DEV418 – Programming Dynamic CFN Templates

Overview

In this lab you will be creating two AWS CloudFormation templates to create two stacks. One stack will deploy a CloudFormation macro in your account and region, and the second stack will use and get benefit from the deployed macro, which is an AWS Lambda function.

Prerequisites

- A working Internet connection.
- An AWS account with minimum access to create a CloudFormation stack and S3 bucket.
- pip and awscli installed on your machine.
    - If this does not work on your workstation, use Cloud9.
    - https://docs.aws.amazon.com/cloud9/latest/user-guide/create-environment.html#create-environment-main
- The AWS CLI configured with your account information.
- run the following command to download the lab files to your workstation or Cloud9

```
wget http://s3.amazonaws.com/cfnda-datalake/dev418/dev418.zip
```

Lab 1

Overview

In this lab, we will be create a simple macro called Count and a stack that will use it.

The Count macro creates x number of resources. In this case, we will be using it to create a stack with three S3 buckets.

Instructions

1.1 cd into the Lab1 folder in the downloaded DEV418 file.

```
cd DEV418/Lab1
```

1.2 Create a new S3 bucket. Replace <*my-bucket-name*> with a name of your choice.

```
aws s3 mb s3://<my-bucket-name>
```

1.3 Create the packaged template file for your macro. Replace <*my-bucket-name*> with the name you declared above.

MAC/Linux

```
aws cloudformation package --template-file lab1-macro.yaml \
    --s3-bucket <my-bucket-name> --output-template-file lab1-packaged.yaml
```

Windows

```
aws cloudformation package --template-file lab1-macro.yaml ^
    --s3-bucket <my-bucket-name> --output-template-file lab1-packaged.yaml
```

1.4 Create your macro to register it in CloudFormation.

MAC/Linux

```
aws cloudformation deploy --stack-name count-macro \
    --template-file lab1-packaged.yaml --capabilities CAPABILITY_IAM
```

Windows

```
aws cloudformation deploy --stack-name count-macro ^
    --template-file lab1-packaged.yaml --capabilities CAPABILITY_IAM
```

1.5 Call describe stack on the above macro to ensure it has been created.

```
aws cloudformation describe-stacks --stack-name count-macro
```

1.6 Create a stack that will use the macro created above.

MAC/Linux

```
aws cloudformation deploy --stack-name count-test \
    --template-file lab1-example.yaml --capabilities CAPABILITY_IAM
```

Windows

```
aws cloudformation deploy --stack-name count-test ^
    --template-file lab1-example.yaml --capabilities CAPABILITY_IAM
```

1.7 Call list-resources on your created stack to verify three S3 buckets were created.

```
aws cloudformation list-stack-resources --stack-name count-test
```

Lab 2

Overview

In this lab, we will explore a more complex macro. We'll create a new macro, S3Objects.

It adds a new resource type: AWS::S3::Object which you can use to populate a given S3 bucket. A typical use case for this macro might be, as an example, populating an S3-driven website with static assets.

You can either create new S3 objects or copy S3 objects from other buckets that you have permission to access. As with any other resource created by CloudFormation, if you delete a stack containing S3 objects defined with this macro, those objects will also be deleted.

Instructions

2.1 cd to the Lab2 folder.

```
cd ../Lab2
```

2.2 Using the same S3 bucket from above, create the packaged template for the macro. Replace <*my-bucket-name*> with the name you declared above.

MAC/Linux

```
aws cloudformation package --template-file lab2-macro.yaml \
    --s3-bucket <my-bucket-name> --output-template-file lab2-packaged.yaml
```

Windows

```
aws cloudformation package --template-file lab2-macro.yaml ^
    --s3-bucket <my-bucket-name> --output-template-file lab2-packaged.yaml
```

2.3 Create your macro to register it in CloudFormation.

MAC/Linux

```
aws cloudformation deploy --stack-name s3objects-macro \
    --template-file lab2-packaged.yaml --capabilities CAPABILITY_IAM
```

Windows

```
aws cloudformation deploy --stack-name s3objects-macro ^
    --template-file lab2-packaged.yaml --capabilities CAPABILITY_IAM
```

2.4 Call describe stack on the above macro to ensure it has been created.

```
aws cloudformation describe-stacks --stack-name s3objects-macro
```

2.5 Create a stack that will use the macro created above.

MAC/Linux

```
aws cloudformation deploy --stack-name s3objects-test \
    --template-file lab2-example.yaml --capabilities CAPABILITY_IAM
```

Windows

```
aws cloudformation deploy --stack-name s3objects-test ^
    --template-file lab2-example.yaml --capabilities CAPABILITY_IAM
```

2.6 Call describe-stack-resources on your created stack to verify it contains one s3 bucket and 3 resources corresponding to s3 object.

```
aws cloudformation describe-stack-resources --stack-name s3objects-test
```

2.7 Find the bucket-name from the above command and verify it exists.

```
aws s3 ls <bucket-name-from-above-stack>
```

Clean up

C.1 Delete objects in Bucket from Lab 2 in step 2.7. Replace *<lab2-bucket>* with the bucket name.

```
aws s3 rm s3://<lab2-bucket>/ --recursive
```

C.2 Delete stacks created in Lab 1.

```
aws cloudformation delete-stack --stack-name count-test
aws cloudformation delete-stack --stack-name count-macro
```

C.3 Delete stacks created in Lab 2.

```
aws cloudformation delete-stack --stack-name s3objects-test
aws cloudformation delete-stack --stack-name s3objects-macro
```

C.4 Delete S3 bucket. Replace *<my-bucket-name>* with the name you declared above.

```
aws s3 rb s3://<my-bucket-name> --force
```

Lab 3

Overview

In this lab, you will learn how to run validations via command line, as well as validating against a macro. Also, you will learn how to override CloudFormation specifications. Finally, you'll revalidate a macro with a new version of the specifications that will recognize the macro.

The CloudFormation Linter is a tool to validate CloudFormation templates beyond checking for valid YAML or JSON syntax. It will look into the CloudFormation resource specifications and determine if there are any issues. You also have the ability to use the Linter to create custom rules to validate your templates against your company's custom requirements, like internal business rules and regulatory restrictions.

Instructions

3.1 Install CloudFormation Linter

```
pip install cfn-lint
```

OR

```
pip3 install cfn-lint
```

3.1 Verify Linter installation

```
cfn-lint --version
```

3.3 cd to the Lab3 folder

```
cd ../Lab3
```

3.4 Validate a CloudFormation template

```
cfn-lint lab3-basic.yaml
```

3.5 Validate an erroneous CloudFormation template

```
cfn-lint lab3-invalid-basic.yaml
```

It should fail with the following error: could not find expected ':'

3.6 Validate a CloudFormation template containing an unsupported resource-type

```
cfn-lint lab3-s3-object.yaml
```

It should fail with the following error: Invalid or unsupported Type AWS::S3::Object for resource NewObject in us-east-1

3.7 Validate the same template again by using a customized resource-specification for AWS::S3::Object

MAC/Linux

```
cfn-lint --template lab3-s3-object.yaml \
    --override-spec lab3-s3-object-spec.json
```

Windows

```
cfn-lint --template lab3-s3-object.yaml ^
    --override-spec lab3-s3-object-spec.json
```

The command above should not fail, it should exit with an empty output.

YOU ARE DONE WITH THE PLANNED LABS FOR THIS WORKSHOP – CONGRATULATIONS!!

Alternative Instructions Using Console

Overview

There are the alternative instructions using the CloudFormation Console in case the AWS CLI failed to install.

You will still need all the source files from above and need to run same (or equivalent) commands via console. The main difference when using Console that equivalents of ***'aws cloudformation package'*** and '***aws cloudformation deploy'*** commands are not available . Here is the workaround for that -

Lab 1

1. 1 Instead of running 'aws cloudformation package', do the following -

1.2 Go to the folder on where you extracted the contents of the source zip folder.

1.3 Use the 'index.py' file and create a zip using the following command -

```
zip -r index.zip index.py
```

1.4 Upload the zip file to your S3 bucket.

1.7 Open the 'lab1-packaged-console.yaml' file present in your folder. Change the 'CodeUri' properties of 'MacroFunction' and 'ResourceFunction' resources to point to the respective .zip files you added in previous step.

1.8 Instead of running 'aws cloudformation deploy', instead run create-change-set followed by execute-change-set using Console.

Lab 2

2. 1 Instead of running 'aws cloudformation package', do the following -

2.2 Go to the folder on where you extracted the contents of the source zip folder.

2.3 Use the 'macro.py' file and create a zip using the following command -

```
zip -r macro.zip macro.py
```

2.4 Similarly, use the 'resource.py' file and create a zip using the following command -

```
zip -r resource.zip resource.py
```

2.5 Upload these two zip files to your S3 bucket.

2.6 Open the 'lab2-packaged-console.yaml' file present in your folder. Change the 'CodeUri' properties of 'MacroFunction' and 'ResourceFunction' resources to point to the respective .zip files you added in previous step.

2.7 Instead of running 'aws cloudformation deploy', instead run create-change-set followed by execute-change-set using Console.