

# Table of Contents

## Preface

## Introduction

- Introduction to AWS developer tools
- What is Continuous Integration?
- What is Continuous Delivery?
- What is DevOps?
- What do you mean by Blue/Green deployment?

## AWS Code Commit

- Introduction to AWS CodeCommit
- IAM for CodeCommit
- HTTPS credentials for Git repository
- Creating and Deleting repositories
- Clone repositories
- Pushing and Pulling Code
- CodeCommit to SNS Part-1
- CodeCommit to SNS Part-2
- CodeCommit to AWS Lambda
- Working with branches
- Working with pull requests
- Pull Request notifications
- Migrating to CodeCommit
- AWS KMS and Encryption for CodeCommit Repositories

## AWS CodeBuild

- What is CodeBuild?
- Introduction
- Code Build Demo
- CodeBuild Phases
- CodeBuild buildspec.yml
- Monitoring CodeBuild - Logs
- Monitoring CodeBuild - Metrics and Alarms
- Triggering CodeBuild

# Table of Contents

## AWS CodeDeploy

- What is AWS CodeDeploy?
- Introduction
- Create Instance - Demo Part 1
- Create Instance IAM Role - Demo Part 2
- Create Service Role - Demo Part 3
- Create Application and Deployment Group - Demo Part 4
- Upload code archive to S3 - Demo Part 5
- Deploy the application - Demo Part 6
- View the application - Demo Part 7
- Review appsec.yml - Demo Part 8
- CodeDeploy Life Cycle Events
- CodeDeploy Demo using AutoScaling Group
- CodeDeploy Events Trigger - SNS
- AWS CodePipeline

## AWS CodePipeline

- Introduction
- What is CodePipeline?
- CodePipeline Demo
- Creating a Multi Stage Pipeline Part 1
- Creating a Multi Stage Pipeline Part 2
- Creating a Multi Stage Pipeline Part 3
- Creating a Multi Stage Pipeline Part 4
- Adding Approval Steps to the Pipeline

## Summary

## Review

Provide the required information below and review this role before you create it.

**Role name\***

Use alphanumeric and '+=,@-\_' characters. Maximum 64 characters.

**Role description**

Maximum 1000 characters. Use alphanumeric and '+=,@-\_' characters.

**Trusted entities** AWS service: lambda.amazonaws.com

**Policies**  [AWSCodeCommitReadOnly](#)   
 [AWSLambdaBasicExecutionRole](#) 

**Permissions boundary** Permissions boundary is not set

\* Required

[Cancel](#)

[Previous](#)

[Create role](#)

8) Once you have reviewed everything you can click on the **Create role** button.

You will now be able to see that your **LambdaCodeCommitReadOnly** role is now created. In order to create a Lambda function as trigger you can either use the CodeCommit console or create Lambda function from the Lambda console.

For this exercise, we are going to use the Lambda console to create a Lambda function for our trigger.

- 1) Go to the Lambda console, by selecting Lambda from the services menu.
- 2) Click on **Create Function** button
- 3) For the Lambda function name give it the name *CodeCommitTrigger* and chose the runtime as Node.js 12.x
- 4) For the Lambda permissions click on **Use an existing role** and then select the Lambda function *LambdaCodeCommitReadOnly* we created in the previous section.

## Function name

Enter a name that describes the purpose of your function.

CodeCommitTrigger

Use only letters, numbers, hyphens, or underscores with no spaces.

## Runtime [Info](#)

Choose the language to use to write your function.

Node.js 12.x

## Permissions [Info](#)

Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can configure and modify permissions further when you add triggers.

### ▼ Choose or create an execution role

#### Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

- Create a new role with basic Lambda permissions
- Use an existing role
- Create a new role from AWS policy templates

#### Existing role

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

LambdaCodeCommitReadOnly

[View the LambdaCodeCommitReadOnly role](#) on the IAM console.

5) Click on the **Create Function** button.

6) Once the function is created in the designer section you will be able to see an option to Add trigger . We are going to use CodeCommit as a trigger and then configure the required parameters.

The screenshot shows the AWS Lambda console configuration page for a function named "CodeCommitTrigger". At the top, there are buttons for "Throttle", "Qualifiers", "Actions", "Select a test event", "Test", and "Save". Below these are tabs for "Configuration", "Permissions", and "Monitoring". The "Designer" section is expanded, showing a box for the function "CodeCommitTrigger" with a "Layers" section below it showing "(0)". There are buttons for "+ Add trigger" and "+ Add destination".

## Trigger configuration



CodeCommit

aws developer-tools git



### Repository name

Select the repository to add a trigger to.

ToDoApplication



### Trigger name

Provide a name for the trigger that will invoke this function.

LambdaTrigger

### Events

Choose one or more events to listen for. If you choose "All repository events", you cannot choose other event types.



All repository events

### Branch names

This trigger will be configured for all repository branches and tags by default. For a more specific configuration, choose up to 10 branches. If you choose "All branches", you cannot choose specific branches.



All branches

### Custom data - *optional*

Custom data is additional contextual information used to distinguish this trigger from other triggers that run for the same event, refer to external resources, or group triggers from different repositories. For example, you could include the channel ID # for a chat room used by your team to collaborate on development.

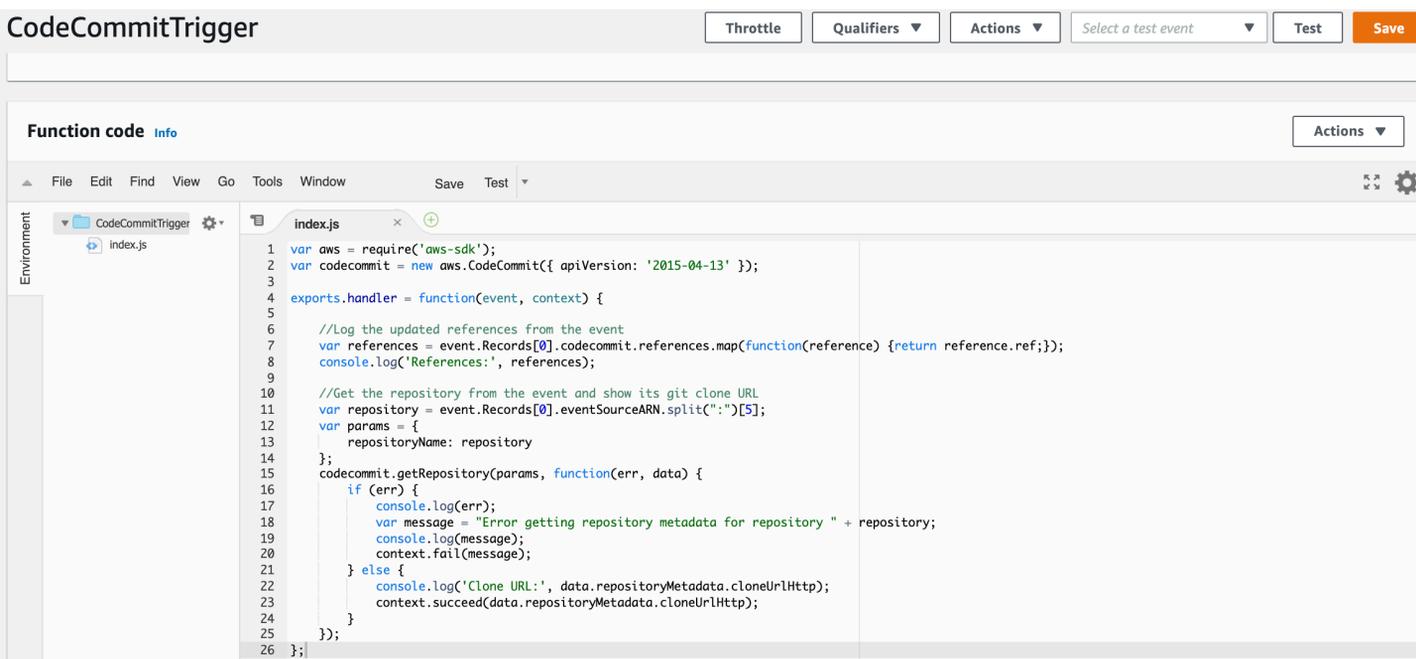
*For example, an IRC channel ID #*

- 5) For the CodeCommit trigger configuration, you can select the Repository name as the *ToDoApplication* and enter the Trigger name as *LambdaTrigger*
- 6) You have option to select specific repository events such as Create branch, push or delete branch or or select **All repository events**.
- 7) For the branch name you can select any specific branch, however for the purpose of this exercise we are going to use **All branches**.
- 8) Click on the **Add** button.

This add a Lambda function as a trigger for CodeCommit. However we still need to write a function code for Lambda to execute something.

You can download the sample Node.js code from the link [https://github.com/kloudgems/awscodcommit/blob/master/lambda\\_trigger.js](https://github.com/kloudgems/awscodcommit/blob/master/lambda_trigger.js) which we are going to use for our lambda function

Go to the **Function code** section on the Lambda function and then remove the existing default code from it and replace it with the code downloaded from the above link



```
1 var aws = require('aws-sdk');
2 var codecommit = new aws.CodeCommit({ apiVersion: '2015-04-13' });
3
4 exports.handler = function(event, context) {
5
6     //Log the updated references from the event
7     var references = event.Records[0].codecommit.references.map(function(reference) {return reference.ref;});
8     console.log('References:', references);
9
10    //Get the repository from the event and show its git clone URL
11    var repository = event.Records[0].eventSourceARN.split(":")[5];
12    var params = {
13        repositoryName: repository
14    };
15    codecommit.getRepository(params, function(err, data) {
16        if (err) {
17            console.log(err);
18            var message = "Error getting repository metadata for repository " + repository;
19            console.log(message);
20            context.fail(message);
21        } else {
22            console.log('Clone URL:', data.repositoryMetadata.cloneUrlHttp);
23            context.succeed(data.repositoryMetadata.cloneUrlHttp);
24        }
25    });
26 };
```

Using this function, every time there is a CodeCommit event which triggers the Lambda function, we are going to log that event through this function. While this is just an example, there is no limit what you could do with the Lambda function, such as invoke Jenkins built or sending slack/email notification etc.

Once you have added the code, click on the save button

In order to test if the function works fine, we need to first configure a test event. A test event is like an input to the Lambda function which simulates a real trigger. You can click on the drop down menu before the **Test** and select **Configure test events**.

In the Configure test events pop up form, click on the drop down menu for **Event template**, select the **codecommit-repository** template from the list. This will show a template with the sample event which would be generated to trigger the lambda function from CodeCommit. In order to customize it further to, we are going to change the repository arn from

*arn:aws:codecommit:us-east-1:123456789012:my-repo* to *arn:aws:codecommit:us-east-1:123456789012:ToDoApplication*

Enter a name for the test event and click on Create button

### 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  
codecommit-repository ▾

Event name  
TestCodeCommit

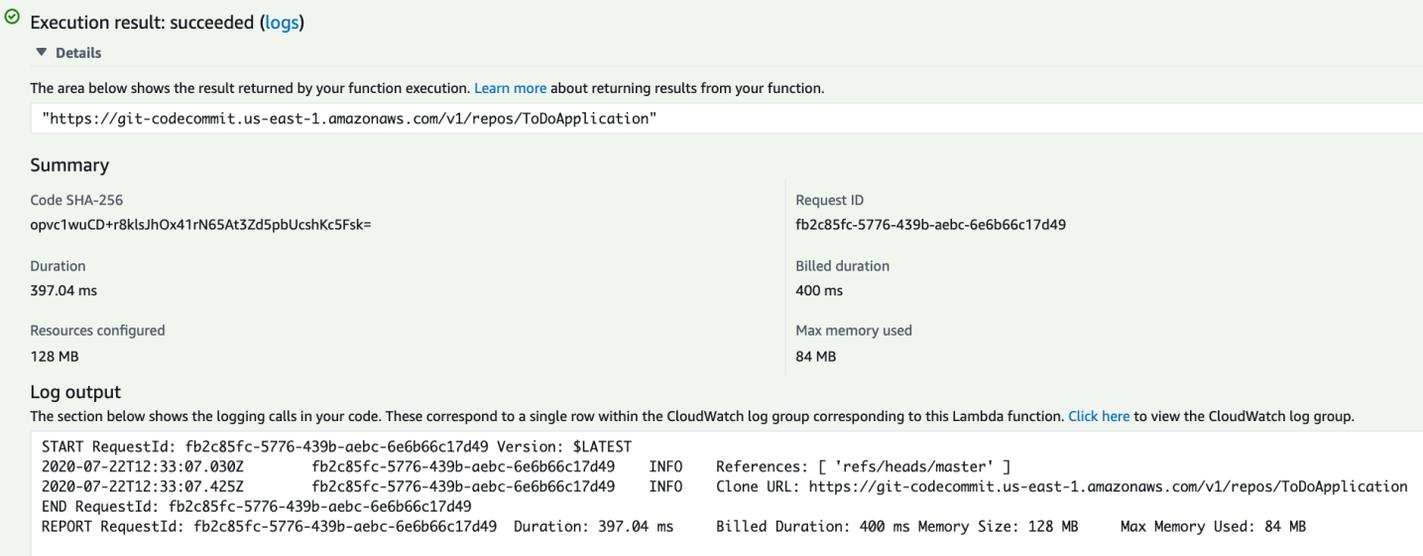
```
1 {
2   "Records": [
3     {
4       "awsRegion": "us-east-1",
5       "codecommit": {
6         "references": [
7           {
8             "commit": "5c4ef1049f1d27deadbeeff313e0730018be182b",
9             "ref": "refs/heads/master"
10          }
11        ]
12      },
13      "customData": "this is custom data",
14      "eventId": "5a824061-17ca-46a9-bbf9-114edeadbeef",
15      "eventName": "TriggerEventTest",
16      "eventPartNumber": 1,
17      "eventSource": "aws:codecommit",
18      "eventSourceARN": "arn:aws:codecommit:us-east-1:123456789012:ToDoApplication",
19      "eventTime": "2016-01-01T23:59:59.000+0000",
20      "eventTotalParts": 1,
21      "eventTriggerConfigId": "5a824061-17ca-46a9-bbf9-114edeadbeef",
22      "eventTriggerName": "my-trigger",
23      "eventVersion": "1.0",
24      "userIdentityARN": "arn:aws:iam::123456789012:root"
25    }
26  ]
27 }
```

41 Cancel Format JSON Create

Now lets test our function with the test event we just created by selecting it from the drop down menu and clicking on the **Test** button.



You should see “Execution result: succeeded” and by clicking on the details button you will see the logs of the function indicating the output of the Lambda function execution.



Execution result: succeeded (logs)

▼ Details

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

```
"https://git-codecommit.us-east-1.amazonaws.com/v1/repos/ToDoApplication"
```

### Summary

Code SHA-256	Request ID
opvc1wuCD+r8klsJhOx41rN65At3Zd5pbUcshKc5Fsk=	fb2c85fc-5776-439b-aebc-6e6b66c17d49
Duration	Billed duration
397.04 ms	400 ms
Resources configured	Max memory used
128 MB	84 MB

### Log output

The section below shows the logging calls in your code. These correspond to a single row within the CloudWatch log group corresponding to this Lambda function. [Click here](#) to view the CloudWatch log group.

```
START RequestId: fb2c85fc-5776-439b-aebc-6e6b66c17d49 Version: $LATEST
2020-07-22T12:33:07.030Z      fb2c85fc-5776-439b-aebc-6e6b66c17d49      INFO      References: [ 'refs/heads/master' ]
2020-07-22T12:33:07.425Z      fb2c85fc-5776-439b-aebc-6e6b66c17d49      INFO      Clone URL: https://git-codecommit.us-east-1.amazonaws.com/v1/repos/ToDoApplication
END RequestId: fb2c85fc-5776-439b-aebc-6e6b66c17d49
REPORT RequestId: fb2c85fc-5776-439b-aebc-6e6b66c17d49  Duration: 397.04 ms   Billed Duration: 400 ms Memory Size: 128 MB   Max Memory Used: 84 MB
```

You can also see additional Lambda execution logs in CloudWatch logs. In order to see the log group you can either go to the **Monitoring** tab in the function configuration and then click View logs in CloudWatch logs or by going to the CloudWatch console directly and clicking on the `/aws/lambda/CodeCommitTrigger` Log group in the logs section.