

Workshop: Deploying Lambda Functions using AWS Code Build and Code Pipeline CI/CD Tools

This workshop is intended to take you on a short tour of using AWS CodeBuild and CodePipeline to automate your code deployment. The workshop will guide you through the process of taking your code from a GitHub repository all the way through to a fully deployed Lambda function. Note that this fully automated pipeline will use no servers and will have no infrastructure to maintain. The pipeline will use CloudFormation templates to define and manage the process. In this workshop you will explore the sections of the CloudFormation template, identify your buildspec.yaml file, and deploy your pipeline automation.

Note: This workshop will build on the Lambda 101 Workshop Artifacts. If you do not have these Artifacts, please redo The Lambda Functions 101 workshop or run this CloudFormation Template: [<Link Here>](#).

This workshop is divided in to 3 sections as follows:

Section 1: Login to github.com and create a personal access token. Also, fork the source repo

Section 2: Create CloudFormation Stack

Section 3: Test deployed Lambda Function

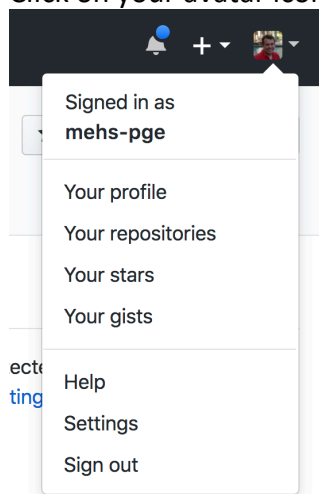
Section 4: Update index.js in github.com

Section 5: Confirm updated Lambda Function

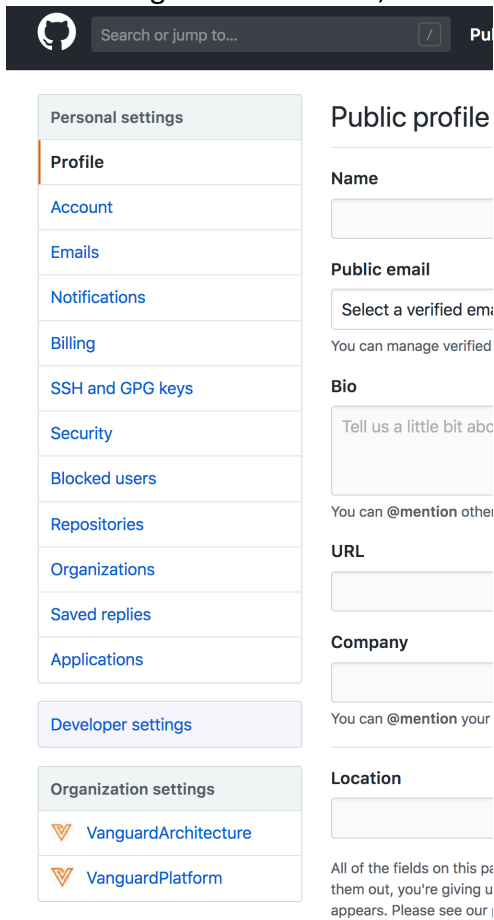
Important: Please note that this workshop should be done in your personal account using us-east-1 (N. Virginia) region. In addition, you must have an account on github.com.

Section 1: Login to github.com and create a personal access token. Also, fork the source repo

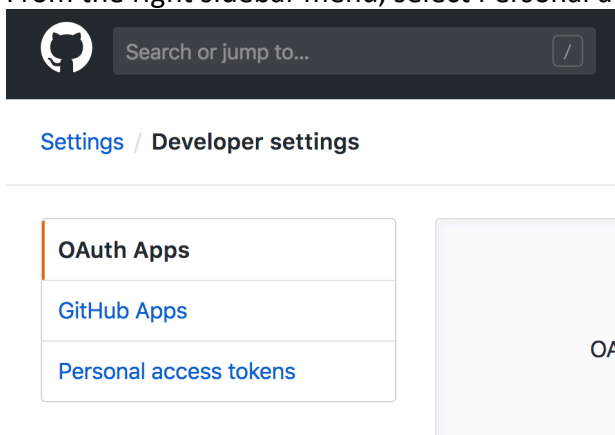
1. Click on your avatar icon and select Settings:



2. From the right sidebar menu, select developer settings:



3. From the right sidebar menu, select Personal access tokens:



4. Select the Generate new token button:



5. Give the token a description such as 'WorkshopDemo'

Token description

WorkshopDemo

What's this token for?

6. For this example, let's just use full repo access as follows:

<input checked="" type="checkbox"/>	repo	Full control of private repositories
<input checked="" type="checkbox"/>	repo:status	Access commit status
<input checked="" type="checkbox"/>	repo_deployment	Access deployment status
<input checked="" type="checkbox"/>	public_repo	Access public repositories
<input checked="" type="checkbox"/>	repo:invite	Access repository invitations

7. Click Generate Token

Generate token

8. On the next screen, Make sure to copy the token for use in the next session as this is the ONLY place you will ever see this token:



Personal access tokens

Generate new token

Revoke all

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your new personal access token now. You won't be able to see it again!

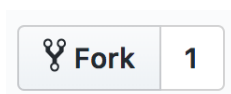
✓  

Delete

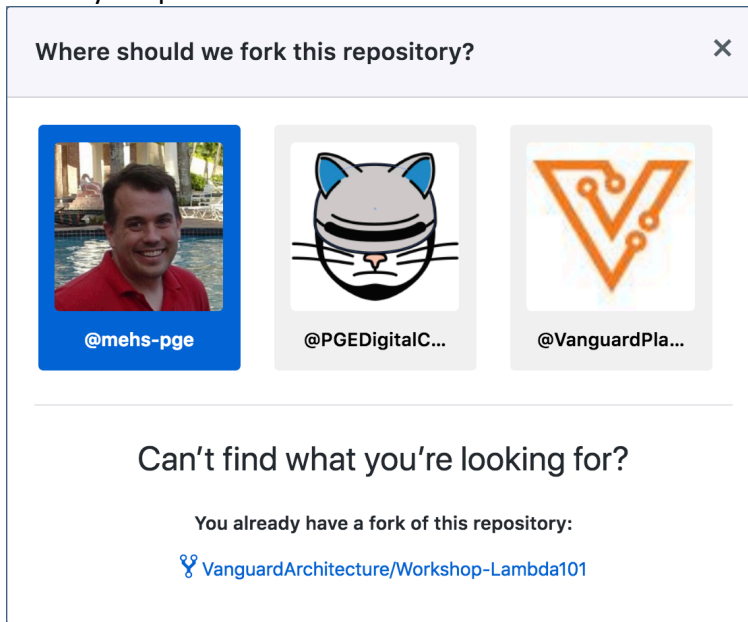
9. Navigate to the following public repo:

<https://github.com/snowake4me/Workshop-Lambda101>

10. Click the Fork button to Fork a copy of the Repo to your account



11. Select your personal account:



At this point, you should be redirected to your new repo and you should have a personal access token.

Section 2: Create CloudFormation Stack

1. Navigate to <https://github.com/VanguardArchitecture/Workshop-LambdaCICD>

2. Click the Launch Stack Button:

Getting Started



3. The next screen should be pre-populated so you can just click Next: Select Template

Select the template that describes the stack that you want to create. A stack is a group of related resources that you manage as a single unit.

Design a template Use AWS CloudFormation Designer to create or modify an existing template. [Learn more.](#)
Design template

Choose a template A template is a JSON/YAML-formatted text file that describes your stack's resources and their properties. [Learn more.](#)

☐ Select a sample template

☐ Upload a template to Amazon S3

Choose File No file chosen

☒ Specify an Amazon S3 template URL

[View/Edit template in Designer](#)

Cancel

Next

4. Remember that you have a forked repo that looks something like this:

<https://github.com/mehs-pge/Workshop-Lambda101>

You will need to use this to fill out the next screen as follows:

Specify Details

Specify a stack name and parameter values. You can use or change the default parameter values, which are defined in the AWS CloudFormation template. [Learn more.](#)

Stack name

Parameters

pApplicationName The name of the application

pCFNOwnerTag Enter CorpID of the owner of this stack (for tagging of resources)

pEnv

PROD

 The environment

pGitHubAccount Enter the GitHub Account Name

pGitHubRepo Enter the GitHub Repo Name

pGitHubRepoBranch Enter the Branch (dev, master, feature, etc) you wish to pull from the GitHub repo

pGitHubToken Enter the GitHub Access Token

pPipelineBucketName The name of the S3 bucket that will contain the pipeline artifacts

Cancel

Previous

Next

- Click Next



- The following screen, just accept the defaults and click Next again:



- At the bottom of the next screen, check the acknowledgement box and click Create:

Capabilities

i The following resource(s) require capabilities: [AWS::IAM::Role]

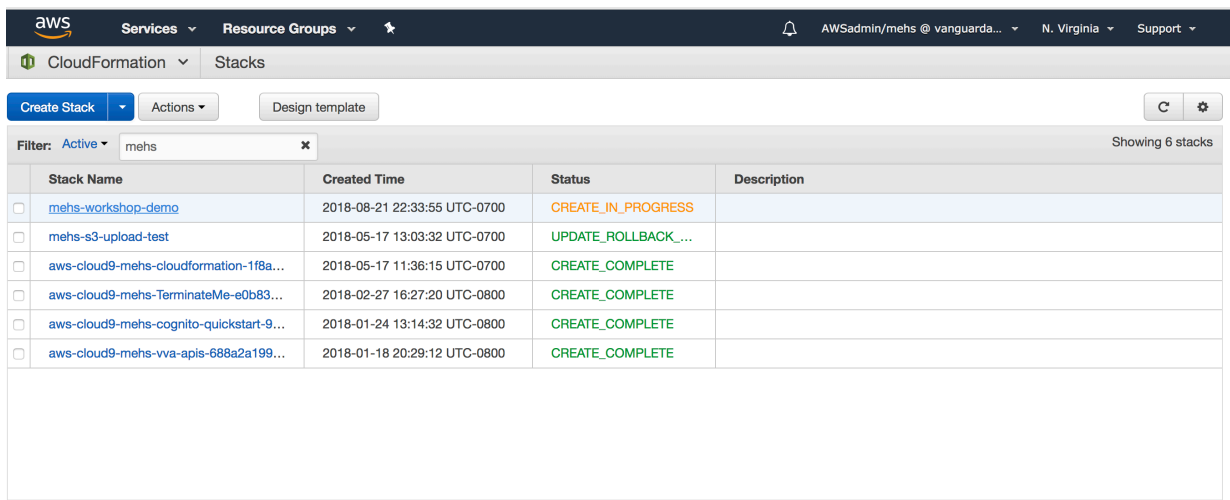
This template contains Identity and Access Management (IAM) resources. Check that you want to create each of these resources and that they have the minimum required permissions. In addition, they have custom names. Check that the custom names are unique within your AWS account. [Learn more.](#)

☒ I acknowledge that AWS CloudFormation might create IAM resources with custom names.

[Quick Create Stack](#) (Create stacks similar to this one, with most details auto-populated)

[Cancel](#) [Previous](#) [Create](#)

- On the following screen, type in your Land ID and do a filter search for your Stack. Click on your Stack once found:

The screenshot shows the AWS CloudFormation console. At the top, there's a navigation bar with the AWS logo, "Services", "Resource Groups", and a star icon. Below that, a breadcrumb shows "CloudFormation" > "Stacks". A toolbar contains "Create Stack", "Actions", and "Design template". A filter bar shows "Filter: Active" and a search box with "mehs". The table lists six stacks. The first stack, "mehs-workshop-demo", is highlighted. The table has columns for Stack Name, Created Time, Status, and Description. The status for the first stack is "CREATE_IN_PROGRESS", while the others are "CREATE_COMPLETE".

	Stack Name	Created Time	Status	Description
<input type="checkbox"/>	mehs-workshop-demo	2018-08-21 22:33:55 UTC-0700	CREATE_IN_PROGRESS	
<input type="checkbox"/>	mehs-s3-upload-test	2018-05-17 13:03:32 UTC-0700	UPDATE_ROLLBACK_...	
<input type="checkbox"/>	aws-cloud9-mehs-cloudformation-1f8a...	2018-05-17 11:36:15 UTC-0700	CREATE_COMPLETE	
<input type="checkbox"/>	aws-cloud9-mehs-TerminateMe-e0b83...	2018-02-27 16:27:20 UTC-0800	CREATE_COMPLETE	
<input type="checkbox"/>	aws-cloud9-mehs-cognito-quickstart-9...	2018-01-24 13:14:32 UTC-0800	CREATE_COMPLETE	
<input type="checkbox"/>	aws-cloud9-mehs-vva-apis-688a2a199...	2018-01-18 20:29:12 UTC-0800	CREATE_COMPLETE	

9. Notice that clicking on your stack allows you to see the stack creation events. Monitor this window for errors.

Overview	Outputs	Resources	Events	Template	Parameters	Tags	Stack Policy	Change Sets	Rollback Triggers
Filter by: Status ▾ Search events									
2018-08-21									
		Status	Type	Logical ID	Status Reason				
▶	22:34:45 UTC-0700	CREATE_IN_PROGRESS	AWS::IAM::Role	rPipelineRole	Resource creation Initiated				
▶	22:34:44 UTC-0700	CREATE_IN_PROGRESS	AWS::IAM::Role	rPipelineRole					
▶	22:34:41 UTC-0700	CREATE_COMPLETE	AWS::CodeBuild::Project	rBuildProject	Resource creation Initiated				
▶	22:34:40 UTC-0700	CREATE_IN_PROGRESS	AWS::CodeBuild::Project	rBuildProject					
▶	22:34:37 UTC-0700	CREATE_IN_PROGRESS	AWS::CodeBuild::Project	rBuildProject	Resource creation Initiated				
▶	22:34:34 UTC-0700	CREATE_COMPLETE	AWS::IAM::Role	rBuildProjectRole					
▶	22:34:23 UTC-0700	CREATE_IN_PROGRESS	AWS::IAM::Role	rBuildProjectRole	Resource creation Initiated				
▶	22:34:23 UTC-0700	CREATE_IN_PROGRESS	AWS::IAM::Role	rBuildProjectRole					
▶	22:34:21 UTC-0700	CREATE_COMPLETE	AWS::S3::Bucket	rPipelineBucket	Resource creation Initiated				
▶	22:34:14 UTC-0700	CREATE_COMPLETE	AWS::Lambda::Function	rLambdaFunction					
▶	22:34:14 UTC-0700	CREATE_IN_PROGRESS	AWS::Lambda::Function	rLambdaFunction					
▶	22:34:13 UTC-0700	CREATE_IN_PROGRESS	AWS::Lambda::Function	rLambdaFunction					

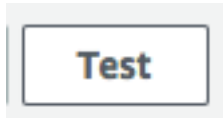
10. Once the creation is complete, click on the Resources Tab to see a list of what was created by the stack.

Overview	Outputs	Resources	Events	Template	Parameters	Tags	Stack Policy	Change Sets	Rollback Triggers
Logical ID	Physical ID	Type	Status						
rBuildProject	mehs-workshop-demo-BUILDPROJECT-PROD	AWS::CodeBuild::Project	CREATE_COMPLETE						
rBuildProjectRole	mehs-workshop-demo-BUILDPROJECT-ROLE-PROD	AWS::IAM::Role	CREATE_COMPLETE						
rLambdaFunction	mehs-workshop-demo-LAMBDA-PROD	AWS::Lambda::Function	CREATE_COMPLETE						
rLambdaFunctionRole	mehs-workshop-demo-LAMBDA-ROLE-PROD	AWS::IAM::Role	CREATE_COMPLETE						
rPipeline	mehs-workshop-demo-PIPELINE-PROD	AWS::CodePipeline::Pipeline	CREATE_COMPLETE						
rPipelineBucket	mehs-workshop-demo-artifacts	AWS::S3::Bucket	CREATE_COMPLETE						
rPipelineRole	mehs-workshop-demo-PIPELINE-ROLE-PROD	AWS::IAM::Role	CREATE_COMPLETE						

11. Click your Lambda function so we can begin the next section:

Section 3: Test your generated Lambda function

1. Click the Test Button in the upper right corner



2. Give your event a name and change value1 of key1 to something unique to you:

Configure test event ✕

A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events.

☒ Create new test event
☐ Edit saved test events

Event template
Hello World ▼

Event name
MyTestEvent

```
1 {  
2   "key3": "value3",  
3   "key2": "value2",  
4   "key1": "Michael's first Lambda"  
5 }
```

3. Click the Create Button:

Create

4. Click the test button again:

Test

5. In the Execution result section, expand the Details by click on it:

✓ Execution result: succeeded (logs)
▶ Details

6. Your custom message should appear along with details about the execution of the function:

✓ Execution result: succeeded ([logs](#))

▼ Details

The area below shows the result returned by your function execution. [Learn more](#) about returning results from your function.

```
{
  "Items": [
    {
      "mobile": "805-305-8438",
      "firstName": "Michael",
      "lastName": "Hansen",
      "lanID": "mehs"
    }
  ],
  "Count": 1,
  "ScannedCount": 1
}
```

Summary

Code SHA-256	Request ID
XSLJBKhdnyff1HuYbYwyENpeo65Da4tEIScvlHm/vl=	ba356ea7-a5ce-11e8-a31e-a38969be74c1
Duration	Billed duration
1072.19 ms	1100 ms
Resources configured	Max memory used
128 MB	34 MB

Section 4: Update index.js in github.com

1. Go to your forked repo. The URL should look something like this:

<https://github.com/VanguardArchitecture/Workshop-Lambda101>

2. Click on index.js.
3. Click the edit icon:



4. On line 8, change 'mehs' to 'jva2'

5. Click 'Commit Changes'

Commit changes

6. Optional: You could now navigate to CodePipeline, search for your pipeline, and watch the progress:

The screenshot shows the AWS CodePipeline console interface. At the top, there's a header with the AWS CodePipeline logo and the pipeline name 'mehs-workshop-demo-PIPELINE-PROD'. Below the header, there's a sub-header 'View progress and manage your pipeline.' and two buttons: 'Edit' and 'Release change'. The main content area displays the pipeline stages. The first stage is 'Source', which contains a 'SourceAction' using 'GitHub' as the provider. This action is marked as 'Succeeded' with a green checkmark and the commit ID 'bf5d6c0'. Below this, there's a 'SourceAction: Update ind...' entry. An arrow points from the 'Source' stage to the 'BuildDeploy' stage. The 'BuildDeploy' stage contains a 'CodeBuild' action using 'AWS CodeBuild' as the provider. This action is marked as 'In Progress' with a blue circular arrow icon. Below this, there's another 'SourceAction: Update ind...' entry.

Section 5: Verify that your changes were applied

Make sure that the CodeBuild from the prior section has completed before starting this section

1. Navigate to the AWS Lambda service in the console and find your Lambda function
2. Click Test

3. You should now see a different result as follows:

mehs-workshop-demo-LAMBDA-PROD Throttle Qualifiers ▼ Actions ▼ fefdf

This function belongs to the CloudFormation stack **mehs-workshop-demo**. Visit the [CloudFormation console](#) to manage this stack.

Execution result: succeeded (logs)

▼ **Details**

The area below shows the result returned by your function execution. [Learn more](#) about returning results from your function.

```
{
  "Items": [
    {
      "mobile": "805-555-1212",
      "firstName": "John",
      "lastName": "Adkins",
      "lanID": "java2"
    }
  ],
  "Count": 1,
  "ScannedCount": 1
}
```

Summary

Code SHA-256	Request ID
+PV1vamAYddFTucnEbG08ji8dC2s0b8y0Dbi9MCFBOA=	64c0d65d-a5d0-11e8-a52a-198785e16abc
Duration	Billed duration
319.17 ms	400 ms
Resources configured	Max memory used
128 MB	37 MB