AWS Lambda と Parameter Store の環境を CDK で構築する

2021年5月23日　engineering

aws　lambda　parameter_store　cdk　aws-cdk

こんにちは、 @kz_morita です。

今回は，AWS CDK を使って Parameter Store のデータを参照する Lambda の環境を構築します．

事前準備

awc-cli のインストールと Credential の設定をしておきます．

自分の環境は以下のような感じです．

```
$ aws --version
aws-cli/1.18.69 Python/3.6.9 Linux/5.4.0-72-generic botocore/1.16.19
```

具体的なインストール方法などは，以下を参考にしてみてください．

https://docs.aws.amazon.com/ja_jp/cli/latest/userguide/cli-chap-welcome.html

CDK のインストールとプロジェクトの作成

CDK は AWS のインフラをプログラミング言語で記述できるものです．ここでは TypeScript を使ってインフラを構築します． npm install で cdk をインストールします．

```
$ npm install -g aws-cdk

$ cdk --versoin
1.105.0 (build 4813992)
```

インストールが完了したら，プロジェクトを作成します．

```
$ mkdir lambda-cdk-sample
$ cd lambda-cdk-sample
$ cdk init app --language typescript
```

上記を実行すると以下のようないろいろなファイルが生成されます．

```
.
├─── README.md
├─── bin
│    └─── lambda-cdk-sample.ts
├─── cdk.json
├─── cdk.out/
│    ├─── LambdaCdkSampleStack.template.json
│    ├─── cdk.out
│    ├─── manifest.json
│    └─── tree.json
```

```
├── jest.config.js
├── lib
│   └── lambda-cdk-sample-stack.ts
├── node_modules/
├── package-lock.json
├── package.json
├── test
│   └── lambda-cdk-sample.test.ts
└── tsconfig.json
```

主に触っていくのが，bin ディレクトリ配下と，lib ディレクトリ配下になります．

また，cdk を初めて利用する場合は，bootstrap が必要なので，以下のコマンドで実行しておきます．

```
$ cdk bootstrap

# profile を指定する場合は以下
$ cdk bootstrap --profile {profile_name}
```

Lambda Stackの実装

上記で作成されたファイルのなかで，EntryPoint として実行されるのは，bin ディレクトリ以下のファイルです．

上記の例だと，`lambda-cdk-sample.ts` というファイルがそれに当たります．

lambda-cdk-sample.ts を以下のように変更します．

```
#!/usr/bin/env node
import 'source-map-support/register';
import * as cdk from '@aws-cdk/core';
import { LambdaStack } from '../lib/lambda-stack';

const app = new cdk.App();

new LambdaStack(app, 'SampleLambdaStack', {});
```

このファイルから読み出す，`lib/lambda-stack.ts` を以下のように作成します．

```
import * as cdk from '@aws-cdk/core';
import * as iam from '@aws-cdk/aws-iam';
import * as lambda from '@aws-cdk/aws-lambda';

export interface LambdaStackProps extends cdk.StackProps {
}

export class LambdaStack extends cdk.Stack {

  constructor(scope: cdk.Constuct, id: string, props: LambdaStackPro
    super(scope, id, props);

    this.createLambdaFunction(props)
  }

  private createLambdaFunction(props: LambdaStackProps) {
    const executionLambdaRole = new iam.Role(
      this,
      'SampleLambdaExecutionRole',
      {
        roleName: 'sample-lambda-execution-role',
        assumedBy: new iam.ServicePrincipal('lambda.amazonaws.com'),
        managedPolicies: [
```

```
            iam.ManagedPolicy.fromAwsManagedPolicyName(
              'service-role/AWSLambdaBasicExecutionRole'
            )
          ],
        }
      );

      new lambda.Function(this, 'LambdaFunction', {
        functionName: 'sample-lambda',
        runtime: lambda.Runtime.PYTHON_3_8,
        code: lambda.AssetCode.fromAsset('src/sample-lambda/lambda_fun
        handler: 'lambda_function.lambda_handler',
        role: executionLambdaRole
      });
    }
  }
```

やっていることは，シンプルで Lambda を実行するための IAM Role を作成し，その Role を紐付けた Lambda Function を生成しています．

Role の作成

```
  const executionLambdaRole = new iam.Role(
    this,
    'SampleLambdaExecutionRole',
    {
      roleName: 'sample-lambda-execution-role',
      assumedBy: new iam.ServicePrincipal('lambda.amazonaws.com'),
      managedPolicies: [
        iam.ManagedPolicy.fromAwsManagedPolicyName(
          'service-role/AWSLambdaBasicExecutionRole'
        )
```

```
      ],
    }
  );
```

Lambda Function の作成

今回は，Python 3.8 の環境を構築しています． これは `runtime` に指定しています．

また，実行するLambdaのソースコードは， `src/sample-lambda` ディレクトリを作成し，その中に lambda_function.py を作成しました． `code` に，`lambda.AssetCode.fromAsset('src/sample-lambda')` という形式でディレクトリを指定しています．

実行される関数は `handler` に指定しています．

`role` には先程生成した IAM Role を指定します．

```
new lambda.Function(this, 'LambdaFunction', {
  functionName: 'sample-lambda',    // 関数名
  runtime: lambda.Runtime.PYTHON_3_8,
  code: lambda.AssetCode.fromAsset('src/sample-lambda'),
  handler: 'lambda_function.lambda_handler',
  role: executionLambdaRole
});
}
```

上記のコードでは，Lambda と IAM Role を作成したので，ライブラリをインストールする必要があります．

```
$ npm i @aws-cdk/aws-lambda @aws-cdk/aws-iam
```

実行される Lambda Funtion の中身はとりあえず以下のようにします．

```
def lambda_handler(event, context):
    print("Lambda Sample")

    return {
        'statusCode': 200,
        'body': 'hello'
    }
```

この状態でビルドしてみます．

```
$ npm run build
```

とりあえず，最低限のコードができたので，デプロイしていきます．

Lambda をデプロイする

デプロイする前に，現在AWS上にあるリソースと今回 CDK で構築した内容の差分を確認できます．

```
$ npx cdk diff
```

```
# profile を指定する場合は以下
$ npx cdk diff --profile {profile_name}
```

確認して問題なさそうであれば以下コマンドでデプロイします．

```
$ npx cdk deploy

# profile を指定する場合は以下
$ npx cdk deploy --profile {profile_name}
```

デプロイが完了したら，AWS Console にサインインして，Lambda を確認すると上記で作成した Lambda が実際に作られていることを確認できます．

Lambda 上で Parameter Store からデータを読み込む

それでは，Lambda 上から Parameter Store のデータを読み込むようにコードを修正していきます．

全体の方針としては，以下のとおりです．

aws-cli で パラメータストアにデータを保存

パラメータの名前を CDK から，Lambda の環境変数として渡す

渡された パラメータ名を使って，Lambda 上でデータを取得する

パラメータストアにデータ保存

`/sample-lambda/parameter-1`, `/sample-lambda/parameter-2` という名前で適当な文字を保存します.

コマンドは以下のような感じです.

```
$ aws ssm put-parameter --name "/sample-lambda/parameter-1" --type "
$ aws ssm put-parameter --name "/sample-lambda/parameter-2" --type "
```

CDK 側の設定

CDK 側では以下の２つを行います.

- Lambda にパラメータストアからデータを読む権限を付ける
- Lambda に環境変数としてパラメータ名を渡す

lambda-stack.ts

コードは以下のような感じです． IAM Role にパラメータストアからの読み込み権限を与え，パラメータ名を環境変数でわたします.

```
  private createLambdaFunction(props: LambdaStackProps) {
    const executionLambdaRole = new iam.Role(
      this,
      'SampleLambdaExecutionRole',
      {
        roleName: 'sample-lambda-execution-role',
        assumedBy: new iam.ServicePrincipal('lambda.amazonaws.com'),
        managedPolicies: [
```

```
            iam.ManagedPolicy.fromAwsManagedPolicyName(
              'service-role/AWSLambdaBasicExecutionRole'
            ),
            // SSM から読み込む権限を付与
            iam.ManagedPolicy.fromAwsManagedPolicyName(
              'AmazonSSMReadOnlyAccess'
            ),
          ],
        }
      );

      new lambda.Function(this, 'LambdaFunction', {
        functionName: 'sample-lambda',
        runtime: lambda.Runtime.PYTHON_3_8,
        code: lambda.AssetCode.fromAsset('src/sample-lambda'),
        handler: 'lambda_function.lambda_handler',
        role: executionLambdaRole,
        // パラメータ名を環境変数で渡す
        environment: {
          Param_1: `/sample-lambda/parameter-1`,
          Param_2: `/sample-lambda/parameter-2`
        }
      });
    }
  }
```

Lambda からパラメータを読み込む

Lambda側では以下の処理を行います.

環境変数からパラメータ名を取得する

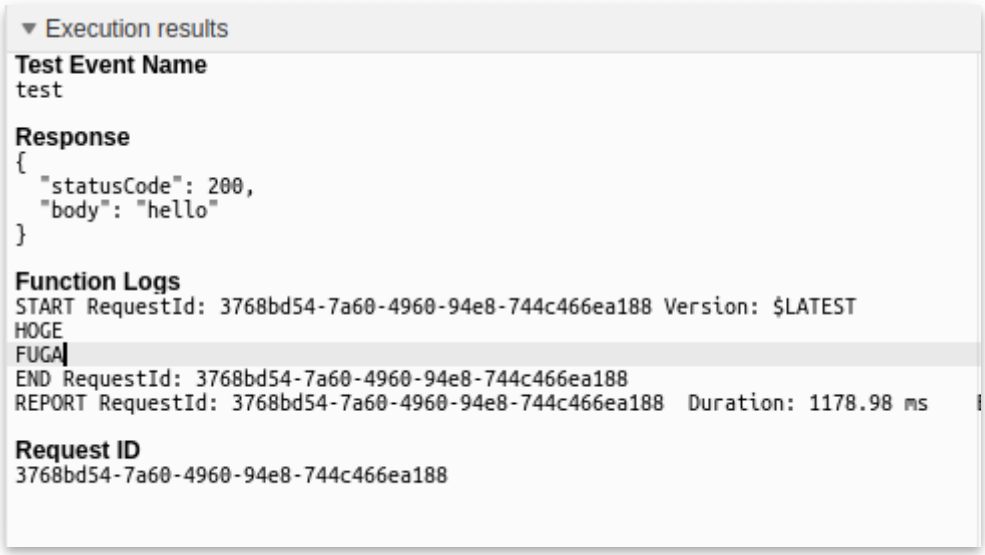
パラメータ名をもとに パラメータストアから取得する

表示

```python
import boto3
import os

def get_parameters(param_names):
    ssm = boto3.client('ssm')
    ssm_response = ssm.get_parameters(
        Names = param_names,
        WithDecryption = True
    )

    params = {}
    for param in ssm_response['Parameters']:
        params[ param['Name'] ] = param['Value']

    if len( ssm_response['InvalidParameters'] ) > 0:
        return {
            "error": "Invalid Parameter Name"
    }

    return params


def lambda_handler(event, context):
    param_1_name = ''
    param_2_name = ''

    # 環境変数から，パラメータ名を取得
    try:
        param_1_name = os.environ['Param_1']
        param_2_name = os.environ['Param_2']
    except KeyError as e:
        return {
            "error": "Parameter name not exists"
        }

    # パラメータストアからデータ取得
    params = get_parameters([param_1_name, param_2_name])
```

```
    # データ表示
    print(params[param_1_name])
    print(params[param_2_name])

    return {
        'statusCode': 200,
        'body': 'hello'
    }
```

これらをデプロイして AWS Console でテスト実行すると以下のように，パラメータストア内のデータが取得できることが確認できます．



まとめ

今回は，CDK を用いて Lambda を作成する方法を紹介し，その中でパラメータストアのデータを読み込むコードの実装も行いました．

インフラもコード化することで，バージョン管理やレビューが可能になるので積極的に使っていきたいと思います．