Bring your own script to the cloud

This tutorial explains how you can create and execute a custom script using AWS. AWS supports multiple ways to achieve this. One option, the AWS Lambda compute service, allows you to run code without provisioning or managing servers. It executes your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time you consume - there is no charge when your code is not running. With AWS Lambda, you can run code for virtually any type of application or backend service - all with zero administration. AWS Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, code monitoring and logging.

Using the Lambda Function

You can use AWS Lambda to run your code in response to events, such as changes to data in an Amazon S3 bucket or an Amazon DynamoDB table; to run your code in response to HTTP requests using Amazon API Gateway; or invoke your code using API calls made using AWS Software Development Kits (SDKs). With these capabilities, you can use Lambda to easily build data processing triggers for AWS services like Amazon S3 and Amazon DynamoDB, process streaming data stored in Kinesis, or create your own back end that operates at AWS scale, performance, and security.

Create a Lambda function

1. Sign into the AWS Management Console.
2. Navigate to the AWS Lambda Console by typing “Lambda console” in the “Find Services” area of the Management Console (see Figure 8.1).

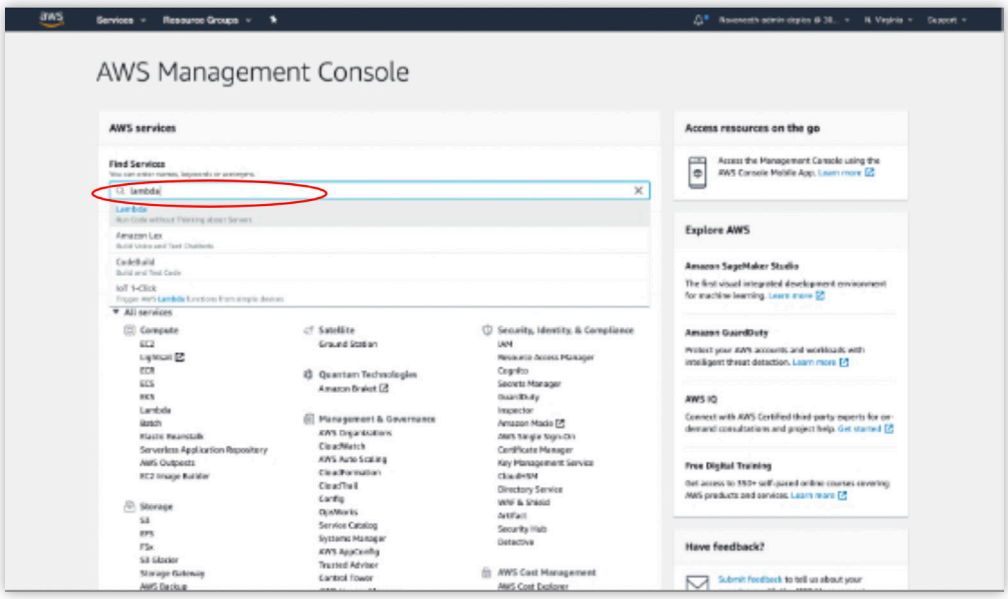

Figure 8.1

Cloud technology is evolving so fast that it is likely that some details in the primer may no longer match reality when you are trying to use it. If you find mismatches (e.g. broken third-party links), please send them to support@earthdata.gov so that we can feed them into the next release of the primer.

If this is the first time you have navigated to the AWS Lambda Console, you will see the page shown in Figure 8.2.

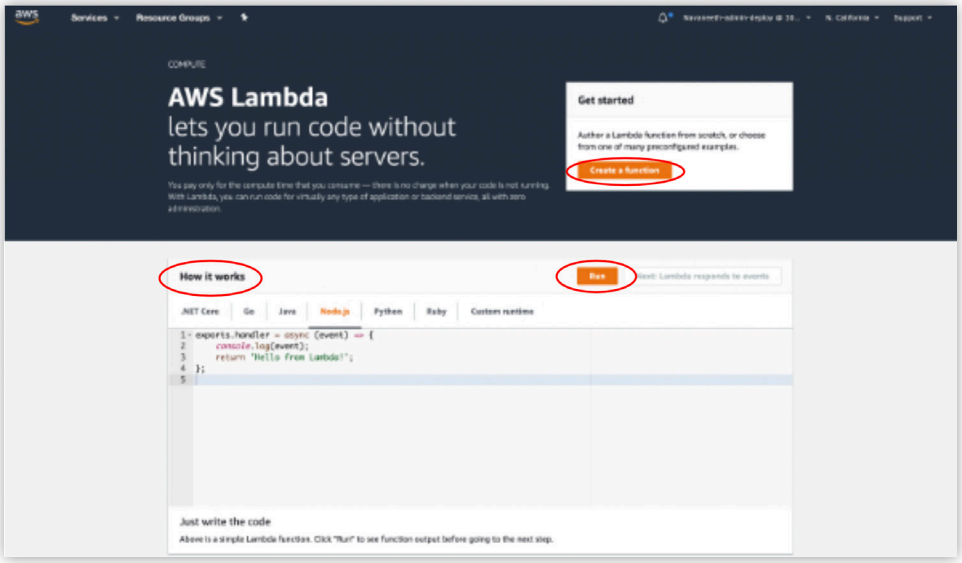

Figure 8.2

> **Note:** AWS Lambda offers a simple "Hello World" function upon introduction under the **How it works** label and includes a **Run** option (Figure 8.1), allowing you to invoke the function as a general introduction.

3. Create a function on the Lambda Functions list page by clicking on **Create Function** (Figure 8.3), to go to the *Create function* page.

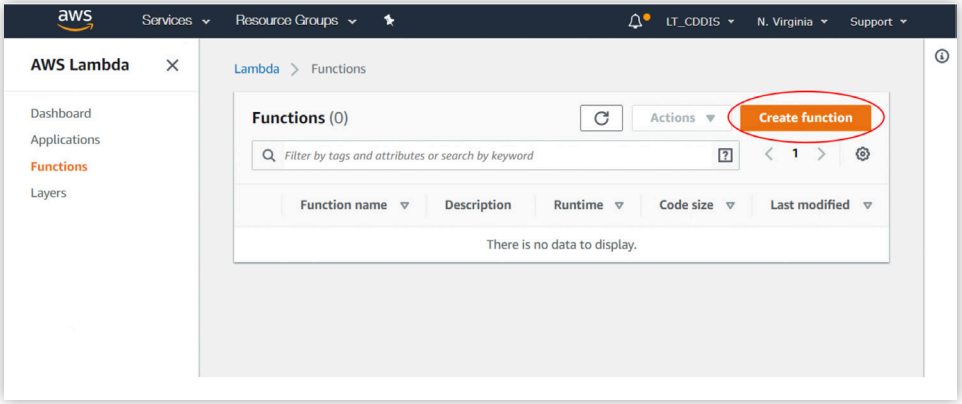

Figure 8.3

4. Choose **Author from scratch** on the *Create function* page (Figure 8.4)

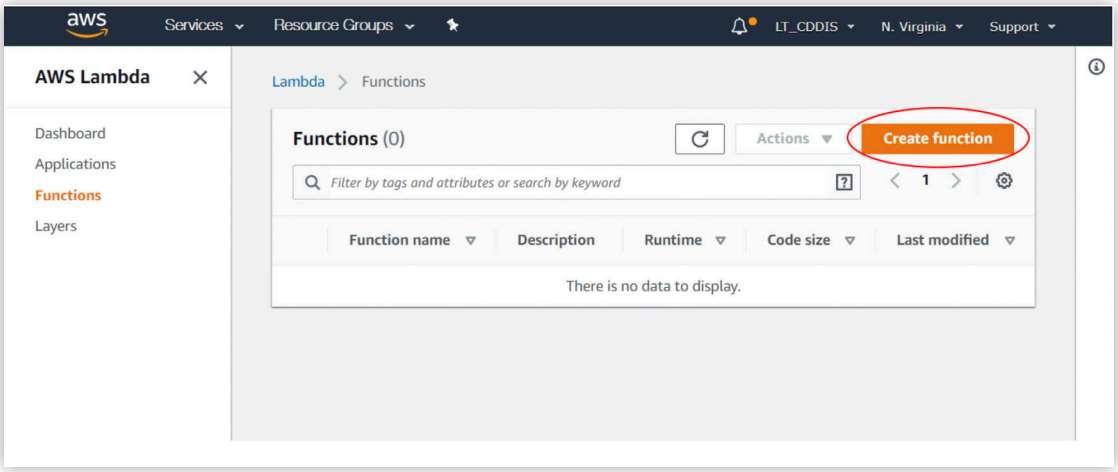

Figure 8.4

5. Give your Lambda function a name under **Basic information**.
6. Choose Python 3.7 from the drop-down list labeled **Runtime**.
7. Reveal the options under **Choose or create an execution role** by clicking on the arrow to the left of the words.

You will then see the options shown in Figure 8.5.

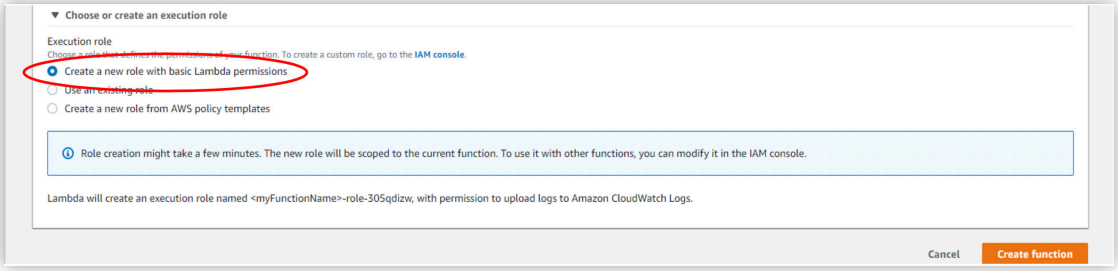

Figure 8.5

8. Create a new role by selecting **Create new role from AWS policy template(s)** under the Execution role heading to reveal more options (Figure 8.6).

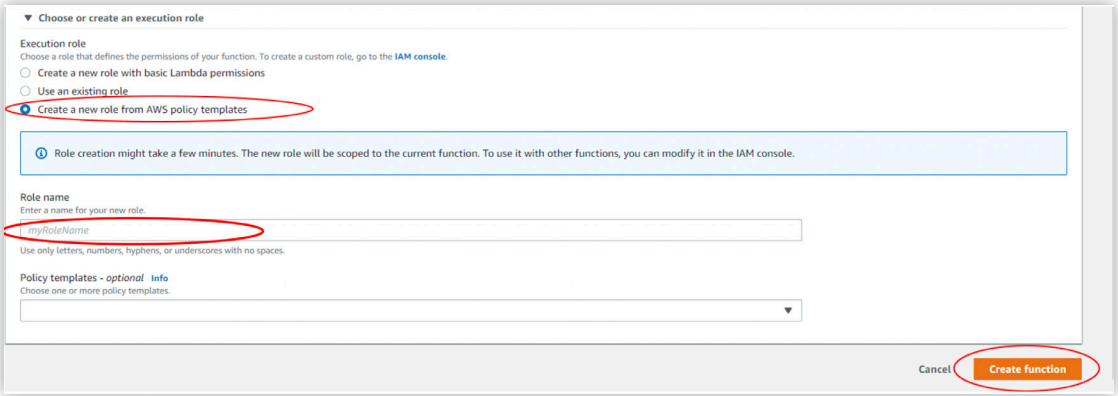

Figure 8.6

9. Enter a name for your new role under Role name. You must provide a name for your role.

Leave the **Policy templates** field blank.

10. Create your function by clicking on the **Create Function** button.

This will automatically create a role and will associate it with your lambda function (permissions required by your lambda function to run)

A new page for your Lambda function will appear containing many options for setting up your function (Figure 8.7).

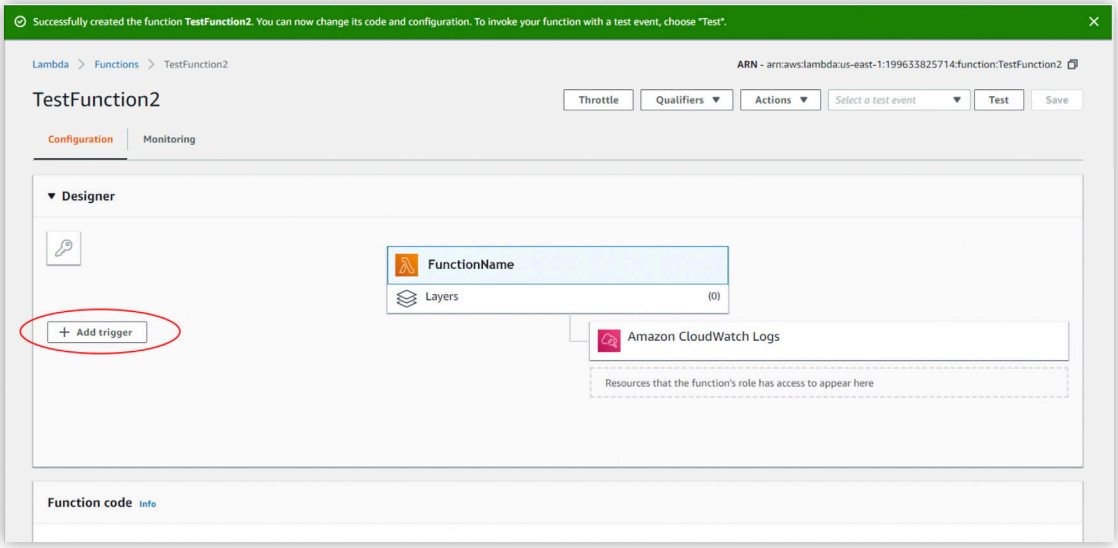

Figure 8.7

We will not be using triggers in this tutorial, but clicking on the **Add trigger** button in Figure 8.7 will take you to a window with several options from which you can choose to trigger your Lambda function (Figure 8.8)

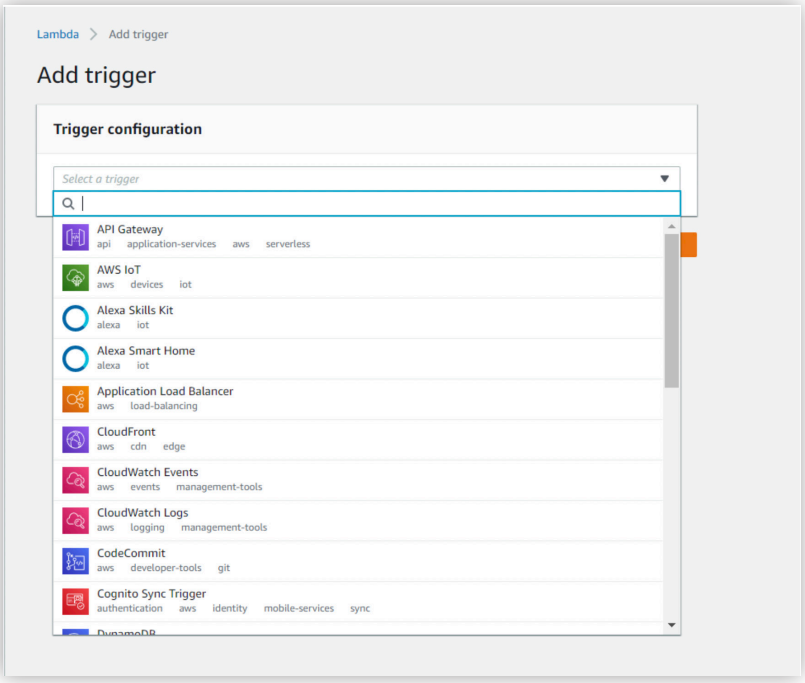

Figure 8.8

Depending on which service you select, you are prompted to provide relevant information for that service. For example, if you select DynamoDB, you need to provide the following:

- The name of the DynamoDB table
- Batch size
- Starting position

Take a look at the window with the options for setting up your function; the second vertical panel is titled Function Code (toward the bottom of Figure 8.7). Figure 8.9 shows the rest of the panel not seen in Figure 8.7.

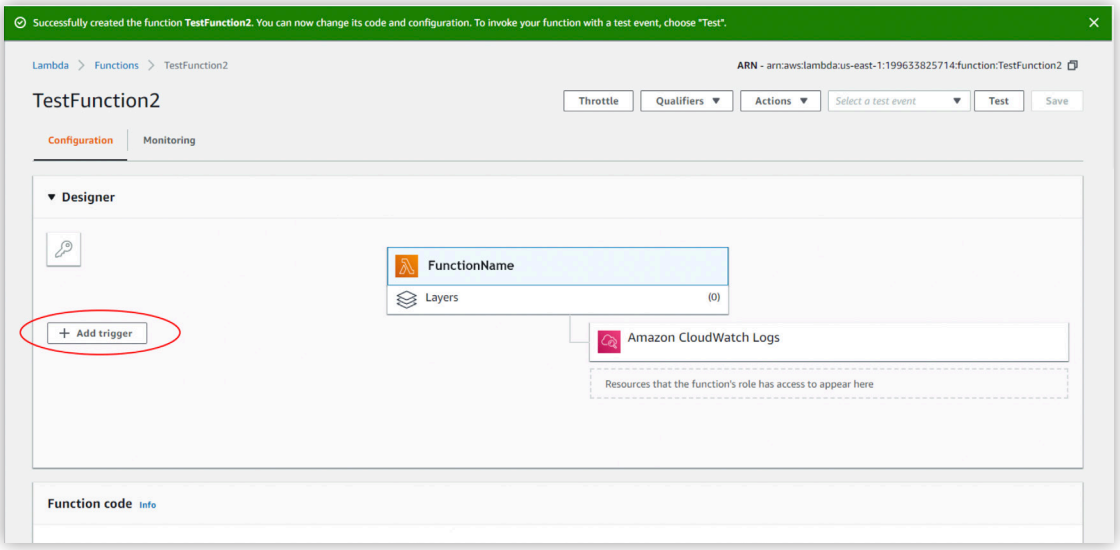

Figure 8.9

Note the code in the second column; when run, it returns a simple "Hello from Lambda" greeting. The **Handler** field on the top right shows a value of "lambda_function.lambda_handler". This is the filename.handler function. The console saves the sample code in the lambda_function.py file. The function name that receives the event as a parameter when the Lambda function is invoked is in the lambda_handler.

Other configuration options on this page include:

- **Environment variables** – enables you dynamically pass settings to your Lambda functions code and libraries without making changes to your code.
- **Tags** – allows you to attach key-value pairs to AWS resources to better organize them.
- **Execution role** – allows you to administer security on your function, using defined roles and policies or creating new ones.
- **Basic settings** – allows you to dictate the memory allocation and timeout limit for your Lambda function.
- **Network** – allows you to select a VPC (Virtual Private Cloud) your function will access.
- **Debugging and error handling** – allows you to select a AWS Lambda Function Dead Letter Queues resource to analyze failed function invocation retries. It also allows you to enable active tracing.
- **Concurrency** – allows you to allocate a specific limit of concurrent executions allowed for this function.
- **Auditing and compliance** – logs function invocations for operational and risk auditing, governance and compliance.

Invoke the Lambda Function and Verify Results, Logs, and Metrics

Follow these steps to invoke your Lambda function using the sample event data provided in the console.

1. Click on **Test** on the Function options page (Figure 8.10).

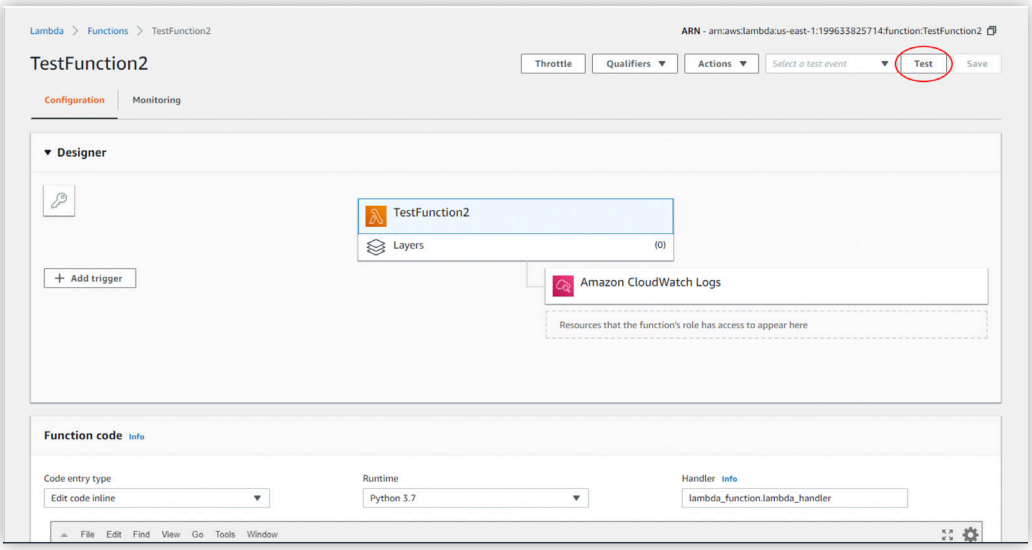

Figure 8.10

2. Click on the circle next to **Create new test event** in the *Configure test event* window (Figure 8.11).

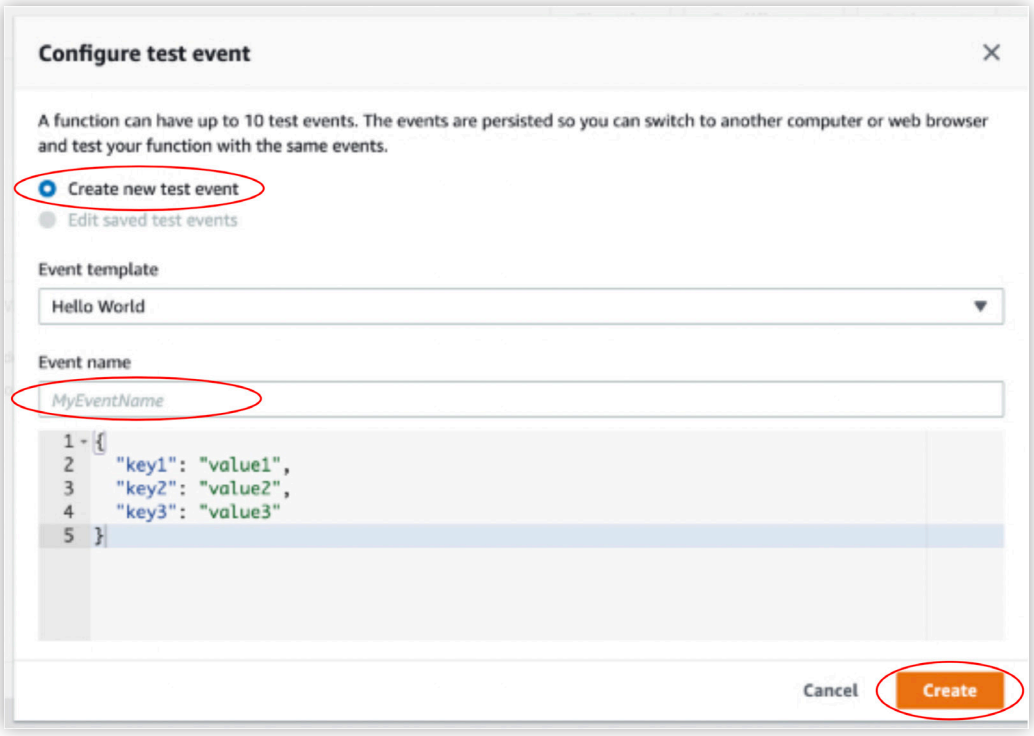

Figure 8.11

Leave the Hello World option in the Event template field.

3. Enter an **Event name**

In the code section, you can change keys and values in the sample JSON, but don't change the event structure. If you do change any keys and values, you must update the sample code accordingly.

4. Click on **Create.**

Each user can create up to 10 test events per function. Those test events are not available to other users.

AWS Lambda executes your function on your behalf. The handler in your Lambda function receives and then processes the sample event.

You will then see the window in Figure 8.12

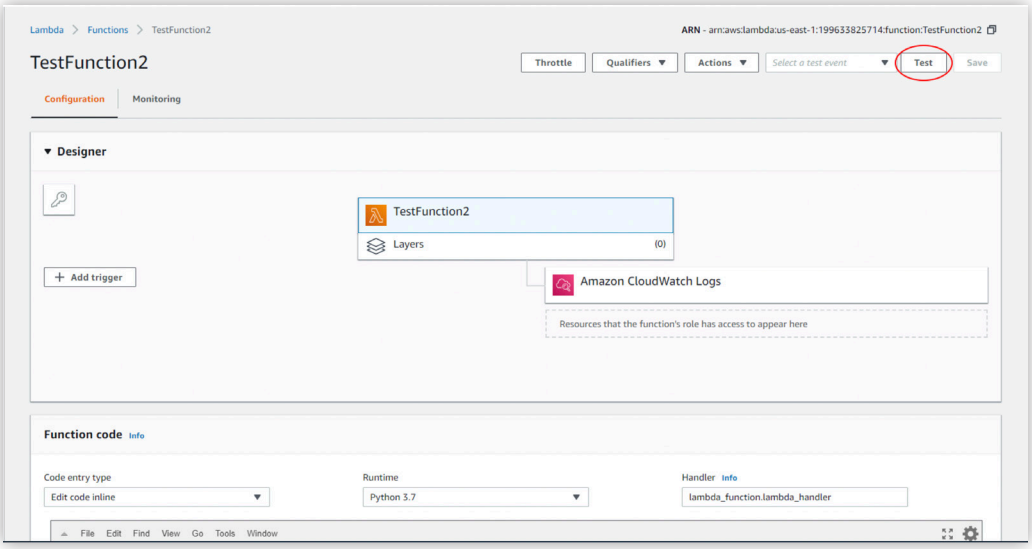

Figure 8.12

5. Click on **Details** to see the full results.

The **Execution result** section shows the execution status as **succeeded** and also shows the function execution results, returned by the return statement.

The **Summary** section shows the key information reported in the **Log output** section in a nice format. This includes important information such as **memory used and billed duration**.

The **Log output** section shows the log AWS Lambda generates for each execution.

These are the logs written to CloudWatch by the Lambda function. The AWS Lambda console shows these logs for your convenience.

Note that clicking on the **Click here** link in Figure 8.12 shows logs in the AWS CloudWatch console. The function then adds logs to Amazon CloudWatch in the log group that corresponds to the Lambda function.

Run the Lambda function a few more times by clicking on the Test button to gather some metrics.

View the metrics by selecting the **Monitoring** tab under the green results box (Figure 8.13).

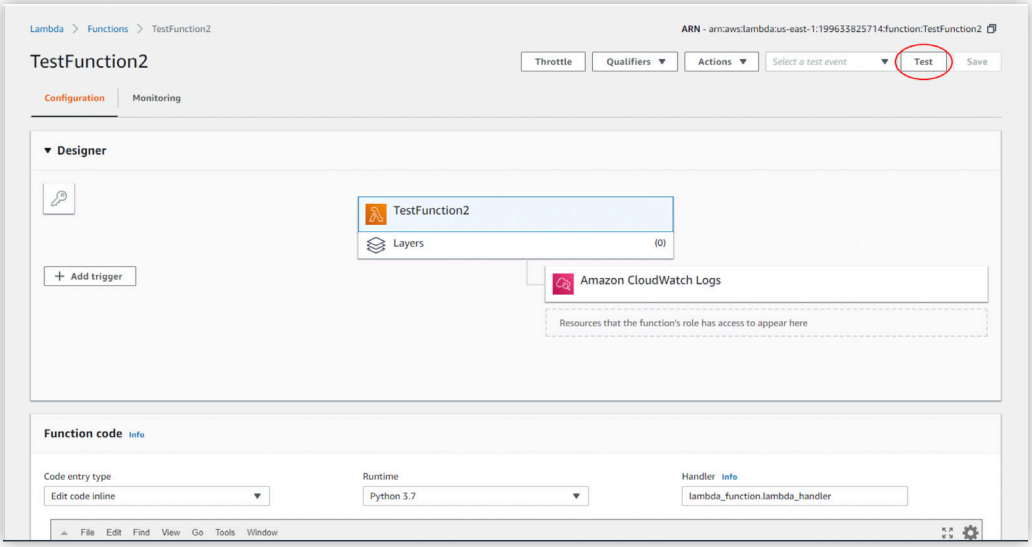

Figure 8.12

You will then be able to view metrics about your script executions (Figure 8.14) including

- Number of times invoked
- Duration
- Error count and Success rate (%)
- Number of Throttles
- Iterator Age
- Dead letter errors

You may have to refresh each graph separately using the small dropdown menu in the upper right of each graph (Figure 9.14) after each run to update the graphs.

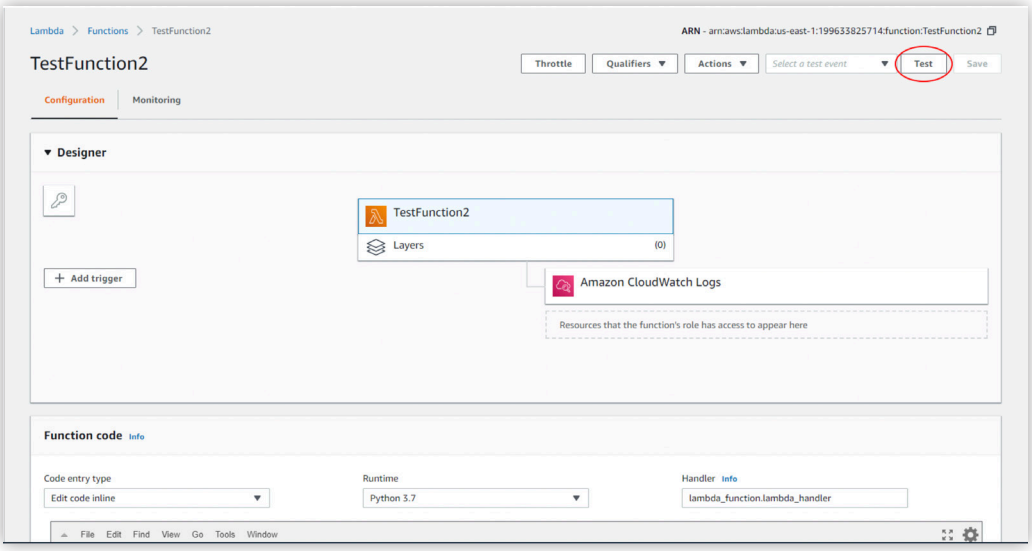

Figure 8.14

Delete a Lambda function

When you no longer need the lambda function, you can delete your lambda function in the *AWS Lambda Functions* page.

Select the function you want to delete by clicking on the small circle next to its name; in this tutorial, it is “lambda_test” Figure 8.15).

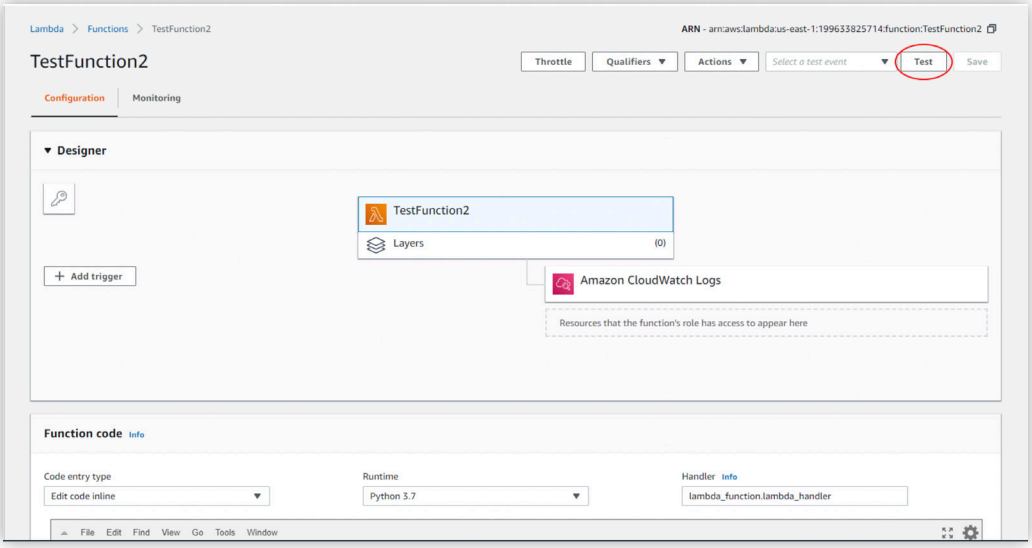

Figure 8.15

2. Select **Delete** from the **Actions** dropdown menu in Figure 8.15 to delete it.

A confirmation prompt is displayed to make sure the you want to delete the lambda function. Click delete. Successful deletion will be notified in the console (Figure 8.16).

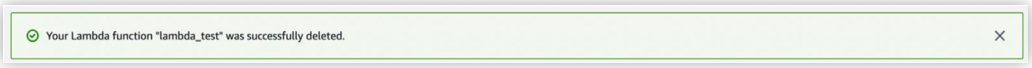

Figure 8.16

> **Note:** The associated role(s) will not be deleted automatically. You have to delete the role(s) manually in Identity and Access management (IAM). Search with the role name and click delete from the menu.