

# EXPERIMENT NO. 11

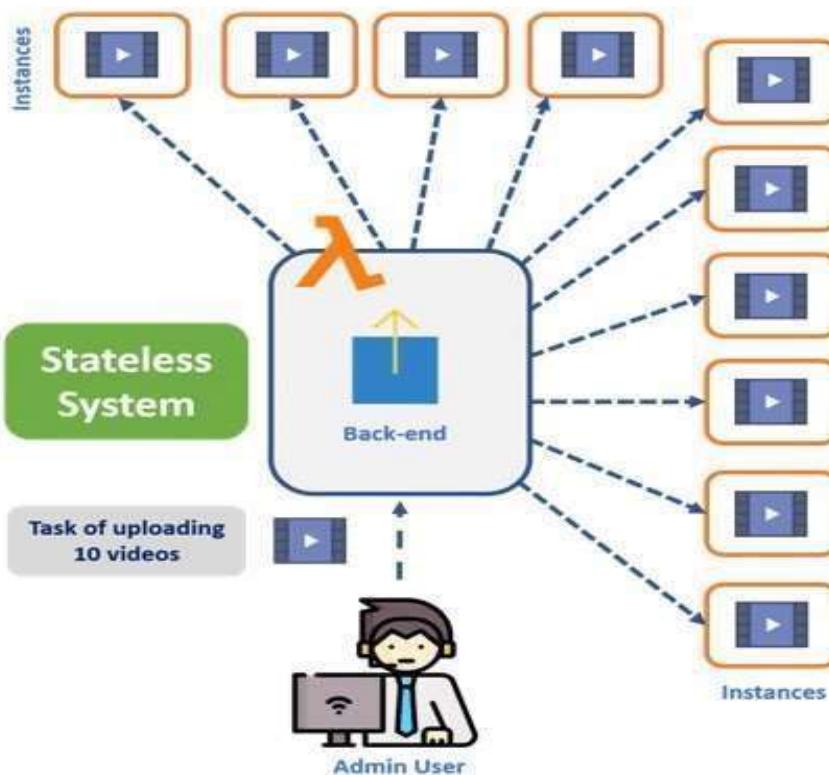
DATE –

## Aim -

To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.

## Theory -

AWS Lambda is one of the computing services provided by AWS, which is event- driven and serverless. It is a stateless serverless system that helps us run our background tasks in the most efficient manner possible.



Here, being server less doesn't mean that servers are nowhere in the play. Instead, it means that we don't have to worry about the provisioning or management of our servers or instances; it just helps us focus on our main goal, i.e., coding. We just have to put our code in AWS Lambda, and we're good to go! Whatever resources are required for our code in response to our events, AWS Lambda automatically provides us. The best feature of it is that we just have to pay for every single request made during the time.

## Events that Trigger AWS Lambda

Here, are Events which will be triggered when you use AWS Lambda.

- Insert, updating and deleting data Dynamo DB table
- To include push notifications in SNS
- To search for log history in CloudTrail
- Entry into an S3 object
- DynamoDB can trigger AWS Lambda whenever there is data added, modified, and deleted in the table.
- Helps you to schedule the event to carry out the task at regular time pattern.
- Modifications to objects in S3 buckets
- Notifications sent from Amazon SNS.
- AWS Lambda can be used to process the CloudTrail logs
- API Gateway allows you to trigger AWS Lambda on GET/POST methods.

## AWS Lambda Concepts

Function:

A function is a program or a script which runs in AWS Lambda. Lambda passes invocation events into your function, which processes an event and returns its response.

Runtimes:

Runtime allows functions in various languages which runs on the same base execution environment. This helps you to configure your function in runtime. It also matches your selected programming language.

Event source:

An event source is an AWS service, such as Amazon SNS, or a custom service. This triggers function helps you to execute its logic.

Lambda Layers:

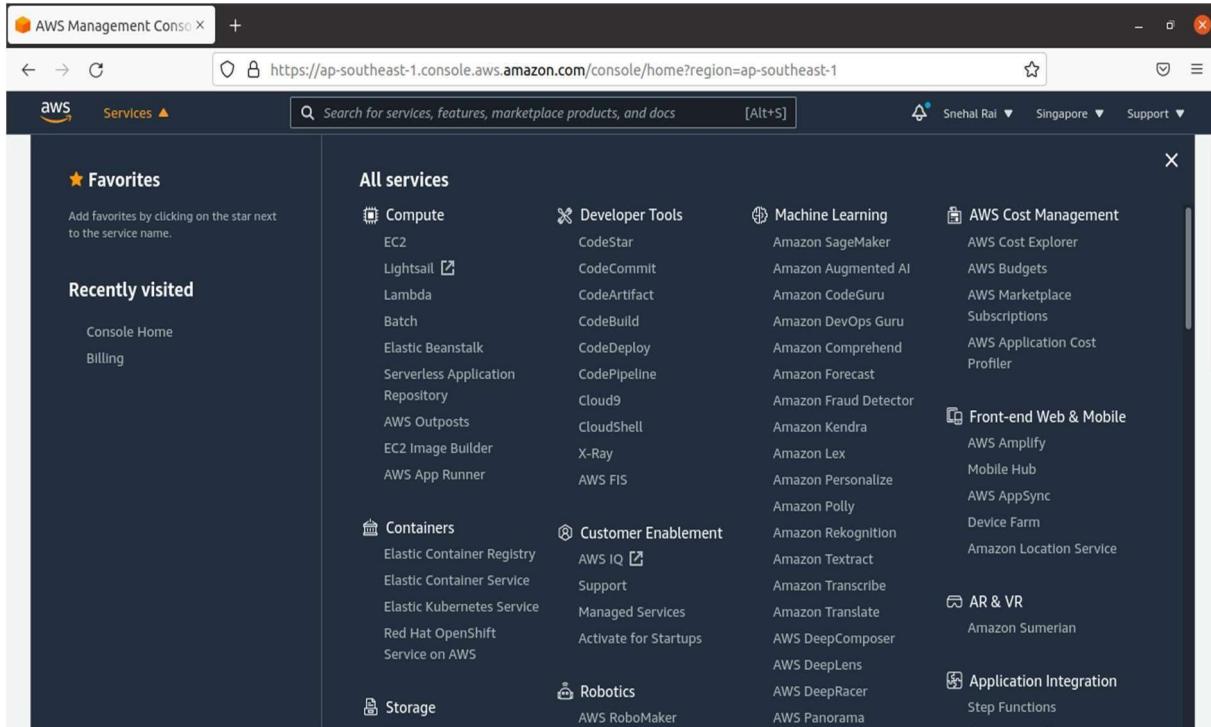
Lambda layers are an important distribution mechanism for libraries, custom runtimes, and other important function dependencies. This AWS component also helps you to manage your development function code separately from the unchanging code and resources that it uses.

Log streams:

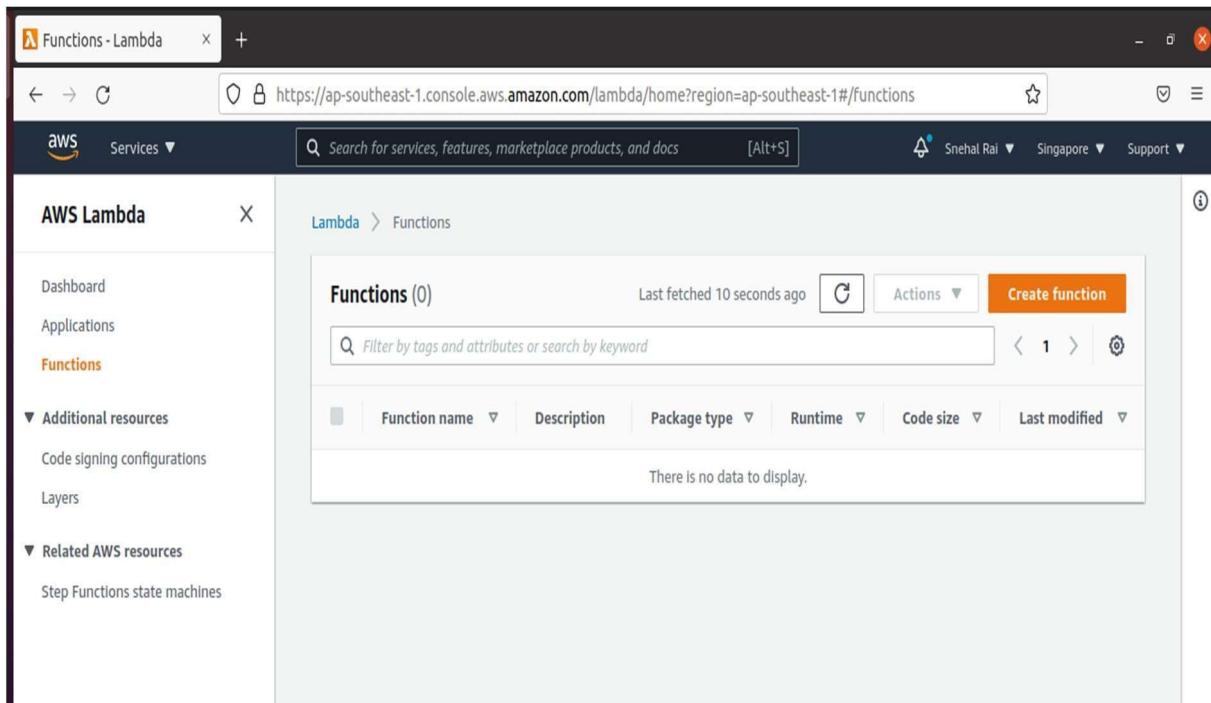
Log stream allows you to annotate your function code with custom logging statements which helps you to analyze the execution flow and performance of your AWS Lambda functions.

## Steps:

### Step 1 - Open AWS Lambda URL in AWS



The screenshot shows the AWS Management Console homepage. The URL in the address bar is <https://ap-southeast-1.console.aws.amazon.com/console/home?region=ap-southeast-1>. The page displays a grid of services categorized into groups like Favorites, All services, Compute, Developer Tools, Machine Learning, AWS Cost Management, etc. The Lambda service is listed under the Compute category.



The screenshot shows the AWS Lambda service page. The URL in the address bar is <https://ap-southeast-1.console.aws.amazon.com/lambda/home?region=ap-southeast-1#/functions>. The left sidebar shows navigation links for AWS Lambda, including Dashboard, Applications, Functions (which is selected), Additional resources, and Related AWS resources. The main content area is titled "Functions (0)" and includes a search bar, a toolbar with "Actions" and "Create function" buttons, and a message stating "There is no data to display."

## Step 2 - Create an account Next, Create an account or sign in with your existing account

The screenshot shows the AWS Lambda function creation interface. In the 'Function name' field, 'random-number-generator' is entered. Under 'Runtime', 'Node.js 14.x' is selected. Under 'Architecture', 'x86\_64' is chosen. In the 'Permissions' section, there's a note about creating an execution role and a link to 'Change default execution role'. A 'Create function' button is at the bottom right.

The screenshot shows the AWS Lambda function details page for 'random-number-generator'. A green banner at the top indicates success: 'Successfully created the function random-number-generator. You can now change its code and configuration. To invoke your function with a test event, choose "Test".' The main area shows the function overview, including the ARN: arn:aws:lambda:ap-southeast-1:849631855320:function:random-number-generator. Other tabs like 'Code', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions' are visible at the bottom.

## Step 3-

Edit the code & Click Run, In the next Lambda page,

1. Edit the code
2. Click Run

The screenshot shows the AWS Lambda function configuration page for a function named "random-number-generator". The "Code" tab is selected. The code editor displays the following JavaScript code:

```
1 exports.handler = async (event) => {
2     // TODO implement
3     const response = {
4         statusCode: 200,
5         body: JSON.stringify('Hello from Lambda!'),
6     };
7     return response;
8 };
```

The screenshot shows the AWS Lambda function configuration page for the same function. The "Test" tab is selected. The "Test event" section is visible, showing a "New event" template named "hello-world" and a "Name" field containing "lambdademo". Below the template, a JSON event payload is displayed:

```
1 ~ []
2   "key1": "value1",
3   "key2": "value2",
4   "key3": "value3"
5 }
```

The screenshot shows the AWS Lambda function configuration page for 'random-number-generator'. At the top, a green success message states: 'Successfully created the function random-number-generator. You can now change its code and configuration. To invoke your function with a test event, choose "Test".'. Below this, there are three main sections: 'Runtime settings', 'Layers', and 'Code source'. The 'Runtime settings' section shows 'Runtime: Node.js 14.x', 'Handler: index.handler', and 'Architecture: x86\_64'. The 'Layers' section shows a table with columns: Merge order, Name, Layer version, Compatible runtimes, and Compatible architectures. A note at the bottom of this section says 'There is no data to display.' The 'Code source' section is currently empty, showing a file tree for 'random-number-generator' with 'index.js' selected. A 'Test' button is highlighted in orange.

## Step 4 - Check output

You will see output

The screenshot shows the AWS Lambda function test results page for 'random-number-generator'. At the top, a green success message states: 'Successfully created the function random-number-generator. You can now change its code and configuration. To invoke your function with a test event, choose "Test".'. Below this, there are two tabs: 'Execution results' and 'Function Logs'. The 'Execution results' tab is selected, showing a successful execution with status: Succeeded, Max memory used: 53 MB, and Time: 2.28 ms. The 'Response' field contains the JSON object: { "statusCode": 200, "body": "{\"Hello from Lambda!\""} }. The 'Request ID' field shows the value c7999b89-4281-4a5c-ae1d-58195e87cb22. The 'Function Logs' tab shows detailed log entries for the execution.

**AWS Lambda**

**Resources for Asia Pacific (Singapore)**

Lambda function(s)	Code storage	Full account concurrency	Unreserved account concurrency
1	304.0 byte (0% of 75.0 GB)	1000	1000

**Create function**

**Account-level metrics**

The charts below show metrics across all your Lambda functions in this AWS Region.

1h 3h 12h 1d 3d 1w Custom Add to dashboard

Error count and success rate	Throttles	Invocations
Count: 1 No unit: * 100	Count: 1 0.925	Count: 2

## Step 5- Delete Function.

**AWS Lambda**

**Functions (1)**

Last fetched 10 seconds ago

Actions	Create function
Filter by tags and attributes or search by keyword	< 1 >

Function name	Description	Package type	Runtime	Code size	Last modified
random-number-generator	-	Zip	Node.js 14.x	304.0 byte	9 minutes ago

## **Conclusion -**

Lambda enables the creation of new micro services to access the data stream by decoupling the product engineering efforts from the platform analytics pipeline, eliminating the need to be bundled with the main analytics applications.

From this experiment, I got a detailed understanding of AWS Lambda, its workflow, various functions, how to create Lambda functions, its need, and various use cases.