
AWS SAM で始める
CI/CD
インフラ1G 川越 鉄郎

自己紹介

名前：Nekotto-Chan

職種：AWS

趣味：The Sandbox

目標：NFT頑張る

目次

1. SAMとは
2. SAM Template
3. SAM CLI
4. Lambdaデプロイの自動化

1. SAMとは

AWS CloudFormationの拡張機能
現在は、9つのAWSリソースが作成できる

- AWS::Serverless::Api
  ⇒ API Gateway
- AWS::Serverless::Function
  ⇒ Lambda
- AWS::Serverless::StateMachine
  ⇒ Step Functions

1. SAMとは

特徴

- 書式はCloudFormationと同じYAMLもしくはJSONで記載できる
- 書式がCloudFormationよりシンプル
- CloudFormationと共存可能

2. SAM Template

SAMの場合

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Transform: 'AWS::Serverless-2016-10-31'
Description: HelloWorld
Resources:
  HelloWorld:
    Type: 'AWS::Serverless::Function'
    Properties:
      Handler: index.handler
      Runtime: nodejs8.10
      CodeUri: s3://xxx-bucket/xxx.zip
      Description: HelloWorld
      MemorySize: 128
      Timeout: 3
      Events:
        GetResource:
          Type: Api
          Properties:
            Path: /hello
            Method: get
```

CloudFormationの場合

2. SAM Template

通常のCloudFormationと共存可能!!

```yaml
AWSTemplateFormatVersion: '2010-09-09'
Transform: 'AWS::Serverless-2016-10-31'
Description: HelloWorld
Resources:
  HelloWorld:
    Type: 'AWS::Serverless::Function'
    Properties:
      Handler: index.handler
      Runtime: nodejs10.x
      CodeUri: s3://xxx-bucket/xxx.zip
      Description: HelloWorld
      MemorySize: 128
      Timeout: 3
  S3Bucket:
    Type: AWS::S3::Bucket
    Properties:
      BucketName: my-bucket
```

SAM

CloudFormation

3. SAM CLI

SAM Templateに対して、SAMコマンドを
実行することで、ビルド、パッケージング、
デプロイ等を行うことができる!!
以下の基本的なSAM CLIについて解説するよ!!

- sam validate
- sam build
- sam package
- sam deploy

3. SAM CLI

sam validate

SAM Templateの文法をチェックできる

例 : sam validate --lint -t template.yaml

```
[cloudshell-user@ip-10-4-95-235 lambda-repo]$ sam validate --lint -t template.yaml
E3030 You must specify a valid value for Runtime (python3.12). Valid values are ["dotnet6",
4.3", "nodejs4.3-edge", "nodejs6.10", "nodejs8.10", "provided", "provided.al2", "python2.7",
/home/cloudshell-user/tmp/lambda-repo/template.yaml:16:3

Error: Linting failed. At least one linting rule was matched to the provided template.
```

3. SAM CLI

sam build

アプリケーションをビルドしたり、
外部ライブラリをインストールできる
例 : sam build -t template.yaml

```
[Container] 2024/01/25 21:56:28.122228 Running command sam build --template-file template.yaml

Building layer 'LambdaLayer'
 Running PythonPipBuilder:ResolveDependencies
 Running PythonPipBuilder:CopySource
Building codeuri: /codebuild/output/src2040242340/src/app_dir runtime: python3.12 metadata: {} architecture: x86_64 functions: LambdaFunction
requirements.txt file not found. Continuing the build without dependencies.
 Running PythonPipBuilder:CopySource

Build Succeeded

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

3. SAM CLI

sam package

Lambdaにデプロイするコードをzip化し、
指定したS3にアップロードする

例 : sam package --template-file .aws-sam/build/template.yaml
--output-template-file package.yml --s3-bucket「S3BucketName」

```
[Container] 2024/01/25 22:47:26.076197 Running command sam package --template-file .aws-sam/build/template.yaml --output-template-file package.yml --s3-bucket "${S3BucketName}"

    Uploading to 2c7f739271557ac8a641d3fcd58917be  262144 / 978853  (26.78%)
    Uploading to 2c7f739271557ac8a641d3fcd58917be  524288 / 978853  (53.56%)
    Uploading to 2c7f739271557ac8a641d3fcd58917be  786432 / 978853  (80.34%)
    Uploading to 2c7f739271557ac8a641d3fcd58917be  978853 / 978853  (100.00%)
File with same data already exists at 02a6b05ac854582da2f5f90a785e908e, skipping upload

Successfully packaged artifacts and wrote output template to file package.yml.
Execute the following command to deploy the packaged template
sam deploy --template-file /codebuild/output/src2304752109/src/package.yml --stack-name <YOUR STACK NAME>
```

3. SAM CLI

sam deploy

S3にあるコードをCloudFormation Stackを利用し、Lambdaにデプロイする

例 : sam deploy --template-file package.yml --stack-name sam-app-stack --capabilities CAPABILITY_IAM

```
[Container] 2024/01/25 23:23:18.132430 Running command sam deploy --template-file package.yml --stack-name sam-app-stack --capabilities CAPABILITY_IAM

    Deploying with following values
    ===============================
    Stack name                   : sam-app-stack
    Region                       : None
    Confirm changeset            : False
    Disable rollback             : False
    Deployment s3 bucket         : None
    Capabilities                 : ["CAPABILITY_IAM"]
    Parameter overrides          : {}
    Signing Profiles             : {}

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

Waiting for changeset to be created..

CloudFormation stack changeset
-----------------------------------------------------------------------------------------
Operation                LogicalResourceId        ResourceType             Replacement
-----------------------------------------------------------------------------------------
+ Add                    LambdaFunctionRole       AWS::IAM::Role           N/A
+ Add                    LambdaFunction           AWS::Lambda::Function    N/A
+ Add                    LambdaLayer7ff05f957a    AWS::Lambda::LayerVers   N/A
                                                  ion
-----------------------------------------------------------------------------------------
```

```
CloudFormation events from stack operations (refresh every 5.0 seconds)
-------------------------------------------------------------------------------------------------
ResourceStatus           ResourceType             LogicalResourceId        ResourceStatusReason
-------------------------------------------------------------------------------------------------
CREATE_IN_PROGRESS       AWS::CloudFormation::S   sam-app-stack            User Initiated
                         tack
CREATE_IN_PROGRESS       AWS::Lambda::LayerVers   LambdaLayer7ff05f957a    -
                         ion
CREATE_IN_PROGRESS       AWS::IAM::Role           LambdaFunctionRole       -
CREATE_IN_PROGRESS       AWS::IAM::Role           LambdaFunctionRole       Resource creation
                                                                           Initiated
CREATE_IN_PROGRESS       AWS::Lambda::LayerVers   LambdaLayer7ff05f957a    Resource creation
                         ion                                               Initiated
CREATE_COMPLETE          AWS::Lambda::LayerVers   LambdaLayer7ff05f957a    -
                         ion
CREATE_COMPLETE          AWS::IAM::Role           LambdaFunctionRole       -
CREATE_IN_PROGRESS       AWS::Lambda::Function    LambdaFunction           -
CREATE_IN_PROGRESS       AWS::Lambda::Function    LambdaFunction           Resource creation
                                                                           Initiated
CREATE_COMPLETE          AWS::Lambda::Function    LambdaFunction           -
CREATE_COMPLETE          AWS::CloudFormation::S   sam-app-stack            -
                         tack
-------------------------------------------------------------------------------------------------

Successfully created/updated stack - sam-app-stack in None
```

4. Lambdaデプロイの自動化

背景

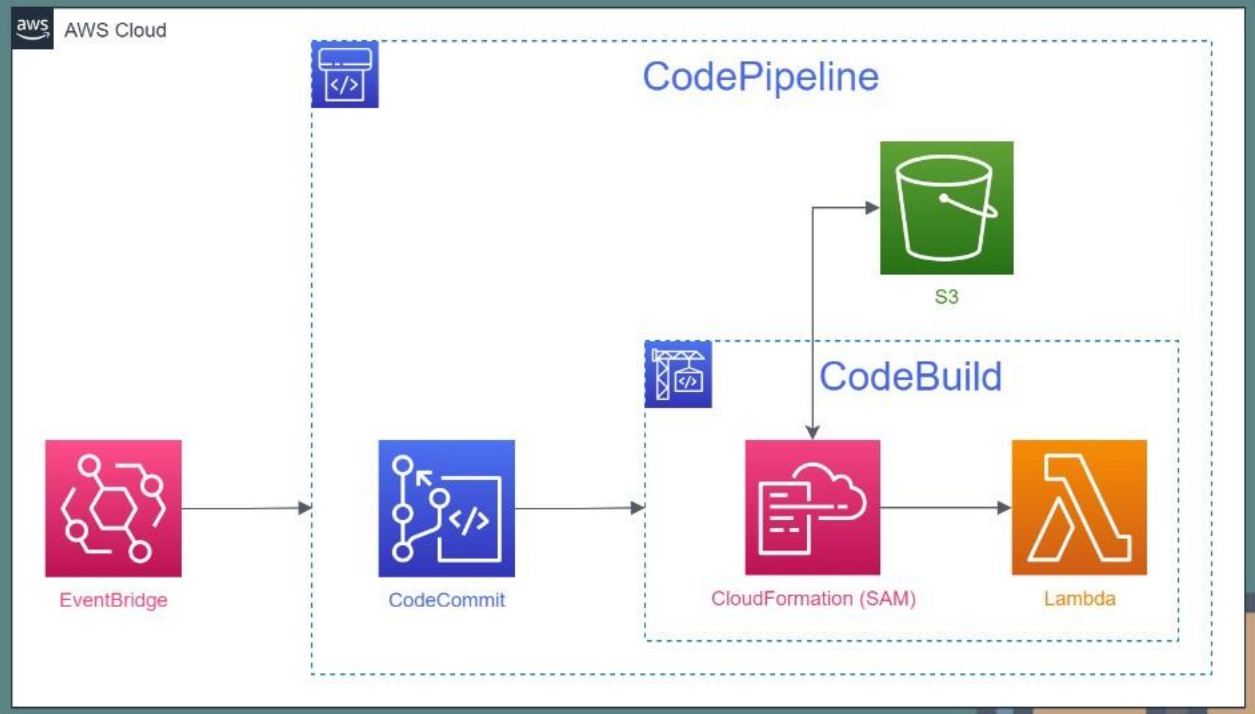
現場でLambdaのコード管理ができていない...

目標

CodeCommitでコード管理し、
Lambdaへ同期させる!!

4. Lambdaデプロイの自動化

構成図



4. Lambdaデプロイの自動化

特徴

- トリガーは、CodeCommitのコード変更
- CodeBuildで、SAMを実行し、CodeCommitのコードをLambdaにデプロイする
- CodeBuildで、外部ライブラリをインストールし、Lambda Layerとして設定可能

最後に

いかがだったでしょうか?
1年を通して、楽しく! 面白く! 学べるような記事を
投稿してきたつもりです (*゜▽゜)ノ
みなさんのお役に立っていれば幸いです!
今年度、お疲れ様でした (o_ _)o

※https://github.com/Flupinochan/CodeCommitToLambda