.

실행 ARN은 간접 호출에서 반환하거나 실행 로그에 있는 형식이어야 합니다.

> **Note**
>
> 이 명령은 AWS 자격 증명에 대한 액세스 권한이 필요합니다.

사용법

```
$ sam remote execution get DURABLE_EXECUTION_ARN [OPTIONS]
```

필수 인수

`DURABLE_EXECUTION_ARN`

세부 정보를 검색할 내구성 함수 실행의 Amazon 리소스 이름(ARN)입니다.

옵션

`--format [summary|json]`

출력 형식. 기본값: `summary`

`--region TEXT`

서비스의 AWS 리전을 설정합니다(예: `us-east-1`).

`--profile TEXT`

자격 증명 파일에서 특정 프로필을 선택하여 AWS 자격 증명을 가져옵니다.

`--config-env TEXT`

구성 파일에서 기본 파라미터 값을 지정하는 환경 이름입니다. 기본값: `default`

`--config-file TEXT`

기본 파라미터 값이 포함된 구성 파일입니다. 기본값: `samconfig.toml`

`--save-params`

명령줄을 통해 제공된 파라미터를 구성 파일에 저장합니다.

`--beta-features / --no-beta-features`

베타 기능을 활성화/비활성화합니다.

`--debug`

디버그 로깅을 켜서 AWS SAM CLI에서 생성된 디버그 메시지를 인쇄하고 타임스탬프를 표시합니다.

`-h, --help`

이 메시지를 표시한 후 종료합니다.

예제

기본 요약 형식으로 실행 세부 정보를 가져옵니다.

```
$ sam remote execution get 'arn:aws:lambda:us-east-1:123456789012:function:my-function:
$LATEST/durable-execution/c63eec67-3415-4eb4-a495-116aa3a86278/1d454231-a3ad-3694-aa03-
c917c175db55'
```

JSON 형식으로 실행 세부 정보를 가져옵니다.

```
$ sam remote execution get 'arn:aws:lambda:us-east-1:123456789012:function:my-function:
$LATEST/durable-execution/c63eec67-3415-4eb4-a495-116aa3a86278/1d454231-a3ad-3694-aa03-
c917c175db55' --format json
```

> **Note**
>
> 셸 명령을 사용할 때와 같은 실행 ARN의 제어 문자$가 제대로 이스케이프 처리되었는지 확인해야 합니다.

sam remote execution history

AWS Lambda에서 특정 내구성 함수 실행의 실행 기록을 검색합니다.

실행 ARN은 StartDurableExecution에서 반환하거나 실행 로그에 있는 형식이어야 합니다.

> **Note**
> 이 명령은 AWS 자격 증명에 대한 액세스 권한이 필요합니다.

사용법

```
$ sam remote execution history DURABLE_EXECUTION_ARN [OPTIONS]
```

필수 인수

`DURABLE_EXECUTION_ARN`

기록을 검색할 내구성 있는 함수 실행의 Amazon 리소스 이름(ARN)입니다.

옵션

`--format [table|json]`

출력 형식. 기본값: `table`

`--region TEXT`

서비스의 AWS 리전을 설정합니다(예: `us-east-1`).

`--profile TEXT`

자격 증명 파일에서 특정 프로필을 선택하여 AWS 자격 증명을 가져옵니다.

`--config-env TEXT`

구성 파일에서 기본 파라미터 값을 지정하는 환경 이름입니다. 기본값: `default`

`--config-file TEXT`

기본 파라미터 값이 포함된 구성 파일입니다. 기본값: `samconfig.toml`

`--save-params`

명령줄을 통해 제공된 파라미터를 구성 파일에 저장합니다.

`--beta-features / --no-beta-features`

베타 기능을 활성화/비활성화합니다.

`--debug`

디버그 로깅을 켜서 AWS SAM CLI에서 생성된 디버그 메시지를 인쇄하고 타임스탬프를 표시합니다.

`-h, --help`

이 메시지를 표시한 후 종료합니다.

예제

기본 테이블 형식으로 실행 기록을 가져옵니다.

```
$ sam remote execution history arn:aws:lambda:us-east-1:123456789012:function:MyFunction:$LATEST/durable-execution/my-execution-name/my-execution-id
```

JSON 형식으로 실행 기록 가져오기:

```
$ sam remote execution history arn:aws:lambda:us-east-1:123456789012:function:MyFunction:$LATEST/durable-execution/my-execution-name/my-execution-id --format json
```

sam remote execution stop

AWS Lambda에서 실행 중인 내구성 함수 실행을 중지합니다.

실행 ARN은 StartDurableExecution에서 반환하거나 실행 로그에 있는 형식이어야 합니다.

> **Note**
> 이 명령은 AWS 자격 증명에 대한 액세스 권한이 필요합니다.

사용법

```
$ sam remote execution stop DURABLE_EXECUTION_ARN [OPTIONS]
```

필수 인수

`DURABLE_EXECUTION_ARN`

중지할 내구성 함수 실행의 Amazon 리소스 이름(ARN)입니다.

옵션

`--error-message TEXT`

중지된 실행과 연결할 오류 메시지입니다.

`--error-type TEXT`

중지된 실행과 연결할 오류 유형입니다.

`--error-data TEXT`

중지된 실행과 연결할 오류 데이터입니다.

`--stack-trace TEXT`

트레이스 항목을 스택합니다. 이 옵션은 여러 번 지정할 수 있습니다.

`--region TEXT`

서비스의 AWS 리전을 설정합니다(예: `us-east-1`).

`--profile TEXT`

자격 증명 파일에서 특정 프로필을 선택하여 AWS 자격 증명을 가져옵니다.

`--config-env TEXT`

구성 파일에서 기본 파라미터 값을 지정하는 환경 이름입니다. 기본값: `default`

`--config-file TEXT`

기본 파라미터 값이 포함된 구성 파일입니다. 기본값: `samconfig.toml`

`--save-params`

명령줄을 통해 제공된 파라미터를 구성 파일에 저장합니다.

`--beta-features / --no-beta-features`

베타 기능을 활성화/비활성화합니다.

`--debug`

디버그 로깅을 켜서 AWS SAM CLI에서 생성된 디버그 메시지를 인쇄하고 타임스탬프를 표시합니다.

`-h, --help`

이 메시지를 표시한 후 종료합니다.

예제

오류 세부 정보 없이 실행 중지:

```
$ sam remote execution stop arn:aws:lambda:us-east-1:123456789012:function:MyFunction:
$LATEST/durable-execution/my-execution-name/my-execution-id
```

오류 메시지 및 유형으로 실행 중지:

```
$ sam remote execution stop arn:aws:lambda:us-east-1:123456789012:function:MyFunction:
$LATEST/durable-execution/my-execution-name/my-execution-id --error-message "Execution
 cancelled" --error-type "UserCancellation"
```

전체 오류 세부 정보 및 스택 추적으로 실행을 중지합니다.

```
$ sam remote execution stop arn:aws:lambda:us-east-1:123456789012:function:MyFunction:
$LATEST/durable-execution/my-execution-name/my-execution-id --error-message "Task
 failed" --error-type "TaskFailure" --error-data '{"reason":"timeout"}' --stack-trace
 "at function1()" --stack-trace "at function2()"
```

sam remote invoke

이 페이지에서는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAMCLI) `sam remote invoke` 명령에 대한 참조 정보를 제공합니다.

- 에 대한 소개는 단원을 AWS SAMCLI참조하십시오. 란 무엇입니까 AWS SAMCLI?
- `sam remote invoke` 명령 사용에 대한 설명서는 섹션을 참조하세요 AWS SAMCLIsam remote invoke를 사용하여 클라우드에서 테스트 소개.

이 `sam remote invoke` 명령은 AWS 클라우드에 속한 지원되는 리소스를 간접 호출합니다.

사용법

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

인수

리소스 ID

호출할 지원 리소스의 ID.

이 인수는 다음 값으로 지정할 수 있습니다.

- Amazon 리소스 이름(ARN) – 리소스의 ARN입니다.

> **Tip**
>
> 귀하의 리소스의 ARN을 구하는데 `sam list stack-outputs --stack-name <stack-name>`을 사용합니다.

- 논리적 ID - 리소스의 논리적 ID 또한 `--stack-name` 옵션을 사용하여 AWS CloudFormation 스택 이름을 제공해야 합니다.
- 물리적 ID – 리소스의 물리적 ID. 이 ID는를 사용하여 리소스를 배포할 때 생성됩니다 CloudFormation.

> **Tip**
>
> 귀하의 리소스의 물리적 ID를 얻는데 `sam list resources --stack-name <stack-name>`을 사용합니다.

ARN 또는 물리적 ID를 제공하는 경우:

ARN 또는 물리적 ID를 제공하는 경우 스택 이름을 제공하지 마십시오. `--stack-name` 옵션을 사용하여 스택 이름을 제공하거나 구성 파일에 AWS SAM CLI 스택 이름을 정의하면가 자동으로 리소스 ID를 CloudFormation 스택의 논리적 ID 값으로 처리합니다.

귀하가 리소스 ID를 제공하지 않는 경우:

리소스 ID를 제공하지 않지만 `--stack-name` 옵션과 함께 스택 이름을 제공하는 경우 AWS SAM CLI는 다음 로직을 사용하여 CloudFormation 스택의 리소스를 자동으로 호출하려고 시도합니다.

1. 는 AWS SAM CLI 다음 순서로 리소스 유형을 식별하고 스택에서 리소스 유형이 발견되면 다음 단계로 이동합니다.
   a. Lambda
   b. 단계 함수
   c. Amazon SQS
   d. Kinesis Data Streams
2. 리소스 유형에 스택에 단일 리소스가 있는 경우 AWS SAM CLI가 해당 리소스를 호출합니다. 스택에 리소스 유형의 리소스가 여러 개 있는 경우 AWS SAM CLI는 오류를 반환합니다.

다음은에서 수행할 작업의 예입니다. AWS SAM CLI

- 두 개의 Lambda 함수와 Amazon SQS 대기열이 포함된 스택 - 스택에 둘 이상의 Lambda 함수가 포함되어 있으므로는 AWS SAM CLI Lambda 리소스 유형을 찾고 및 오류를 반환합니다.
- Lambda 함수 1개와 Amazon Kinesis Data Streams 애플리케이션 2개가 포함된 스택 - 스택에 단일 Lambda 리소스가 포함되어 있으므로는 AWS SAM CLI Lambda 함수를 찾아 호출합니다.
- 단일 Amazon SQS 대기열과 두 개의 Kinesis Data Streams 애플리케이션이 포함된 스택 - 스택에 단일 Amazon SQS 대기열이 포함되어 있으므로는 AWS SAM CLI Amazon SQS 대기열을 찾아 호출합니다.

옵션

`--beta-features | --no-beta-features`

베타 기능을 허용 또는 거부합니다.

`--config-env` *TEXT*

귀하의 AWS SAMCLI 구성 파일로부터 사용할 환경을 지정합니다.

기본값: `default`

`--config-file` *FILENAME*

귀하의 구성 파일의 경로 및 이름을 지정합니다.

구성 파일에 대한 자세한 내용은 AWS SAM CLI 구성 섹션을 참조하세요.

기본값: 프로젝트 디렉터리의 루트에 있는 `samconfig.toml`

`--debug`

디버그 로깅을 활성화합니다. 이것은 AWS SAMCLI가 생성한 타임스탬프와 디버그 메시지를 인쇄합니다.

`--event, -e` *TEXT*

대상 리소스에 전송할 이벤트입니다.

`--event-file` *FILENAME*

대상 리소스에 보낼 이벤트가 포함된 파일의 경로입니다.

`--help, -h`

도움말 메시지를 표시한 후 종료합니다.

`--output` *[text | json]*

호출 결과를 특정 출력 형식으로 출력합니다.

`json` – 요청 메타데이터와 리소스 응답이 JSON 구조로 반환됩니다. 응답에는 전체 SDK 출력 결과가 포함됩니다.

`text` – 요청 메타데이터가 텍스트 구조로 반환됩니다. 리소스 응답은 간접 호출된 리소스의 출력 형식으로 반환됩니다.

`--parameter`

간접 호출 중인 리소스에 귀하가 전달할 수 있는 추가 Boto3파라미터.

Amazon Kinesis Data Streams

Kinesis 데이터 스트림에 기록을 넣는 데 다음과 같은 추가 파라미터가 사용될 수 있습니다.

- `ExplicitHashKey='`*string*`'`
- `PartitionKey='`*string*`'`
- `SequenceNumberForOrdering='`*string*`'`
- `StreamARN='`*string*`'`

각 파라미터에 대한 설명은 Kinesis.Client.put_record를 참조하세요.

AWS Lambda

다음과 같은 추가 파라미터를 사용하여 Lambda 리소스를 간접 호출하고 버퍼링된 응답을 받을 수 있습니다.

- `ClientContext='base64-encoded string'`
- `InvocationType='[ DryRun | Event | RequestResponse ]'`
- `LogType='[ None | Tail ]'`
- `Qualifier='string'`

다음 추가 파라미터를 사용하여 응답 스트리밍으로 Lambda 리소스를 간접 호출할 수 있습니다.

- `ClientContext='base64-encoded string'`
- `InvocationType='[ DryRun | RequestResponse ]'`
- `LogType='[ None | Tail ]'`
- `Qualifier='string'`

각 파라미터에 대한 설명은 다음을 참조하세요.

- 버퍼링된 응답을 포함하는 Lambda – Lambda.Client.invoke
- 응답 스트리밍을 지원하는 Lambda – Lambda.Client.invoke_with_response_stream

Amazon Simple Queue Service(Amazon SQS)

Amazon SQS 대기열로 메시지를 보내는 데 다음 추가 파라미터가 사용될 수 있습니다.

- `DelaySeconds=integer`
- `MessageAttributes='json string'`
- `MessageDeduplicationId='string'`
- `MessageGroupId='string'`
- `MessageSystemAttributes='json string'`

각 파라미터에 대한 설명은 SQS.Client.send_message를 참조하세요.

AWS Step Functions

다음 추가 파라미터는 상태 시스템 실행을 시작하는 데 사용할 수 있습니다.

- `name='string'`
- `traceHeader='string'`

각 파라미터에 대한 설명은 SFN.Client.start_execution를 참조하세요.

`--profile TEXT`

자격 증명을 가져올 자격 증명 파일의 특정 프로필 AWS 입니다.

`--region TEXT`

리소스 AWS 리전 의 입니다. 예를 들어 `us-east-1`입니다.

`--stack-name TEXT`

리소스가 속한 CloudFormation 스택의 이름입니다.

`--tenant-id TEXT`

다중 테넌트 Lambda 함수의 테넌트 ID입니다. 서로 다른 테넌트 간의 컴퓨팅 격리를 보장하는 데 사용됩니다. 테넌트 격리 모드로 구성된 함수를 호출할 때 필요합니다.

`--test-event-name NAME`

Lambda 함수에 전달할 공유 가능한 테스트 이벤트의 이름.

> **Note**
>
> 이 옵션은 Lambda 함수만 지원합니다.

예제

다음 예제에서는 AWS 클라우드에서 지원되는 리소스를 호출하고 디버그 로깅을 활성화하여에서 생성된 디버그 메시지와 타임스탬프를 인쇄합니다. AWS SAMCLI

```
$ sam remote invoke--debug
```

sam remote test-event

이 페이지는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) `sam remote test-event` 명령에 대한 참조 정보를 제공합니다.

- AWS SAM CLI에 대한 소개는 [란 무엇입니까 AWS SAMCLI?](#) 섹션을 참조하세요.
- AWS SAM CLI `sam remote test-event` 명령 사용에 대한 가이드는 [sam remote test-event를 사용한 클라우드 테스트 소개](#) 섹션을 참조하세요.

이 `sam remote test-event` 명령은 Amazon EventBridge 스키마 레지스트리의 공유 가능한 테스트 이벤트와 상호 작용합니다.

사용법

```
$ sam remote test-event <options> <subcommand>
```

옵션

`--help, -h`

도움말 메시지를 표시한 후 종료합니다.

하위 명령

`delete`

EventBridge 스키마 레지스트리에서 공유 가능한 테스트 이벤트를 삭제합니다. 자세한 내용은 sam remote test-event delete 섹션을 참조하세요.

`get`

EventBridge 스키마 레지스트리에서 공유 가능한 테스트 이벤트를 가져옵니다. 자세한 내용은 sam remote test-event get 섹션을 참조하세요.

`list`

AWS Lambda 함수의 기존 공유 가능한 테스트 이벤트를 나열합니다. 자세한 내용은 sam remote test-event list 섹션을 참조하세요.

`put`

로컬 파일의 이벤트를 EventBridge 스키마 레지스트리에 저장합니다. 자세한 내용은 sam remote test-event put 섹션을 참조하세요.

sam remote test-event delete

이 페이지는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) `sam remote test-event delete` 하위 명령에 대한 참조 정보를 제공합니다.

- AWS SAM CLI에 대한 소개는 란 무엇입니까 AWS SAMCLI? 섹션을 참조하세요.
- AWS SAM CLI `sam remote test-event` 명령 사용에 대한 가이드는 sam remote test-event를 사용한 클라우드 테스트 소개 섹션을 참조하세요.

`sam remote test-event delete` 하위 명령은 Amazon EventBridge 스키마 레지스트리에서 공유 가능한 테스트 이벤트를 삭제합니다.

사용법

```
$ sam remote test-event delete <arguments> <options>
```

인수

리소스 ID

공유 가능한 테스트 이벤트와 관련된 AWS Lambda 함수의 ID입니다.

논리적 ID를 제공하는 경우 AWS CloudFormation 옵션을 사용하여 Lambda 함수와 연결된 `--stack-name` 스택의 값도 제공해야 합니다.

유효한 값: 리소스의 논리적 ID 또는 리소스 ARN.

옵션

`--config-env` *TEXT*

사용할 구성 파일의 기본 매개변수 값을 지정하는 환경 이름입니다. 기본값은 "기본값"입니다. 구성 파일에 대한 자세한 내용은 AWS SAMCLI구성 파일 섹션을 참조하세요.

`--config-file` *PATH*

사용할 기본 매개변수 값이 들어 있는 구성 파일의 경로 및 파일 이름입니다. 기본값은 프로젝트 디렉터리의 루트에 있는 "`samconfig.toml`"입니다. 구성 파일에 대한 자세한 내용은 AWS SAMCLI 구성 파일 섹션을 참조하세요.

`--help, -h`

도움말 메시지를 표시한 후 종료합니다.

`--name` *TEXT*

공유 가능한 삭제할 테스트 이벤트의 이름입니다.

`--stack-name` *TEXT*

Lambda 함수와 연결되는 CloudFormation 스택의 이름입니다.

Lambda 함수 논리 ID를 인수로 제공하는 경우 이 옵션이 필요합니다.

sam remote test-event get

이 페이지는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) `sam remote test-event get` 하위 명령에 대한 참조 정보를 제공합니다.

- AWS SAM CLI에 대한 소개는 [란 무엇입니까 AWS SAMCLI?] 섹션을 참조하세요.
- AWS SAM CLI `sam remote test-event` 명령 사용에 대한 가이드는 [sam remote test-event를 사용한 클라우드 테스트 소개] 섹션을 참조하세요.

`sam remote test-event get` 하위 명령은 Amazon EventBridge 스키마 레지스트리에서 공유 가능한 테스트 이벤트를 가져옵니다.

사용법

```
$ sam remote test-event get <arguments> <options>
```

인수

리소스 ID

가져올 공유 가능한 테스트 이벤트와 관련된 AWS Lambda 함수의 ID입니다.

논리적 ID를 제공하는 경우 AWS CloudFormation 옵션을 사용하여 Lambda 함수와 연결된 `--stack-name` 스택의 값도 제공해야 합니다.

유효한 값: 리소스의 논리적 ID 또는 리소스 ARN.

옵션

`--config-env` *TEXT*

사용할 구성 파일의 기본 매개변수 값을 지정하는 환경 이름입니다. 기본값은 "기본값"입니다. 구성 파일에 대한 자세한 내용은 [AWS SAMCLI구성 파일] 섹션을 참조하세요.

`--config-file` *PATH*

사용할 기본 매개변수 값이 들어 있는 구성 파일의 경로 및 파일 이름입니다. 기본값은 프로젝트 디렉터리의 루트에 있는 "`samconfig.toml`"입니다. 구성 파일에 대한 자세한 내용은 [AWS SAMCLI 구성 파일] 섹션을 참조하세요.

`--help, -h`

도움말 메시지를 표시한 후 종료합니다.

`--name TEXT`

가져올 공유 가능한 테스트 이벤트의 이름입니다.

`--output-file FILENAME`

로컬 기기에 이벤트를 저장할 파일 경로와 이름입니다.

이 옵션을 제공하지 않으면 AWS SAM CLI는 공유 가능한 테스트 이벤트의 콘텐츠를 콘솔에 출력합니다.

`--stack-name TEXT`

Lambda 함수와 연결되는 CloudFormation 스택의 이름입니다.

Lambda 함수 논리 ID를 인수로 제공하는 경우 이 옵션이 필요합니다.

sam remote test-event list

이 페이지는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) `sam remote test-event list` 하위 명령에 대한 참조 정보를 제공합니다.

- AWS SAM CLI에 대한 소개는 [란 무엇입니까 AWS SAMCLI?] 섹션을 참조하세요.
- AWS SAM CLI `sam remote test-event` 명령 사용에 대한 가이드는 [sam remote test-event를 사용한 클라우드 테스트 소개] 섹션을 참조하세요.

`sam remote test-event list` 하위 명령은 Amazon EventBridge 스키마 레지스트리의 특정 AWS Lambda 함수에 대한 기존의 공유 가능한 테스트 이벤트를 나열합니다.

사용법

```
$ sam remote test-event list <arguments> <options>
```

인수

리소스 ID

공유 가능한 테스트 이벤트와 관련된 Lambda 함수의 ID입니다.

논리적 ID를 제공하는 경우 AWS CloudFormation 옵션을 사용하여 Lambda 함수와 연결된 `--stack-name` 스택의 값도 제공해야 합니다.

유효한 값: 리소스의 논리적 ID 또는 리소스 ARN.

옵션

`--config-env` *TEXT*

사용할 구성 파일의 기본 매개변수 값을 지정하는 환경 이름입니다. 기본값은 “기본값”입니다. 구성 파일에 대한 자세한 내용은 AWS SAMCLI구성 파일 섹션을 참조하세요.

`--config-file` *PATH*

사용할 기본 매개변수 값이 들어 있는 구성 파일의 경로 및 파일 이름입니다. 기본값은 프로젝트 디렉터리의 루트에 있는 “`samconfig.toml`“입니다. 구성 파일에 대한 자세한 내용은 AWS SAMCLI 구성 파일 섹션을 참조하세요.

`--help, -h`

도움말 메시지를 표시한 후 종료합니다.

`--stack-name` *TEXT*

Lambda 함수와 연결되는 CloudFormation 스택의 이름입니다.

Lambda 함수 논리 ID를 인수로 제공하는 경우 이 옵션이 필요합니다.

예시

이 명령을 사용하는 예제는 공유 가능한 테스트 이벤트 나열 섹션을 참조하세요.

sam remote test-event put

이 페이지는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) `sam remote test-event put` 하위 명령에 대한 참조 정보를 제공합니다.

- AWS SAM CLI에 대한 소개는 란 무엇입니까 AWS SAMCLI? 섹션을 참조하세요.
- AWS SAM CLI `sam remote test-event` 명령 사용에 대한 가이드는 sam remote test-event를 사용한 클라우드 테스트 소개 섹션을 참조하세요.

`sam remote test-event put` 하위 명령은 로컬 기기의 공유 가능한 테스트 이벤트를 Amazon EventBridge 스키마 레지스트리에 저장합니다.

사용법

```
$ sam remote test-event put <arguments> <options>
```

인수

리소스 ID

공유 가능한 테스트 이벤트와 관련된 AWS Lambda 함수의 ID입니다.

논리적 ID를 제공하는 경우 AWS CloudFormation 옵션을 사용하여 Lambda 함수와 연결된 `--stack-name` 스택의 값도 제공해야 합니다.

유효한 값: 리소스의 논리적 ID 또는 리소스 ARN.

옵션

`--config-env` *TEXT*

사용할 구성 파일의 기본 매개변수 값을 지정하는 환경 이름입니다. 기본값은 “기본값”입니다. 구성 파일에 대한 자세한 내용은 AWS SAMCLI구성 파일 섹션을 참조하세요.

`--config-file` *PATH*

사용할 기본 매개변수 값이 들어 있는 구성 파일의 경로 및 파일 이름입니다. 기본값은 프로젝트 디렉터리의 루트에 있는 “`samconfig.toml`“입니다. 구성 파일에 대한 자세한 내용은 AWS SAMCLI 구성 파일 섹션을 참조하세요.

`--file` *FILENAME*

로컬 기기에 있는 이벤트의 파일 경로 및 이름입니다.

`-`로부터 읽어들일 파일 이름 값으로 `stdin`를 입력합니다.

이 옵션은 필수입니다.

`--force`, `-f`

공유 가능한 테스트 이벤트를 같은 이름으로 덮어씁니다.

`--help, -h`

도움말 메시지를 표시한 후 종료합니다.

`--name` *`TEXT`*

공유 가능한 테스트 이벤트를 저장할 이름입니다.

EventBridge 스키마 레지스트리에 이름이 같은 공유 가능한 테스트 이벤트가 있는 경우 AWS SAM CLI는 이를 덮어쓰지 않습니다. 덮어쓰려면 `--force` 옵션을 추가하십시오.

`--output-file` *`FILENAME`*

로컬 기기에 이벤트를 저장할 파일 경로와 이름입니다.

이 옵션을 제공하지 않으면 AWS SAM CLI는 공유 가능한 테스트 이벤트의 콘텐츠를 콘솔에 출력합니다.

`--stack-name` *`TEXT`*

Lambda 함수와 연결되는 CloudFormation 스택의 이름입니다.

Lambda 함수 논리 ID를 인수로 제공하는 경우 이 옵션이 필요합니다.

예시

이 명령을 사용하는 예제는 [공유 가능한 테스트 이벤트 저장](#) 섹션을 참조하세요.

sam sync

이 페이지는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) `sam sync` 명령에 대한 참조 정보를 제공합니다.

- AWS SAM CLI에 대한 소개는 [란 무엇입니까 AWS SAMCLI?](#) 섹션을 참조하세요.
- AWS SAM CLI 사용에 대한 설명서는 [AWS SAM CLI](#) 섹션을 참조하세요.

이 `sam sync` 명령은 로컬 애플리케이션 변경 내용을 AWS 클라우드에 동기화합니다.

사용법

```
$ sam sync <options>
```

옵션

`--base-dir, -s DIRECTORY`

이 디렉터리에 대해 함수 또는 계층의 소스 코드에 대한 상대 경로를 확인합니다. 이 옵션을 사용하여 소스 코드 폴더의 상대 경로를 확인하는 방법을 변경할 수 있습니다. 기본적으로 상대 경로는 AWS SAM 템플릿 위치를 기준으로 확인됩니다.

이 옵션은 빌드 중인 루트 애플리케이션 또는 스택의 리소스 외에도 중첩된 애플리케이션 또는 스택에도 적용됩니다. 또한 이 옵션은 다음 리소스 유형 및 속성에도 적용됩니다.

- 리소스 유형: `AWS::Serverless::Function` 속성: `CodeUri`
- 리소스 유형: `AWS::Serverless::Function` 리소스 속성: `Metadata` 항목: `DockerContext`
- 리소스 유형: `AWS::Serverless::LayerVersion` 속성: `ContentUri`
- 리소스 유형: `AWS::Lambda::Function` 속성: `Code`
- 리소스 유형: `AWS::Lambda::LayerVersion` 속성: `Content`

`--build-image TEXT`

애플리케이션을 빌드할 때 사용할 [컨테이너 이미지]의 URI. 기본적으로 AWS SAM은 [Amazon Elastic Container Registry(Amazon ECR) Public]의 컨테이너 이미지 리포지토리 URI를 사용합니다. 다른 이미지를 사용하려면 이 옵션을 지정합니다.

단일 명령에서 이 옵션을 여러 번 사용할 수 있습니다. 각 옵션은 문자열 또는 키-값 쌍을 허용합니다.

- 문자열 - 애플리케이션의 모든 리소스가 사용할 컨테이너 이미지의 URI를 지정합니다. 다음은 그 예제입니다.

```
$ sam sync --build-image amazon/aws-sam-cli-build-image-python3.8
```

- 키-값 쌍 - 리소스 이름을 키로 지정하고 해당 리소스와 함께 사용할 컨테이너 이미지 URI를 값으로 지정합니다. 이 형식을 사용하면 애플리케이션의 각 리소스에 대해 서로 다른 컨테이너 이미지 URI를 지정할 수 있습니다. 다음은 그 예제입니다.

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

이 옵션은 `--use-container` 옵션이 지정된 경우에만 적용되며, 그렇지 않으면 오류가 발생합니다.

`--build-in-source | --no-build-in-source`

소스 폴더에서 프로젝트를 직접 빌드하기 위한 `--build-in-source`를 제공합니다.

`--build-in-source` 옵션은 다음과 같은 런타임과 빌드 메서드를 지원합니다.

- 런타임 - `sam init --runtime` 옵션에서 지원하는 모든 Node.js 런타임.
- 빌드 메서드 - `Makefile`, `esbuild`.

`--build-in-source` 옵션은 다음 옵션과 호환되지 않습니다.

- `--use-container`

기본값: `--no-build-in-source`

`--capabilities` *LIST*

CloudFormation가 특정 스택을 생성할 수 있도록 지정하는 기능 목록. 일부 스택 템플릿은 AWS 계정의 권한에 영향을 줄 수 있는 리소스를 포함합니다. 예로는 새 AWS Identity and Access Management(IAM)사용자를 생성하는 경우가 있습니다. 기본값을 재정의하려면 이 옵션을 지정합니다. 유효한 값은 다음과 같습니다.

- CAPABILITY_IAM
- CAPABILITY_NAMED_IAM
- CAPABILITY_RESOURCE_POLICY
- CAPABILITY_AUTO_EXPAND

기본값: `CAPABILITY_NAMED_IAM` 및 `CAPABILITY_AUTO_EXPAND`

`--code`

기본적으로 AWS SAM은 애플리케이션의 모든 리소스를 동기화합니다. 다음을 포함하는 코드 리소스만 동기화하려면 이 옵션을 지정합니다.

- `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`

코드 리소스를 동기화하기위해 AWS SAM은 AWS CloudFormation을 통해 배포하는 대신 AWS 서비스 API를 직접 사용합니다. CloudFormation 스택을 업데이트하려면 sam sync --watch 또는 sam deploy를 실행합니다.

`--config-env` *TEXT*

사용할 구성 파일의 기본 매개변수 값을 지정하는 환경 이름입니다. 기본값은 "기본값"입니다. 구성 파일에 대한 자세한 내용은 [AWS SAMCLI구성 파일](#) 섹션을 참조하세요.

`--config-file` *PATH*

사용할 기본 매개변수 값이 들어 있는 구성 파일의 경로 및 파일 이름입니다. 기본값은 프로젝트 디렉터리의 루트에 있는 "`samconfig.toml`"입니다. 구성 파일에 대한 자세한 내용은 [AWS SAMCLI 구성 파일](#) 섹션을 참조하세요.

`--dependency-layer | --no-dependency-layer`

동기화 프로세스의 속도를 높이기 위해 개별 함수의 종속성을 다른 계층으로 분리할지 여부를 지정합니다.

기본값: `--dependency-layer`

`--image-repository` *TEXT*

이 명령이 함수의 이미지를 업로드하는 Amazon Elastic Container Registry(Amazon ECR) 리포지토리의 이름. `Image` 패키지 유형으로 선언된 함수에 필요합니다.

`--image-repositories` *TEXT*

Amazon ECR 리포지토리 URI에 대한 함수의 매핑. 논리적 ID의 함수 참조. 다음은 그 예제입니다.

```
$ sam sync --image-repositories Function1=123456789012.dkr.ecr.us-east-1.amazonaws.com/my-repo
```

이 옵션은 하나의 명령에서 여러 번 지정할 수 있습니다.

`--kms-key-id` *TEXT*

Amazon S3 버킷에 저장된 아티팩트를 암호화하는 데 사용되는 AWS Key Management Service(AWS KMS) 키의 ID입니다. 이 옵션을 지정하지 않을 경우 AWS SAM은 Amazon S3 관리형 암호화 키를 사용합니다.

`--metadata`

템플릿에서 참조하는 모든 아티팩트에 연결할 메타데이터 맵.

`--notification-arns` *LIST*

CloudFormation이 스택과 열결하는 Amazon Simple Notification Service(Amazon SNS) 주제 ARN의 목록.

`--no-use-container`

IDE 툴킷이 기본 동작을 설정하는 데 사용할 수 있는 옵션입니다.

`--parameter-overrides`

키-값 페어로 인코딩된 CloudFormation 매개변수를 포함하는 스트링이 우선합니다. AWS Command Line Interface(AWS CLI)와 같은 형식을 사용합니다. AWS SAM CLI 형식은 명시적 키 및 값 키워드이며 각 재정의는 공백으로 구분됩니다. 다음은 두 가지 예제입니다.

- `--parameter-overrides ParameterKey=hello,ParameterValue=world`
- `--parameter-overrides ParameterKey=hello,ParameterValue=world ParameterKey=example1,ParameterValue=example2 ParameterKey=apple,ParameterValue=banana`

`--resource` *TEXT*

동기화할 리소스 유형을 지정합니다. 여러 리소스를 동기화하기 위해 이 옵션을 여러 번 지정할 수 있습니다. 이 옵션은 `--code` 옵션과 함께 지원됩니다. 이 값은 `--code`에 나열된 리소스 중 하나여야 합니다. 예를 들어 `--resource AWS::Serverless::Function --resource AWS::Serverless::LayerVersion`입니다.

`--resource-id` *TEXT*

동기화할 리소스 ID를 지정합니다. 여러 리소스를 동기화하기 위해 이 옵션을 여러 번 지정할 수 있습니다. 이 옵션은 `--code` 옵션과 함께 지원됩니다. 예를 들어 `--resource-id Function1 --resource-id Function2`입니다.

`--role-arn` *TEXT*

변경 세트를 적용할 때 CloudFormation이 맡은 IAM 역할의 Amazon 리소스 이름(ARN).

`--s3-bucket` *TEXT*

이 명령이 CloudFormation 템플릿을 업로드하는 Amazon Simple Storage Service(Amazon S3) 버킷의 이름. 템플릿이 51,200바이트를 초과하는 경우 `--s3-bucket` 또는 `--resolve-s3` 옵션이 필요합니다. `--s3-bucket` 및 `--resolve-s3` 옵션을 모두 지정하면 오류가 발생합니다.

`--s3-prefix` *TEXT*

Amazon S3 버킷에 업로드하는 아티팩트의 이름에 추가되는 접두사. 접두사 이름은 Amazon S3 버킷의 경로 이름(폴더 이름)입니다. 이는 `Zip` 패키지 유형으로 선언된 함수에만 적용됩니다.

`--save-params`

명령줄에서 AWS SAM 구성 파일에 제공한 파라미터를 저장합니다.

`--skip-deploy-sync | --no-skip-deploy-sync`

필요하지 않은 경우 초기 인프라 동기화를 건너뛰는 `--skip-deploy-sync`를 지정합니다. AWS SAM CLI는 로컬 AWS SAM 템플릿을 배포된 CloudFormation 템플릿과 비교하여 변경 사항이 감지된 경우에만 배포를 수행합니다.

`sam sync`가 실행될 때마다 CloudFormation 배포를 수행하도록 `--no-skip-deploy-sync`를 지정합니다.

자세한 내용은을 참조하십시오[초기 CloudFormation 배포는 건너뜁니다.](#).

기본값: `--skip-deploy-sync`

`--stack-name` *TEXT*

애플리케이션의 CloudFormation 스택 이름.

이 옵션은 필수입니다.

`--tags` *LIST*

생성되거나 업데이트된 스택과 연결할 태그 목록입니다. 또한 CloudFormation은 이러한 태그를 지원하는 스택의 리소스로 전파합니다.

`--template-file, --template, -t` *PATH*

AWS SAM 템플릿이 위치한 경로 및 파일 이름입니다.

> **Note**
> 이 옵션을 지정하는 경우 AWS SAM은 템플릿과 템플릿이 가리키는 로컬 리소스만에만 배포합니다.

`--use-container, -u`

함수가 기본적으로 컴파일된 종속성이 있는 패키지를 사용하는 경우, 이 옵션을 사용하여 AWS Lambda 유사 Docker 컨테이너 내에 함수를 빌드합니다.

> **Note**
>
> 현재 이 옵션은 `--dependency-layer`와 호환되지 않습니다. `--use-container`를 `--dependency-layer`와 함께 사용하면 AWS SAM CLI는 이를 알리고 `--no-dependency-layer`를 계속 사용합니다.

`--watch`

로컬 애플리케이션의 변경 내용을 감시하고 이를 AWS 클라우드에 자동으로 동기화하는 프로세스를 시작합니다. 기본적으로 이 옵션을 지정하면 AWS SAM은 업데이트 시 애플리케이션의 모든 리소스를 동기화합니다. 이 옵션을 사용하면 AWS SAM은 초기 CloudFormation 배포를 수행합니다. 그런 다음 AWS SAM은 AWS 서비스 API를 사용하여 코드 리소스를 업데이트합니다. AWS SAM은 CloudFormation을 사용하여 AWS SAM 템플릿을 업데이트할 때 인프라 리소스를 업데이트합니다.

`--watch-exclude` *TEXT*

파일 또는 폴더를 파일 변경 관찰 대상에서 제외합니다. 이 옵션을 사용하려면 `--watch` 도 제공해야 합니다.

이 옵션은 다음과 같은 키-값 페어를 수신합니다.

- 키 — 애플리케이션에 있는 Lambda 함수의 논리적 ID.
- 값 - 제외할 관련 파일 이름 또는 폴더.

`--watch-exclude` 옵션으로 지정된 파일이나 폴더를 업데이트하면 AWS SAM CLI가 동기화가 시작되지 않습니다. 하지만 다른 파일이나 폴더에 대한 업데이트로 인해 동기화가 시작하면 해당 파일 또는 폴더가 해당 동기화에 포함됩니다.

단일 명령에서 이 옵션을 여러 번 제공할 수 있습니다.

예시

이 명령을 사용하는 예제는 sam sync 명령의 옵션 섹션을 참조하세요.

sam traces

이 페이지는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) `sam traces` 명령에 대한 참조 정보를 제공합니다.

AWS SAM CLI에 대한 소개는 [란 무엇입니까 AWS SAMCLI?](#) 섹션을 참조하세요.

이 `sam traces` 명령은 AWS 리전 내에 있는 귀하의 AWS 계정에서 AWS X-Ray 추적을 가져옵니다.

사용법

```
$ sam traces <options>
```

옵션

`--config-env` *TEXT*

사용할 구성 파일의 기본 매개변수 값을 지정하는 환경 이름입니다. 기본값은 “기본값”입니다. 구성 파일에 대한 자세한 내용은 [AWS SAMCLI구성 파일](#) 섹션을 참조하세요.

`--config-file` *PATH*

사용할 기본 매개변수 값이 들어 있는 구성 파일의 경로 및 파일 이름입니다. 기본값은 프로젝트 디렉터리의 루트에 있는 “`samconfig.toml`“입니다. 구성 파일에 대한 자세한 내용은 [AWS SAMCLI 구성 파일](#) 섹션을 참조하세요.

`--end-time` *TEXT*

이 시간까지의 추적을 가져옵니다. 시간은 ‘5분 전’, ‘내일’과 같은 상댓값이거나 ‘2018-01-01 10:10:10’과 같은 형식이 지정된 타임스탬프일 수 있습니다.

`--output` *TEXT*

로그에 대한 출력 형식을 지정합니다. 형식이 지정된 로그를 인쇄하려면 `text`을 지정합니다. 로그를 JSON으로 인쇄하려면 `json`을 지정합니다.

`--save-params`

명령줄에서 AWS SAM 구성 파일에 제공한 매개변수를 저장합니다.

`--start-time` *TEXT*

이때부터 추적을 가져옵니다. 시간은 ‘5분 전’, ‘어제’와 같은 상댓값이거나 ‘2018-01-01 10:10:10’과 같은 형식이 지정된 타임스탬프일 수 있습니다. 기본값은 ‘10분 전’입니다.

`--tail`

추적 결과를 업데이트합니다. 이렇게 하면 엔드타임 인수가 무시되고 변경이 반영 가능할 때마다 계속 표시됩니다.

`--trace-id` *TEXT*

X-Ray 추적의 고유 식별자.

예시

다음 명령을 실행하여 ID별로 X-Ray 추적을 가져옵니다.

```
$ sam traces --trace-id tracing-id-1 --trace-id tracing-id-2
```

다음 명령을 실행하여 X-Ray 추적을 사용할 수 있게 되면 이를 추적합니다.

```
$ sam traces --tail
```

sam validate

이 페이지는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) `sam validate` 명령에 대한 참조 정보를 제공합니다.

AWS SAM CLI에 대한 소개는 [란 무엇입니까 AWS SAMCLI?](#) 섹션을 참조하세요.

이 `sam validate` 명령은 AWS SAM 템플릿 파일이 유효한지 여부를 확인합니다.

사용법

```
$ sam validate <options>
```

옵션

`--config-env` *TEXT*

사용할 구성 파일의 기본 매개변수 값을 지정하는 환경 이름입니다. 기본값은 “기본값”입니다. 구성 파일에 대한 자세한 내용은 AWS SAMCLI구성 파일 섹션을 참조하세요.

`--config-file PATH`

사용할 기본 매개변수 값이 들어 있는 구성 파일의 경로 및 파일 이름입니다. 기본값은 프로젝트 디렉터리의 루트에 있는 “samconfig.toml”입니다. 구성 파일에 대한 자세한 내용은 AWS SAMCLI구성 파일 섹션을 참조하세요.

`--debug`

디버그 로깅을 켜서 AWS SAM CLI 에 의해 생성된 디버그 메시지를 인쇄하고 타임스탬프를 표시합니다.

`--lint`

cfn-lint를 통해 템플릿에서 린팅검증을 실행합니다. `cfnlintrc` 구성 파일을 생성하여 추가 파라미터를 지정합니다. 자세한 내용은 CloudFormation GitHub 리포지토리의 cfn-lint를 참조하세요.

`--profile TEXT`

AWS 보안 인증 정보를 가져오는 보안 인증 파일의 특정 프로필입니다.

`--region TEXT`

배포할 AWS 지역입니다. 예를 들어 us-east-1입니다.

`--save-params`

명령줄에서 AWS SAM 구성 파일에 제공한 매개변수를 저장합니다.

`--template-file, --template, -t PATH`

AWS SAM 템플릿 파일입니다. 기본값은 `template.[yaml|yml]`입니다.

템플릿이 현재 작업 디렉터리에 있고 `template.[yaml|yml|json]`으로 이름이 지정된 경우에는 이 옵션이 필요하지 않습니다.

방금 sam build를 실행한 경우에는 이 옵션이 필요하지 않습니다.

예시

이 명령을 사용하여 템플릿을 검증하는 예제는 AWS SAM 템플릿 파일 검증 섹션을 참조하세요.

cfn-lint에서 이 명령을 사용하는 예제는 CloudFormation Linter를 사용하여 AWS SAM 애플리케이션을 검증합니다. 섹션을 참조하세요.

AWS SAM CLI 관리

이 섹션에는 AWS SAM CLI 버전을 관리하고 사용자 지정할 수 있는 방법에 대한 정보가 포함되어 있습니다. 여기에는 프로젝트 수준 구성 파일을 사용하여 AWS SAM CLI 명령 파라미터 값을 구성하는 방법에 대한 정보가 포함됩니다. 또한 AWS SAM CLI의 다양한 버전을 관리하고, AWS SAM이 사용자를 대신하여 AWS 서비스를 호출할 수 있게 해주는 AWS 보안 인증 정보를 설정하고, AWS SAM을 사용자 지정할 수 있는 다양한 방법에 대한 정보도 포함되어 있습니다. 이 섹션은 일반적인 AWS SAM 문제 해결 섹션으로 끝납니다.

주제

- AWS SAMCLI구성 파일
- AWS SAM CLI 버전 관리
- AWS 보안 인증 설정
- AWS SAM CLI 내 텔레메트리
- AWS SAMCLI 문제 해결

AWS SAMCLI구성 파일

AWS Serverless Application Model명령줄 인터페이스(AWS SAMCLI)는 AWS SAM CLI 명령 파라미터 값을 구성하는 데 사용할 수 있는 프로젝트 수준 구성 파일을 지원합니다.

구성 파일 작성 및 사용에 대한 설명서는 AWS SAM CLI 구성 섹션을 참조하세요.

주제

- 기본 구성 파일 설정
- 지원되는 구성 파일 형식
- 구성 파일을 지정합니다.
- 구성 파일 기본 사항
- 파라미터 값 규칙
- 구성 우선 순위
- 구성 파일 생성 및 수정

기본 구성 파일 설정

AWS SAM은 다음과 같은 기본 구성 파일 설정을 사용합니다.

- 이름 – samconfig.
- 위치 - 프로젝트의 루트에 있습니다. 이 위치는 귀하의 template.yaml 파일과 같은 위치입니다.
- 포맷 - TOML. 자세한 내용은 TOML설명서 내 TOML를 참조하세요.

다음은 기본 구성 파일 이름 및 위치가 포함된 예제 프로젝트 구조입니다.

```
sam-app
### README.md
### __init__.py
### events
### hello_world
### samconfig.toml
### template.yaml
### tests
```

다음은 samconfig.toml 파일의 예입니다.

```
...
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
```

지원되는 구성 파일 형식

TOML 및 [YAML|YML] 형식이 지원됩니다. 다음 기본 명령문을 참조하세요.

TOML

```
version = 0.1
[environment]
[environment.command]
[environment.command.parameters]
option = parameter value
```

YAML

```
version: 0.1
environment:
  command:
    parameters:
      option: parameter value
```

구성 파일을 지정합니다.

기본적으로 AWS SAM CLI는 다음과 같은 순서로 구성 파일을 찾습니다.

1. 사용자 지정 구성 파일 - `--config-file` 옵션을 사용하여 파일 이름과 위치를 지정하는 경우, AWS SAM CLI는 이 파일을 먼저 찾습니다.
2. 기본 **`samconfig.toml`** 파일 - 이는 귀하의 프로젝트의 루트에 있는 기본 구성 파일 이름 및 형식입니다. 사용자 지정 구성 파일을 지정하지 않는 경우 AWS SAM CLI는 다음으로 이 파일을 찾습니다.
3. **`samconfig.[yaml|yml]`**파일 – 귀하의 프로젝트 루트에 `samconfig.toml`이 없는 경우, AWS SAM CLI는 이 파일을 찾습니다.

다음은 `--config-file` 옵션을 사용하여 사용자 지정 구성 파일을 지정하는 예시입니다.

```
$ sam deploy --config-file myconfig.yaml
```

> **Note**
>
> AWS SAM CLI이(가) 구성이 적용되는 컨텍스트를 결정해야 하므로, `--config-file` 매개변수는 AWS SAM 템플릿 파일의 위치에 상대적이어야 합니다. `samconfig.toml` 파일은 AWS SAM CLI의 버전에 대한 구성 설정을 관리하며, CLI는 `samconfig.toml` 파일(또는 재정의된 구성 파일 매개변수)을 `template.yaml` 파일의 상대 폴더에서 찾습니다.

구성 파일 기본 사항

환경

환경은 고유한 구성 설정 세트를 포함하는 명명된 식별자입니다. 단일 AWS SAM 애플리케이션에 여러 환경을 포함할 수 있습니다.

환경 이름 기본은 `default`입니다.

AWS SAMCLI`--config-env` 옵션을 사용하여 사용할 환경을 지정합니다.

Command

명령은 파라미터 값을 지정하는 AWS SAM CLI 명령입니다.

모든 명령의 파라미터 값을 지정하려면 `global` 식별자를 사용합니다.

AWS SAMCLI명령을 참조할 때는 공백( )과 하이픈(–)을 밑줄(_)로 바꿉니다. 다음 예를 참조하세요.

- `build`
- `local_invoke`
- `local_start_api`

파라미터

파라미터는 키-값 페어로 지정됩니다.

- 키는 AWS SAM CLI명령 옵션 이름입니다.
- 값은 지정할 값입니다.

키를 지정할 때는 긴 형식의 명령 옵션 이름을 사용하고 하이픈(-)을 밑줄(_)로 바꿉니다. 예를 들어, 다음과 같습니다.

- `region`
- `stack_name`
- `template_file`

파라미터 값 규칙

TOML

- 부울 값은 `true` 또는 `false`일 수 있습니다. 예를 들어 `confirm_changeset = true`입니다.
- 문자열 값에는 따옴표("")를 사용합니다. 예를 들어 `region = "us-west-2"`입니다.
- 목록 값에는 따옴표("")를 사용하고 공백( )을 사용하여 각 값을 구분합니다. 예를 들어 `capabilities = "CAPABILITY_IAM CAPABILITY_NAMED_IAM"`입니다.
- 키-값 쌍의 목록이 포함된 값의 경우 쌍은 공백으로 구분되고( ) 각 쌍의 값은 인코딩된 따옴표(\" \") 로 묶습니다. 예를 들어 `tags = "project=\"my-application\" stage=\"production\""`입니다.
- 여러 번 지정할 수 있는 파라미터 값의 경우, 값은 인수의 배열입니다. 예를 들어 `image_repositories = ["my-function-1=image-repo-1", "my-function-2=image-repo-2"]`입니다.

YAML

- 부울 값은 `true` 또는 `false`일 수 있습니다. 예를 들어 `confirm_changeset: true`입니다.
- 단일 문자열 값을 포함하는 항목의 경우 따옴표("")는 선택 사항입니다. 예를 들어 `region: us-west-2`입니다. 여기에는 단일 문자열로 제공되는 여러 키-값 쌍을 포함하는 항목이 포함됩니다. 다음은 그 예제입니다.

```
$ sam deploy --tags "foo=bar hello=world"
```

```
default:
  deploy:
    parameters:
      tags: foo=bar hello=world
```

- 값 목록이 포함된 항목 또는 단일 명령으로 여러 번 사용할 수 있는 항목의 경우 해당 항목을 문자열 목록으로 지정합니다.

다음은 그 예제입니다.

```
$ sam remote invoke --parameter "InvocationType=Event" --parameter "LogType=None"
```

```
default:
  remote_invoke:
    parameters:
        parameter:
        - InvocationType=Event
        - LogType=None
```

구성 우선 순위

값을 구성할 때는 다음과 같은 우선 순위가 적용됩니다.

- 명령줄에서 제공하는 파라미터 값은 템플릿 파일의 구성 파일 및 `Parameters` 섹션에 있는 해당 값보다 우선합니다.
- 명령줄이나 구성 파일에서 `parameter_overrides` 키와 함께 `--parameter-overrides` 옵션을 사용하는 경우 해당 값이 템플릿 파일 `Parameters` 섹션의 값보다 우선합니다.
- 구성 파일에서 특정 명령에 대해 제공된 항목이 글로벌 항목보다 우선합니다. 다음 예제에서 `sam deploy` 명령은 `my-app-stack` 스택 이름을 사용합니다.

  TOML

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

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

  YAML

  ```
  default:
    global:
      parameters:
        stack_name: common-stack
    deploy:
      parameters:
  ```

```
        stack_name: my-app-stack
```

구성 파일 생성 및 수정

구성 파일 생성

`sam init`를 사용하여 애플리케이션을 만들면 기본으로 `samconfig.toml` 파일이 생성됩니다. 구성 파일을 수동으로 생성할 수도 있습니다.

구성 파일 수정

구성 파일을 수동으로 수정할 수 있습니다. 또한 모든 AWS SAM CLI 대화형 흐름 중에 구성된 값은 대괄호([ ]) 안에 표시됩니다. 이 값을 수정하면 AWS SAM CLI는 구성 파일을 업데이트합니다.

다음은 `sam deploy --guided` 명령을 사용하는 대화형 흐름의 예입니다.

```
$ 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
```

구성 파일을 수정할 때 AWS SAM CLI는 다음과 같이 글로벌 값을 처리합니다.

- 파라미터 값이 구성 파일 `global` 섹션에 있는 경우 AWS SAM CLI는 해당 값을 특정 명령 섹션에 기록하지 않습니다.
- 파라미터 값이 특정 명령 `global` 섹션과 특정 명령 섹션 모두에 있는 경우 AWS SAM CLI는 특정 항목을 삭제하고 글로벌 값을 취합니다.

AWS SAM CLI 버전 관리

업그레이드, 다운그레이드, 제거를 통해 AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI) 버전을 관리합니다. 선택에 따라, AWS SAM CLI nightly build로 다운로드하고 설치할 수 있습니다.

주제

- AWS SAMCLI업그레이드
- AWS SAM CLI 제거
- Homebrew 사용에서 AWS SAMCLI 관리로 전환
- AWS SAMCLI의 nightly build 관리
- AWS SAM을 이용하여 가상 환경에 CLIpip 설치
- AWS SAM로 CLIHomebrew 관리
- 문제 해결

AWS SAMCLI업그레이드

Linux

Linux에서 AWS SAM CLI를 업그레이드하려면 AWS SAMCLI의 설치의 설치 지침을 따르되, 다음과 같이 설치 명령에 `--update` 옵션을 추가합니다.

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

macOS

설치 시 사용한 것과 동일한 방법을 통해 AWS SAM CLI를 업그레이드해야 합니다. 패키지 설치 프로그램을 사용하여 AWS SAM CLI를 설치하고 업그레이드하는 것이 좋습니다.

패키지 설치 프로그램을 사용하여 AWS SAM CLI를 업그레이드하려면 최신 패키지 버전을 설치하십시오. 지침은 AWS SAMCLI의 설치 섹션을 참조하세요.

Windows

AWS SAM CLI를 업그레이드하려면 AWS SAM CLI 설치의 Windows 설치 단계를 다시 반복합니다.

AWS SAM CLI 제거

Linux

Linux에서 AWS SAM CLI를 제거하려면 다음 명령을 실행하여 symlink 및 설치 디렉터리를 삭제해야 합니다.

1. symlink 및 설치 경로를 찾습니다.

    - which 명령을 사용하여 symlink를 찾습니다.

    ```
    which sam
    ```

    출력에는 AWS SAM 바이너리가 있는 경로가 표시됩니다. 예를 들면 다음과 같습니다.

    ```
    /usr/local/bin/sam
    ```

    - ls 명령을 사용하여 symlink가 가리키는 디렉터리를 찾습니다.

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

    다음 예제에서 설치 디렉터리는 `/usr/local/aws-sam-cli`입니다.

    ```
    lrwxrwxrwx 1 ec2-user ec2-user 49 Oct 22 09:49 /usr/local/bin/sam -> /usr/local/aws-sam-cli/current/bin/sam
    ```

2. symlink를 삭제합니다.

    ```
    sudo rm /usr/local/bin/sam
    ```

3. 설치 디렉터리를 삭제합니다.

    ```
    sudo rm -rf /usr/local/aws-sam-cli
    ```

macOS

설치에 사용한 것과 동일한 방법을 사용하여 AWS SAM CLI를 제거합니다. 패키지 설치 프로그램을 사용하여 AWS SAM CLI를 설치하는 것이 좋습니다.

패키지 설치 프로그램을 사용하여 AWS SAM CLI를 설치한 경우 다음 단계에 따라 제거하십시오.

AWS SAMCLI 제거하기

1. 다음을 수정하고 실행하여 AWS SAM CLI 프로그램을 제거합니다.

```
$ sudo rm -rf /path-to/aws-sam-cli
```

   a. *sudo* – 사용자에게 AWS SAM CLI 프로그램이 설치된 위치에 대한 쓰기 권한이 있는 경우 sudo는 필요하지 않습니다. 그렇지 않으면 sudo이 필요합니다.

   b. *path-to* - AWS SAM CLI 프로그램을 설치한 위치의 경로. 기본 위치는 `/usr/local`입니다.

2. 다음을 수정하고 실행하여 AWS SAM CLI $PATH을 제거합니다.

```
$ sudo rm -rf /path-to-symlink-directory/sam
```

   a. *sudo* – 사용자에게 $PATH에 대한 쓰기 권한이 있는 경우에는 sudo은 필요하지 않습니다. 그렇지 않으면 sudo이 필요합니다.

   b. *path-to-symlink-directory* – 귀하의 $PATH 환경 변수입니다. 기본 위치는 `/usr/local/bin`입니다.

3. 다음을 실행하여 AWS SAM CLI가 제거되었는지 확인합니다.

```
$ sam --version
command not found: sam
```

Windows

Windows 설정을 사용하여 AWS SAM CLI를 제거하려면 다음 단계를 따르십시오.

1. 시작 메뉴에서 “프로그램 추가 또는 제거”를 검색합니다.

2. AWS SAM 명령줄 인터페이스라는 결과 항목을 선택한 다음 제거를 선택하여 제거 프로그램을 작동시킵니다.

3. AWS SAM CLI를 제거하고자 하는지 의사를 확인합니다.

Homebrew 사용에서 AWS SAMCLI 관리로 전환

Homebrew을 사용해 AWS SAM CLI을 설치 및 업그레이드하는 경우, AWS 지원 방법을 권장합니다. 지원되는 방법으로 전환하려면 다음 지침을 따르십시오.

Homebrew 사용에서 전환하기

1. [Homebrew CLI가 설치된 AWS SAM 제거](#)의 지침에 따라 Homebrew 관리형 버전을 제거합니다.
2. [AWS SAM CLI 설치](#)의 지침에 따라 지원되는 방법을 사용하여 AWS SAM CLI를 설치합니다.

AWS SAMCLI의 nightly build 관리

AWS SAM CLI nightly build를 다운로드 및 설치할 수 있습니다. 여기에는 생산용 버전보다 안정성이 떨어질 수 있는 시험판 버전의 AWS SAM CLI 코드가 포함되어 있습니다. 설치되면 `sam-nightly` 명령과 함께 자동 빌드를 사용할 수 있습니다. AWS SAM CLI의 생산용 버전과 nightly build 버전을 동시에 설치하여 사용할 수 있습니다.

> **Note**
>
> nightly build에는 시험판 버전의 빌드 이미지가 포함되어 있지 않습니다. 따라서 --use-container 옵션을 사용하여 서버리스 애플리케이션을 빌드할 때는 빌드 이미지의 최신 프로덕션 버전을 사용합니다.

AWS SAMCLI nightly build 설치

AWS SAM CLI nightly build를 설치하려면 다음 지침을 따르십시오.

Linux

패키지 설치 프로그램을 사용하여 AWS SAMx86_64 플랫폼에 CLI Linux의 nightly build 버전을 설치할 수 있습니다.

AWS SAMCLI nightly build 설치하기

1. [aws-sam-cli 리포지토리](#)의 sam-cli-nightlyGitHub에서 패키지 설치 프로그램을 다운로드하십시오.
2. [설치 AWS SAMCLI](#)단계를 따라 자동 빌드 패키지를 설치합니다.

macOS

자동 빌드 패키지 설치 프로그램을 사용하여 AWS SAM CLI의 nightly build 버전을 macOS상에 설치할 수 있습니다.

AWS SAMCLI nightly build 설치하기

1. aws-sam-cli 리포지토리의 sam-cli-nightlyGitHub에서 플랫폼을 위한 패키지 설치 프로그램을 다운로드하십시오.
2. 설치 AWS SAMCLI단계를 따라 자동 빌드 패키지를 설치합니다.

Windows

AWS SAM CLI의 nightly build 버전은 AWS SAM CLI 자동 빌드 다운로드 링크에서 이용할 수 있습니다. Windows에 자동 빌드를 설치하려면 AWS SAM CLI 설치에서와 동일한 단계를 수행하되, nightly build 다운로드 링크를 대신 사용하십시오.

자동 빌드 버전을 설치했는지 확인하려면 sam-nightly --version 명령을 실행합니다. 이 명령의 출력 결과는 `1.X.Y.dev<YYYYMMDDHHmm>`의 형식으로 표시됩니다. 예를 들면 다음과 같습니다.

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

Homebrew에서 패키지 설치 프로그램으로 전환

Homebrew을 사용하여 AWS SAM CLI 자동 빌드를 설치 및 업그레이드하다가 패키지 설치 프로그램 사용으로 전환하려면 다음 단계를 따르십시오.

Homebrew에서 패키지 설치 프로그램으로 전환하기

1. Homebrew AWS SAM nightly build가 설치된 CLI을 제거합니다.

   ```
   $ brew uninstall aws-sam-cli-nightly
   ```

2. 다음을 실행하여 AWS SAM CLI nightly build가 제거되었는지 확인하십시오.

   ```
   $ sam-nightly --version
   zsh: command not found: sam-nightly
   ```

3. 이전 섹션의 단계에 따라 AWS SAM CLI 자동 빌드를 설치하십시오.

AWS SAM을 이용하여 가상 환경에 CLIpip 설치

네이티브 패키지 설치 프로그램을 사용하여 AWS SAM CLI를 설치하는 것이 좋습니다. pip을 사용해야 한다면, AWS SAM CLI을 가상 환경으로 설치할 것을 권장합니다. 이렇게 하면 설치 환경이 깔끔해지고 오류 발생 시 격리된 환경이 보장됩니다.

> **Note**
>
> 2023년 10월 24일자로 AWS SAM CLI가 Python 3.7에 대한 지원을 중단합니다. 자세한 내용은 AWS SAMCLI가 Python 3.7에 대한 지원 중단 을 참조하세요.

가상 환경에 AWS SAMCLI 설치하기

1. 선택한 시작 디렉터리를 사용하여 가상 환경을 생성하고 이름을 지정합니다.

   Linux / macOS

   ```
   $ mkdir project
   $ cd project
   $ python3 -m venv venv
   ```

   Windows

   ```
   > mkdir project
   > cd project
   > py -3 -m venv venv
   ```

2. 가상 환경 활성화

   Linux / macOS

   ```
   $ . venv/bin/activate
   ```

   프롬프트가 변경되어 가상 환경이 활성임을 보여줍니다.

   ```
   (venv) $
   ```

Windows

```
> venv\Scripts\activate
```

프롬프트가 변경되어 가상 환경이 활성임을 보여줍니다.

```
(venv) >
```

3. 가상 환경에 AWS SAM CLI을 설치합니다.

```
(venv) $ pip install --upgrade aws-sam-cli
```

4. AWS SAM CLI이 올바르게 설치되었는지 확인합니다.

```
(venv) $ sam --version
SAM CLI, version 1.94.0
```

5. `deactivate` 명령을 사용하여 가상 환경을 종료할 수 있습니다. 새 세션을 시작할 때마다 환경을 다시 활성화해야 합니다.

AWS SAM로 CLIHomebrew 관리

> **Note**
>
> 2023년 9월부터 AWS은 AWS SAM CLI `aws/tap/aws-sam-cli`을 위한 AWS 관리형 Homebrew 설치관리자 프로그램을 더 이상 유지 관리하지 않습니다. Homebrew을 계속 사용하려면, 커뮤니티 관리형 설치 프로그램(`aws-sam-cli`)을 사용하실 수 있습니다. 2023년 9월부터 Homebrew를 참조하는 모든 `aws/tap/aws-sam-cli` 명령이 `aws-sam-cli`으로 리디렉션됩니다.
> 지원되는 설치 및 업그레이드 방법을 사용하는 것이 좋습니다.

AWS SAM을 사용하여 CLIHomebrew 설치

> **Note**
>
> 이 지침은 커뮤니티 관리형 AWS SAM CLI Homebrew 설치 프로그램을 사용합니다. 추가 지원이 필요하면 homebrew-core 리포지토리를 참조하세요.

AWS SAMCLI 설치하기

1. 다음을 실행합니다.

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

2. 설치를 확인합니다.

```
$ sam --version
```

AWS SAM CLI를 성공적으로 설치하면 다음과 같은 출력 결과가 표시됩니다.

```
SAM CLI, version 1.94.0
```

AWS SAM을 사용하여 CLIHomebrew 업그레이드

AWS SAM을 사용하여 CLI Homebrew을 업그레이드하려면, 다음 명령을 실행합니다.

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

Homebrew CLI가 설치된 AWS SAM 제거

AWS SAM을 사용하여 CLI Homebrew가 설치된 경우, 다음 단계에 따라 제거하십시오.

AWS SAM CLI 제거하기

1. 다음을 실행합니다.

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

2. 다음을 실행하여 AWS SAM CLI가 제거되었는지 확인합니다.

```
$ sam --version
command not found: sam
```

커뮤니티 관리형 Homebrew 설치 프로그램으로 전환

AWS 관리형 Homebrew 설치 프로그램(`aws/tap/aws-sam-cli`)을 사용 중이고 Homebrew를 계속 사용하고 싶다면, 커뮤니티 관리형 Homebrew 설치 프로그램(`aws-sam-cli`)으로 전환하는 것이 좋습니다.

단일 명령으로 전환하려면 다음을 실행하십시오.

```
$ brew uninstall aws-sam-cli && brew untap aws/tap && brew cleanup aws/tap && brew
 update && brew install aws-sam-cli
```

다음 지침에 따라 각 명령을 개별적으로 실행하십시오.

커뮤니티 관리형 Homebrew 설치 프로그램으로 전환하기

1. AWS의 Homebrew 관리형 AWS SAM CLI 버전을 제거합니다.

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

2. AWS SAM CLI가 제거되었는지 확인합니다.

```
$ which sam
sam not found
```

3. AWS 관리형 AWS SAM CLI 탭 제거합니다.

```
$ brew untap aws/tap
```

다음과 같은 오류가 발생하는 경우 `--force` 옵션을 추가하고 다시 시도하십시오.

```
Error: Refusing to untap aws/tap because it contains the following installed
 formulae or casks:
aws-sam-cli-nightly
```

4. AWS 관리형 설치 프로그램의 캐시된 파일을 제거합니다.

```
$ brew cleanup aws/tap
```

5. Homebrew 및 모든 공식 업데이트:

```
$ brew update
```

6. AWS SAM CLI의 커뮤니티 관리형 버전을 설치하십시오.

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

7. AWS SAM CLI의 설치가 성공적인지 확인합니다.

```
$ sam --version
SAM CLI, version 1.94.0
```

문제 해결

AWS SAM CLI의 설치 또는 사용 중에 오류가 발생하는 경우 AWS SAMCLI 문제 해결을 참조하세요.

AWS 보안 인증 설정

AWS SAM 명령줄 인터페이스(CLI)를 사용하려면 사용자 대신 AWS 서비스를 호출할 수 있도록 AWS 보안 인증을 설정해야 합니다. 예를 들어 AWS SAM CLI는 Amazon S3 및 CloudFormation를 호출합니다.

AWS SDK나 AWS와 같은 AWS 도구를 사용하기 위한 AWS CLI 보안 인증 정보를 이미 설정했을 것입니다. 아직 설정하지 않은 경우라면 이 주제에서 AWS 보안 인증을 설정하는 권장 방법을 보여드립니다.

AWS자격 증명을 설정하려면 구성하려는 IAM 사용자의 액세스 키 ID와 비밀 액세스 키가 있어야 합니다. 액세스 키 및 비밀 액세스 키에 대한 자세한 내용은 IAM 사용자 가이드의 IAM 사용자를 위한 액세스 키 관리를 참조하세요.

다음으로, AWS CLI을 이미 설치했는지 확인합니다. 그런 다음 아래 항목들 중 하나의 설정 지침을 따릅니다.

AWS CLI 사용

이미 AWS CLI을 설치한 경우, `aws configure` 명령을 사용하고 프롬프트를 따르십시오.

```
$ 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]:
```

aws configure 명령에 대한 자세한 내용은 AWS CLI 사용자 가이드의 AWS Command Line Interface빠른 구성을 참조하세요.

AWS CLI을 사용하지 않음

아직 AWS CLI을 설치하지 않은 경우, 자격 증명 파일을 만들거나 환경 변수를 설정할 수 있습니다.

- 보안 인증 정보 파일 - 로컬 시스템 상의 AWS 보안 인증 정보 파일에 보안 인증을 설정할 수 있습니다. 이 파일은 다음 중 한 곳에 있어야 합니다.
  - Linux 또는 macOS에서 `~/.aws/credentials`
  - Windows에서 `C:\Users\USERNAME\.aws\credentials`

  이 파일에는 다음 형식의 행이 포함되어야 합니다.

  ```
  [default]
  aws_access_key_id = your_access_key_id
  aws_secret_access_key = your_secret_access_key
  ```

- 환경 변수 – `AWS_ACCESS_KEY_ID`및 `AWS_SECRET_ACCESS_KEY`환경 변수를 설정할 수 있습니다.

  Linux 또는 macOS에서 이러한 변수를 설정하려면 내보내기 명령을 사용하십시오.

  ```
  export AWS_ACCESS_KEY_ID=your_access_key_id
  export AWS_SECRET_ACCESS_KEY=your_secret_access_key
  ```

  Windows에서 이러한 변수를 설정하려면 설정 명령을 사용하십시오.

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

AWS SAM CLI 내 텔레메트리

에서는 고객과의 상호 작용에서 배운 내용을 기반으로 서비스를 AWS개발하고 시작합니다. 우리는 고객 피드백을 바탕으로 제품을 반복적으로 개선해 나갑니다. 원격 측정 데이터는 가 사용자의 요구 사항을 더 잘 이해하고, 문제를 진단하고, 의 경험을 개선할 기능을 제공하는 데 도움이 되는 추가 정보입니다.

AWS SAM 명령줄 인터페이스(CLI)는 일반 사용 지표, 시스템 및 환경 정보, 오류와 같은 원격 측정을 수집합니다. 수집되는 텔레메트리 유형에 대한 자세한 내용은 수집되는 정보의 유형 섹션을 참조하세요.

AWS SAMCLI는 사용자 이름이나 이메일 주소와 같은 개인 정보는 수집하지 않습니다. 또한 민감한 프로젝트 수준 정보를 추출하지 않습니다.

고객은 텔레메트리 사용 여부를 제어하고 언제든지 설정을 변경할 수 있습니다. 텔레메트리가 켜져 있는 경우 AWS SAMCLI는 추가 고객 상호 작용 없이 백그라운드에서 텔레메트리 데이터를 전송합니다.

세션에 대한 텔레메트리 끄기

macOS 및 Linux 운영 체제에서는 단일 세션에 대한 텔레메트리를 끌 수 있습니다. 현재 세션의 텔레메트리를 끄려면 다음 명령을 실행하여 환경 변수 `SAM_CLI_TELEMETRY`를 `false`로 설정합니다. 각 새 터미널 또는 세션에 대해 명령을 반복합니다.

```
export SAM_CLI_TELEMETRY=0
```

모든 세션에서 사용자 프로필에 대한 텔레메트리 끄기

운영 체제에서 AWS SAMCLI를 실행할 때 다음 명령을 실행하여 모든 세션의 텔레메트리를 끕니다.

Linux에서 텔레메트리를 끄려면

1. 실행합니다.

   ```
   echo "export SAM_CLI_TELEMETRY=0" >>~/.profile
   ```

2. 실행합니다.

   ```
   source ~/.profile
   ```

macOS에서 텔레메트리를 끄려면

1. 실행합니다.

```
echo "export SAM_CLI_TELEMETRY=0" >>~/.profile
```

2. 실행합니다.

```
source ~/.profile
```

Windows에서 텔레메트리를 끄려면

다음 명령을 사용하여 터미널 창의 수명 동안 일시적으로 환경 변수를 설정할 수 있습니다.

명령 프롬프트를 사용하는 경우:

```
set SAM_CLI_TELEMETRY=0
```

PowerShell을 사용하는 경우:

```
$env:SAM_CLI_TELEMETRY=0
```

명령 프롬프트 또는 PowerShell에서 환경 변수를 영구적으로 설정하려면 다음 명령을 사용합니다.

```
setx SAM_CLI_TELEMETRY 0
```

> **Note**
> 터미널을 닫고 다시 열기 전에는 변경 사항이 적용되지 않습니다.

수집되는 정보의 유형

- 사용 정보 - 고객이 실행하는 일반 명령 및 하위 명령.
- 오류 및 진단 정보 - 종료 코드, 내부 예외 이름, Docker 연결 시 실패 등 고객이 실행하는 명령의 상태 및 기간.
- 시스템 및 환경 정보 - Python 버전, 운영 체제(Windows, Linux 또는 macOS),가 AWS SAMCLI 실행되는 환경(예: AWS CodeBuild AWS IDE 도구 키트 또는 터미널), 사용 속성의 해시 값입니다.

자세히 알아보기

가 AWS SAMCLI 수집하는 원격 측정 데이터는 AWS 데이터 개인 정보 보호 정책을 준수합니다. 자세한 내용은 다음 자료를 참조하세요.

- AWS 서비스 약관
- 데이터 프라이버시 FAQ

AWS SAMCLI 문제 해결

이 섹션에서는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAM CLI)의 사용, 설치 및 관리 시 발생하는 오류 메시지를 해결하는 방법에 대한 세부 정보를 제공합니다.

주제

- 문제 해결
- 오류 메시지
- 경고 메시지

문제 해결

AWS SAM CLI 관련 문제 해결 지침은 설치 오류 문제 해결 섹션을 참조하세요.

오류 메시지

curl 오류: “curl: (6) 해결할 수 없습니다:...”

API Gateway 엔드포인트를 호출하려고 하면 다음 오류가 표시됩니다.

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

이는 유효하지 않은 도메인으로 요청을 보내려고 시도했음을 의미합니다. 이는 서버리스 애플리케이션이 성공적으로 배포되지 않았거나 curl 명령에 오타가 있는 경우 발생할 수 있습니다. CloudFormation 콘솔 또는 AWS CLI를 사용하여 애플리케이션이 성공적으로 배포되었는지 확인하고 curl 명령이 올바른지 확인하세요.

오류: 지정된 스택 이름으로 정확한 리소스 정보를 찾을 수 없습니다.

단일 Lambda 함수 리소스가 포함된 애플리케이션에서 `sam remote invoke` 명령을 실행하면 다음 오류가 표시됩니다.

```
Error: Can't find exact resource information with given <stack-name>. Please provide
 full resource ARN or --stack-name to resolve the ambiguity.
```

가능한 원인: **--stack-name** 옵션을 제공하지 않았습니다.

함수 ARN이 인수로 제공되지 않은 경우 `sam remote invoke` 명령을 실행하려면 `--stack-name` 옵션을 제공해야 합니다.

해결 방법: **--stack-name** 옵션을 제공합니다.

다음은 예제입니다.

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

오류: 스택 이름에서 리소스 정보를 찾을 수 없습니다.

`sam remote invoke` 명령을 실행하고 Lambda 함수 ARN을 인수로 전달하면 다음 오류가 표시됩니다.

```
Error: Can't find resource information from stack name (<stack-name>) and resource id
 (<function-id>)
```

가능한 원인: **samconfig.toml** 파일에 스택 이름 값이 정의되어 있습니다.

AWS SAM CLI는 먼저 `samconfig.toml` 파일에 스택 이름이 있는지 확인합니다. 지정된 경우 인수는 논리적 ID 값으로 전달됩니다.

해결 방법: 함수의 논리적 ID를 대신 전달합니다.

함수의 ARN 대신 함수의 논리적 ID를 인수로 전달할 수 있습니다.

해결 방법: 구성 파일에서 스택 이름 값을 제거합니다.

구성 파일에서 스택 이름 값을 제거할 수 있습니다. 이렇게 하면 AWS SAM CLI가 함수 ARN을 논리적 ID 값으로 전달하는 것을 방지할 수 있습니다.

구성 파일을 수정한 후 `sam build`를 실행합니다.

오류: 관리형 리소스를 생성하지 못했습니다. 자격 증명을 찾을 수 없습니다.

sam deploy 명령을 실행할 때 다음 오류가 표시됩니다.

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

즉, AWS SAM CLI가 AWS 서비스를 호출할 수 있도록 AWS 보안 인증을 설정하지 않았습니다. 이 문제를 해결하려면 AWS 보안 인증을 설정해야 합니다. 자세한 내용은 AWS 보안 인증 설정 섹션을 참조하세요.

오류: Windows의 FileNotFoundError

Windows의 AWS SAM CLI에서 명령을 실행할 때 다음 오류가 발생할 수 있습니다.

```
Error: FileNotFoundError
```

가능한 원인: AWS SAM CLI는 Windows 최대 경로 제한을 초과하는 파일 경로와 상호 작용할 수 있습니다.

해결 방법: 이 문제를 해결하기 위해서는 새로운 긴 경로 동작을 활성화해야 합니다. 이렇게 하려면 Microsoft Windows 앱 개발 설명서의 Windows 10, 버전 1607 이상에서 긴 경로 사용 활성화를 참조하세요.

오류: pip의 종속성해석기...

오류 텍스트 예:

```
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.
```

가능한 원인: pip를 사용하여 패키지를 설치하는 경우 패키지 간 종속성이 충돌할 수 있습니다.

각 `aws-sam-cli` 패키지의 버전은 `aws-sam-translator` 패키지 버전에 따라 달라집니다. 예를 들어 `aws-sam-cli` v1.58.0은 `aws-sam-translator` v1.51.0에 따라 달라질 수 있습니다.

pip를 사용하여 AWS SAM CLI를 설치한 다음 새로운 버전의 `aws-sam-translator`를 사용하는 다른 패키지를 설치하면 다음과 같은 상황이 발생합니다.

- 새로운 버전의 `aws-sam-translator`가 설치됩니다.
- `aws-sam-cli`의 현재 버전과 `aws-sam-translator`의 새로운 버전은 호환되지 않을 수 있습니다.
- AWS SAM CLI를 사용하면 종속성 해석기 오류가 발생합니다.

솔루션

1. AWS SAM CLI 네이티브 패키지 설치 프로그램을 사용합니다.
   a. pip를 사용하여 AWS SAM CLI 제거 지침은 [AWS SAM CLI 제거](#) 섹션을 참조하세요.
   b. 네이티브 패키지 설치 프로그램을 사용하여 AWS SAM CLI를 설치합니다. 지침은 [AWS SAM CLI 설치](#) 섹션을 참조하세요.
   c. 필요한 경우 네이티브 패키지 설치 프로그램을 사용하여 AWS SAM CLI를 업그레이드합니다. 지침은 [AWS SAMCLI업그레이드](#) 섹션을 참조하세요.
2. pip를 사용해야 하는 경우 가상 환경에 AWS SAM CLI를 설치하는 것이 좋습니다. 이렇게 하면 설치 환경이 깔끔해지고 오류 발생 시 격리된 환경이 보장됩니다. 지침은 [AWS SAM을 이용하여 가상 환경에 CLIpip 설치](#) 섹션을 참조하세요.

오류: '원격' 명령이 없습니다.

`sam remote invoke` 명령을 실행할 때 다음 오류가 표시됩니다.

```
$ 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'.
```

가능한 원인: AWS SAM CLI의 버전이 오래된 버전입니다.

이 AWS SAM CLI `sam remote invoke` 명령은 AWS SAM CLI 버전 1.88.0과 함께 릴리스되었습니다. 버전은 `sam --version` 명령을 통해 확인할 수 있습니다.

해결 방법: AWS SAM CLI를 최신 버전으로 업그레이드합니다.

지침은 AWS SAMCLI업그레이드 섹션을 참조하세요.

오류: AWS SAM 프로젝트를 로컬에서 실행해야 Docker 합니다. 설치하셨습니까?

sam local start-api 명령을 실행할 때 다음 오류가 표시됩니다.

```
Error: Running AWS SAM projects locally requires Docker. Have you got it installed?
```

Docker가 제대로 설치되지 않았음을 의미합니다. 애플리케이션을 로컬에서 테스트하는 데 Docker가 필요합니다. 이 문제를 해결하려면 개발 호스트용 Docker 설치 지침을 따르세요. 자세한 내용은 Docker 설치 섹션을 참조하세요.

오류: 보안 제약 조건이 충족되지 않음

sam deploy --guided 실행 중에 질문 `Function may not have authorization defined, Is this okay? [y/N]`을 묻는 메시지가 표시됩니다. 이 프롬프트에 **N**(기본 응답)으로 응답하면 다음 오류가 표시됩니다.

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

이 프롬프트는 배포하려는 애플리케이션에 공개적으로 액세스할 수 있는 Amazon API Gateway API가 승인 없이 구성되어 있을 수 있다는 메시지를 표시합니다. 이 프롬프트에 **N**으로 응답하는 것은 괜찮지 않다고 말하는 것입니다.

이 문제를 해결할 수 있도록 다음 옵션이 제공됩니다.

- 권한 부여를 통해 애플리케이션을 구성하세요. 권한 부여 구성에 대한 자세한 내용은을 참조하세요AWS SAM 템플릿으로 API 액세스 제어
- 승인 없이 공개적으로 액세스할 수 있는 API 엔드포인트를 만들려면 배포를 다시 시작하고 이 질문에 **Y**로 응답하여 배포해도 괜찮다는 의사를 표시하세요.

인증 토큰 누락

API Gateway 엔드포인트를 호출하려고 하면 다음 오류가 표시됩니다.

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

이는 올바른 도메인으로 요청을 보내려고 했지만 URI를 인식할 수 없음을 의미합니다. 이 문제를 해결하려면 전체 URL을 확인하고 올바른 URL로 curl 명령을 업데이트합니다.

경고 메시지

경고:... AWS더 이상 Homebrew 설치 프로그램을 유지 관리하지 않습니다... AWS SAM

Homebrew를 사용하여 AWS SAM CLI를 설치할 때 다음과 같은 경고 메시지가 나타납니다.

```
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 ...
```

잠재적 원인: AWS가 Homebrew 지원을 더 이상 유지할 수 없습니다.

2023년 9월부터 AWS는 더 이상 AWS SAM CLI의 Homebrew 설치 프로그램을 유지 관리하지 않습니다.

해결 방법: AWS 지원 설치 방법을 사용합니다.

- AWS 지원 설치 방법은 AWS SAM CLI 설치에서 찾을 수 있습니다.

해결 방법: 계속 Homebrew를 사용하려면 커뮤니티 관리 설치 프로그램을 사용합니다.

- 재량에 따라 커뮤니티 관리 Homebrew 설치 프로그램을 사용할 수 있습니다. 지침은 AWS SAM로 CLIHomebrew 관리 단원을 참조하세요.

AWS SAM 커넥터 참조

이 섹션에는 AWS Serverless Application Model(AWS SAM) Connector 리소스 유형에 대한 참조 정보가 포함되어 있습니다. 커넥터에 대한 소개는 AWS SAM 커넥터를 사용한 리소스 권한 관리 섹션을 참조하세요.

커넥터에 지원되는 소스 및 대상 리소스 유형

`AWS::Serverless::Connector` 리소스 유형은 소스 리소스와 대상 리소스 간에 선택한 수의 연결을 지원합니다. 귀하의 AWS SAM 템플릿에서 커넥터를 구성할 때 다음 표를 사용하여 지원되는 연결과 각 소스 및 대상 리소스 유형에 정의해야 하는 속성을 참조하세요. 템플릿에서 커넥터를 구성하는 방법에 대한 자세한 내용은 AWS::Serverless::Connector 섹션을 참조하세요.

소스 및 대상 리소스 모두에 대해 동일한 템플릿 내에 정의된 경우 `Id` 속성을 사용합니다. 선택적으로 `Qualifier`을 추가하여 정의된 리소스의 범위를 좁힐 수 있습니다. 리소스가 동일한 템플릿 내에 있지 않은 경우 지원되는 속성을 조합하여 사용합니다.

새 연결을 요청하려면 서버리스 애플리케이션 모델 AWS GitHub 리포지토리에 새 문제를 제기하십시오.

소스 유형	대상 유형	권한	소스 속성	대상 속성
`AWS::ApiGateway::RestApi`	`AWS::Lambda::Function`	`Write`	`Id`또는 `Qualifier` , `ResourceId` , 및 `Type`	`Id`또는 `Arn` 및 `Type`
`AWS::ApiGateway::RestApi`	`AWS::Serverless::Function`	`Write`	`Id`또는 `Qualifier` , `ResourceId` , 및 `Type`	`Id`또는 `Arn` 및 `Type`
`AWS::ApiGatewayV2::Api`	`AWS::Lambda::Function`	`Write`	`Id`또는 `Qualifier` , `ResourceId` , 및 `Type`	`Id`또는 `Arn` 및 `Type`
`AWS::ApiGatewayV2::Api`	`AWS::Serverless::Function`	`Write`	`Id`또는 `Qualifier` , `ResourceId` , 및 `Type`	`Id`또는 `Arn` 및 `Type`

소스 유형	대상 유형	권한	소스 속성	대상 속성
AWS::AppSync::DataSource	AWS::DynamoDB::Table	Read	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::AppSync::DataSource	AWS::DynamoDB::Table	Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::AppSync::DataSource	AWS::Events::EventBus	Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::AppSync::DataSource	AWS::Lambda::Function	Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::AppSync::DataSource	AWS::Serverless::Function	Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::AppSync::DataSource	AWS::Serverless::SimpleTable	Read	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::AppSync::DataSource	AWS::Serverless::SimpleTable	Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::AppSync::GraphQLApi	AWS::Lambda::Function	Write	Id또는 ResourceId 및 Type	Id또는 Arn 및 Type
AWS::AppSync::GraphQLApi	AWS::Serverless::Function	Write	Id또는 ResourceId 및 Type	Id또는 Arn 및 Type

소스 유형	대상 유형	권한	소스 속성	대상 속성
AWS::DynamoDB::Table	AWS::Lambda::Function	Read	Id또는 Arn 및 Type	Id또는 RoleName 및 Type
AWS::DynamoDB::Table	AWS::Serverless::Function	Read	Id또는 Arn 및 Type	Id또는 RoleName 및 Type
AWS::Events::Rule	AWS::Events::EventBus	Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::Events::Rule	AWS::Lambda::Function	Write	Id또는 Arn 및 Type	Id또는 Arn 및 Type
AWS::Events::Rule	AWS::Serverless::Function	Write	Id또는 Arn 및 Type	Id또는 Arn 및 Type
AWS::Events::Rule	AWS::Serverless::StateMachine	Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::Events::Rule	AWS::SNS::Topic	Write	Id또는 Arn 및 Type	Id또는 Arn 및 Type
AWS::Events::Rule	AWS::SQS::Queue	Write	Id또는 Arn 및 Type	Id또는 Arn, QueueUrl, 및 Type
AWS::Events::Rule	AWS::StepFunctions::StateMachine	Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type

소스 유형	대상 유형	권한	소스 속성	대상 속성
AWS::Lambda::Function	AWS::DynamoDB::Table	Read, Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::Lambda::Function	AWS::Events::EventBus	Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::Lambda::Function	AWS::Lambda::Function	Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::Lambda::Function	AWS::Location::PlaceIndex	Read	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::Lambda::Function	AWS::S3::Bucket	Read, Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::Lambda::Function	AWS::Serverless::Function	Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::Lambda::Function	AWS::Serverless::SimpleTable	Read, Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::Lambda::Function	AWS::Serverless::StateMachine	Read, Write	Id또는 RoleName 및 Type	Id또는 Arn, Name, 및 Type
AWS::Lambda::Function	AWS::SNS::Topic	Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type

소스 유형	대상 유형	권한	소스 속성	대상 속성
AWS::Lambda::Function	AWS::SQS::Queue	Read, Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::Lambda::Function	AWS::StepFunctions::StateMachine	Read, Write	Id또는 RoleName 및 Type	Id또는 Arn, Name, 및 Type
AWS::S3::Bucket	AWS::Lambda::Function	Write	Id또는 Arn 및 Type	Id또는 Arn 및 Type
AWS::S3::Bucket	AWS::Serverless::Function	Write	Id또는 Arn 및 Type	Id또는 Arn 및 Type
AWS::Serverless::Api	AWS::Lambda::Function	Write	Id또는 Qualifier , ResourceId , 및 Type	Id또는 Arn 및 Type
AWS::Serverless::Api	AWS::Serverless::Function	Write	Id또는 Qualifier , ResourceId , 및 Type	Id또는 Arn 및 Type
AWS::Serverless::Function	AWS::DynamoDB::Table	Read, Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::Serverless::Function	AWS::Events::EventBus	Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type

소스 유형	대상 유형	권한	소스 속성	대상 속성
AWS::Serverless::Function	AWS::Lambda::Function	Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::Serverless::Function	AWS::S3::Bucket	Read, Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::Serverless::Function	AWS::Serverless::Function	Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::Serverless::Function	AWS::Serverless::SimpleTable	Read, Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::Serverless::Function	AWS::Serverless::StateMachine	Read, Write	Id또는 RoleName 및 Type	Id또는 Arn, Name, 및 Type
AWS::Serverless::Function	AWS::SNS::Topic	Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::Serverless::Function	AWS::SQS::Queue	Read, Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::Serverless::Function	AWS::StepFunctions::StateMachine	Read, Write	Id또는 RoleName 및 Type	Id또는 Arn, Name, 및 Type

소스 유형	대상 유형	권한	소스 속성	대상 속성
AWS::Serverless::HttpApi	AWS::Lambda::Function	Write	Id또는 Qualifier , ResourceId , 및 Type	Id또는 Arn 및 Type
AWS::Serverless::HttpApi	AWS::Serverless::Function	Write	Id또는 Qualifier , ResourceId , 및 Type	Id또는 Arn 및 Type
AWS::Serverless::SimpleTable	AWS::Lambda::Function	Read	Id또는 Arn 및 Type	Id또는 RoleName 및 Type
AWS::Serverless::SimpleTable	AWS::Serverless::Function	Read	Id또는 Arn 및 Type	Id또는 RoleName 및 Type
AWS::Serverless::StateMachine	AWS::DynamoDB::Table	Read, Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::Serverless::StateMachine	AWS::Events::EventBus	Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::Serverless::StateMachine	AWS::Lambda::Function	Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type

소스 유형	대상 유형	권한	소스 속성	대상 속성
AWS::Serverless::StateMachine	AWS::S3::Bucket	Read, Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::Serverless::StateMachine	AWS::Serverless::Function	Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::Serverless::StateMachine	AWS::Serverless::SimpleTable	Read, Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::Serverless::StateMachine	AWS::Serverless::StateMachine	Read, Write	Id또는 RoleName 및 Type	Id또는 Arn, Name, 및 Type
AWS::Serverless::StateMachine	AWS::SNS::Topic	Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::Serverless::StateMachine	AWS::SQS::Queue	Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::Serverless::StateMachine	AWS::StepFunctions::StateMachine	Read, Write	Id또는 RoleName 및 Type	Id또는 Arn, Name, 및 Type

소스 유형	대상 유형	권한	소스 속성	대상 속성
AWS::SNS::Topic	AWS::Lambda::Function	Write	Id또는 Arn 및 Type	Id또는 Arn 및 Type
AWS::SNS::Topic	AWS::Serverless::Function	Write	Id또는 Arn 및 Type	Id또는 Arn 및 Type
AWS::SNS::Topic	AWS::SQS::Queue	Write	Id또는 Arn 및 Type	Id또는 Arn, QueueUrl, 및 Type
AWS::SQS::Queue	AWS::Lambda::Function	Read, Write	Id또는 Arn 및 Type	Id또는 RoleName 및 Type
AWS::SQS::Queue	AWS::Serverless::Function	Read, Write	Id또는 Arn 및 Type	Id또는 RoleName 및 Type
AWS::StepFunctions::StateMachine	AWS::DynamoDB::Table	Read, Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::StepFunctions::StateMachine	AWS::Events::EventBus	Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::StepFunctions::StateMachine	AWS::Lambda::Function	Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type

소스 유형	대상 유형	권한	소스 속성	대상 속성
AWS::StepFunctions::StateMachine	AWS::S3::Bucket	Read, Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::StepFunctions::StateMachine	AWS::Serverless::Function	Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::StepFunctions::StateMachine	AWS::Serverless::SimpleTable	Read, Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::StepFunctions::StateMachine	AWS::Serverless::StateMachine	Read, Write	Id또는 RoleName 및 Type	Id또는 Arn, Name, 및 Type
AWS::StepFunctions::StateMachine	AWS::SNS::Topic	Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::StepFunctions::StateMachine	AWS::SQS::Queue	Write	Id또는 RoleName 및 Type	Id또는 Arn 및 Type
AWS::StepFunctions::StateMachine	AWS::StepFunctions::StateMachine	Read, Write	Id또는 RoleName 및 Type	Id또는 Arn, Name, 및 Type

커넥터에서 생성한 IAM 정책

이 섹션에서는 커넥터를 사용할 때 AWS SAM에 의해 생성되는 AWS Identity and Access Management(IAM) 정책에 대해 설명합니다.

AWS::DynamoDB::Table~AWS::Lambda::Function

정책 유형

AWS::Lambda::Function 역할에 연결된 고객 관리형 정책.

액세스 카테고리

Read

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:DescribeStream",
        "dynamodb:GetRecords",
        "dynamodb:GetShardIterator",
        "dynamodb:ListStreams"
      ],
      "Resource": [
        "%{Source.Arn}/stream/*"
      ]
    }
  ]
}
```

AWS::Events::Rule~AWS::SNS::Topic

정책 유형

AWS::SNS::Topic에 연결된 AWS::SNS::TopicPolicy.

액세스 카테고리

Write

```
{
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "events.amazonaws.com"
      },
      "Resource": "%{Destination.Arn}",
      "Action": "sns:Publish",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "%{Source.Arn}"
        }
      }
    }
  ]
}
```

AWS::Events::Rule~AWS::Events::EventBus

정책 유형

AWS::Events::Rule 역할에 연결된 고객 관리형 정책.

액세스 카테고리

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::Events::Rule~AWS::StepFunctions::StateMachine

정책 유형

AWS::Events::Rule 역할에 연결된 고객 관리형 정책.

액세스 카테고리

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::Events::Rule~AWS::Lambda::Function

정책 유형

AWS::Lambda::Function에 연결된 AWS::Lambda::Permission.

액세스 카테고리

Write

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "events.amazonaws.com",
  "SourceArn": "%{Source.Arn}"
}
```

AWS::Events::Rule~AWS::SQS::Queue

정책 유형

AWS::SQS::Queue에 연결된 AWS::SQS::QueuePolicy.

액세스 카테고리

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "events.amazonaws.com"
      },
      "Resource": "%{Destination.Arn}",
      "Action": "sqs:SendMessage",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "%{Source.Arn}"
        }
      }
    }
  ]
}
```

AWS::Lambda::Function~AWS::Lambda::Function

정책 유형

AWS::Lambda::Function 역할에 연결된 고객 관리형 정책.

액세스 카테고리

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeAsync",
        "lambda:InvokeFunction"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
```

```
}
```

AWS::Lambda::Function~AWS::S3::Bucket

정책 유형

AWS::Lambda::Function 역할에 연결된 고객 관리형 정책.

액세스 카테고리

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~AWS::DynamoDB::Table`

정책 유형

`AWS::Lambda::Function` 역할에 연결된 고객 관리형 정책.

액세스 카테고리

`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~AWS::SQS::Queue

정책 유형

AWS::Lambda::Function 역할에 연결된 고객 관리형 정책.

액세스 카테고리

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~AWS::SNS::Topic

정책 유형

AWS::Lambda::Function 역할에 연결된 고객 관리형 정책.

액세스 카테고리

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
```

```
          "sns:Publish"
        ],
        "Resource": [
          "%{Destination.Arn}"
        ]
      }
    ]
  }
```

AWS::Lambda::Function~AWS::StepFunctions::StateMachine

정책 유형

AWS::Lambda::Function 역할에 연결된 고객 관리형 정책.

액세스 카테고리

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~AWS::Events::EventBus

정책 유형

AWS::Lambda::Function 역할에 연결된 고객 관리형 정책.

액세스 카테고리

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
```

```
          "events:PutEvents"
        ],
        "Resource": [
          "%{Destination.Arn}"
        ]
      }
    ]
}
```

AWS::Lambda::Function~AWS::Location::PlaceIndex

정책 유형

AWS::Lambda::Function 역할에 연결된 고객 관리형 정책.

액세스 카테고리

Read

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "geo:DescribePlaceIndex",
        "geo:GetPlace",
        "geo:SearchPlaceIndexForPosition",
        "geo:SearchPlaceIndexForSuggestions",
        "geo:SearchPlaceIndexForText"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::ApiGatewayV2::Api~AWS::Lambda::Function

정책 유형

AWS::Lambda::Function에 연결된 AWS::Lambda::Permission.

액세스 카테고리

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~AWS::Lambda::Function

정책 유형

AWS::Lambda::Function에 연결된 [AWS::Lambda::Permission].

액세스 카테고리

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~AWS::SQS::Queue

정책 유형

AWS::SQS::Queue에 연결된 [AWS::SQS::QueuePolicy].

액세스 카테고리

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Resource": "%{Destination.Arn}",
```

```
        "Action": "sqs:SendMessage",
        "Condition": {
          "ArnEquals": {
            "aws:SourceArn": "%{Source.Arn}"
          }
        }
      }
    ]
}
```

AWS::SNS::Topic~AWS::Lambda::Function

정책 유형

AWS::Lambda::Function에 연결된 AWS::Lambda::Permission.

액세스 카테고리

Write

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "sns.amazonaws.com",
  "SourceArn": "%{Source.Arn}"
}
```

AWS::SQS::Queue~AWS::Lambda::Function

정책 유형

AWS::Lambda::Function 역할에 연결된 고객 관리형 정책.

액세스 카테고리

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~AWS::Lambda::Function

정책 유형

AWS::Lambda::Function에 연결된 AWS::Lambda::Permission.

액세스 카테고리

Write

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "s3.amazonaws.com",
  "SourceArn": "%{Source.Arn}",
  "SourceAccount": "${AWS::AccountId}"
}
```

AWS::StepFunctions::StateMachine~AWS::Lambda::Function

정책 유형

`AWS::StepFunctions::StateMachine` 역할에 연결된 고객 관리형 정책.

액세스 카테고리

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeAsync",
        "lambda:InvokeFunction"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

`AWS::StepFunctions::StateMachine~AWS::SNS::Topic`

정책 유형

`AWS::StepFunctions::StateMachine` 역할에 연결된 고객 관리형 정책.

액세스 카테고리

`Write`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
```

```
}
```

AWS::StepFunctions::StateMachine~AWS::SQS::Queue

정책 유형

AWS::StepFunctions::StateMachine 역할에 연결된 고객 관리형 정책.

액세스 카테고리

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:SendMessage"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::StepFunctions::StateMachine~AWS::S3::Bucket

정책 유형

AWS::StepFunctions::StateMachine 역할에 연결된 고객 관리형 정책.

액세스 카테고리

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~AWS::DynamoDB::Table

정책 유형

AWS::StepFunctions::StateMachine 역할에 연결된 고객 관리형 정책.

액세스 카테고리

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~AWS::StepFunctions::StateMachine

정책 유형

AWS::StepFunctions::StateMachine 역할에 연결된 고객 관리형 정책.

액세스 카테고리

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~AWS::Events::EventBus

정책 유형

AWS::StepFunctions::StateMachine 역할에 연결된 [고객 관리형 정책](#).

액세스 카테고리

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::AppSync::DataSource~AWS::DynamoDB::Table

정책 유형

AWS::AppSync::DataSource 역할에 연결된 고객 관리형 정책.

액세스 카테고리

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~AWS::Lambda::Function

정책 유형

AWS::AppSync::DataSource 역할에 연결된 고객 관리형 정책.

액세스 카테고리

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeAsync",
        "lambda:InvokeFunction"
      ],
      "Resource": [
```

```
        "%{Destination.Arn}",
        "%{Destination.Arn}:*"
      ]
    }
  ]
}
```

AWS::AppSync::DataSource~AWS::Events::EventBus

정책 유형

AWS::AppSync::DataSource 역할에 연결된 고객 관리형 정책.

액세스 카테고리

Write

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "events:PutEvents"
      ],
      "Resource": [
        "%{Destination.Arn}"
      ]
    }
  ]
}
```

AWS::AppSync::GraphQLApi~AWS::Lambda::Function

정책 유형

AWS::Lambda::Function에 연결된 AWS::Lambda::Permission.

액세스 카테고리

Write

```
{
  "Action": "lambda:InvokeFunction",
  "Principal": "appsync.amazonaws.com",
```

```
  "SourceArn": "arn:${AWS::Partition}:appsync:${AWS::Region}:${AWS::AccountId}:apis/
%{Source.ResourceId}"
}
```

AWS SAMCLI와 함께 사용할 Docker 설치

Docker은 컴퓨터에서 컨테이너를 실행하는 애플리케이션입니다. Docker를 사용하면, AWS SAM는 AWS Lambda과 유사한 로컬 환경을 서버리스 애플리케이션을 구축, 테스트 및 디버그하기 위한 컨테이너로서 제공할 수 있습니다.

> **Note**
>
> Docker은 애플리케이션을 로컬에서 테스트하고 `--use-container` 옵션을 사용하여 배포 패키지를 구축하는 경우에만 필요합니다.

주제

- Docker 설치
- 다음 단계

Docker 설치

Docker을 운영 체제 상에 설치하려면 이 지침을 따르십시오.

Linux

Docker는 CentOS, Debian, Ubuntu등 최신 Linux 배포 버전을 비롯하여 많은 서로 다른 운영 체제에서 사용할 수 있습니다. 특정 운영 체제에 Docker를 설치하는 방법에 대한 자세한 내용은 Docker Docs 웹사이트에서 Get Docker를 참조하세요.

Amazon Linux 2 또는 Amazon Linux 2023 상에 Docker을 설치하기

1. 인스턴스에 설치한 패키지 및 패키지 캐시를 업데이트합니다.

   ```
   $ sudo yum update -y
   ```

2. 최신 Docker Community Edition 패키지를 설치합니다.

   - Amazon Linux 2의 경우 다음을 실행합니다.

```
$ sudo amazon-linux-extras install docker
```

- Amazon Linux 2023의 경우 다음을 실행합니다.

```
$ sudo yum install -y docker
```

3. Docker 서비스를 시작합니다.

```
$ sudo service docker start
```

4. `ec2-user`를 사용하지 않고도 `docker` 명령을 실행할 수 있도록 Docker 그룹에 sudo를 추가합니다.

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

5. 로그아웃 후 다시 로그인해서 새 `docker` 그룹 권한을 취득합니다. 이를 위해 현재 SSH 터미널 창을 닫고 새 창에서 인스턴스를 다시 연결할 수 있습니다. 새 SSH 세션은 적절한 `docker` 그룹 권한을 가져야 합니다.

6. `ec2-user` 없이도 sudo가 Docker 명령을 실행할 수 있는지 확인합니다.

```
$ docker ps
```

다음과 같은 출력결과를 보고 Docker가 설치 및 실행 중임을 확인할 수 있어야 합니다.

```
CONTAINER ID        IMAGE               COMMAND             CREATED
STATUS              PORTS               NAMES
```

> **Note**
>
> Linux에서 호스트 머신과 다른 명령 세트 아키텍처로 Lambda 함수를 빌드하고 실행하려면 추가로 Docker을 구성하는 단계가 있습니다. 예를 들어, `arm64` 기기에서 `x86_64` 함수를 실행하려면 다음 Docker 명령을 실행하여 `docker run --rm --privileged multiarch/qemu-user-static --reset -p yes` 대몬을 구성할 수 있습니다.

Docker의 설치에 문제가 발생하면 [설치 오류 문제 해결]을 잠조하세요. 또는 Docker Docs 웹 사이트에서 [Linux용 사후 설치 단계]의 문제 해결 섹션을 잠조하세요.

macOS

> **Note**
>
> Docker 데스크톱은 공식적으로 지원되지만 AWS SAM CLI 버전 1.47.0부터는 Docker 런타임을 사용하는 경우 대체 버전을 사용할 수 있습니다.

1. Docker 설치

   AWS SAM CLI는 macOS Sierra 10.12 이상에서 Docker 실행을 지원할 수 있습니다. Docker의 설치 방법은 Docs 웹 사이트에서 DockerMac용 데스크톱 Docker 설치를 잠조하세요.

2. 공유 드라이브 구성

   AWS SAM CLI를 사용하려면 프로젝트 디렉터리 또는 상위 디렉터리가 공유 드라이브에 나열되어 있어야 합니다. 이 경우 파일 공유가 필요할 수 있습니다. 자세한 내용은 문서의 Docker볼륨 마운팅에 필요한 파일 공유 문제 해결 항목을 잠조하세요.

3. 설치 확인

   Docker을 설치한 후 제대로 작동하는지 확인하십시오. 또한 명령줄에서 Docker 명령을 실행할 수 있는지 확인하십시오(예: `docker ps`). 컨테이너를 설치하거나 가져오거나 가져올 필요가 없습니다. 필요에 따라 AWS SAM CLI가 자동으로 이 작업을 수행합니다.

Docker의 설치 문제가 발생하는 경우 추가 문제 해결 팁은 Docs 웹 사이트의 Docker 문제 해결 및 진단 섹션을 잠조하세요.

Windows

> **Note**
>
> AWS SAM은 Docker 데스크톱을 공식적으로 지원합니다. 그러나 AWS SAM CLI 버전 1.47.0부터는 Docker 런타임을 사용하는 한 대체 제품을 사용할 수 있습니다.

1. Docker을 설치합니다.

   Docker 데스크톱은 최신 Windows 운영 체제를 지원합니다. 레거시 버전의 Windows의 경우 Docker Toolbox를 사용할 수 있습니다. 올바른 Docker 설치 단계를 위해 사용 중인 Windows 버전을 선택하십시오.

- Docker을 [Windows 10용으로 설치하려면 Docker Docs 웹 사이트]에서 Docker Windows용 데스크톱 설치를 참조하세요.
- Docker을 이전 버전의 Windows용으로 설치하려면 [Toolbox GitHubDocker 리포지토리]의 DockerToolbox를 참조하세요.

2. 공유 드라이브를 구성하십시오.

   AWS SAM CLI를 사용하려면 프로젝트 디렉터리 또는 상위 디렉터리가 공유 드라이브에 나열되어 있어야 합니다. 경우에 따라 Docker이 제대로 작동하게 하려면 드라이브를 공유해야 합니다.

3. 설치를 확인합니다.

   Docker을 설치한 후 제대로 작동하는지 확인하십시오. 또한 명령줄에서 Docker 명령을 실행할 수 있는지 확인하십시오(예: docker ps). 컨테이너를 설치하거나 가져오거나 가져올 필요가 없습니다. 필요에 따라 AWS SAM CLI가 자동으로 이 작업을 수행합니다.

Docker의 설치 문제가 발생하는 경우 추가 문제 해결 팁은 Docs 웹 사이트의 Docker 문제 해결 및 진단 섹션을 참조하세요.

다음 단계

AWS SAM CLI의 설치 방법은 AWS SAM CLI 설치을 참조하세요.

AWS SAM CLI와 함께 사용할 Finch 설치하기

Finch은(는) Docker에 대한 대안, 특히 macOS 및 Linux 시스템에 유용한의 AWS 컨테이너 개발 도구입니다. 자세한 내용은 Finch on GitHub을 참조하세요. Finch를 사용하면, AWS SAM는 AWS Lambda과 유사한 로컬 환경을 서버리스 애플리케이션을 구축, 테스트 및 디버그하기 위한 컨테이너로서 제공할 수 있습니다.

> **Note**
>
> 애플리케이션을 로컬로 테스트하고, `--use-container` 옵션을 사용하여 배포 패키지를 빌드하며, OCI 이미지 함수를 빌드하려면 Finch이(가) 필요합니다. AWS SAM CLI 버전 1.145.0부터 AWS SAM CLI는 Docker의 대체 컨테이너 런타임으로 Finch을(를) 지원합니다..

주제

- Finch 설치하기
- AWS SAM CLI로 Finch 구성하기
- 설치 확인하기
- Finch 중지하기
- 문제 해결
- 다음 단계

Finch 설치하기

Finch을 운영 체제 상에 설치하려면 이 지침을 따르십시오.

사전 조건

> **Note**
>
> AWS SAM CLI 내 Finch 통합은 macOS 및 Linux 시스템에서 사용할 수 있습니다. Windows 사용자의 경우 Finch 기능을 위해 가상 머신 또는 Linux 환경을 사용하는 것이 좋습니다.

macOS 버전:

- macOS 15 Sequoia
- macOS 14 Sonoma

> **Note**
>
> Finch은(는) 최신 두 버전의 macOS를 지원합니다.

(Linux):

- Linux 커널 v4.x+

Finch 설치하기

macOS

macOS에 Finch를 설치하려면

1. `brew` CLI를 사용하여 Finch을(를) 설치합니다. 프롬프트가 표시되면 macOS 암호를 입력합니다.

   ```
   $ brew install finch
   ```

2. 설치된 Finch 버전을 확인합니다.

   ```
   $ finch --version
   ```

3. Finch 가상 머신을 초기화합니다(처음 설정).

   ```
   $ finch vm init
   ```

4. Finch 가상 머신을 시작합니다.

   ```
   $ finch vm start
   ```

5. Finch 가상 머신의 상태를 확인합니다.

   ```
   $ finch vm status
   ```

6. Finch이(가) 올바르게 실행되는지 확인합니다.

   ```
   $ finch --info
   $ finch ps
   ```

자세한 내용은 Finch on macOS 설치하기를 참조하세요.

Linux

Amazon Linux, Ubuntu 또는 일반 설치 방법을 사용하여 Finch을(를) 설치할 수 있습니다. 자세한 내용은 Finch on Linux 설치하기를 참조하세요.

AWS SAM CLI로 Finch 구성하기

Finch을(를) 설치한 후, AWS SAM CLI는 Finch이(가) 실행되지 않을 때 Docker을(를) 컨테이너 런타임으로 자동으로 감지하고 사용합니다.

> **Note**
>
> Finch은(는) Docker에 대한 폴백으로 작동합니다. Docker 및 Finch 모두 설치 및 실행 중인 경우, AWS SAM CLI는 기본적으로 Docker에 우선순위를 둡니다.

관리자 환경 설정 구성(macOS)

관리자 환경 설정 파일을 생성하여 AWS SAM CLI가 특정 컨테이너 런타임을 기본값으로 사용하도록 구성할 수 있습니다. 이 시스템 전체 환경 설정은 기본 대체 동작을 재정의합니다.

사용 가능한 구성 옵션

- Finch을(를) 선호하는 컨테이너 런타임으로 설정:

```
$ sudo /usr/libexec/PlistBuddy -c "Add :DefaultContainerRuntime string finch" /
Library/Preferences/com.amazon.samcli.plist
```

- Docker을(를) 선호하는 컨테이너 런타임으로 설정:

```
$ sudo /usr/libexec/PlistBuddy -c "Add :DefaultContainerRuntime string docker" /
Library/Preferences/com.amazon.samcli.plist
```

- 환경 설정 제거(기본 동작으로 돌아가기):

```
$ sudo /usr/libexec/PlistBuddy -c "Remove :DefaultContainerRuntime" /Library/
Preferences/com.amazon.samcli.plist
```

> **Note**
>
> 관리자 환경 설정이 설정되지 않은 경우, AWS SAM CLI는 기본적으로 Docker을(를) 사용하며, Docker이(가) 실행되지 않을 때 Finch을(를) 폴백으로 사용합니다.

디렉터리 탑재 차이점(macOS)

macOS에서 AWS SAM 로컬 명령은 프로젝트를 홈 디렉터리 (~) 또는 `/Volumes` 밖에 두면 실패합니다. 이 디자인은 보안상의 이유로 사용자 모르게 자동 디렉터리 탑재를 제한합니다.

완화 작업

- 옵션 1: Finch 구성을 텍스트 편집기 `~/.finch/finch.yaml`에서 열고, `additional_directories` 섹션 아래에 프로젝트 경로를 추가한 후, Finch 가상 머신을 다시 시작하세요. 추가 정보는 Finch 설명서를 참조하세요.

  ```
  $ finch vm stop
  $ finch vm start
  ```

- 옵션 2: 프로젝트를 홈 디렉터리로 이동합니다.

설치 확인하기

Finch을(를) 설치하고 시작한 후 AWS SAM CLI에서 작동하는지 확인합니다.

기본 확인

AWS SAM CLI가 Finch을(를) 사용할 수 있는지 확인하려면 다음 명령을 실행하세요:

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

컨테이너 환경에서 애플리케이션을 빌드하려면 Finch을(를) 사용하는 AWS SAM CLI를 확인해야 합니다.

> **Note**
>
> Finch은(는) Docker에 대한 폴백으로 작동합니다. Docker이(가) 실행 중인 경우, AWS SAM CLI는 기본적으로 Docker에 우선순위를 둡니다.

Finch을(를) 폴백으로 테스트하기

Docker이(가)가 실행 중일 때 Finch을(를) 폴백으로 테스트하려면, 먼저 Docker을(를) 중지해야 합니다.

Finch 폴백 기능 테스트하기

1. 운영 체제에 맞는 적절한 명령어를 사용하여 Docker을(를) 중지하세요.

   macOS:

   ```
   $ docker stop $(docker ps -q)
   $ pkill -f "Docker Desktop"
   ```

   Docker이(가) 중지되었는지 확인하세요.

   ```
   $ docker ps
   ```

   (Linux):

   ```
   $ docker stop $(docker ps -q)
   $ sudo systemctl stop docker
   ```

   Docker이(가) 중지되었는지 확인하세요.

   ```
   $ sudo systemctl status docker
   ```

2. AWS SAM CLI가 자동으로 Finch을(를) 폴백으로 사용하는지 확인하려면 `sam build --use-container` 명령어를 실행하세요.

Finch 중지하기

Finch을(를) 중지해야 하는 경우, 다음 명령어를 사용하세요:

macOS:

```
$ finch vm stop
$ sudo pkill -f finch
```

Finch이(가) 중지되었는지 확인하세요.

```
$ finch vm status
```

(Linux):

```
$ sudo systemctl stop finch
$ sudo pkill -f finch
```

Finch이(가) 중지되었는지 확인하세요.

```
$ sudo systemctl status finch
```

문제 해결

Finch에서 문제가 발생하면:

- 운영 체제가 필수 조건을 충족하는지 확인하세요.
- 시스템에서 가상화가 활성화되었는지 확인하세요.
- macOS의 경우, 가상 머신에 충분한 디스크 스페이스가 있는지 확인하세요.
- Linux의 경우, `systemctl` 명령어를 실행할 수 있는 적절한 권한이 있는지 확인하세요.
- Amazon Elastic Container Registry (Amazon ECR)에 다중 아키텍처 이미지를 푸시할 때 "미디어 유형 지원 안됨" 오류와 함께 CloudFormation 배포 실패가 발생하면, 다중 아키텍처 이미지를 사용하는 대신 단일 이미지 아티팩트를 생성하세요. 이것은 Finch이(가) 이미지 인덱스와 이미지를 모두 Amazon ECR로 푸시하는 반면, Docker은(는) 이미지만 푸시하기 때문에 발생합니다.
- 대상 함수 아키텍처가 Linux 호스트 머신 아키텍처와 일치하지 않아 교차 플랫폼 빌드가 실패하면, `sudo finch run --privileged --rm tonistiigi/binfmt:master --install all`을(를) 실행하여 교차 플랫폼 에뮬레이션을 활성화하세요.

추가 문제 해결은 Finch 설명서 또는 섹션을 참조하세요AWS SAMCLI 문제 해결.

다음 단계

이제 Finch이(가) 설치되었으므로, AWS SAM CLI와 함께 사용하여 다음을 수행할 수 있습니다.

- `sam build --use-container`을(를) 사용하여 컨테이너로 서버리스 애플리케이션 빌드 자세한 내용은 the section called “sam build” 섹션을 참조하십시오.
- `sam local invoke`을(를) 사용하여 로컬에서 Lambda 함수 테스트 자세한 내용은 the section called “sam local invoke” 섹션을 참조하십시오.
- `sam local start-api`을(를) 사용하여 로컬 API Gateway 시작 자세한 내용은 the section called “sam local start-api” 섹션을 참조하십시오.

- `sam local start-lambda`을(를) 사용하여 로컬 Lambda 엔드포인트 시작 자세한 내용은 the section called “sam local start-lambda” 섹션을 참조하십시오.

컨테이너와 함께 AWS SAM CLI를 사용하는 방법에 대한 자세한 내용은 AWS SAM을 사용한 기본 빌드을(를) 참조하세요..

의 이미지 리포지토리 AWS SAM

AWS SAM 는 컨테이너 이미지 빌드를 통해 서버리스 애플리케이션의 지속적 통합 및 지속적 전송(CI/CD) 작업을 간소화합니다. 에서 AWS SAM 제공하는 이미지에는 AWS SAM 명령줄 인터페이스(CLI)와 지원되는 여러 AWS Lambda 런타임을 위한 빌드 도구가 포함됩니다. 따라서 AWS SAMCLI를 사용하여 서버리스 애플리케이션을 쉽게 구축하고 패키징할 수 있습니다. 이러한 이미지를 CI/CD 시스템과 함께 사용하여 AWS SAM 애플리케이션 구축 및 배포를 자동화할 수 있습니다. 예시는 CI/CD 시스템 및 파이프라인을 사용하여 배포 섹션을 참조하세요.

AWS SAM 빌드 컨테이너 이미지 URIs 해당 이미지에 AWS SAMCLI 포함된 버전의 로 태그가 지정됩니다. 태그 없는 URI를 지정하면, 최신 버전이 사용됩니다. 예를 들어 `public.ecr.aws/sam/build-nodejs20.x`는 최신 이미지를 사용합니다. 하지만 `public.ecr.aws/sam/build-nodejs20.x:1.24.1`는 AWS SAM CLI 버전 1.24.1을 포함하는 이미지를 사용합니다.

버전 1.33.0부터 지원되는 런타임에 `x86_64` 및 `arm64` 컨테이너 이미지를 AWS SAMCLI모두 사용할 수 있습니다. 자세한 내용은 개발자 가이드의 AWS Lambda Lambda 런타임을 참조하세요.

> **Note**
>
> 버전 1.22.0 이전에는가 컨테이너 이미지를 AWS SAMCLI 가져온 기본 리포지토리가 AWS SAMCLI DockerHub였습니다. 버전 1.22.0부터 기본 리포지토리가 Amazon Elastic Container Registry 퍼블릭 (Amazon ECR 퍼블릭)으로 변경되었습니다. 현재 기본 사항이 아닌 다른 리포지토리에서 컨테이너 이미지를 가져오려면 sam build 명령을 --build-image 옵션과 함께 사용할 수 있습니다. 이 항목의 끝에 있는 예제는 DockerHub 리포지토리 이미지를 사용하여 애플리케이션을 구축하는 방법을 보여줍니다.

이미지 리포지토리 URI

다음 표에는 서버리스 애플리케이션을 빌드하고 패키징하는 데 사용할 수 있는 Amazon ECR 퍼블릭 빌드 컨테이너 이미지의 URIs가 나열되어 있습니다 AWS SAM.

> **Note**
>
> Amazon ECR 퍼블릭은 DockerHub AWS SAM버전 1.22.0부터 CLI을 대체했습니다. 이전 버전의를 사용하는 경우 업그레이드하는 AWS SAMCLI것이 좋습니다.

런타임	Amazon ECR 퍼블릭
사용자 지정 런타임(AL2023)	public.ecr.aws/sam/build-provided.al2023
사용자 지정 런타임(AL2)	public.ecr.aws/sam/build-provided.al2
사용자 지정 런타임	public.ecr.aws/sam/build-provided
Java 25	public.ecr.aws/sam/build-java25
Java 21	public.ecr.aws/sam/build-java21
Java 17	public.ecr.aws/sam/build-java17
Java 11	public.ecr.aws/sam/build-java11
Java 8	public.ecr.aws/sam/build-java8
.NET 9	public.ecr.aws/sam/build-dotnet9
.NET 8	public.ecr.aws/sam/build-dotnet8
.NET 7	public.ecr.aws/sam/build-dotnet7
.NET 6	public.ecr.aws/sam/build-dotnet6
Node.js 24	public.ecr.aws/sam/build-nodejs24.x
Node.js 22	public.ecr.aws/sam/build-nodejs22.x
Node.js 20	public.ecr.aws/sam/build-nodejs20.x
Node.js 18	public.ecr.aws/sam/build-nodejs18.x
Node.js 16	public.ecr.aws/sam/build-nodejs16.x

런타임	Amazon ECR 퍼블릭
Python 3.14	public.ecr.aws/sam/build-python3.14
Python 3.13	public.ecr.aws/sam/build-python3.13
Python 3.12	public.ecr.aws/sam/build-python3.12
Python 3.11	public.ecr.aws/sam/build-python3.11
Python 3.10	public.ecr.aws/sam/build-python3.10
Python 3.9	public.ecr.aws/sam/build-python3.9
Python 3.8	public.ecr.aws/sam/build-python3.8
Ruby 3.4	public.ecr.aws/sam/build-ruby3.4
Ruby 3.3	public.ecr.aws/sam/build-ruby3.3
Ruby 3.2	public.ecr.aws/sam/build-ruby3.2

예제

다음 두 가지 예시 명령어는 이미지 리포지토리의 컨테이너 이미지를 사용하여 애플리케이션을 빌드합니다.

Amazon ECR에서 가져온 컨테이너 이미지를 사용하여 Node.js 24 애플리케이션을 빌드하세요

```
$ sam build --use-container --build-image public.ecr.aws/sam/build-nodejs24.x
```

Amazon ECR에서 가져온 Python 3.14 컨테이너 이미지를 사용하여 함수 리소스를 구축하세요

```
$ sam build --use-container --build-image Function1=public.ecr.aws/sam/build-python3.14
```

AWS SAM에서 서버리스 애플리케이션의 점진적 배포

AWS Serverless Application Model(AWS SAM)는 CodeDeploy와 함께 기본 제공되므로 점진적인 AWS Lambda 배포를 제공합니다. 몇 줄의 구성만으로 AWS SAM은 귀하를 위해 다음을 수행합니다.

- Lambda 함수의 새 버전을 배포하고 새 버전을 가리키는 별칭을 자동으로 생성합니다.
- 예상대로 작동한다고 확신할 때까지 고객 트래픽을 새 버전으로 점진적으로 이동시킵니다. 업데이트가 제대로 작동하지 않는 경우 변경 내용을 롤백할 수 있습니다.
- 새로 배포된 코드가 올바르게 구성되었고 애플리케이션이 예상대로 작동하는지 확인하는 사전 트래픽 및 사후 트래픽 테스트 함수를 정의합니다.
- CloudWatch 경보가 트리거되면 배포를 자동으로 롤백합니다.

> **Note**
>
> AWS SAM 템플릿을 통해 점진적 배포를 활성화하면 CodeDeploy 리소스가 자동으로 생성됩니다. AWS Management 콘솔를 통해 CodeDeploy 리소스를 직접 볼 수 있습니다.

예시

다음 예제는 CodeDeploy를 사용하여 고객을 새로 배포된 Lambda 함수 버전으로 점진적으로 전환하는 방법을 보여줍니다.

```
Resources:
MyLambdaFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: index.handler
    Runtime: nodejs20.x
    CodeUri: s3://bucket/code.zip

    AutoPublishAlias: live

    DeploymentPreference:
      Type: Canary10Percent10Minutes
      Alarms:
        # A list of alarms that you want to monitor
        - !Ref AliasErrorMetricGreaterThanZeroAlarm
        - !Ref LatestVersionErrorMetricGreaterThanZeroAlarm
      Hooks:
        # Validation Lambda functions that are run before & after traffic shifting
        PreTraffic: !Ref PreTrafficLambdaFunction
        PostTraffic: !Ref PostTrafficLambdaFunction
```

AWS SAM 템플릿의 이러한 수정 내용은 다음을 수행합니다.

- `AutoPublishAlias`: 이 속성을 추가하고 별칭 이름을 지정하면, AWS SAM:
  - Lambda 함수의 Amazon S3 URI에 대한 변경 내용을 기반으로 새 코드가 배포되는 시기를 감지합니다.
  - 최신 코드를 사용하여 해당 함수의 업데이트된 버전을 생성하고 게시합니다.
  - 제공한 이름으로 별칭을 생성하고(별칭이 이미 존재하지 않는 경우) Lambda 함수의 업데이트된 버전을 가리킵니다. 이를 활용하려면 함수 호출 시 별칭 한정자를 사용해야 합니다. 귀하가 Lambda 함수 버전 관리 및 별칭에 익숙하지 않은 경우, 함수 버전 관리 및 별칭AWS Lambda을 참조하세요.
- `Deployment Preference Type`: 이전 예시에서는 귀하의 고객 트래픽의 10%가 즉시 새 버전으로 이동합니다. 10분 후 모든 트래픽이 새 버전으로 이동합니다. 하지만 사전 트래픽 또는 사후 트래픽 테스트가 실패하거나 CloudWatch 경보가 트리거되는 경우 CodeDeploy는 배포를 롤백합니다. 다음과 같은 방법으로 버전 간에 트래픽을 이동하는 방법을 지정할 수 있습니다.
  - `Canary`: 트래픽이 2 증분씩 이동합니다. 사전 정의된 canary 옵션 중에서 선택할 수 있습니다. 이 옵션은 나머지 트래픽이 두 번째 증분으로 이동하기 전에 첫 증분에서 업데이트된 Lambda 함수 버전으로 이동할 트래픽 비율(%)과 간격(분)을 지정합니다.
  - `Linear`: 트래픽이 동일한 증분으로 이동하며 각 증분 간 시간(분)은 동일합니다. 각 증분에서 이동할 트래픽 비율(%)과 각 증분 간의 시간 간격(분)을 지정하는 사전 정의된 선형 옵션에서 선택할 수 있습니다.
  - `AllAtOnce`: 모든 트래픽이 기존 Lambda 함수에서 업데이트된 Lambda 함수 버전으로 한번에 이동합니다.

다음 표에는 예제에 사용된 옵션 외에 사용할 수 있는 기타 트래픽 이동 옵션이 요약되어 있습니다.

배포 기본 설정 유형
Canary10Percent30Minutes
Canary10Percent5Minutes
Canary10Percent10Minutes
Canary10Percent15Minutes
Linear10PercentEvery10Minutes
Linear10PercentEvery1Minute

배포 기본 설정 유형
Linear10PercentEvery2Minutes
Linear10PercentEvery3Minutes
AllAtOnce

- `Alarms`: 이들은 배포에서 발생한 오류로 인해 트리거되는 CloudWatch 경보입니다. 이들은 발생하면 귀하의 배포를 자동으로 롤백합니다. 예를 들어, 배포하려는 업데이트된 코드로 인해 애플리케이션 내에서 오류가 발생하는 경우입니다. 또 다른 예는 AWS Lambda 또는 귀하가 지정한 사용자 지정 CloudWatch 지표가 경보 임계값을 위반한 경우입니다.
- `Hooks`: 이들은 새 버전으로 트래픽 이동이 시작되기 전과 트래픽 이동이 완료된 후에 검사를 실행하는 사전 트래픽 및 사후 트래픽 테스트 함수입니다.
  - `PreTraffic`: 트래픽 이동이 시작되기 전에 CodeDeploy는 사전 트래픽 후크 Lambda 함수를 호출합니다. 이 Lambda 함수는 CodeDeploy를 다시 호출하고 성공 또는 실패를 표시해야 합니다. 함수가 실패하면 함수가 중단되고 실패를 다시 CloudFormation에 보고합니다. 함수가 성공하면 CodeDeploy는 트래픽 이동을 진행합니다.
  - `PostTraffic`: 트래픽 이동이 완료되면 CodeDeploy는 사후 트래픽 후크 Lambda 함수를 호출합니다. 이는 함수가 CodeDeploy를 다시 호출하여 성공 또는 실패를 보고해야 하는 사전 트래픽 후크와 유사합니다. 사후 트래픽 후크를 사용하여 통합 테스트 또는 기타 검증 작업을 실행할 수 있습니다.

자세한 내용을 알아보려면 안전 배포 SAM 참조를 참조하세요.

Lambda 함수를 처음으로 점진적으로 배포

Lambda 함수를 점진적으로 배포하는 경우 CodeDeploy에서 트래픽을 이동하려면 이전에 배포한 함수 버전이 필요합니다. 따라서 첫 번째 배포는 다음 두 단계로 수행해야 합니다.

- 1단계: Lambda 함수를 배포하고 `AutoPublishAlias`를 사용하여 자동으로 별칭을 생성합니다.
- 2단계: `DeploymentPreference`를 사용하여 점진적 배포를 수행합니다.

첫 번째 점진적 배포를 두 단계로 수행하면 CodeDeploy는 트래픽을 이동할 수 있는 이전 Lambda 함수 버전을 제공합니다.

1단계: Lambda 함수 배포

```
Resources:
MyLambdaFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: index.handler
    Runtime: nodejs20.x
    CodeUri: s3://bucket/code.zip

    AutoPublishAlias: live
```

2단계: 점진적 배포 수행

```
Resources:
MyLambdaFunction:
  Type: AWS::Serverless::Function
  Properties:
    Handler: index.handler
    Runtime: nodejs20.x
    CodeUri: s3://bucket/code.zip

    AutoPublishAlias: live

    DeploymentPreference:
      Type: Canary10Percent10Minutes
      Alarms:
        # A list of alarms that you want to monitor
        - !Ref AliasErrorMetricGreaterThanZeroAlarm
        - !Ref LatestVersionErrorMetricGreaterThanZeroAlarm
      Hooks:
        # Validation Lambda functions that are run before and after traffic shifting
        PreTraffic: !Ref PreTrafficLambdaFunction
        PostTraffic: !Ref PostTrafficLambdaFunction
```

자세히 알아보기

점진적 배포를 구성하는 실제 예제는 전체 AWS SAM 워크숍의 [모듈 5 - Canary 배포]를 참조하세요.

AWS SAM에 대한 중요 참조 정보

이 섹션에는 AWS Serverless Application Model(AWS SAM)에 대한 중요 참고 및 공지 사항이 포함되어 있습니다.

주제

- 2023년 중요 정보

2023년 중요 정보

2023년 10월

AWS SAMCLI가 Python 3.7에 대한 지원 중단

2023년 10월 20일 게시

Python 3.7는 2023년 6월에 수명 종료 상태를 지정 받았습니다. AWS SAM CLI는 2023년 10월 24일에 Python 3.7에 대한 지원을 중단할 예정입니다. 자세한 내용은 aws-sam-cli 리포지토리의 GitHub공지를 참조하세요.

이 변경은 다음 사용자에게 영향을 미칩니다.

- Python 3.7를 사용하고 AWS SAM CLI를 통해 `pip`를 설치하는 경우입니다.
- `aws-sam-cli`을 라이브러리로 사용하고 Python 3.7를 사용하여 애플리케이션을 구축하는 경우입니다.

다른 방법을 통해 AWS SAM CLI를 설치하고 관리하는 경우에는 영향을 받지 않습니다.

영향을 받는 사용자의 경우 개발 환경을 Python 3.8 또는 신규 버전으로 업그레이드하는 것이 좋습니다.

이 변경은 Python 3.7 AWS Lambda 런타임 환경에 대한 지원에는 영향을 미치지 않습니다. 자세한 내용은 개발자 가이드의 AWS Lambda런타임 중단 정책을 참조하세요.

AWS SAM CLI Terraform 지원

이 섹션에서는 Terraform 프로젝트 및 Terraform 클라우드에서 AWS Serverless Application Model 명령줄 인터페이스(AWS SAMCLI)를 사용하는 방법을 다룹니다.

피드백을 제공하고 기능 요청을 제출하려면 GitHub 문제를 생성하세요.

주제

- AWS SAM CLI에 대한 Terraform 지원과 함께 시작하기
- 로컬 디버깅 및 테스트에 Terraform과 함께 AWS SAMCLI 사용하기
- 로컬 디버깅 및 테스트에 Serverless.tf와 함께 AWS SAMCLI 사용
- Terraform 참조로AWS SAMCLI 사용
- Terraform에 대한 AWS SAMCLI지원이란 무엇입니까?

AWS SAM CLI에 대한 Terraform 지원과 함께 시작하기

이 주제에서는에서 AWS Serverless Application Model 명령줄 인터페이스(AWS SAMCLI) 사용을 시작하는 방법을 다룹니다Terraform.

피드백을 제공하고 기능 요청을 제출하려면 GitHub 문제를 생성하세요.

주제

- AWS SAM CLI Terraform사전 조건
- Terraform과 함께 AWS SAMCLI 명령 사용
- Terraform 프로젝트 설정
- Terraform Cloud에 대해 설정

AWS SAM CLI Terraform사전 조건

Terraform 프로젝트에서 AWS SAMCLI를 사용하려면 모든 사전 조건을 완료하세요.

1. AWS SAM CLI 설치 또는 업그레이드

   AWS SAMCLI가 설치되어 있는지 확인하려면 다음을 실행합니다.

```
$ sam --version
```

AWS SAMCLI가 이미 설치된 경우 출력에 버전이 표시됩니다. 최신 버전으로 업그레이드하려면 AWS SAMCLI업그레이드 섹션을 참조하세요.

모든 사전 조건과 함께 AWS SAMCLI를 설치하는 것에 대한 지침은 AWS SAM CLI 설치 섹션을 참조하세요.

2. Terraform 설치

   Terraform 설치 여부를 확인하려면 다음을 실행합니다.

```
$ terraform -version
```

   Terraform을 설치하려면 Terraform 레지스트리에서 Terraform 설치를 참조하세요.

3. 로컬 테스트를 위한 Docker 설치

   AWS SAMCLI에는 로컬 테스트를 위해 Docker가 필요합니다. Docker을 설치하려면 AWS SAMCLI와 함께 사용할 Docker 설치 섹션을 참조하세요.

Terraform과 함께 AWS SAMCLI 명령 사용

지원되는 AWS SAMCLI 명령을 실행할 때는 `--hook-name` 옵션을 사용하고 `terraform` 값을 제공합니다. 다음은 예제입니다.

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

다음과 같이 AWS SAMCLI 구성 파일에서 이 옵션을 구성할 수 있습니다.

```
hook_name = "terraform"
```

Terraform 프로젝트 설정

Terraform 프로젝트에서 AWS SAM CLI를 사용하려면 이 항목의 단계를 완료합니다.

Terraform 프로젝트 외부에서 AWS Lambda 아티팩트를 빌드하는 경우 추가 설정이 필요하지 않습니다. 로컬 디버깅 및 테스트에 Terraform과 함께 AWS SAMCLI 사용하기 사용을 시작하려면 단원을 참조하십시오 AWS SAMCLI.

Terraform 프로젝트 내에 Lambda 아티팩트을 빌드하는 경우 다음을 수행해야 합니다.

1. Python 3.8 이상 버전 설치
2. Make 도구를 설치합니다.
3. Terraform 프로젝트 내에서 Lambda 아티팩트 빌드 로직을 정의합니다.
4. 빌드 로직의 AWS SAMCLI를 알릴 `sam metadata` 리소스를 정의합니다.
5. `sam build` 명령을 사용하여 Lambda 아티팩트를 AWS SAMCLI 빌드합니다.

Python 3.8 이상 버전 설치

Python에서 사용하려면 3.8 이상이 필요합니다 AWS SAMCLI. `sam build`를 실행하면 AWS SAMCLI가 Lambda 아티팩트를 빌드하기 위한 Python 명령이 포함된 `makefiles`를 생성합니다.

설치 지침은 Python 초급 가이드의 [Python 다운로드] 섹션을 참조하세요.

다음을 실행하여 Python 3.8 이상이 컴퓨터 경로에 추가되었는지 확인합니다.

```
$ python --version
```

출력에는 3.8 이상의 Python 버전이 표시되어야 합니다.

Make 도구를 설치합니다.

GNU Make(은)는 프로젝트의 실행 파일 및 기타 비소스 파일 생성을 제어하는 도구입니다. AWS SAMCLI는 이 도구를 사용하여 Lambda 아티팩트를 빌드하는 `makefiles`를 생성합니다.

Make가 로컬 시스템에 아직 설치되지 않은 경우, 계속 진행하기 전에 설치하세요.

Windows의 경우 Chocolatey를 사용하여 설치할 수 있습니다. 자세한 지침은 Windows에서 'Make'를 설치하고 사용하는 방법의 Chocolatey 사용하기 섹션을 참조하세요.

Lambda 아티팩트 빌드 로직 정의

`null_resource` Terraform 리소스 유형을 사용하여 Lambda 빌드 로직을 정의합니다. 다음은 사용자 지정 빌드 스크립트를 사용하여 Lambda 함수를 빌드하는 예제입니다.

```
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"
    }
}
```

sam metadata 리소스를 정의합니다.

`sam metadata` 리소스는 AWS SAMCLI에 Lambda 아티팩트를 찾는 데 필요한 정보를 제공하는 `null_resource` Terraform 리소스 유형입니다. 프로젝트의 각 Lambda 함수 또는 계층에는 고유한 `sam metadata` 리소스가 필요합니다. 이 리소스 유형에 대해 자세히 알아보려면 Terraform 레지스트리의 null_resource를 참조하세요.

sam metadata 리소스를 정의하려면

1. 리소스가 sam metadata 리소스임을 식별하는 `sam_metadata_`로 시작하는 리소스의 이름을 지정합니다.
2. 리소스의 `triggers` 블록 내에서 Lambda 아티팩트 속성을 정의합니다.
3. depends_on 인수와 함께 Lambda 빌드 로직을 포함하는 `null_resource`를 지정합니다.

   다음은 템플릿의 예입니다.

```
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
  ]
}
```

   다음은 `sam metadata` 리소스의 예입니다.

```
resource "null_resource" "sam_metadata_aws_lambda_function_publish_book_review" {
```

```
    triggers = {
        resource_name = "aws_lambda_function.publish_book_review"
        resource_type = "ZIP_LAMBDA_FUNCTION"
        original_source_code = "${local.lambda_src_path}"
        built_output_path = "${local.building_path}/${local.lambda_code_filename}"
    }
    depends_on = [
        null_resource.build_lambda_function
    ]
}
```

`sam metadata` 리소스의 콘텐츠는 Lambda 리소스 유형(함수 또는 계층)과 패키지 유형(ZIP 또는 이미지)에 따라 달라집니다. 예에 사용된 SSML에 대한 자세한 내용은 sam 메타데이터 리소스 섹션을 참조하세요.

`sam metadata` 리소스를 구성하고 지원되는 AWS SAMCLI 명령을 사용하면 AWS SAMCLI는 AWS SAMCLI 명령을 실행하기 전에 메타데이터 파일을 생성합니다. 이 파일을 생성한 후에는 미래 AWS SAMCLI 명령과 함께 `--skip-prepare-infra` 옵션을 사용하여 메타데이터 생성 프로세스를 건너뛰고 시간을 절약할 수 있습니다. 이 옵션은 새 Lambda 함수 또는 새 API 엔드포인트 생성과 같이 인프라를 변경하지 않은 경우에만 사용해야 합니다.

AWS SAMCLI를 사용하여 Lambda 아티팩트를 빌드합니다.

`sam build` 명령을 사용하여 Lambda 아티팩트를 AWS SAMCLI 빌드합니다. `sam build`를 실행하면 AWS SAMCLI는 다음과 같은 작업을 수행합니다.

1. Lambda 리소스에 대해 알아보고 위치를 찾기 위해 Terraform 프로젝트에서 `sam metadata` 리소스를 찾습니다.
2. Lambda 빌드 로직을 시작하여 Lambda 아티팩트를 빌드합니다.
3. `sam local` 명령과 함께 사용할 Terraform 프로젝트를 구성하는 `.aws-sam` 디렉터리를 AWS SAMCLI 생성합니다.

sam build로 빌드하려면

1. Terraform 루트 모듈이 포함된 디렉터리에서 다음을 실행합니다.

```
$ sam build --hook-name terraform
```

2. 특정 Lambda 함수 또는 계층을 빌드하려면 다음을 실행합니다.

```
$ sam build --hook-name terraform lambda-resource-id
```

Lambda 리소스 ID는 aws_lambda_function.list_books 또는 module.list_book_function.aws_lambda_function.this[0]과 같은 Lambda 함수 이름 또는 전체 Terraform 리소스 주소일 수 있습니다.

함수 소스 코드나 기타 Terraform 구성 파일이 Terraform 루트 모듈이 포함된 디렉터리 외부에 있는 경우 위치를 지정해야 합니다. --terraform-project-root-path 옵션을 사용하여 이러한 파일이 포함된 최상위 디렉터리의 절대 경로 또는 상대 경로를 지정합니다. 다음은 예제입니다.

```
$ sam build --hook-name terraform --terraform-project-root-path ~/projects/terraform/demo
```

컨테이너를 사용하여 빌드하기

sam build 명령을 실행할 AWS SAMCLI 때 로컬 Docker 컨테이너를 사용하여 애플리케이션을 빌드 AWS SAMCLI하도록를 구성할 수 있습니다.

> **Note**
>
> Docker를 설치하고 구성합니다. 지침은 [AWS SAMCLI와 함께 사용할 Docker 설치](#) 섹션을 참조하세요.

컨테이너를 사용하여 빌드하려면

1. Terraform,Python, Make 도구가 포함된 Dockerfile를 만듭니다. Lambda 함수 런타임도 포함해야 합니다.

   다음은 한 Dockerfile 예입니다.

```
FROM public.ecr.aws/amazonlinux/amazonlinux:2

RUN yum -y update \
    && yum install -y unzip tar gzip bzip2-devel ed gcc gcc-c++ gcc-gfortran \
    less libcurl-devel openssl openssl-devel readline-devel xz-devel \
    zlib-devel glibc-static libcxx libcxx-devel llvm-toolset-7 zlib-static \
    && rm -rf /var/cache/yum
```

```
RUN yum -y install make \
    && yum -y install zip

RUN yum install -y yum-utils \
    && yum-config-manager --add-repo https://rpm.releases.hashicorp.com/
AmazonLinux/hashicorp.repo \
    && yum -y install terraform \
    && terraform --version

# AWS Lambda Builders
RUN amazon-linux-extras enable python3.8
RUN yum clean metadata && yum -y install python3.8
RUN curl -L get-pip.io | python3.8
RUN pip3 install aws-lambda-builders
RUN ln -s /usr/bin/python3.8 /usr/bin/python3
RUN python3 --version

VOLUME /project
WORKDIR /project

ENTRYPOINT ["sh"]
```

2. [docker build](https://docs.docker.com/engine/reference/commandline/build/)를 사용하여 Docker 이미지를 빌드합니다.

   다음은 예제입니다.

   ```
   $ docker build --tag terraform-build:v1 <path-to-directory-containing-Dockerfile>
   ```

3. `--use-container` 및 `--build-image` 옵션을 사용하여 `sam build` 명령을 실행합니다 AWS SAMCLI.

   다음은 예제입니다.

   ```
   $ sam build --use-container --build-image terraform-build:v1
   ```

다음 단계

Terraform 프로젝트에서 로컬 디버깅 및 테스트에 Terraform과 함께 AWS SAMCLI 사용하기 AWS SAM를 사용하려면 CLI 섹션을 참조하세요.

Terraform Cloud에 대해 설정

Terraform v1.6.0 버전 이상을 사용하는 것이 좋습니다. 이전 버전을 사용하는 경우 로컬에서 Terraform 계획 파일을 생성해야 합니다. 로컬 계획 파일은 로컬 테스트 및 디버깅을 수행하는 데 필요한 정보를에 제공합니다 AWS SAM CLI.

로컬 계획 파일을 생성하려면

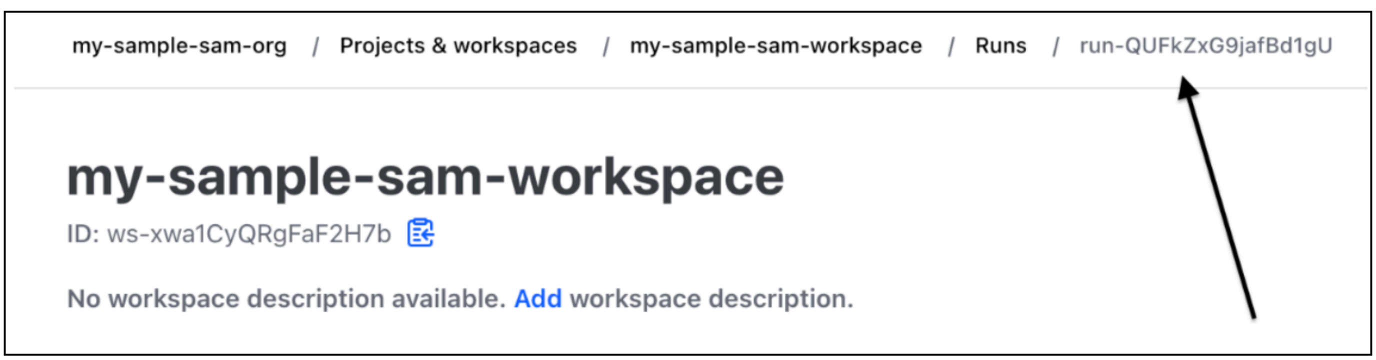
> **Note**
>
> Terraform v1.6.0 이후의 버전에는 이러한 단계가 필요하지 않습니다. 에서 AWS SAM CLI 사용을 시작하려면 단원을 Terraform Cloud참조하십시오Terraform과 함께 AWS SAM CLI 사용.

1. API 토큰 구성 - 토큰 유형은 액세스 수준에 따라 달라집니다. 자세한 내용은 API 토큰의 Terraform Cloud 배문서를 참조하세요.
2. API 토큰 환경 변수 설정 - 다음은 명령줄의 예입니다.

```
$ export TOKEN="<api-token-value>"
```

3. 실행 ID 가져오기 - Terraform Cloud 콘솔에서에 사용할 실행의 Terraform 실행 ID를 찾습니다 AWS SAMCLI.

   실행 ID는 실행의 브레드크럼 경로에 있습니다.



4. 계획 파일 가져오기 - API 토큰을 사용하여 로컬 계획 파일을 가져옵니다. 다음은 명령줄의 예입니다.

```
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
```

이제 Terraform Cloud와 함께 AWS SAM CLI를 사용할 준비가 되었습니다. 지원되는 AWS SAMCLI 명령을 사용하는 경우 `--terraform-plan-file` 옵션을 사용하여 로컬 계획 파일의 이름과 경로를 지정합니다. 다음은 예제입니다.

```
$ sam local invoke --hook-name terraform --terraform-plan-file custom-plan.json
```

다음은 `sam local start-api`를 사용한 명령의 예입니다.

```
$ sam local start-api --hook-name terraform --terraform-plan-file custom-plan.json
```

이러한 예제와 함께 사용할 수 있는 샘플 애플리케이션은 aws-samples GitHub 리포지토리의 api_gateway_v2_tf_cloud를 참조하세요.

다음 단계

Terraform Cloud와 함께 AWS SAMCLI를 시작하려면 로컬 디버깅 및 테스트에 Terraform과 함께 AWS SAMCLI 사용하기 섹션을 참조하세요.

로컬 디버깅 및 테스트에 Terraform과 함께 AWS SAMCLI 사용하기

이 주제에서는 지원되는 AWS Serverless Application Model 명령줄 인터페이스(AWS SAMCLI) 명령을 Terraform 프로젝트 및와 함께 사용하는 방법을 다룹니다Terraform Cloud.

피드백을 제공하고 기능 요청을 제출하려면 GitHub 문제를 생성하세요.

주제

- sam local invoke를 사용한 로컬 테스트
- sam local start-api를 사용한 로컬 테스트
- sam local start-lambda를 사용한 로컬 테스트
- Terraform 제한 사항

sam local invoke를 사용한 로컬 테스트

> **Note**
>
> AWS SAMCLI를 사용하여 로컬에서 테스트하려면 Docker를 설치하고 구성해야 합니다. 지침은 AWS SAMCLI와 함께 사용할 Docker 설치 섹션을 참조하세요.

다음은 이벤트를 전달하여 로컬에서 Lambda 함수를 테스트하는 예입니다.

```
$ sam local invoke --hook-name terraform hello_world_function -e events/event.json -
```

이 함수 사용에 대한 자세한 내용은 sam local invoke를 사용한 테스트 소개 섹션을 참조하세요.

sam local start-api를 사용한 로컬 테스트

Terraform과 함께 `sam local start-api`를 사용하려면 다음이 필요합니다.

```
$ sam local start-api --hook-name terraform
```

다음은 예제입니다.

```
$ 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)
```

sam local start-api를 사용한 테스트 소개 명령에 대한 자세한 내용은 섹션을 참조하세요.

Lambda 권한 부여자를 사용하는 Lambda 함수

Lambda 권한 부여자를 사용하도록 구성된 Lambda 함수의 경우 AWS SAMCLI는 Lambda 함수 엔드포인트를 간접 호출하기 전에 Lambda 권한 부여자를 자동으로 간접 호출합니다.

- 에서이 기능에 대해 자세히 알아보려면 섹션을 AWS SAMCLI참조하세요Lambda 권한 부여자를 사용하는 Lambda 함수.
- Terraform에서 Lambda 권한 부여자를 사용하는 방법에 대한 자세한 내용은 Terraform 레지스트리에서 Resource: aws_api_gateway_authorizer 섹션을 참조하세요.

sam local start-lambda를 사용한 로컬 테스트

다음은 AWS Command Line Interface ()를 사용하여 로컬에서 Lambda 함수를 테스트하는 예제입니다 AWS CLI.

1. AWS SAMCLI를 사용하여 로컬 테스트 환경을 생성합니다.

```
$ sam local start-lambda --hook-name terraform hello_world_function
```

2. AWS CLI 를 사용하여 로컬에서 함수를 호출합니다.

```
$ 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
```

sam local start-lambda를 사용한 테스트 소개 명령에 대한 자세한 내용은 섹션을 참조하세요.

Terraform 제한 사항

Terraform와 함께 AWS SAM CLI를 사용할 때의 제한 사항은 다음과 같습니다.

- Lambda 함수는 여러 계층에 연결되어 있습니다.
- 리소스 간 링크를 정의하는 Terraform 로컬 변수.
- 아직 생성되지 않은 Lambda 함수를 참조합니다. 여기에는 REST API 리소스의 본문 속성에 정의된 함수가 포함됩니다.

이러한 제한을 피하려면 새 리소스가 추가될 때 `terraform apply`를 실행할 수 있습니다.

로컬 디버깅 및 테스트에 Serverless.tf와 함께 AWS SAMCLI 사용

AWS Serverless Application Model 명령줄 인터페이스(AWS SAMCLI)는 AWS Lambda 함수 및 계층의 로컬 디버깅 및 테스트를 위해 Serverless.tf 모듈과 함께 사용할 수 있습니다. 다음 AWS SAMCLI 명령이 지원됩니다.

- `sam build`
- `sam local invoke`
- `sam local start-api`

- `sam local start-lambda`

> **Note**
>
> Serverless.tf 버전 4.6.0 이상은 AWS SAMCLI 통합을 지원합니다.

Serverless.tf 모듈에서 AWS SAMCLI 사용을 시작하려면 최신 버전의 Serverless.tf 및 로 업데이트합니다 AWS SAMCLI.

serverless.tf version 6.0.0부터 `create_sam_metadata` 파라미터를 `true`로 설정해야 합니다. 이렇게 하면 `sam build` 명령에 필요한 메타데이터 리소스가 AWS SAMCLI 생성됩니다.

Serverless.tf에 대해 자세히 알아보려면 terraform-aws-lambda-module을 참조하세요.

Terraform 참조로AWS SAMCLI 사용

이 섹션은 로컬 디버깅 및 테스트를 위해에서 AWS Serverless Application Model 명령줄 인터페이스(AWS SAMCLI)를 사용하기 Terraform 위한 참조입니다.

피드백을 제공하고 기능 요청을 제출하려면 GitHub 문제를 생성하세요.

AWS SAM 지원되는 기능 참조

Terraform 사용에 지원되는 AWS SAMCLI 기능에 대한 참조 설명서는 다음에서 찾을 수 있습니다.

- sam build
- sam local invoke
- sam local start-api
- sam local start-lambda

Terraform 특정 참조

Terraform과 함께 AWS SAMCLI 사용과 관련된 참조 문서는 다음에서 찾을 수 있습니다.

- sam 메타데이터 리소스

sam 메타데이터 리소스

이 페이지에는 Terraform 프로젝트에 사용되는 sam metadata resource 리소스 유형에 대한 참조 정보를 포함합니다.

- 에서 AWS Serverless Application Model 명령줄 인터페이스(AWS SAMCLI)를 사용하는 방법에 대한 소개는 섹션을 Terraform참조하세요Terraform에 대한 AWS SAMCLI지원이란 무엇입니까?.
- Terraform과 함께 AWS SAM CLI을 사용하려면 로컬 디버깅 및 테스트에 Terraform과 함께 AWS SAMCLI 사용하기 섹션을 참조하세요.

주제

- 인수
- 예제

인수

인수	설명
`built_output_path`	AWS Lambda 함수의 빌드된 아티팩트 경로입니다.
`docker_build_args`	Docker 빌드 인수 JSON 객체의 디코딩된 문자열. 이 인수는 선택 사항입니다.
`docker_context`	도커 이미지 빌드 컨텍스트가 포함된 디렉터리의 경로.
`docker_file`	Docker 파일에 대한 경로입니다. 이 경로는 `docker_context` 경로와 상대적입니다. 이 인수는 선택 사항입니다. 기본값은 `Dockerfile` 입니다.
`docker_tag`	생성된 도커 이미지 태그의 값. 이 값은 선택 사항입니다.
`depends_on`	Lambda 함수 또는 계층의 빌딩 리소스 경로. 자세히 알아보려면Terraform 레지스트리의 depends_on 인수를 참조하세요.
`original_source_code`	Lambda 함수가 정의된 경로입니다. 이 값은 문자열, 문자열 배열 또는 문자열로 디코딩된 JSON 객체일 수 있습니다.

인수	설명
	• 문자열 배열의 경우 여러 코드 경로가 지원되지 않으므로 첫 번째 값만 사용됩니다. • JSON 객체의 경우, `source_code_property` 도 정의해야 합니다.
`resource_name`	Lambda 함수의 이름입니다.
`resource_type`	Lambda 함수 패키지 유형의 형식. 사용 가능한 값은 다음과 같습니다. • `IMAGE_LAMBDA_FUNCTION` • `LAMBDA_LAYER` • `ZIP_LAMBDA_FUNCTION`
`source_code_property`	JSON 객체에 있는 Lambda 리소스 코드의 경로입니다. `original_source_code` 가 JSON 객체인 경우 이 속성을 정의합니다.

예제

ZIP 패키지 유형을 사용하는 Lambda 함수를 참조하는 sam metadata resource

```
# 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
  ]
}
```

이미지 패키지 유형을 사용하여 Lambda 함수를 참조하는 sam 메타데이터 리소스

```
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"
  }
}
```

Lambda 계층을 참조하는 sam 메타데이터 리소스

```
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]
}
```

Terraform에 대한 AWS SAMCLI지원이란 무엇입니까?

AWS Serverless Application Model 명령줄 인터페이스(AWS SAMCLI)를 Terraform 프로젝트 또는와 함께 사용하여 다음에 대한 로컬 디버깅 및 테스트를 Terraform Cloud 수행합니다.

- AWS Lambda 함수 및 계층.
- Amazon API Gateway API

Terraform에 대한 소개는 HashiCorp Terraform 웹사이트에서 Terraform란 무엇입니까? 섹션을 참조하세요.

피드백을 제공하고 기능 요청을 제출하려면 GitHub 문제를 생성하세요.

> **Note**
>
> 의 통합에 대한 구문 분석 단계의 일환으로 AWS SAMCLI는 사용자 명령을 AWS SAMCLI 처리하여 프로젝트 파일과 데이터를 생성합니다. 명령 출력은 변경되지 않아야 하지만 특정 환경에서는 환경 또는 러너가 출력에 추가 로그나 정보를 삽입할 수 있습니다.

주제

- 란 무엇입니까 AWS SAMCLI?
- Terraform와 함께 AWS SAM CLI를 사용하려면 어떻게 해야 합니까?
- 다음 단계

란 무엇입니까 AWS SAMCLI?

AWS SAMCLI는 AWS SAM 템플릿 및와 같은 지원되는 타사 통합과 함께 사용하여 서버리스 애플리케이션을 Terraform빌드하고 실행할 수 있는 명령줄 도구입니다. 에 대한 소개는 단원을 AWS SAMCLI 참조하십시오란 무엇입니까 AWS SAMCLI?.

는 Terraform에 대해 다음 명령을 AWS SAMCLI 지원합니다.

- `sam local invoke` - 로컬에서 AWS Lambda 함수 리소스의 일회성 호출을 시작합니다. sam local invoke를 사용한 테스트 소개 명령에 대한 자세한 내용은 섹션을 참조하세요.
- `sam local start-api` - Lambda 리소스를 로컬에서 실행하고 로컬 HTTP 서버 호스트를 통해 테스트합니다. 이 유형의 테스트는 API Gateway 엔드포인트에서 간접 호출되는 Lambda 함수에 유용합니다. sam local start-api를 사용한 테스트 소개 명령에 대한 자세한 내용은 섹션을 참조하세요.
- `sam local start-lambda` - AWS Command Line Interface (AWS CLI) 또는 SDKs를 사용하여 함수를 로컬로 호출하려면 Lambda 함수의 로컬 엔드포인트를 시작합니다. sam local start-lambda를 사용한 테스트 소개 명령에 대한 자세한 내용은 섹션을 참조하세요.

Terraform와 함께 AWS SAM CLI를 사용하려면 어떻게 해야 합니까?

핵심 Terraform 워크플로는 작성, 계획, 적용의 3단계로 구성됩니다. 에 대한 AWS SAMCLI 지원을 통해 Terraform 워크플로를 AWS SAMCLI 계속 사용하여에서 애플리케이션을 관리하면서 명령 `sam local` 세트를 활용할 Terraform수 있습니다 AWS. 일반적으로 이는 다음을 의미합니다.

- 작성 - Terraform를 사용하여 인프라를 코드로 작성합니다.
- 테스트 및 디버그 - AWS SAMCLI를 사용하여 애플리케이션을 로컬에서 테스트하고 디버깅합니다.
- 계획 - 적용하기 전에 변경 사항을 미리 볼 수 있습니다.
- 적용 - 인프라를 프로비저닝합니다.

와 AWS SAMCLI 함께를 사용하는 예제는 AWS 컴퓨팅 블로그의 함께 개선: AWS SAMCLI 및 HashiCorp Terraform 섹션을 Terraform참조하세요.

다음 단계

모든 사전 조건을 완료하고 Terraform을 설정하려면 AWS SAM CLI에 대한 Terraform 지원과 함께 시작하기 섹션을 참조하세요.

AWS SAMCLI를 사용하여 애플리케이션 게시

다른 사용자가 AWS SAM 애플리케이션을 찾고 배포할 수 있도록 하려면를 사용하여 AWS SAMCLI에 게시할 수 있습니다 AWS Serverless Application Repository. 를 사용하여 애플리케이션을 게시하려면 AWS SAM 템플릿을 사용하여 애플리케이션을 정의 AWS SAMCLI해야 합니다. 또한 로컬 또는 AWS 클라우드에서 테스트해야 합니다.

이 주제의 지침에 따라 새 애플리케이션을 만들거나, 기존 애플리케이션의 새로운 버전을 만들거나, 기존 애플리케이션의 메타데이터를 업데이트합니다. (사용자가 수행하는 작업은 애플리케이션이에 이미 존재하는지 AWS Serverless Application Repository여부와 애플리케이션 메타데이터가 변경되고 있는지 여부에 따라 달라집니다.) 메타데이터에 대한 자세한 정보는 [AWS SAM 템플릿 메타데이터 섹션 속성](#) 섹션을 참조하세요.

사전 조건

를 AWS Serverless Application Repository 사용하여에 애플리케이션을 게시 AWS SAMCLI하기 전에 다음이 있어야 합니다.

- AWS SAM CLI이 설치됨 자세한 내용은 [AWS SAM CLI 설치](#) 단원을 참조하십시오. AWS SAM CLI 설치 여부를 알아보려면 다음 명령을 실행합니다.

  ```
  sam --version
  ```

- 유효한 AWS SAM 템플릿입니다.
- AWS SAM 템플릿이 참조하는 애플리케이션 코드 및 종속성입니다.
- 애플리케이션을 공개적으로 공유하는 데에만 필요한 시맨틱 버전입니다. 이 값은 1.0만큼 간단할 수 있습니다.
- 애플리케이션의 소스 코드를 가리키는 URL.
- `README.md` 파일. 이 파일은 고객이 어떻게 애플리케이션을 사용할 수 있는지와 AWS 계정에 배포하기 전에 구성하는 방법을 설명할 것입니다.
- 애플리케이션을 공개적으로 공유하는 데에만 필요한 `LICENSE.txt` 파일입니다.
- 애플리케이션에 중첩된 애플리케이션이 포함되어 있는 경우 AWS Serverless Application Repository에 이미 게시한 상태여야 합니다.
- 애플리케이션을 패키징할 때 Amazon S3에 업로드된 아티팩트에 대한 서비스 읽기 권한을 부여하는 유효한 Amazon Simple Storage Service(S3) 버킷 정책. 이 정책에는 다음 설정이 포함되어 있습니다.

1. https://console.aws.amazon.com/s3/에서 S3 콘솔을 엽니다.
2. 애플리케이션을 패키징하는 데 사용한 Amazon S3 버킷을 선택합니다.
3. 권한을 선택합니다.
4. 권한(Permissions) 탭의 버킷 정책(Bucket policy)에서 편집(Edit)을 선택합니다.
5. 버킷 정책 편집기 페이지에서 다음 정책 문을 정책 편집기에 붙여 넣습니다. 정책 설명에서 `Resource` 요소에는 버킷 이름을, `Condition` 요소에는 AWS 계정 ID를 사용해야 합니다. `Condition` 요소의 표현식은 AWS Serverless Application Repository 가 지정된 AWS 계정의 애플리케이션에만 액세스할 수 있는 권한을 갖도록 합니다. 요소에 대한 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service":  "serverlessrepo.amazonaws.com"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::<your-bucket-name>/*",
            "Condition" : {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                }
            }
        }
    ]
}
```

6. 변경 사항 저장을 선택합니다.

애플리케이션 게시()

1단계: AWS SAM 템플릿에 **Metadata** 섹션 추가

먼저 AWS SAM 템플릿에 `Metadata` 섹션을 추가합니다. AWS Serverless Application Repository에 게시할 애플리케이션 정보를 제공합니다.

다음은 `Metadata` 섹션의 예입니다.

```
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
        ...
```

AWS SAM 템플릿의 `Metadata` 섹션에 대한 자세한 내용은 섹션을 참조하세요AWS SAM 템플릿 메타데이터 섹션 속성.

2단계: 애플리케이션 패키징

다음 AWS SAMCLI 명령을 실행합니다. 그러면 애플리케이션의 아티팩트를 Amazon S3에 업로드하고 `packaged.yaml`라는 새 템플릿 파일이 출력됩니다.

```
sam package --output-template-file packaged.yaml --s3-bucket <your-bucket-name>
```

다음 단계에서 `packaged.yaml` 템플릿 파일을 사용하여 애플리케이션을 AWS Serverless Application Repository에 게시합니다. 이 파일은 원본 템플릿 파일(`template.yaml`)과 비슷하지만, 중요한 차이점이 있습니다. `CodeUri`, `LicenseUrl` 및 `ReadmeUrl` 속성인 Amazon S3 버킷과 해당 아티팩트가 포함된 객체를 가리킵니다.

예제 `packaged.yaml` 템플릿 파일의 다음 코드 조각은 `CodeUri` 속성을 보여 줍니다.

```
MySampleFunction:
    Type: AWS::Serverless::Function
```

```
    Properties:
      CodeUri: s3://bucketname/fbd77a3647a4f47a352fcObjectGUID

...
```

3단계: 애플리케이션 게시

AWS SAM 애플리케이션의 프라이빗 버전을에 게시하려면 다음 AWS SAMCLI 명령을 AWS Serverless Application Repository실행합니다.

```
sam publish --template packaged.yaml --region us-east-1
```

`sam publish` 명령 출력에는 AWS Serverless Application Repository에 있는 애플리케이션으로 연결되는 링크가 포함됩니다. 또는 AWS Serverless Application Repository 시작 페이지로 이동하여 애플리케이션을 검색할 수도 있습니다.

4단계: 애플리케이션 공유(선택 사항)

기본적으로 애플리케이션은 비공개로 설정되어 있으므로 다른 AWS 계정에서는 볼 수 없습니다. 애플리케이션을 다른 사용자와 공유하려면 애플리케이션을 퍼블릭으로 설정하거나 특정 AWS 계정 목록에 권한을 부여해야 합니다.

를 사용하여 애플리케이션을 공유하는 방법에 대한 자세한 내용은 AWS Serverless Application Repository 개발자 안내서의 AWS Serverless Application Repository 리소스 기반 정책 예제를 AWS CLI참조하세요. AWS Management 콘솔을 사용하여 애플리케이션을 공유하는 방법에 대한 자세한 내용은 AWS Serverless Application Repository 개발자 안내서의 애플리케이션 공유를 참조하세요.

기존 애플리케이션의 새 버전 게시

에 애플리케이션을 게시한 후 새 버전을 게시할 AWS Serverless Application Repository수 있습니다. 예를 들어, Lambda 함수 코드를 변경하거나 애플리케이션 아키텍처에 새 구성 요소를 추가했을 수 있습니다.

이전에 게시한 애플리케이션을 업데이트하려면 앞서 설명한 것과 동일한 프로세스를 사용하여 애플리케이션을 다시 게시합니다. AWS SAM 템플릿 파일의 `Metadata` 섹션에 원래 게시할 때 사용한 것과 동일한 애플리케이션 이름을 제공하되 새 `SemanticVersion` 값을 포함합니다.

예를 들어, 이름 `SampleApp`와 `1.0.0`의 `SemanticVersion`을 사용하여 게시된 애플리케이션을 생각해 보겠습니다. 해당 애플리케이션을 업데이트하려면 AWS SAM 템플릿에 애플리케이션 이름 `SampleApp`과 `1.0.1`의 `SemanticVersion`(또는 `1.0.0`이외의 다른 것)가 있어야 합니다.

추가 주제

- AWS SAM 템플릿 메타데이터 섹션 속성

AWS SAM 템플릿 메타데이터 섹션 속성

`AWS::ServerlessRepo::Application`은 귀하가 AWS Serverless Application Repository에 게시하려는 애플리케이션 정보를 지정하는 데 사용할 수 있는 메타데이터 키입니다.

> **Note**
>
> CloudFormation 내장 함수는 `AWS::ServerlessRepo::Application` 메타데이터 키에서 지원되지 않습니다.

속성

이 표는 AWS SAM 템플릿 `Metadata` 섹션의 속성에 대한 정보를 제공합니다. 이 섹션은를 AWS Serverless Application Repository 사용하여에 애플리케이션을 게시하는 데 필요합니다 AWS SAMCLI.

속성	유형	필수	설명
`Name`	문자열	TRUE	애플리케이션의 이름입니다. 최소 길이: 1. 최대 길이: 140. 패턴: `"[a-zA-Z0-9\\-]+";`
`Description`	문자열	TRUE	애플리케이션에 대한 설명입니다. 최소 길이: 1. 최대 길이: 256.
`Author`	문자열	TRUE	애플리케이션을 게시하는 작성자의 이름입니다. 최소 길이: 1. 최대 길이: 127. 패턴: `"^[a-z0-9](([a-z0-9]|-(?!-))*[a-z0-9])?$";`

속성	유형	필수	설명
`SpdxLicenseId`	문자열	FALSE	유효한 라이선스 식별자. 유효한 라이선스 식별자 목록을 보려면 소프트웨어 패키지 데이터 교환(SPDX) 웹사이트의 SPDX 라이선스 목록을 참조하세요.
`LicenseUrl`	문자열	FALSE	애플리케이션의 SPDXLicenseID 값과 일치하는 로컬 라이선스 파일에 대한 참조 또는 Amazon S3 라이선스 파일에 대한 링크입니다. `sam package` 명령을 사용하여 패키징되지 않은 AWS SAM 템플릿 파일에는이 속성의 로컬 파일에 대한 참조가 있을 수 있습니다. 하지만 `sam publish` 명령을 사용하여 애플리케이션을 게시하려면 이 속성이 Amazon S3 버킷에 대한 참조여야 합니다. 최대 크기 = 5 MB 애플리케이션을 공개로 설정하려면 이 속성의 값을 지정해야 합니다. 단, 애플리케이션이 게시된 후에는 이 속성을 업데이트할 수 없습니다. 따라서 애플리케이션에 라이선스를 추가하려면 먼저 애플리케이션을 삭제하거나 다른 이름으로 새 애플리케이션을 게시해야 합니다.
`ReadmeUrl`	문자열	FALSE	로컬 readme 파일에 대한 참조 또는 Readme 파일에 대한 Amazon S3 링크입니다. 이 링크에는 애플리케이션 및 작동 방식에 대한 자세한 설명이 들어 있습니다. `sam package` 명령을 사용하여 패키징되지 않은 AWS SAM 템플릿 파일에는이 속성의 로컬 파일에 대한 참조가 있을 수 있습니다. 하지만 `sam publish` 명령을 사용하여 게시하려면 이 속성이 Amazon S3 버킷에 대한 참조여야 합니다. 최대 크기 = 5 MB

속성	유형	필수	설명
`Labels`	문자열	FALSE	검색 결과에서 애플리케이션의 검색을 향상시키는 레이블입니다. 최소 길이: 1. 최대 길이: 127. 최대 레이블 수: 10. 패턴: `"^[a-zA-Z0-9+\\-_:\\/@]+$";`
`HomePageUrl`	문자열	FALSE	애플리케이션에 대한 자세한 정보가 들어 있는 URL(예: 애플리케이션의 GitHub 리포지토리 위치)
`SemanticVersion`	문자열	FALSE	애플리케이션의 의미 체계 버전입니다. 시맨틱 버전 관리 사양은 시맨틱 버전 관리 웹사이트를 참조하세요. 애플리케이션을 공개로 설정하려면 이 속성의 값을 지정해야 합니다.
`SourceCodeUrl`	문자열	FALSE	애플리케이션의 소스 코드가 있는 퍼블릭 리포지토리의 링크입니다.

사용 사례

이 섹션에는 애플리케이션을 게시하는 사용 사례와 해당 사용 사례에 대해 처리된 `Metadata` 속성이 나열되어 있습니다. 특정 사용 사례에 대해 나열되지 않은 속성은 무시됩니다.

- 새 애플리케이션 생성 -에 계정 이름과 AWS Serverless Application Repository 일치하는 애플리케이션이 없는 경우 새 애플리케이션이 생성됩니다.
  - `Name`
  - `SpdxLicenseId`
  - `LicenseUrl`
  - `Description`
  - `Author`
  - `ReadmeUrl`
  - `Labels`

- `HomePageUrl`
- `SourceCodeUrl`
- `SemanticVersion`
- AWS SAM 템플릿의 내용(예: 이벤트 소스, 리소스 및 Lambda 함수 코드)

- 애플리케이션 버전 생성 - AWS Serverless Application Repository 내에 계정과 일치하는 이름을 가진 애플리케이션 버전이 이미 존재하고 그에 더하여 SemanticVersion이 변경 중인 경우, 애플리케이션 버전이 생성됩니다.
  - `Description`
  - `Author`
  - `ReadmeUrl`
  - `Labels`
  - `HomePageUrl`
  - `SourceCodeUrl`
  - `SemanticVersion`
  - AWS SAM 템플릿의 내용(예: 이벤트 소스, 리소스 및 Lambda 함수 코드)

- 애플리케이션 업데이트 -에 계정과 AWS Serverless Application Repository 일치하는 이름의 애플리케이션이 이미 있고 SemanticVersion이 변경되지 않는 경우 애플리케이션이 업데이트됩니다.
  - `Description`
  - `Author`
  - `ReadmeUrl`
  - `Labels`
  - `HomePageUrl`

예제

다음은 `Metadata` 섹션의 예입니다.

```
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
```

AWS SAM에 대한 문서 기록

다음 표는 AWS Serverless Application Model 개발자 안내서의 각 릴리스에서 변경된 중요 사항에 대해 설명합니다. 이 설명서에 대한 업데이트 알림을 받으려면 RSS 피드를 구독하면 됩니다.

- 최종 설명서 업데이트: 2024년 6월 20일

변경 사항	설명	날짜
개발자 안내서 전체에서 콘텐츠 재구성 및 업데이트	검색 편의성과 사용성을 개선하기 위해 안내서를 개편하고 재구성했습니다. 제목을 업데이트하고 개선했습니다. 주제 및 개념을 소개할 때 세부 정보를 추가했습니다.	2024년 6월 20일
Ruby 3.3에 대한 AWS SAM CLI 지원 추가	이제 Ruby 3.3을 런타임 및 이미지 리포지토리로 사용할 수 있습니다. 자세한 내용은 이미지 리포지토리 및 sam init를 참조하세요.	2024년 4월 4일
AWS SAMCLI 명령 옵션 추가	sam local start-api 명령에서 새 옵션 `--ssl-cert-file` PATH, `--ssl-key-file` PATH를 사용할 수 있습니다. 또한 sam local invoke, sam local start-api 및 sam local start-lambda에서 새 명령줄 옵션 `--add-host LIST` 를 사용할 수 있습니다.	2024년 3월 20일
.NET 8에 대한 AWS SAMCLI 지원 추가	이제 .NET 8을 런타임 및 이미지 리포지토리로 사용할 수 있습니다. .NET Core 3.1, Node.js 14, Node.js 12, Python 3.7, Ruby 2.7에 대한	2024년 2월 22일

	런타임 및 이미지 리포지토리는 더 이상 지원되지 않습니다. 이미지 리포지토리 및 sam init를 참조하세요.	
Linux에 대한 AWS SAMCLI arm64 패키지 설치관리자 프로그램 추가	자세한 내용은 AWS SAMCLI 설치를 참조하세요.	2023년 12월 6일
AWS SAM CLI sam sync 명령에 대한 --watch-exclude 옵션 추가	동기화를 시작할 때 파일 및 폴더를 제외합니다. 자세히 알아보려면 동기화를 시작하지 않는 파일 및 폴더 지정을 참조하세요.	2023년 12월 6일
AWS SAM CLI sam sync 명령에 대한 --build-in-source 옵션 추가	소스 폴더에서 프로젝트를 빌드하여 빌드 프로세스의 속도를 높이세요. 자세히 알아보려면 소스 폴더에서 프로젝트를 빌드하여 빌드 시간 단축을 참조하세요.	2023년 12월 6일
AWS SAM CLI sam build 명령에 대한 --build-in-source 옵션 추가	소스 폴더에서 프로젝트를 빌드하여 빌드 프로세스의 속도를 높이세요. 자세히 알아보려면 소스 폴더에서 프로젝트를 빌드하여 빌드 시간 단축을 참조하세요.	2023년 12월 6일
AWS SAM CLI remote invoke 명령에 대한 새 리소스 지원 추가	Kinesis Data Streams 애플리케이션, Amazon SQS 대기열 및 Step Functions 상태 머신과 함께 `sam remote invoke` 사용. 자세히 알아보려면 sam remote invoke 사용을 참조하세요.	2023년 11월 15일

공유 가능한 테스트 이벤트를 위한 새 AWS SAM CLI remote test-event 명령 추가	AWS SAMCLI를 사용하여 EventBridge 스키마 레지스트리에서 공유 가능한 테스트 이벤트에 액세스하고 관리하여 AWS 클라우드에서 Lambda 함수를 테스트할 수 있습니다. 자세히 알아보려면 sam 원격 테스트 이벤트 사용을 참조하세요.	2023년 10월 3일
Terraform에 대한 AWS SAM CLI의 지원이 이제 통상적으로 제공됩니다.	Terraform을 위한 AWS SAMCLI지원에 대한 자세한 내용은 AWS SAM CLITerraform 지원을 참조하세요.	2023년 9월 5일
Terraform Cloud에 대한 AWS SAMCLI 지원 추가	AWS SAM CLI는 현재 Terraform Cloud의 로컬 테스트를 지원합니다. 자세히 알아보려면 Terraform Cloud의 설정을 참조하세요.	2023년 9월 5일
AWS SAM CLI 구성 파일에 대한 YAML 파일 형식 지원 추가	는 AWS SAM CLI 이제 `[.yaml|.yml]` 파일 형식을 지원합니다. AWS SAMCLI 및 AWS SAMCLI구성 파일 페이지가 업데이트되었습니다.	2023년 7월 18일
Terraform에 대한 AWS SAM CLIsam local start-api 명령 지원 추가	Terraform를 위한 AWS SAMCLI지원이란 무엇입니까? 단원이 Terraform을 위한 AWS SAM CLI `sam local start-api` 명령 지원을 포함하도록 업데이트됨.	2023년 7월 6일
새 AWS SAM CLI remote invoke 명령 추가	`sam remote invoke` 사용을 시작하려면 sam remote invoke 사용을 참조하세요.	2023년 6월 22일

AWS AppSyncGraphQL API 서버리스 리소스 유형 추가	AWS SAM를 사용하는 GraphQL API 리소스를 정의하는 방법을 설명하는 새 `AWS::Serverless::GraphQLApi` 섹션을 생성합니다	2023년 6월 22일
Ruby 3.2에 대한 AWS SAMCLI 지원 추가	새 기본 이미지 및 런타임 값을 포함하도록 sam 초기화 페이지를 업데이트합니다. Ruby3.2 Amazon ECR URI로 이미지 리포지토리 페이지를 업데이트합니다.	2023년 6월 6일
AWS SAMCLI 패키지 설치관리자 프로그램의 무결성 검증을 위한 선택적 단계 추가	선택적 단계를 반영하도록 AWS SAMCLI 설치 페이지를 업데이트합니다. 단계를 문서화하기 위하여 AWS SAMCLI 설치관리자 프로그램 페이지의 무결성 확인을 생성합니다.	2023년 5월 31일
인프라 동기화를 건너뛰기 위해 sam sync 옵션 추가	실행할 때마다 `sam sync` CloudFormation 배포가 필요한지 여부 사용자 지정합니다. 자세히 알아보려면 초기 CloudFormation 배포 건너뛰기를 참조하세요.	2023년 3월 23일
DocumentDB 이벤트 소스 유형에 대한 지원 추가	이제 AWS SAM 템플릿 사양이 `AWS::Serverless::Function` 리소스의 `DocumentDB` 이벤트 소스 유형을 지원합니다. 자세한 내용은 DocumentDB를 참조하세요.	2023년 3월 10일

를 사용하여 Rust Lambda 함수 빌드Cargo Lambda	AWS SAMCLI를 사용하여 Cargo Lambda을 사용한 Rust Lambda 함수 빌드. 자세히 알아보려면 Cargo Lambda를 사용한 Rust Lambda 함수 빌드를 참조하세요.	2023년 2월 23일
의 외부에서 함수 리소스 빌드 AWS SAM	sam build 명령 사용 시 함수 건너뛰기에 대한 지침이 추가되었습니다. 자세히 알아보려면 AWS SAM외부에서 함수 빌드를 참조하세요.	2023년 2월 14일
새로운 임베디드 커넥터 명령문	새 내장 커넥터 명령문을 사용하여 귀하의 `AWS::Serverless::Connector` 리소스를 정의합니다. 자세히 알아보려면 AWS SAM커넥터를 사용한 리소스 권한 관리를 참조하세요.	2023년 2월 8일
에 대한 새 sam list 명령이 추가됨AWS SAMCLI	서버리스 애플리케이션의 리소스에 대한 중요한 정보를 보는 데 sam list를 사용합니다. 자세히 알아보려면 sam list를 참조하세요.	2023년 2월 2일
esbuild용 포맷 및 아웃익스텐션 빌드 속성이 추가됨	esbuild로 Node.js Lambda 함수를 빌드하면 이제 `Format` 및 `OutExtension` 빌드 속성을 지원합니다. 자세히 알아보려면 esbuild를 사용한 Node.js Lambda 함수 빌드를 참조하세요.	2023년 2월 2일

AWS SAM템플릿 사양에 런타임 관리 옵션이 추가됨	Lambda 함수의 런타임 관리 옵션을 구성합니다. 자세한 내용은을 참조하십시오`RuntimeManagementConfig` .	2023년 1월 24일
대상 속성이 AWS::Serverless::StateMachine용 EventSource에 추가됨	`AWS::Serverless::StateMachine` 리소스 유형은 `EventBridgeRule` 및 `Schedule` 이벤트 소스의 `Target` 속성을 지원합니다.	2023년 1월 13일
Lambda 함수에 대한 SQS 폴러 스케일링 구성	`AWS::Serverless::Function` 에 관한 `ScalingConfig` 의 속성을 사용하여 SQS 폴러의 크기 조정을 구성합니다. 자세한 내용은을 참조하십시오 `ScalingConfig` .	2023년 1월 12일
cfn-lint로 AWS SAM 애플리케이션 검증	cfn-lint를 사용하여 AWS SAM CLI를 통해 AWS SAM 템플릿의 유효성을 검사를 할 수 있습니다. 자세히 알아보려면 cfn-lint로 유효성 검사를 참조하세요.	2023년 1월 11일
CloudWatch Application Insights를 사용하여 서버리스 애플리케이션 모니터링	Amazon CloudWatch Application Insights를 구성하여 AWS SAM애플리케이션을 모니터링합니다. 자세히 알아보려면 CloudWatch Application Insights를 사용하여 서버리스 애플리케이션 모니터링을 참조하세요.	2022년 12월 19일

macOS용 AWS SAMCLI 패키지 설치관리자 프로그램이 추가됨	새 macOS 패키지 설치관리자 프로그램을 사용하여 AWS SAM CLI를 설치합니다. 자세한 내용은 AWS SAMCLI 설치하기를 참조하세요	2022년 12월 6일
Lambda SnapStart에 대한 지원이 추가됨	Lambda 함수가 초기화된 함수의 캐시 상태인 스냅샷을 생성하도록 SnapStart를 구성합니다. 자세한 내용은 다음 섹션을 참조하세.`AWS::Serverless::Function`	2022년 11월 28일
nodejs18.x에 대한 AWS SAMCLI지원이 추가됨	AWS SAM CLI는 이제 nodejs18.x 런타임을 지원합니다. 자세히 알아보려면 sam init를 참조하세요.	2022년 11월 17일
액세스 및 권한 구성에 대한 지침이 추가됨	AWS SAM은 귀하의 서버리스 애플리케이션의 액세스 및 권한 관리를 간소화하는 두 가지 옵션을 제공합니다. 자세한 내용은 리소스 액세스 및 권한 관리를 참조하세요.	2022년 11월 17일
네이티브 AOT 컴파일을 사용하여 .NET 7 Lambda 함수를 빌드하기 위한 지원이 추가됨	네이티브 어헤드 오브 타임(AOT) 컴파일을 활용하여 Lambda 콜드 스타트 시간을 개선하여 .NET 7 Lambda 함수를 AWS SAM 빌드 및 패키징합니다. 자세히 알아보려면 네이티브 AOT 컴파일을 사용하여 .NET 7 Lambda 함수 빌드를 참조하세요.	2022년 11월 15일

로컬 디버깅 및 테스트에 AWS SAMCLITerraform 지원이 추가됨	Terraform프로젝트 내에서 AWS SAM CLI를 사용하여 Lambda 함수 및 레이어의 로컬 디버깅 및 테스트를 수행합니다. 자세한 내용은 AWS SAMCLI 지원Terraform을 참조하세요.	2022년 11월 14일
EventBridge AWS SAM 스케줄러에 대한 지원이 추가됨	AWS Serverless Application Model(AWS SAM) 템플릿 사양은 AWS Lambda 및 AWS Step Functions에 관한 EventBridge 스케줄러에서 이벤트를 스케줄링하는 데 사용할 수 있는 간단하고 간단한 명령문 제공. 자세한 내용은 EventBridge Scheduler를 사용한 일정 수립을 참조하세요.	2022년 11월 10일
AWS SAMCLI설치 지침을 단순화	AWS SAMCLI사전 요구 사항 및 선택적 단계가 별도의 페이지로 이동됨. 지원되는 운영 체제의 설치 단계는 AWS SAMCLI설치하기에서 찾을 수 있습니다.	2022년 11월 4일
Windows 10 사용자가 긴 경로를 사용할 수 있도록 수정 사항이 추가됨	AWS SAM CLI 앱 템플릿 저장소에는 Windows 10 MAX_PATH 제한으로 인해 `sam init`의 실행 시 오류가 발생할 수 있는 일부 긴 파일 경로가 포함되어 있습니다. 자세한 정보는 AWS SAM CLI 설치하기를 참조하세요.	2022년 11월 4일

최초 배포를 위한 점진적 배포 프로세스가 업데이트됨	AWS CodeDeploy를 사용하여 Lambda 함수를 점진적으로 배포하려면 두 단계가 필요합니다. 자세히 알아보려면 Lambda 함수를 처음으로 점진적으로 배포하기를 참조하세요.	2022년 10월 13일
더 많은 유형의 이벤트에 대한 추가 Lambda 이벤트 필터링 지원	`MSK`, `MQ`, 및 `SelfManagedKafka` 이벤트 소스 유형에 `FilterCritera` 속성이 추가됨	2022년 10월 13일
AWS SAM파이프라인에 OpenID Connect (OIDC) 지원이 추가됨	AWS SAM은 Bitbucket, GitHub Actions에 대한 OpenID Connect (OIDC) 사용자 인증, GitLab의 지속적 통합 및 전달 (CI/CD) 플랫폼을 지원합니다. 자세히 알아보려면 AWS SAM 파이프라인을 통한 OIDC 사용자 계정 사용하기를 참조하세요.	2022년 10월 13일
JWT구성 속성에 대한 참고 사항	`OAuth2Authorizer` 를 위한 `JwtConfiguration` 에 따른 `issuer` 및 `audience` 속성의 정의에 대하여 메모가 추가되었습니다.	2022년 10월 7일

함수 및 스테이트머신 이벤트 소스의 새 속성	Enabled및 State 속성이 AWS::Serverless::Function 를 위한 CloudWatchEvent 이벤트 소스에 추가되었습니다. State 속성은 AWS::Serverless::Function 및 AWS::Serverless::StateMachine 의 Schedule 이벤트 소스에 추가되었습니다.	2022년 10월 6일
AWS SAM이제 커넥터를 통상적으로 사용 가능	커넥터는 AWS::Serverless::Connector 로 식별되는 AWS SAM 추상 리소스 유형으로, 서버리스 애플리케이션 리소스 간에 권한을 프로비저닝하는 간단하고 안전한 방법을 제공합니다. 자세히 알아보려면 AWS Serverless Application Model커넥터를 사용한 리소스 권한 관리를 참조하세요.	2022년 10월 6일
에 새 sam sync 옵션이 추가됨 AWS SAMCLI	sam sync에 --dependency-layer 및 --use-container 옵션이 추가됨	2022년 9월 20일
에 대한 새 sam 배포 옵션이 추가됨AWS SAMCLI	--on-failure 에 옵션이 추가됨sam deploy	2022년 9월 9일
esbuild 지원이 이제 통상적으로 제공됨	Node.js Lambda 함수를 빌드하고 패키징하려면 esbuild 자바스크립트 번들러와 AWS SAM CLI 함께 사용할 수 있습니다.	2022년 9월 1일

AWS SAMCLI업데이트된 텔레메트리	수집된 시스템 및 환경 정보에 대한 설명이 사용 속성의 해시 값을 포함하도록 업데이트되었습니다.	2022년 9월 1일
에 로컬 환경 변수 지원이 추가 AWS SAMCLI	로컬에서 Lambda 함수를 호출할 때와 로컬에서 API Gateway를 실행할 때 AWS SAM CLI와 함께 환경 변수를 사용합니다.	2022년 9월 1일
Lambda 명령 세트 아키텍처 지원	AWS SAMCLI를 사용하여 `x86_64` 혹은 `arm64` 명령어 세트 아키텍처를 위한 Lambda 함수와 Lambda 레이어를 구축합니다. 자세한 내용은 `AWS::Serverless::Function` 리소스 유형의 아키텍처 속성 및 `AWS::Serverless::LayerVersion` 리소스 유형의 호환 가능한 아키텍처 속성을 참조하세요.	2021년 10월 1일
예제 파이프라인 구성 생성	새로운 `sam pipeline bootstrap` 및 `sam pipeline init` 명령어를 사용하여 여러 CI/CD 시스템에 대한 예시 파이프라인을 생성하려면 AWS SAM CLI 을(를) 사용하세요. 자세한 내용은 예제 CI/CD 파이프라인 생성을 참조하세요.	2021년 7월 21일

AWS SAMCLIAWS CDK 통합 (미리 보기, 2단계)	퍼블릭 프리뷰 릴리스의 2단계부터 이제 AWS SAM CLI를 사용하여 AWS CDK 애플리케이션을 패키징하고 배포할 수 있습니다. AWS SAM CLI를 사용하여 샘플 AWS CDK 애플리케이션을 직접 다운로드할 수도 있습니다. 자세한 내용은 AWS 클라우드 개발 키트 (AWS CDK)(미리 보기)을 참조하세요	2021년 7월 13일
함수의 이벤트 소스로 RabbitMQ 지원	서버리스 함수의 이벤트 소스로 RabbitMQ 지원이 추가됨 자세한 내용은`AWS::Serverless::Function` 리소스 유형의 `MQ` 이벤트 소스의 `SourceAccessConfigurations` 속성을 참조하세요.	2021년 7월 7일
Amazon ECR 빌드 컨테이너 이미지를 사용하여 서버리스 애플리케이션 배포	Amazon ECR 빌드 컨테이너 이미지를 사용하여 AWS CodePipeline 젠킨스, GitLab CI/CD, GitHub 액션과 같은 일반적인 CI/CD 시스템으로 서버리스 애플리케이션을 배포할 수 있습니다. 자세한 내용을 알아보려면 서버리스 애플리케이션 배포를 참조하세요.	2021년 6월 24일
AWS툴킷을 사용한 AWS SAM 애플리케이션 디버깅	AWS이제 툴킷은 통합 개발 환경 (IDE) 및 런타임의 다양한 조합을 통해 단계별 디버깅을 지원합니다. 자세한 내용은 AWS 툴킷 사용하기를 참조하세요.	2021년 5월 20일

AWS SAMCLIAWS CDK 통합 (미리 보기)	이제 AWS SAM CLI를 사용하여 AWS CDK 애플리케이션을 로컬에서 테스트하고 빌드할 수 있습니다. 이것은 평가판 미리 보기 릴리스입니다. 자세한 내용은 AWS 클라우드 개발 키트 (AWS CDK)(미리 보기)을 참조하세요	2021년 4월 29일
기본 컨테이너 이미지 리포지토리가 Amazon ECR Public으로 변경됨	기본 컨테이너 이미지 리포지토리가 DockerHub에서 Amazon ECR 퍼블릭으로 변경되었습니다. 자세한 정보는 이미지 리포지토리를 참조하세요.	2021년 4월 6일
AWS SAMCLI자동 빌드	이제 자동으로 빌드되는 AWS SAM CLI의 시험판 버전을 설치할 수 있습니다. 자세한 내용은 설치에서 선택한 OS 하위 항목의 Nightly build를 참조하세요.AWS SAMCLI	2021년 3월 25일
빌드 컨테이너 환경 변수 지원	이제 환경 변수를 전달하여 컨테이너를 빌드할 수 있습니다. 자세한 내용은 `sam build`의 `--container-env-var` 및 `--container-env-var-file` 옵션을 참조하세요.	2021년 3월 4일
새 Linux 설치 프로세스	이제 네이티브 Linux 설치관리자 프로그램을 사용하여 AWS SAM CLI 설치할 수 있습니다. 자세한 내용은 Linux에서 AWS SAM CLI 설치하기를 참조하세요.	2021년 2월 10일

EventBridge용 DLQ(Dead Letter Queue)대기열 지원	EventBridge의 DLQ 대기열과 서버리스 함수 및 상태 머신을 위한 Schedule이벤트 소스에 대한 지원이 추가됨 자세한 내용은 AWS::Serverless::Function 및 AWS::Serverless::StateMachine 리소스 유형 모두에 대한 EventBridgeRule 및 Schedule 이벤트 소스의 DeadLetterConfig 속성을 참조하세요.	2021년 1월 29일
사용자 지정 체크포인트 지원	서버리스 함수를 위한 DynamoDB 및 Kinesis 이벤트 소스의 사용자 지정 체크포인트에 대한 지원이 추가됨 자세한 내용은 AWS::Serverless::Function 리소스 유형의 Kinesis 및 DynamoDB 데이터 형식의 FunctionResponseTypes 속성을 참조하세요.	2021년 1월 29일
텀블링 윈도우 지원	서버리스 함수를 위한 DynamoDB 및 Kinesis 이벤트 소스의 텀블링 윈도우 지원이 추가됨. 자세한 내용은 AWS::Serverless::Function 리소스 유형의 Kinesis 및 DynamoDB 데이터 형식의 TumblingWindowInSeconds 속성을 참조하세요.	2020년 12월 17일

웜 컨테이너 지원	AWS SAM CLI 명령을 사용하여 sam local start-api 및 sam local start-lambda 를 로컬에서 테스트할 때 웜 컨테이너에 대한 지원이 추가됨 자세한 내용은 해당 명령의 --warm-containers 옵션을 참조하세요.	2020년 12월 16일
Lambda 컨테이너 이미지에 대한 지원	Lambda 컨테이너 이미지에 대한 지원이 추가됨 자세한 내용은 애플리케이션 빌드하기를 참조하세요.	2020년 12월 1일
코드 서명 지원	코드 서명 및 서버리스 애플리케이션 코드의 신뢰할 수 있는 배포에 대한 지원이 추가됨 자세한 내용은 AWS SAM애플리케이션을 위한 코드 서명 구성을 참조하세요.	2020년 11월 23일
Parallel 및 캐시된 빌드 지원	sam build명령에 두 가지 옵션을 추가하여 서버리스 애플리케이션 빌드의 성능 개선. 하나는 함수와 레이어를 순차적으로 빌드하지 않고 병렬로 빌드하는 --parallel 이고, 다른 하나는 재빌드가 필요한 변경 사항이 없을 때 이전 빌드의 빌드 아티팩트를 사용하는 --cached입니다.	2020년 11월 10일

Amazon MQ 지원 및 상호 TLS 인증	Amazon MQ에 대한 지원을 서버리스 함수의 이벤트 소스로 추가함. 자세한 내용은 `AWS::Serverless::Function` 리소스 유형의 `EventSource` 및 `MQ` 데이터 유형을 참조하세요. 또한 API Gateway API 및 HTTP API에 대한 상호 전송 레이어 보안(TLS) 인증에 대한 지원이 추가됨 자세한 내용은 `AWS::Serverless::Api` 리소스 유형의 `DomainConfiguration` 데이터 유형 또는 `AWS::Serverless::HttpApi` 리소스 유형의 `HttpApiDomainConfiguration` 데이터 유형을 참조하세요.	2020년 11월 5일
HTTP API용 Lambda 권한 부여자에 대한 지원	`AWS::Serverless::HttpApi` 리소스 유형에 대한 Lambda 권한 부여자에 대한 지원이 추가됨 자세한 내용은 Lambda 권한 부여자 예제(`AWS::Serverless::HttpApi`)를 참조하세요.	2020년 10월 27일
여러 구성 파일 및 환경 지원	AWS SAMCLI 명령의 기본 파라미터 값을 저장하는 여러 구성 파일 및 환경에 대한 지원이 추가됨 자세한 정보는 AWS SAMCLI구성 파일을 참조하세요.	2020년 9월 24일

Step Functions를 통한 X-Ray 지원 및 API에 대한 액세스 제어 시 참조	서버리스 상태 머신의 이벤트 소스로 X-Ray에 대한 지원이 추가됨 자세한 내용은 `AWS::Serverless::StateMachine` 리소스 유형의 `Tracing` 속성을 참조하세요. API에 대한 액세스를 제어할 때 참조에 대한 지원도 추가됨 자세한 정보는 `ResourcePolicyStatement` 데이터 유형을 참조하세요.	2020년 9월 17일
Amazon MSK 지원	Amazon MSK에 대한 지원을 서버리스 함수의 이벤트 소스로 추가함 이렇게 하면 Amazon MSK 주제의 기록이 Lambda 함수를 트리거할 수 있습니다. 자세한 내용은 `AWS::Serverless::Function` 리소스 유형의 `EventSource` 및 `MSK`데이터 유형을 참조하세요.	2020년 8월 13일
Amazon EFS 지원	Amazon EFS 파일 시스템을 로컬 디렉터리에 마운트하기 위한 지원이 추가됨. 이렇게 하면 Lambda 함수 코드가 공유 리소스에 액세스하여 수정할 수 있습니다. 자세한 내용은 `AWS::Serverless::Function` 리소스 유형의 `FileSystemConfigs` 속성을 참조하세요.	2020년 6월 16일

서버리스 애플리케이션 오케스트레이션	AWS SAM를 사용하여 Step Functions 상태 머신을 생성하여 애플리케이션을 오케스트레이션할 수 있는 지원이 추가됨. 자세한 내용은 AWS Step Functions 및 `AWS::Serverless::StateMachine` 리소스 유형을 사용하는 AWS리소스 오케스트레이션을 참조하세요.	2020년 5월 27일
사용자 지정 런타임 구축	사용자 지정 런타임을 구축하는 기능이 추가됨. 자세한 정보는 사용자 지정 런타임 구축을 참조하세요.	2020년 5월 21일
레이어 구축	개별 `LayerVersion` 리소스를 구축할 수 있는 기능이 추가됨. 자세한 내용은 레이어 구축을 참조하세요	2020년 5월 19일
생성된 CloudFormation리소스	AWS SAM생성되는 CloudFormation 리소스와 해당 리소스를 참조하는 방법에 대한 세부 정보가 제공됨. 자세한 내용을 알아보려면 생성된 CloudFormation리소스를 참조하세요.	2020년 4월 8일
AWS보안 인증 설정	다른 AWS 도구(예: AWS SDK 중 하나 혹은 AWS CLI)와 함께 보안 인증의 사용을 아직 설정하지 않은 경우 AWS 보안 인증을 설정하는 지침이 추가됨. 자세한 내용은 AWS보안 인증 설정을 참조하세요.	2020년 1월 17일

AWS SAM 사양 및 AWS SAM CLI 업데이트	GitHub에서 AWS SAM 사양을 마이그레이션함. 자세한 내용은 AWS SAM사양을 참조하세요. 또한 `sam deploy`명령을 변경하여 배포 워크플로가 업데이트됨.	2019년 11월 25일
API Gateway API 및 정책 템플릿 업데이트에 대한 액세스를 제어하기 위한 새로운 옵션	API Gateway API에 대한 액세스를 제어하기 위한 새로운 옵션(IAM 권한, API 키, 리소스 정책)이 추가됨. 자세한 내용은 API Gateway API에 대한 액세스 제어를 참조하세요. RekognitionFacesPolicy와 ElasticSearch HTTP Post 정책이라는 두 개의 정책 템플릿도 업데이트됨. 자세한 내용은 AWS SAM정책 템플릿을 참조하세요.	2019년 8월 29일
시작하기 및 업데이트	시작 장을 업데이트하여 AWS SAM CLI 및 Hello World 사용 지침서의 설치 지침이 개선됨. 자세한 내용은 AWS SAM로 시작하기를 참조하세요.	2019년 7월 25일
API Gateway API에 대한 액세스 제어	API Gateway API에 대한 액세스 제어에 대한 지원이 추가됨. 자세한 내용은 API Gateway API에 대한 액세스 제어를 참조하세요.	2019년 3월 21일

에 sam publish가 추가됨 AWS SAMCLI	AWS SAM CLI의 새 sam publish 명령은 서버리스 애플리케이션을 AWS Serverless Application Repository에 게시하는 프로세스를 간소화합니다. 자세한 내용은 AWS SAM CLI을 사용하여 서버리스 애플리케이션 게시하기를 참조하세요.	2018년 12월 21일
중첩 애플리케이션과 레이어 지원	중첩 애플리케이션과 레이어 지원이 추가됨 자세한 내용은 중첩 애플리케이션 사용 및 레이어로 작업하기를 참조하세요.	2018년 11월 29일
에 sam build가 추가됨AWS SAMCLI	AWS SAM CLI의 새 sam build명령은 종속물이 있는 서버리스 애플리케이션을 컴파일하는 프로세스를 단순화하여 이러한 애플리케이션을 로컬에서 테스트하고 배포할 수 있도록 합니다. 자세한 내용은 애플리케이션 빌드하기를 참조하세요.	2018년 11월 19일
에 대한 새 설치 옵션이 추가됨 AWS SAMCLI	AWS SAM CLI에 대한 Linuxbrew(리눅스), MSI (윈도우), Homebrew(macOS) 설치 옵션이 추가됨. 자세한 정보는 AWS SAM CLI설치하기를 참조하세요	2018년 11월 7일
새 안내서	이 문서는 첫 번째 AWS Serverless Application Model 개발자 안내서 릴리스입니다.	2018년 10월 17일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.