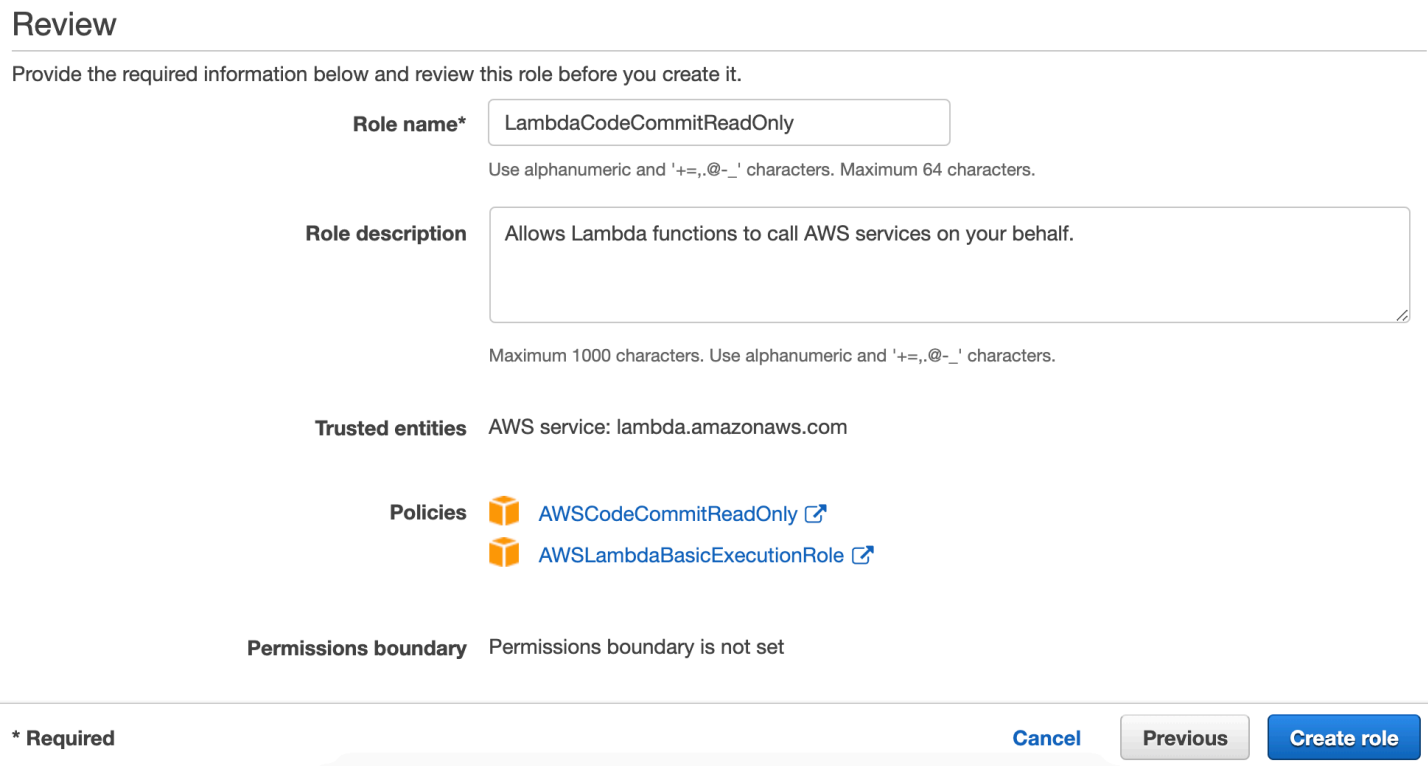
Table of Contents

Table of Contents

AWS CodeDeploy

AWS CodePipeline

Summary



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**.

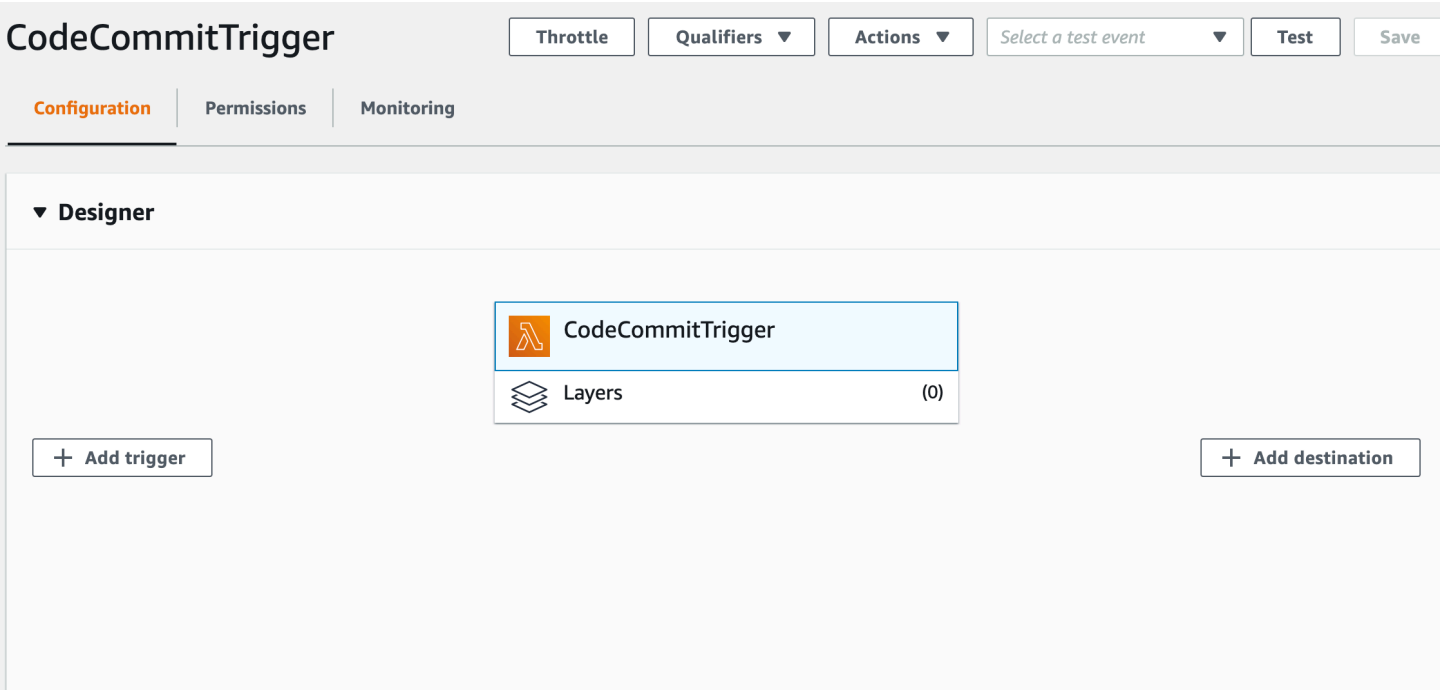
- ○ Create a new role with basic Lambda permissions
- ◉ Use an existing role
- ○ Create a new role from AWS policy templates

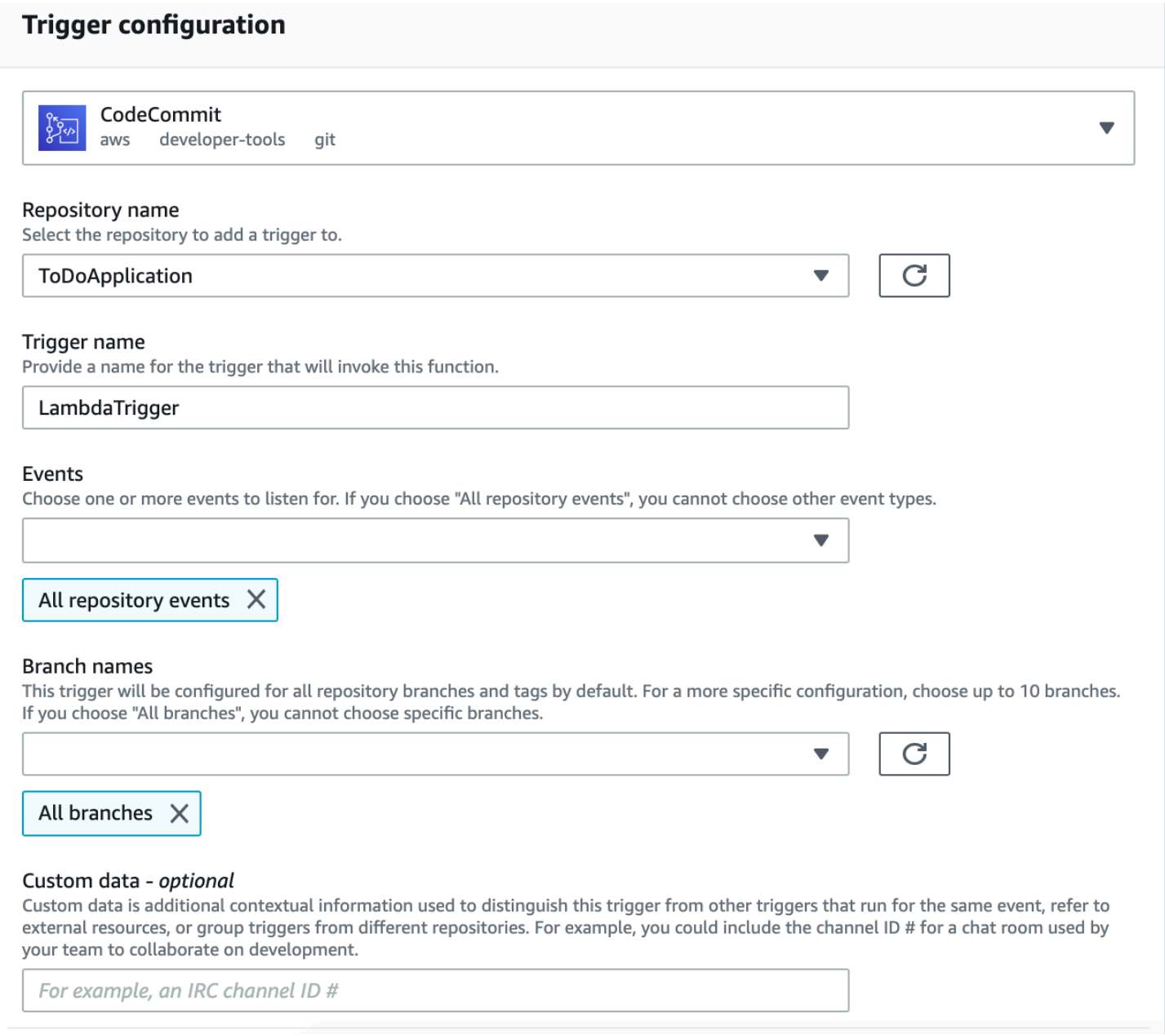
Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

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.



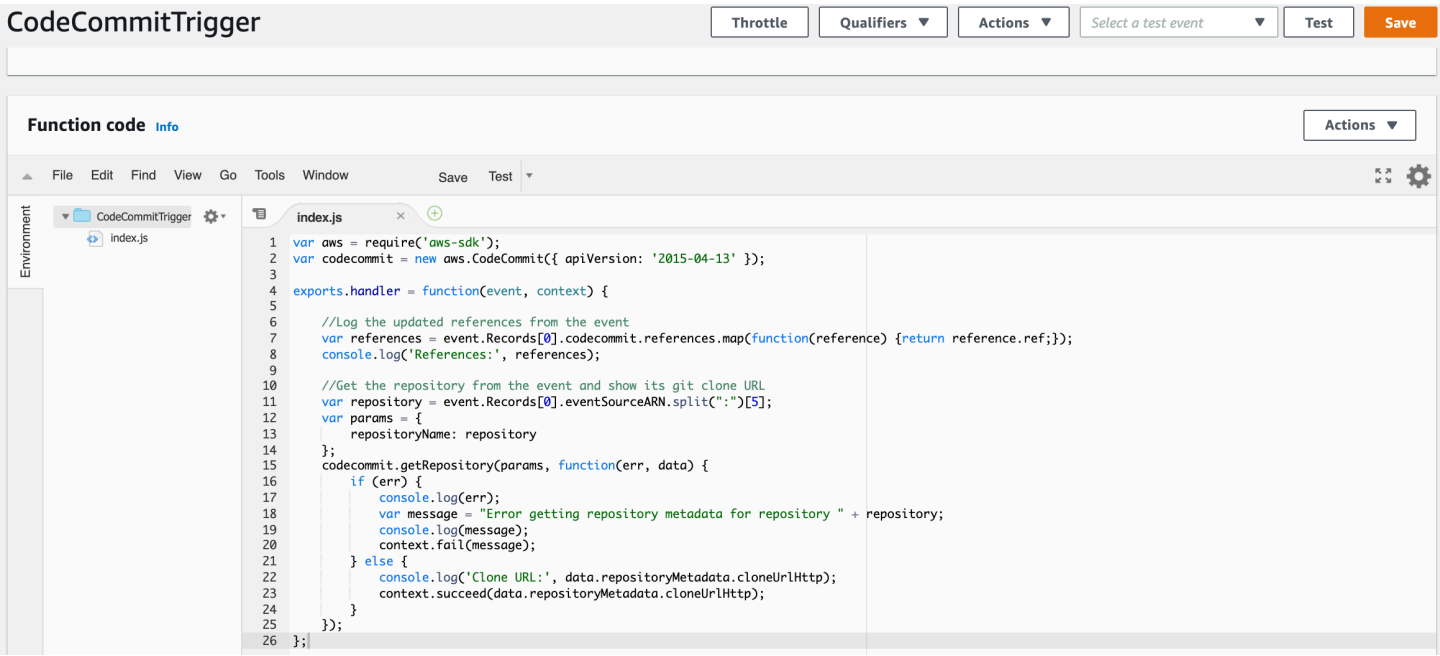


5) For the CodeCommit trigger configuration, you can select the Repository name as the *ToDoApplication* and enter the Trigger name as *LambdaTrigger*
6) You have option to select specific repository events such as Create branch, push or delete branch or or select **All repository events**.
7) For the 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/awscodecommit/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



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.

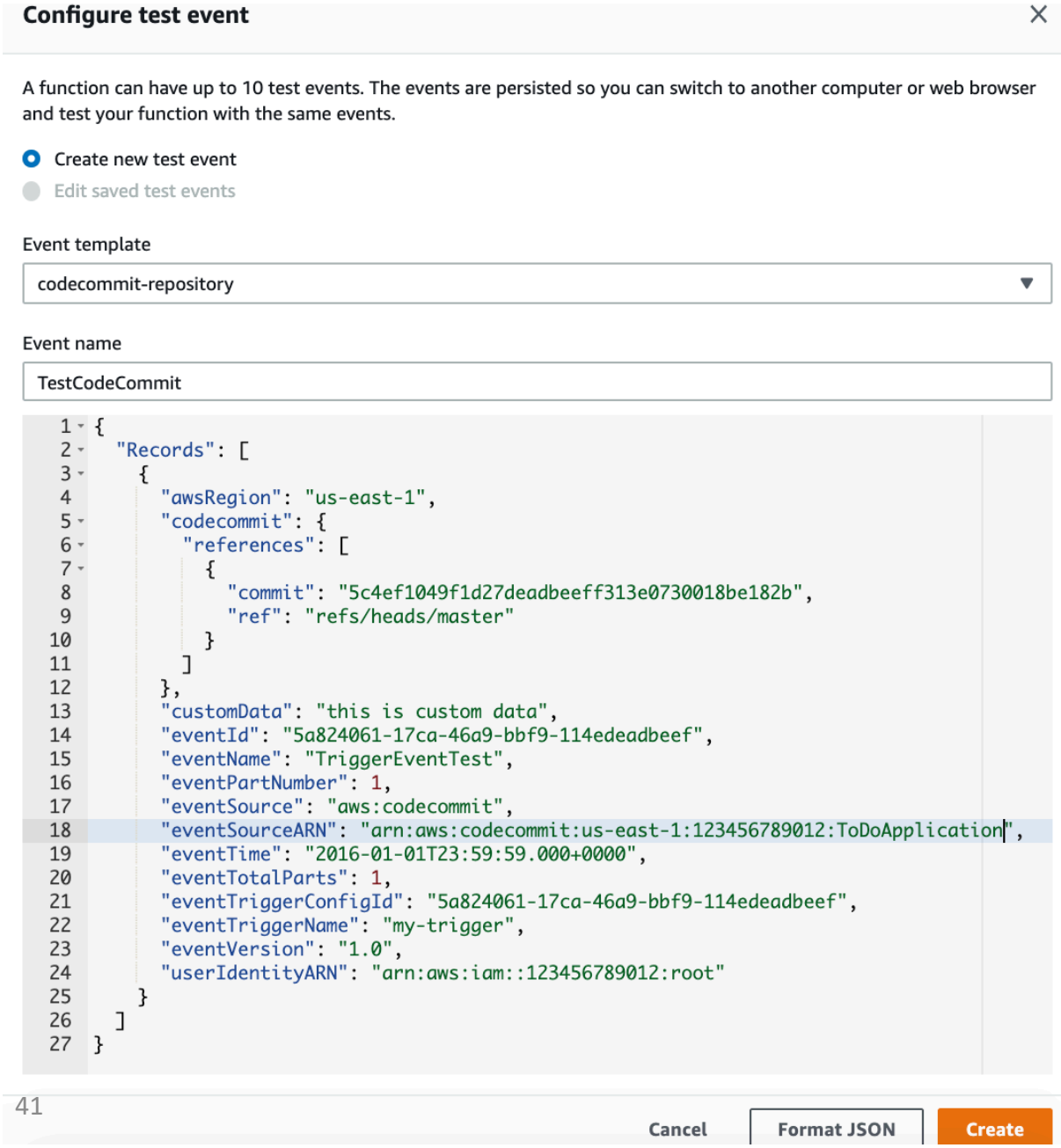
Once you have added the code, click on the save button

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.**

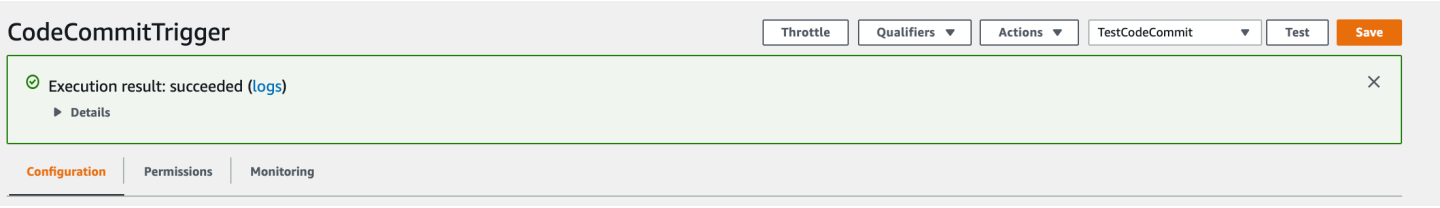
In the Configure test events pop up form, click on the drop down menu for **Event template** , select the **codecommit-repository** template from the list. This will show a template with the sample event which would be generated to trigger the lambda function from CodeCommit. In order to customize it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

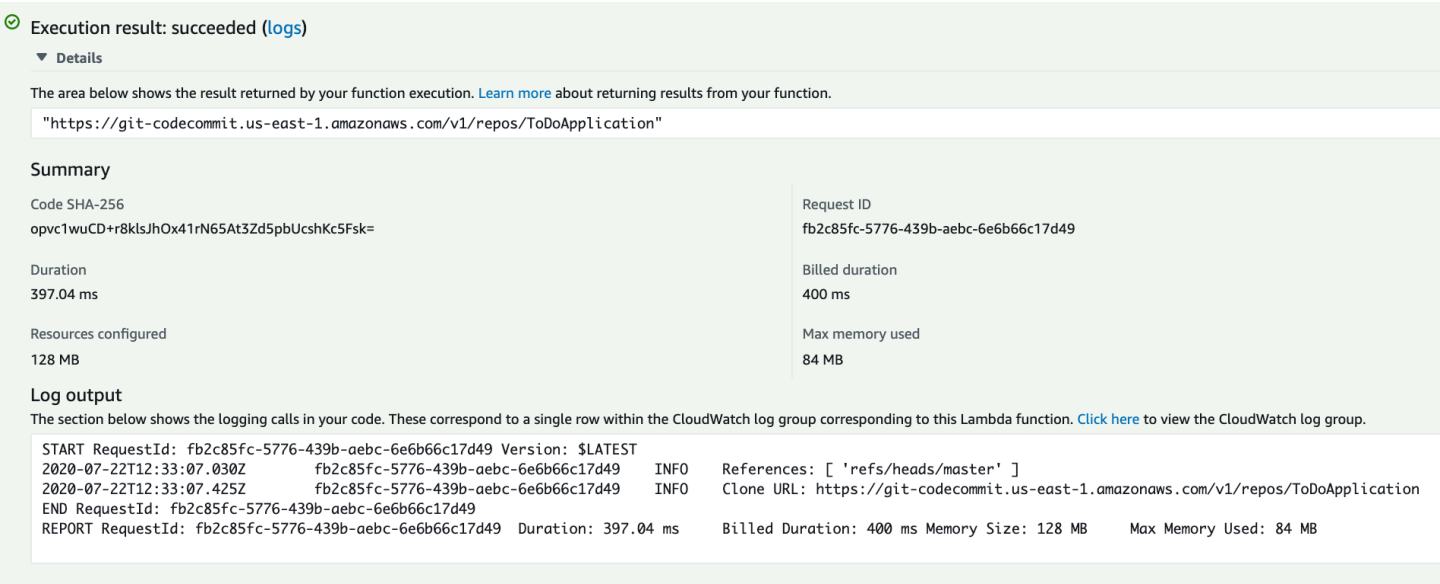
Enter a name for the test event and click on Create button



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.



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.