# Nested Stack Deployment Using CloudFormation Template

Prepared by:

Nurunnubi Talukder, Solutions Architect

David Fei, Senior DevOps Engineer

Profile Setting

vi ~/.aws/credentials

[dev-poweruser]

aws\_access\_key\_id = AKIA4B63WKRT

aws\_secret\_access\_key = maF1JZlWzxhsL/

region=ca-central-1

To use a named profile for multiple commands, you can avoid specifying the profile in every command by setting the AWS\_PROFILE environment variable at the command line.

export AWS\_PROFILE=dev-poweruser [Linux/Mac]

C:\> setx AWS\_PROFILE dev-poweruser [Windows]

To get details about the current IAM identity

aws sts get-caller-identity

To list all your profile names

aws configure list-profiles

<https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-profiles.html>

Prepare the CloudFormation Stack for DEV, UAT and PROD

1. Create three parameter files DEV, UAT, PROD

Graphical user interface, text, application

Description automatically generated

Create “cfn\_templates” and “lambda\_source” in S3 bucket. Upload all the nested stack yaml files to “cfn\_templates” folder and upload lambda-function.zip to “lambda\_source” folder. In local foder, only keep main.yaml and main\*\_parameters.json files. This process can also be implemented by running scripts to invoke AWS CLI to create S3 bucket, create folders and upload files to S3 bucket. Also the parameter files can be modified by script to replace the S3 bucket name to the S3 bucket name that are created by AWS CLI.

~~Create folder for cloudformation template and landa\_source. Upload the cloudformation template and lamda-function.zip~~

Graphical user interface, text, application, email

Description automatically generated

Upload the cloudformation template except main.yaml and paramters file

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

~~Change bucket name and folder two places in parameter file~~

Open main\_<env>\_parameters.json. There are two references for the S3 buck name in each parameter file. Correct them. Failed to correct the bucket name will cause the failure of creating nested stacks.

Text

Description automatically generated

Text

Description automatically generated

**Lambda runtimes**

Change on the lamda.yaml template

Text

Description automatically generated

Change on the main.yaml template

Text

Description automatically generated

Make sure you include all the possible lambda runtimes that supported by AWS.

Based on your lambda code runtime, please update parameterValue in Parameters file

Text

Description automatically generated

<https://docs.aws.amazon.com/lambda/latest/dg/lambda-runtimes.html>

Run the command based on visual studio code

aws cloudformation create-stack --stack-name duobank-iam-stack11ccc --template-body file://main.yaml --parameters file://main\_dev\_parameters.json --profile dev-poweruser --region ca-central-1 --capabilities CAPABILITY\_NAMED\_IAM

If you has set the AWS\_PROFILE environment variable and set the region in the profile, the --profile and –-region parameters can be omitted.

**~~Note: Poweruser does not have capability to create role. Please create this managed policy and attached with the power user.~~**

**Note: Poweruser doesn’t have capacity of creating IAM user/role/policy, access key and S3 replication. Make sure to create/modify the following managed policy to attach to Poweruser IAM user.**

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "VisualEditor0",

"Effect": "Allow",

"Action": [

"iam:AddRoleToInstanceProfile",

"iam:CreatePolicy",

"iam:CreatePolicyVersion",

"iam:GenerateCredentialReport",

"iam:CreateInstanceProfile",

"iam:Get\*",

"iam:List\*",

"iam:GetRole",

"iam:PassRole",

"iam:DeleteRolePolicy",

"iam:GenerateServiceLastAccessedDetails",

"iam:PutUserPolicy",

"iam:CreateRole",

"iam:DeleteRole",

"iam:AttachRolePolicy",

"iam:PutRolePolicy",

"iam:CreateAccessKey",

"iam:UpdateAccessKey",

"iam:ListAccessKey",

"iam:DeleteAccessKey",

"iam:CreateUser",

"iam:ListUsers",

"iam:DeleteUsers",

"s3:PutReplicationConfiguration"

],

"Resource": "\*"

}

]

}

After successful deployment, we can see the below picture from CloudFormation template

Graphical user interface, text, application, email

Description automatically generated