AWS Lambda Function URLs ハンズオン AWS CDK Version.

2022/05/14
和田 健一郎

[AWS CDKとは]

AWS Cloud Development Kit (AWS CDK) v2 はクラウドインフラをコードで定義し、AWS CloudFormationでプロビジョニングするためのフレームワークです。2021年12月に新しいVerson2が一般提供開始となっています。

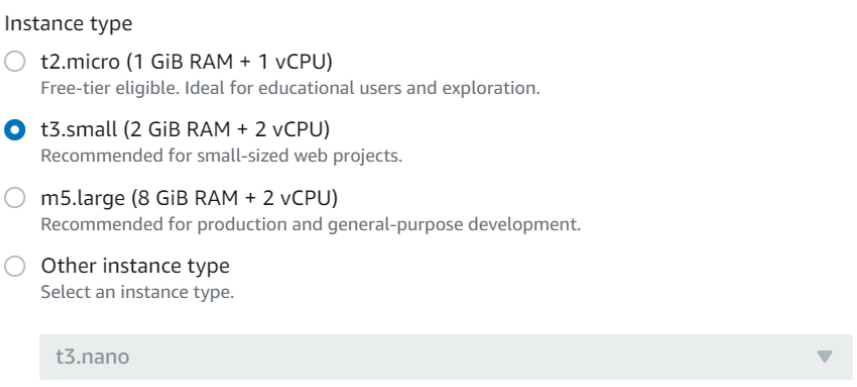
前提条件

このハンズオンテキストは、オレゴンリージョン(us-west-2)で動作確認しています。

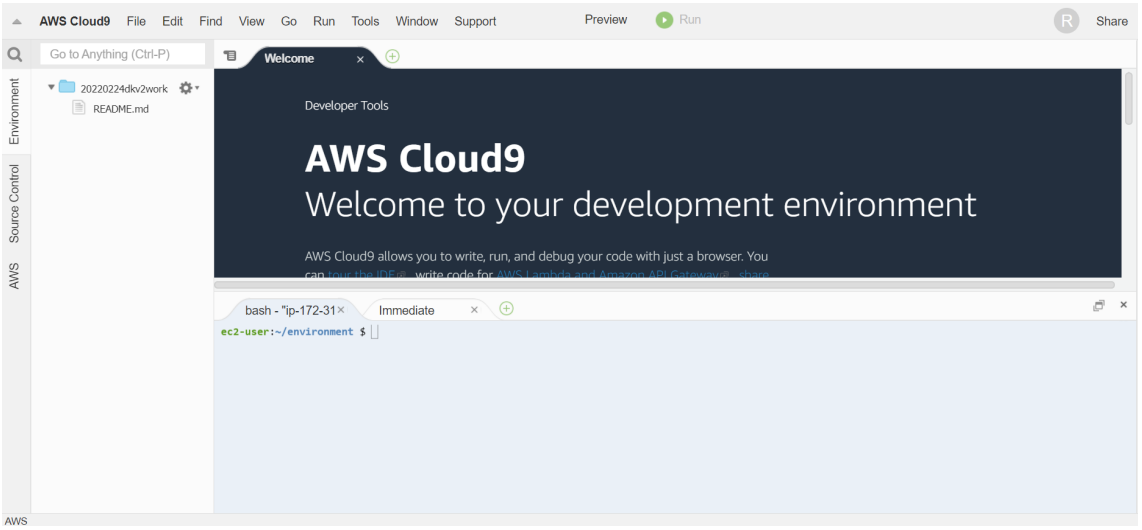
[Cloud9]の起動

AWS Cloud9はAWSが提供しているクラウド型IDEのサービスです。

1. Cloud9マネージメントコンソールにアクセスします（リージョンはどこでも問題ありません。作業は必ず、CDKv2を用いていないリージョンを使ってください。また、CDKv2の初期設定が完了しているAWSアカウントで行わないことを推奨します。）
2. [Create environment]をおします
3. [Name]に適当な値をいれ、[Next Step]をおします
4. [Instance Type]に[t3.small]を選びます



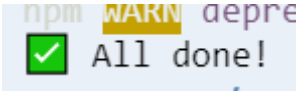
5. [network settings]でdefaultVPCもしくは、任意のVPCのPublic Subnetが指定されていることを確認し、[Next step]をおします
6. 確認画面で[Create environment]をおし、コンソールへアクセス可能となるまで数分間待ちます。
7. 以下の画面が起動すれば完了です。



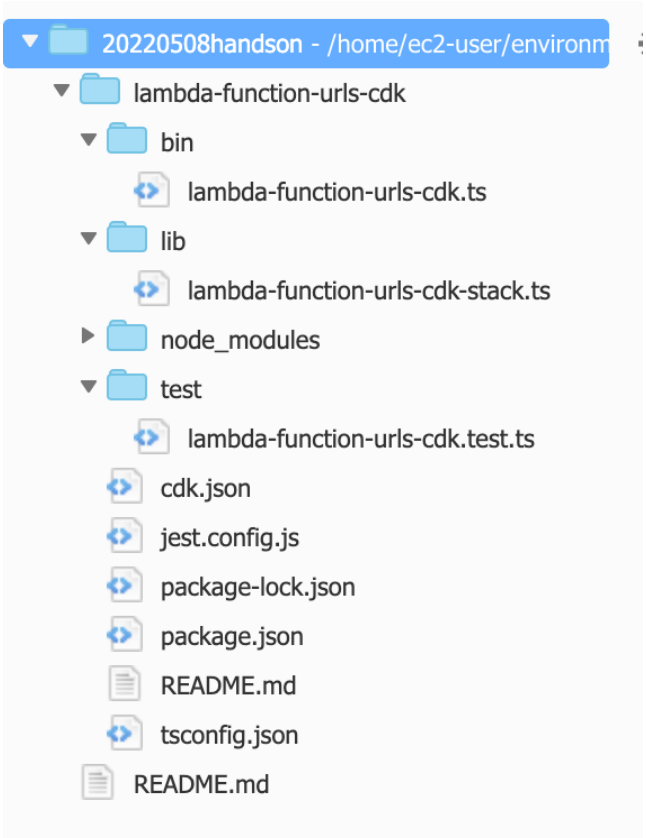
8. 起動したら、下部のコンソール上で、[cdk --version]を実行します。Cloud9はCDKやCDKの実行に必要なモジュールなどがあらかじめインストールされています。別環境を用いる場合のインストール手順は、このページを参考にしてください。また、Cloud9は起動時にマネージメントコンソールにログインしているIAMアカウント情報をもとに一時的に払い出された権限で動作しますが、別IDE環境を用いる場合、別途IAMアカウントの作成と設定(aws configure等)も必要です。
   表示されたバージョンが**2.21.0**以上であれば問題ありません(2022/05/08現在 **2.22.0** になっています)
9. 続いて利用するツール(**jq**)をインストールします。[commands.txt]の1番を実行します。

[L1 Constractで作る]

1. [commands.txt]の2番を実行し、作業ディレクトリを作成します
2. [commands.txt]の3番を実行し、TypeScript用にCDKを初期化します。以下が表示されたら成功です。

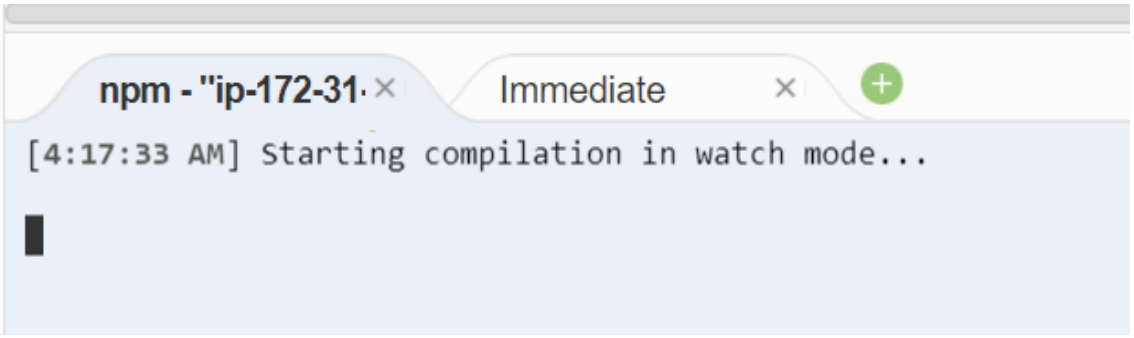


3. 以下のようにいくつかのフォルダやファイルができています。

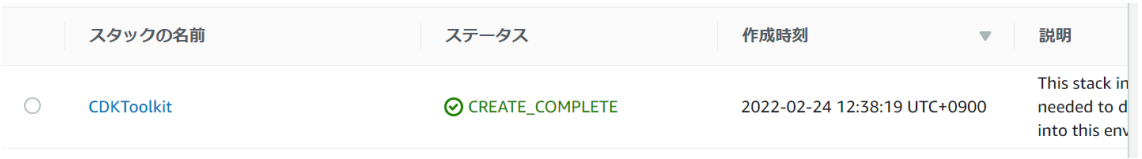


生成されるファイルの解説は、以前行われたAWS CDK v2 ハンズオンの解説をご覧ください。

4. TypeScriptはJavaScriptへ変更が行われる必要があります。ファイルを変更するたびにそれがリアルタイムで行われるよう[npm run watch]コマンドを実行します。
5. そのターミナルを閉じないように、+ボタンで別のターミナルを開きます(New Terminal)（作業用のディレクトリに必ず移動して下さい）



6. [lib/lambda-function-urls-cdk-stack.js]をダブルクリックで開き、[commands.txt]の3番の内容に置換し保存します。（置換のあと、タブを閉じると保存ダイアログが出てきます）
7. ここからlambda関数を作成していきます。まずbinやlibと同じ階層に[lambda]というディレクトリを作成します。
8. [lambda]ディレクトリに[index.ts]というファイルを作成します。
9. 作成したindex.tsを開き、[commands.txt]の5番を貼り付け、保存します。
10. [cdk synth]を実行します。
11. [cdk bootstrap]を実行します。このコマンドはCDKの初期設定に必須であり、リージョンごと、言語ごとに必ず初回作業時の実行が必要です。マネージメントコンソールでCloudFormationを確認すると以下のスタックが生成されています。



これは、CDKが動作に必要とする情報を保存するS3バケットやIAMロール等を含むCDKの動作環境構築を行うスタックです。
なお、すでに同一リージョン、同一言語で[cdk bootstrap]実行済みの場合、以下のように表示されます。

12. [cdk deploy]を実行します。確認が求められますので[y + Enter]を押します。
その後、CloudFormation経由で、Lambda関数が作成されます。
13. +ボタンで別のターミナルを開き、[commands.txt]の6番の[LambdaFunctionUrlsCdkStack.TheLambdaFunctionUrlsL1Urlの値]の部分を、deploy終了時のOutputsの[LambdaFunctionUrlsCdkStack.TheLambdaUrlFunctionL1]に表示されているURLに書き換えた上で実行します。以下のように表示されればOKです。

```
ken-admin:~/environment $ curl https://h5yj2sxekxrgsptahe5drqbq4a0pxhgn.lambda-url.us-west-2.on.aws/ | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    54  100    54    0     0    147      0 --:--:-- --:--:-- --:--:--   147
"Hello from Lambda Functions Urls! you request is GET"
```

これでこちらのパートは終了です。

[L2 Constractで作る]

前のパートでは、L1 Constractで作成してみましたが、
2022/04/22夜(JST)に、L2 Constact対応が盛り込まれたバージョン(v2.21.0)がリリースされています。
このパートではL2 Constactを使ってLambda Function Urlsを設定してみたいと思います。
また、簡易的なAPIを作ってみたいと思います。

1. 再度、[lib/lambda-function-urls-cdk-stack.js]をダブルクリックで開き、[commands.txt]の7番の内容に置換し保存します。
コメントのL2 Constractと書かれた以降が、L2 Constract対応の部分になります。
設定が減っているのがわかると思います。
2. [cdk diff]を実行しますと、L1 Constractと同様にLambdaおよびLambda Function URLsが設定されるのがわかると思います。
3. [cdk deploy]を実行します。同じように、確認が求められますので[y + Enter]を押します。
4. ターミナルで[commands.txt]の8番の[LambdaFunctionUrlsCdkStack.TheLambdaFunctionUrlsL2Urlの値]の部分を、deploy終了時のOutputsの[LambdaFunctionUrlsCdkStack.TheLambdaFunctionUrlsL2Url]に表示されているURLに書き換えた上で実行します。以下のように表示されればOKです。

```
ken-admin:~/environment $ curl -X POST -H "Content-Type: application/json" -d '{"Key":"aaaa", "Value":"bbb"}' https://64gpzlzreutrfvkqlqd6jphdge0cjgfx.lambda-url.us-west-2.on.aws/ | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    84  100    55  100    29    202    106 --:--:-- --:--:-- --:--:--   308
"Hello from Lambda Functions Urls! you request is POST"
```

5. 続いて、このLambdaを簡易的なAPIにしてみたいと思います。
再度、[lib/lambda-function-urls-cdk-stack.js]をダブルクリックで開き、[commands.txt]の9番の内容に置換し保存します。
コメントのDynamoDBと書かれた部分がDynamoDBの設定部分となり、作成したテーブルに対して、Lambdaに権限を与えています。
6. [lambda]ディレクトリに[api.ts]というファイルを作成します。

7. 作成したapi.tsを開き、[commands.txt]の10番を貼り付け、保存します。
8. [cdk diff]を実行しますと、DynamoDBのテーブルが作成され、それに付随する権限追加が行われることがわかると思います。
9. [cdk deploy]を実行します。同じように、確認が求められますので[y + Enter]を押します。
10. ターミナルで[commands.txt]の11番の[LambdaFunctionUrlsCdkStack.TheLambdaFunctionUrlsL2Urlの値]の部分を、deploy終了時のOutputsの[LambdaFunctionUrlsCdkStack.TheLambdaFunctionUrlsL2Url]に表示されているURLに書き換えた上で、上から実行します。
データが登録できること、データが参照できることがわかると思います。

これでこちらのパートは終了です。
基本的な使い方は、上記で完了です。
CORSの設定、認証の設定もCDK上で可能ですが、今回は割愛します。

[VPC LambdaにFunction Urls設定してみる]

Lambda Function URLsですが、VPC Lambdaにも設定することも可能です。
このパートでは、AWS CDKでVPCを構築し、そのVPCにLambdaをアタッチした上で、
Lambda Function URLsを設定してみます。

1. 再度、[lib/lambda-function-urls-cdk-stack.js]をダブルクリックで開き、[commands.txt]の12番の内容に置換し保存します。
コメントのVPCと書かれた以降が、VPCの構築部分、
VPC Lambdaと書かれたあとが、VPC Lambdaの設定の部分になります。
2. [cdk diff]を実行しますと、VPCが構築され、またそれに付随する権限追加が行われることがわかると思います。
3. [cdk deploy]を実行します。同じように、確認が求められますので[y + Enter]を押します。
4. ターミナルで[commands.txt]の13番の[LambdaFunctionUrlsCdkStack.TheLambdaFunctionUrlsVpcUrlの値]の部分を、deploy終了時の[LambdaFunctionUrlsCdkStack.TheLambdaFunctionUrlsVpcUrl]に表示されているURLに書き換えた上で実行します。以下のように表示されればOKです。

```
ken-admin:~/environment/lambda-function-urls-cdk (master) $ curl -X POST -H "Content-Type: application/json" -d '{"Key":"aaaa", "Value":"bbb"}' https://3nz2cmb6ba3jvdejyraye5aj3q0dbafq.lambda-url.us-west-2.on.aws/ | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    84  100    55  100    29    503    265 --:--:-- --:--:-- --:--:--   770
"Hello from Lambda Functions Urls! you request is POST"
```

以上でこのパートは終了です。

[後片付け]

おつかれさまでした！削除は以下を行ってください。

1. cdk destroy を実行し、作成されたリソースの全削除
VPCを削除するのに時間がかかります。
2. CFnスタック(CDKToolkit と Cloud9）
3. S3バケット

4. CloudWatch Logs