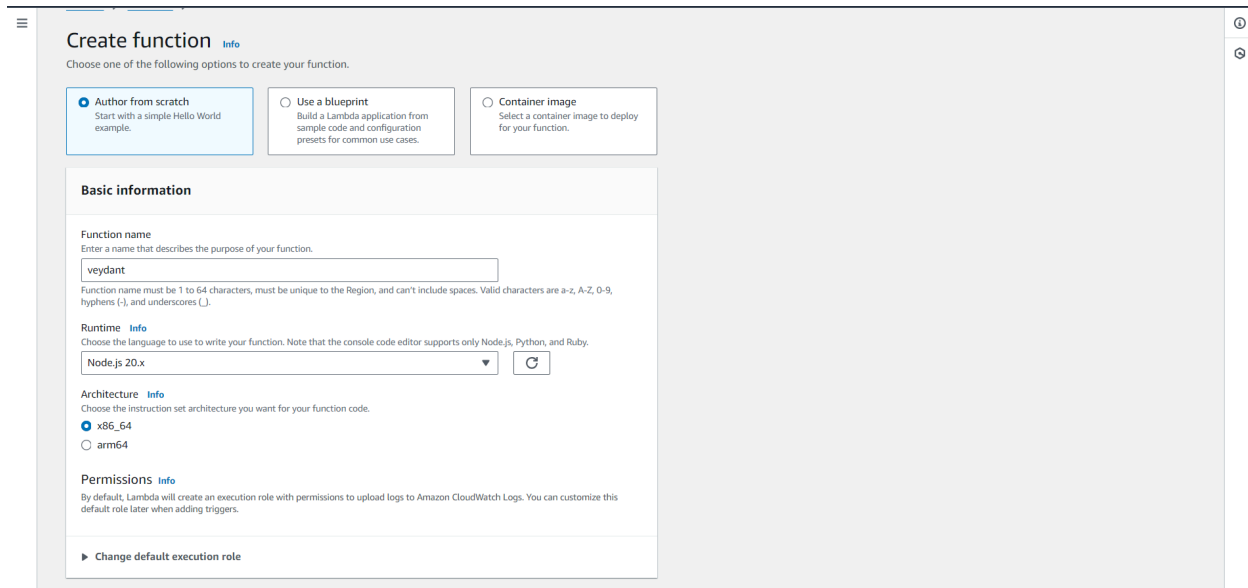


**Aim:** To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs

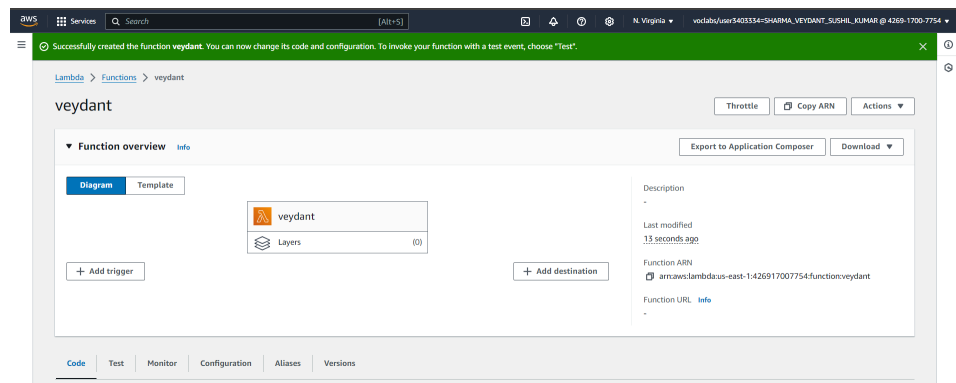
**Step 1:** On your AWS console, click on 'Lambda' in the services section and click on 'Create function'.

**Step 2:** Give your Lambda function a name. Select the language to use to write your function (Node.js is the default and what we will use in this experiment). Keep other options as default.



The screenshot shows the 'Create function' page in the AWS Lambda console. It has three tabs: 'Author from scratch' (selected), 'Use a blueprint', and 'Container image'. Under 'Basic information', the 'Function name' is 'veydant'. The 'Runtime' is set to 'Node.js 20.x'. The 'Architecture' is set to 'x86\_64'. There is a 'Permissions' section with a link to 'Change default execution role'.

Your Lambda function gets created.



**Step 3:** The general configuration of the function is visible in the 'Configuration' tab. To change the configuration, click on 'Edit' Change the timeout to '1'. Then click on 'Save'.

The screenshot shows the 'Configuration' tab of the AWS Lambda console. On the left, a sidebar lists configuration options: General configuration, Triggers, Permissions, Destinations, Function URL, Environment variables, Tags, and VPC. The 'General configuration' section is active, displaying a table with the following details:

General configuration <a href="#">Info</a>		
Description	Memory	Ephemeral storage
-	128 MB	512 MB
Timeout	SnapStart <a href="#">Info</a>	
0 min 3 sec	None	

An 'Edit' button is located in the top right corner of the configuration panel.

The screenshot shows the 'Basic settings' tab of the AWS Lambda console. It contains the following configuration fields and options:

- Description - optional:** A text input field.
- Memory [Info](#):** A text input field set to '128' MB. Below it, a note states: 'Your function is allocated CPU proportional to the memory configured. Set memory to between 128 MB and 10240 MB.'
- Ephemeral storage [Info](#):** A text input field set to '512' MB. Below it, a note states: 'You can configure up to 10 GB of ephemeral storage (/tmp) for your function. [View pricing](#)'. A second note states: 'Set ephemeral storage (/tmp) to between 512 MB and 10240 MB.'
- SnapStart [Info](#):** A dropdown menu set to 'None'. Below it, a note states: 'Reduce startup time by having Lambda cache a snapshot of your function after the function has initialized. To evaluate whether your function code is resilient to snapshot operations, review the [SnapStart compatibility considerations](#)'. Supported runtimes: Java 11, Java 17, Java 21.
- Timeout:** Two input fields for '0' min and '1' sec.
- Execution role:** Radio buttons for 'Use an existing role' (selected) and 'Create a new role from AWS policy templates'. A note states: 'Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#)'.
- Existing role:** A dropdown menu set to 'LabRole' and a refresh button.

Step4: Navigate to code source and click on the arrow near test to create test event.

Step 5: Give your test event a name, keep all other options as default and click on 'Save'

The screenshot shows the 'Code source' tab of the AWS Lambda console. It features a code editor with a file named 'index.mjs' containing the following JavaScript code:

```
1 const test = "Hello D15C50"
2 export const handler = async (event) => {
3   // TODO Implement
4   const response = {
5     statusCode: 200,
6     body: JSON.stringify(test),
7   };
8   return response;
9 }
10
```

At the top right, there is an 'Upload from' button. Below the code editor, there is a section for 'Environment' with a dropdown menu set to 'index.mjs'.

Configure test event

A test event is a JSON object that mocks the structure of requests emitted by AWS services to invoke a Lambda function. Use it to see the function's invocation result.

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action

Create new event

Edit saved event

Event name

lambda

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

Private

This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

Shareable

This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)

Template - optional

hello-world

Event JSON

Format JSON

1

2

3

4

5

"key1": "value1",

"key2": "value2",

"key3": "value3"

Cancel

Invoke

Save

Step 6: Click on the 'Test' button. The following output appears.

▼ Execution results		Status: Succeeded	Max memory used: 62 M
Test Event Name	lambda		
Response	<pre>{   "statusCode": 200,   "body": "\\\"Hello D15C50\\\""} </pre>		
Function Logs	START RequestId: 4cb66e32-9897-4955-a8da-327e4384600d Version: \$LATEST END RequestId: 4cb66e32-9897-4955-a8da-327e4384600d REPORT RequestId: 4cb66e32-9897-4955-a8da-327e4384600d Duration: 13.99 ms Billed Duration: 14 ms Memory Size: 128 MB Max Memory Used: 62 MB Init Duration: 143.30 ms		
Request ID	4cb66e32-9897-4955-a8da-327e4384600d		

## Conclusion:

- We created a lambda function and configured using Node.js
- We setup a test event for the function.
- From the output of the tests, we learn about the working of the Lambda function and how changes in its configuration affects its functionality and outputs.