

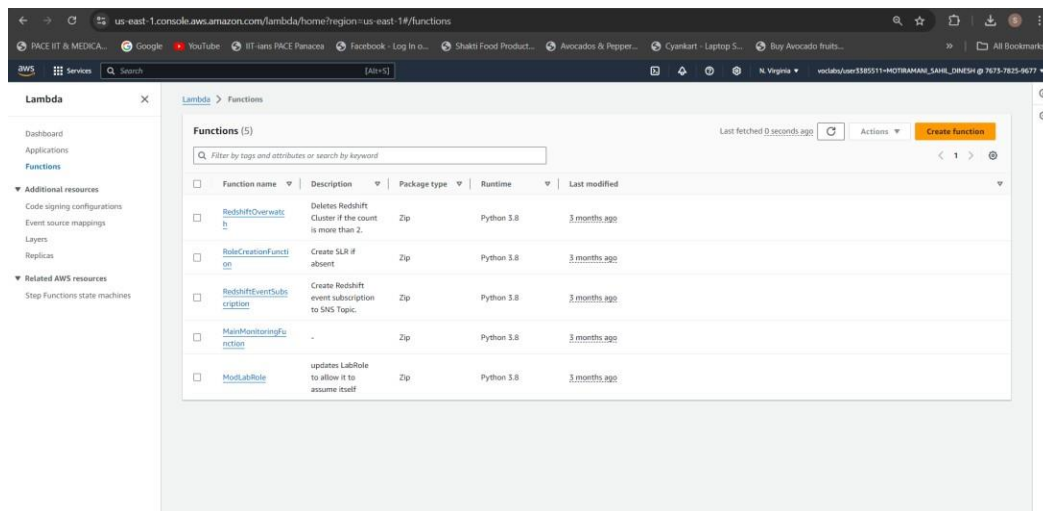
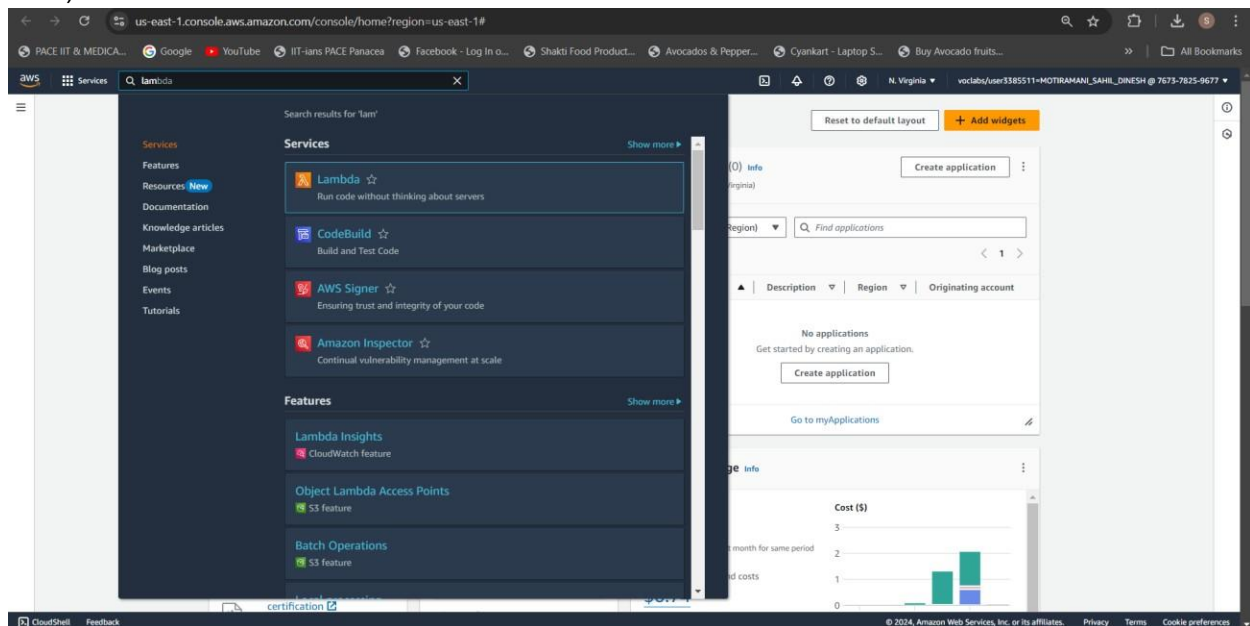
Aim: To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.

Prerequisites:

- 1) AWS account (academy recommended)

Step 1: Set up AWS Lambda Function

- 1) Search for Lambda in the services tab. Click on it once found.



- 2) Click on create functions.
- 3) Give a name to your Lambda function. Select the runtime as Node.js 20.x (You can also use python). Select the architecture as x86_64. Set the default execution role as LabRole if you are doing this on your academy account. (Use an existing role → LabRole)

Create function Info

Choose one of the following options to create your function.

☒ **Author from scratch**
Start with a simple Hello World example.

☐ **Use a blueprint**
Build a Lambda application from sample code and configuration presets for common use cases.

☐ **Container image**
Select a container image to deploy for your function.

Basic information

Function name
Enter a name that describes the purpose of your function.

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Architecture Info
Choose the instruction set architecture you want for your function code.
☒ x86_64
☐ arm64

Permissions Info
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

Change default 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.

View the LabRole role [\[?\]](#) on the IAM console.

Additional Configurations
Use additional configurations to set up code signing, function URL, tags, and Amazon VPC access for your function.

[Cancel](#) [Create function](#)

- 4) Once the function is created, click on the name of the function (myLambda27 in my case).

Functions (7)

Last fetched 0 seconds ago [\[?\]](#) [Actions](#) [Create function](#)

<input type="checkbox"/>	Function name	Description	Package type	Runtime	Last modified
<input type="checkbox"/>	myLambda27	-	Zip	Node.js 20.x	5 days ago
<input type="checkbox"/>	RoleCreationFunction	Create SLR if absent	Zip	Python 3.8	2 months ago
<input type="checkbox"/>	ModLabRole	updates LabRole to allow it to assume itself	Zip	Python 3.8	2 months ago
<input type="checkbox"/>	myLambda27_12	-	Zip	Python 3.12	5 days ago
<input type="checkbox"/>	MainMonitoringFunction	-	Zip	Python 3.8	2 months ago
<input type="checkbox"/>	RedshiftEventSubscription	Create Redshift event subscription to SNS Topic.	Zip	Python 3.8	2 months ago
<input type="checkbox"/>	RedshiftOverwatch	Deletes Redshift Cluster if the count is more than 2.	Zip	Python 3.8	2 months ago

5) This is the dashboard of our lambda function.

Successfully created the function **sahil35**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Lambda > Functions > sahil35

sahil35

Throttle Copy ARN Actions

Function overview Info

Export to Application Composer Download

Diagram Template

sahil35

Layers (0)

+ Add trigger

+ Add destination

Description

Last modified 10 seconds ago

Function ARN arn:aws:lambda:us-east-1:767378259677:function:sahil35

Function URL Info

Code Test Monitor Configuration Aliases Versions

Code source Info

Upload from

File Edit Find View Go Tools Window Test Deploy

Go to Anything (Ctrl-P)

Environment

sahil35 / index.mjs

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

6) This function has the following default code, which is used to print "Hello form Lambda!".

Successfully created the function **sahil35**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Code source Info

Upload from

File Edit Find View Go Tools Window Test Deploy

Go to Anything (Ctrl-P)

Environment

sahil35 / index.mjs

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

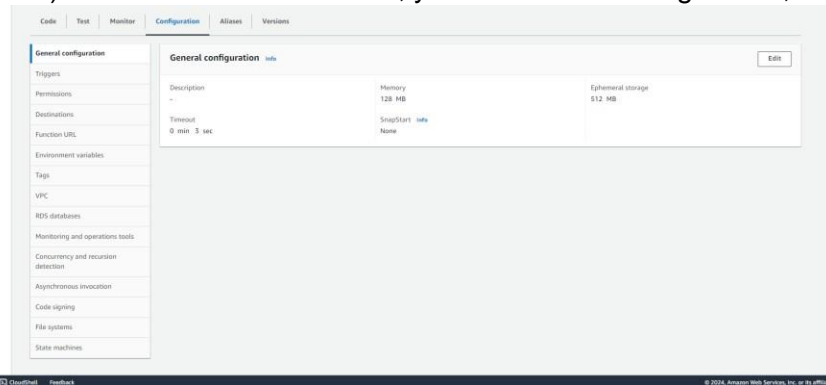
1/1 JavaScript Spaces: 2

Code properties Info

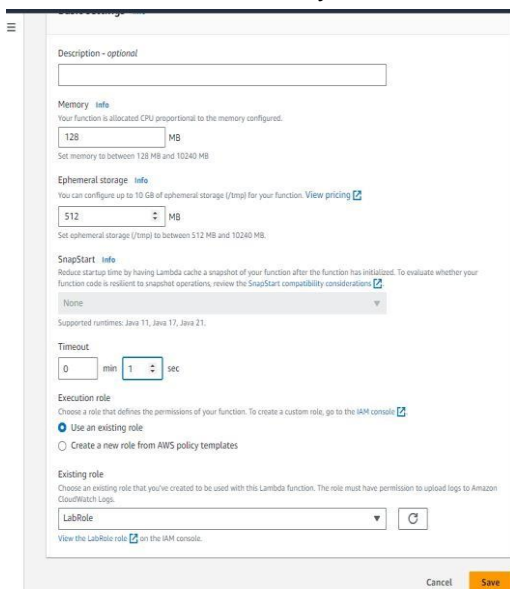
Package size SHA256 hash Last modified

Step 3: Set up configurations and test events

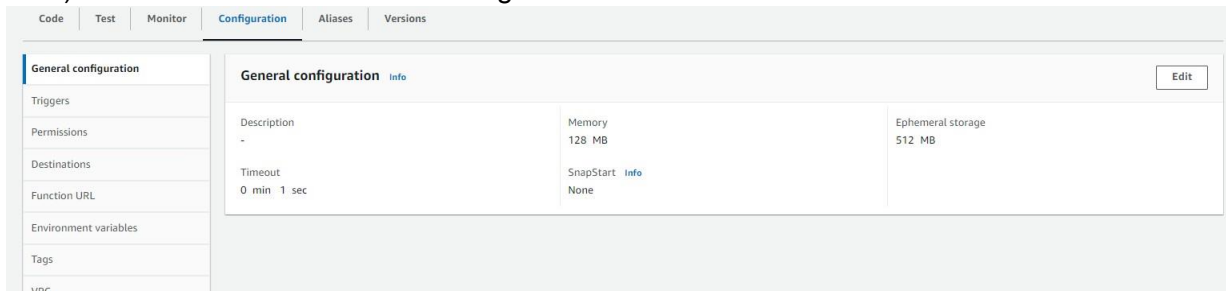
1) Just above the test code, you would find Configuration, click on it. Then click on Edit.



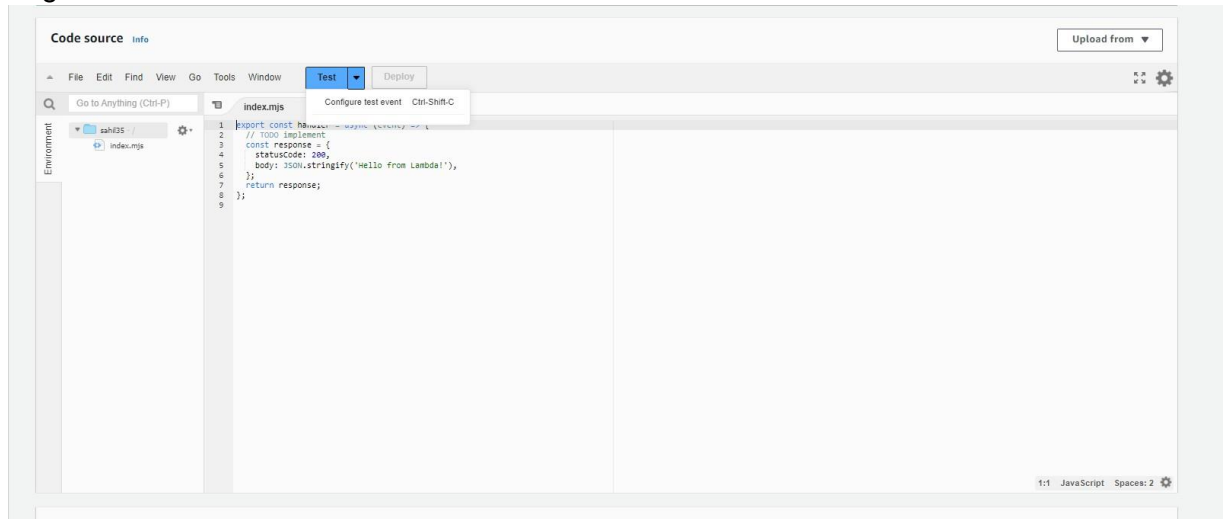
2) Here, change the Timeout to 1 sec. This is the time for which the function can be running before it is forcibly terminated.



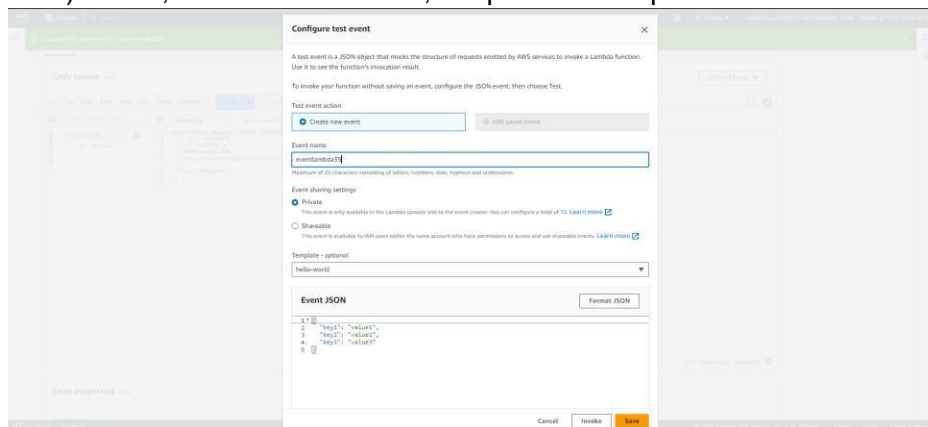
3) We can see the executed changes.



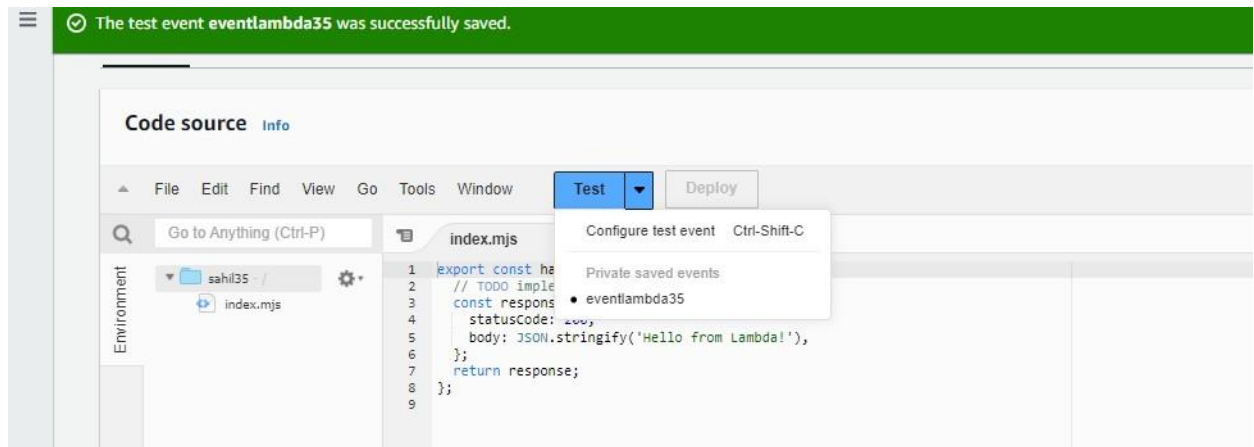
- 4) Switch back to the code tab. Click on the dropdown arrow near test. Then select configure test event.



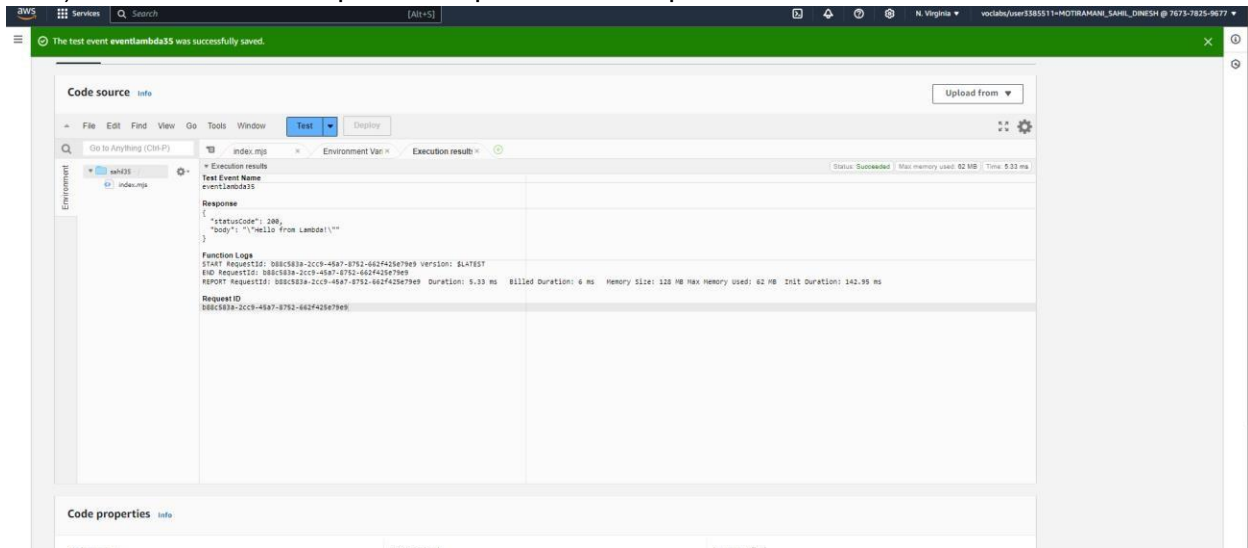
- 5) Here, create a new event, keep the other options default and save the event.



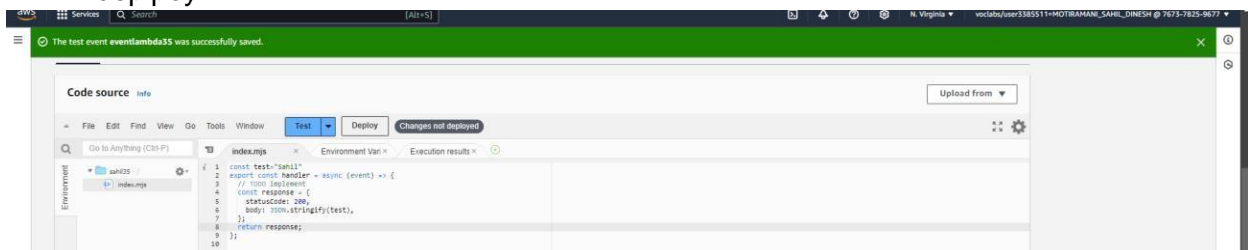
- 6) Now, again click on the dropdown. This time, select the event you have created. Then, click on TEST.



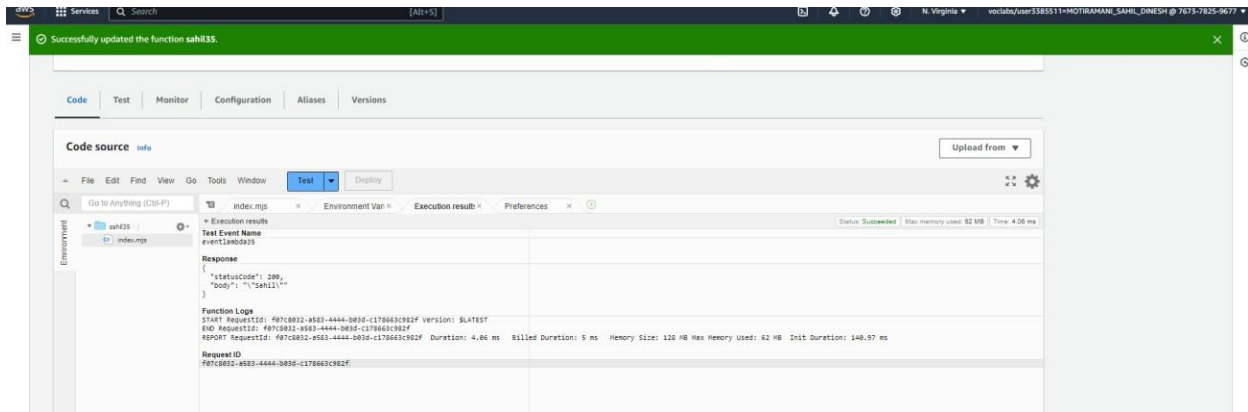
- 7) We can see the expected output for the sample code.



- 8) For a test, declare a string and call it in line 6. After making the changes click on deploy.



- 9) Run the test. We can see that the string we declared has come in the output.



Conclusion:

In this experiment, we effectively investigated the AWS Lambda service by developing and configuring Lambda functions using Python, Java, or Node.js. We discovered how to establish a Lambda function, tweak its settings (like modifying the timeout duration), and evaluate the function with personalized events. Throughout this experience, we noted how Lambda manages executions, including timeout handling and producing expected results according to modifications in the code.