

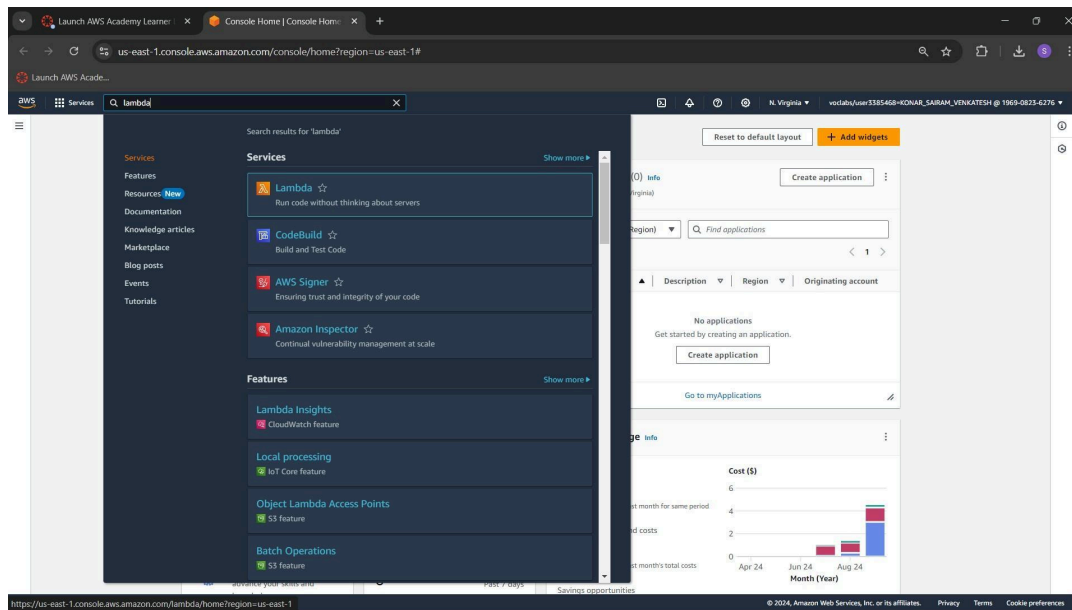
Aim: To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.

Prerequisites:

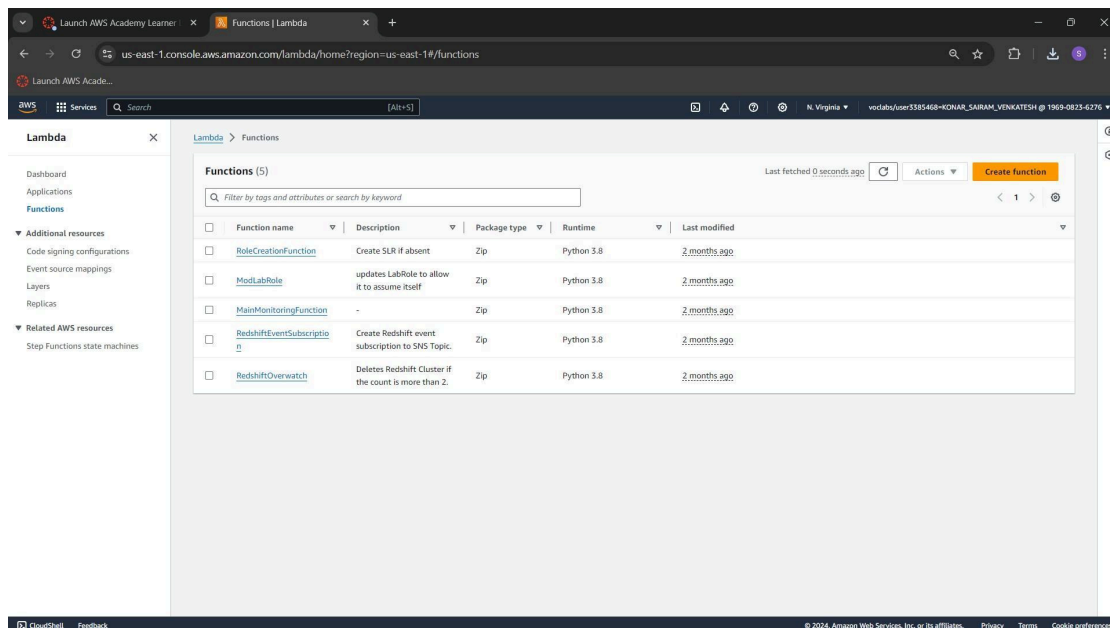
- 1) AWS account (academy recommended)

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)

The screenshot shows the AWS Lambda 'Create function' console page. The 'Basic information' section is visible, showing the function name 'myLambda27', runtime 'Node.js 20.x', architecture 'x86_64', and permissions 'LabRole'.

Basic information

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

Use only letters, numbers, hyphens, or underscores with no spaces.

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 [\[link\]](#).
☐ 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.

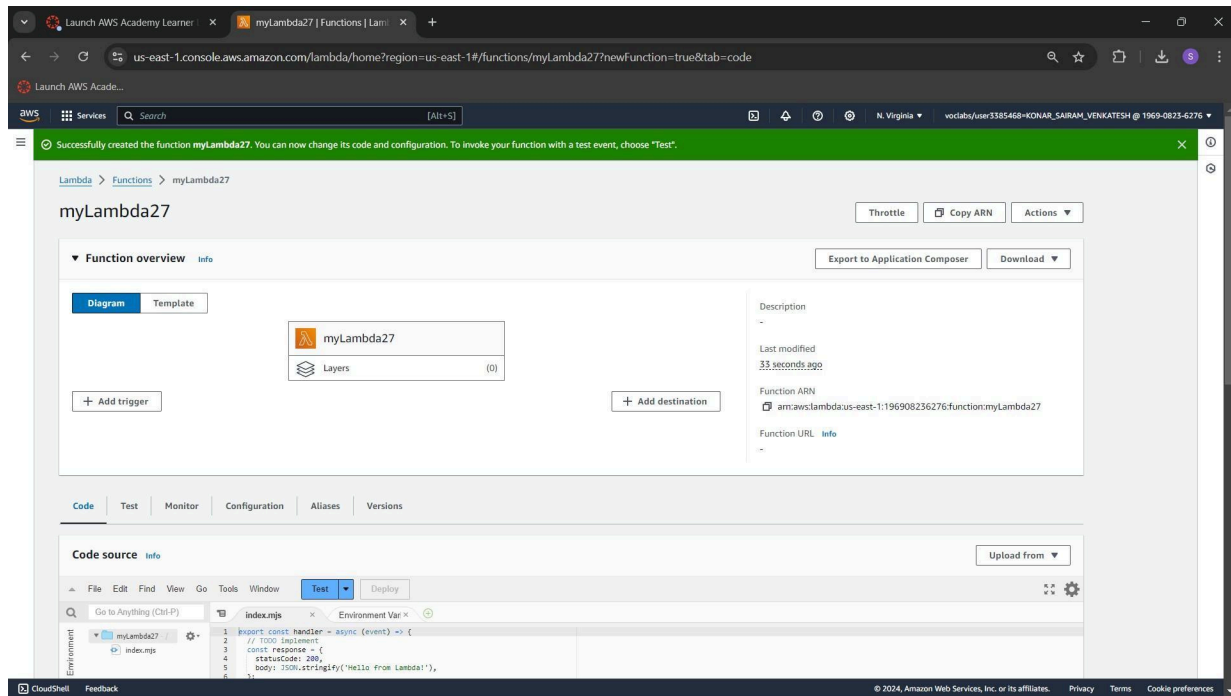
[View the LabRole role \[link\] on the IAM console.](#)

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

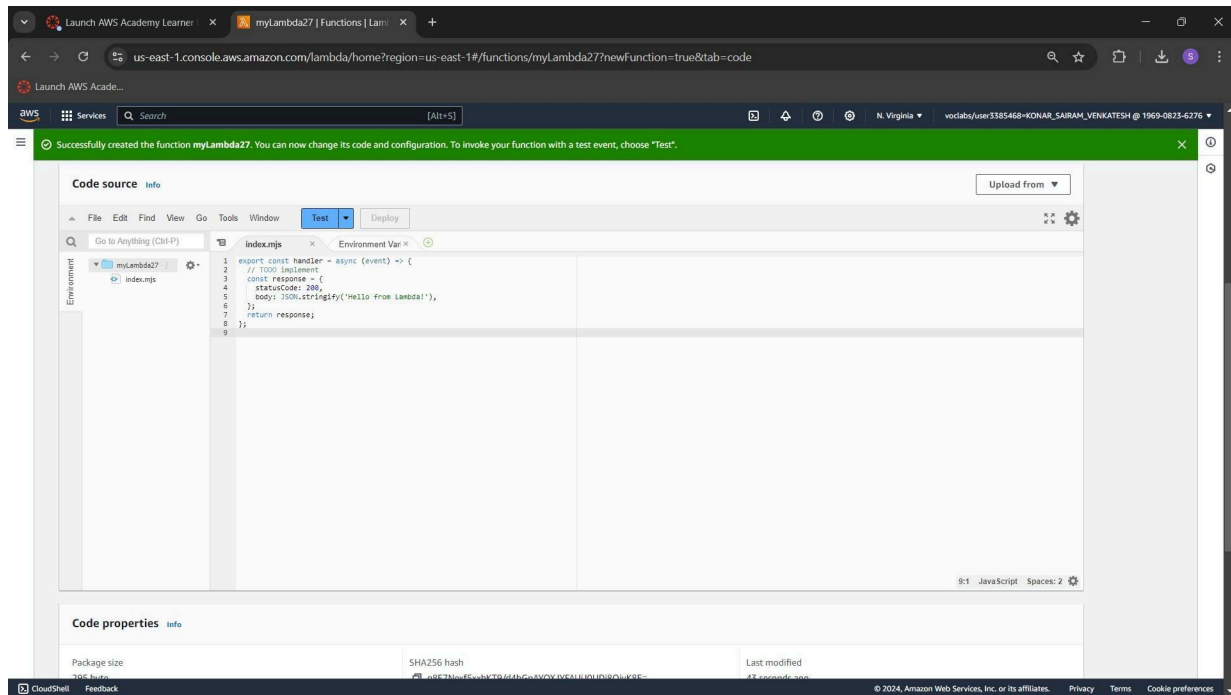
The screenshot shows the AWS Lambda 'Functions' console page. The 'Functions (7)' section is visible, showing a list of functions. The function 'myLambda27' is highlighted.

<input type="checkbox"/>	Function name	Description	Package type	Runtime	Last modified
<input checked="" 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.

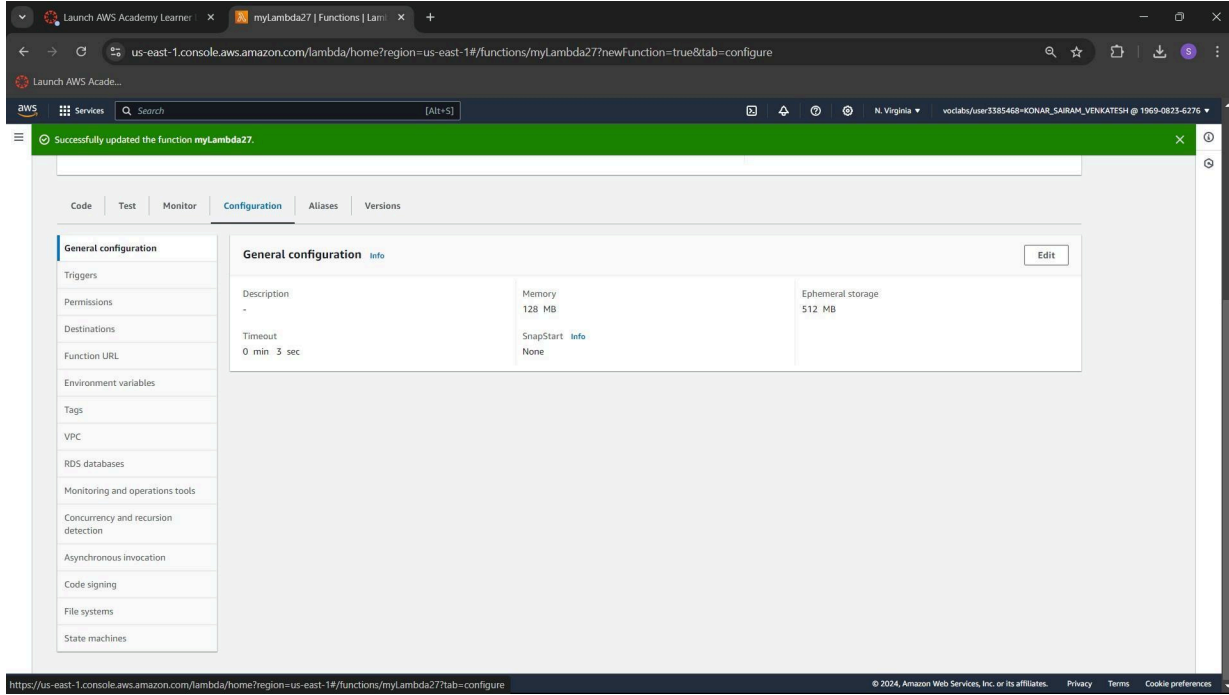


6) This function has the following default code, which is used to print "Hello form Lambda!".

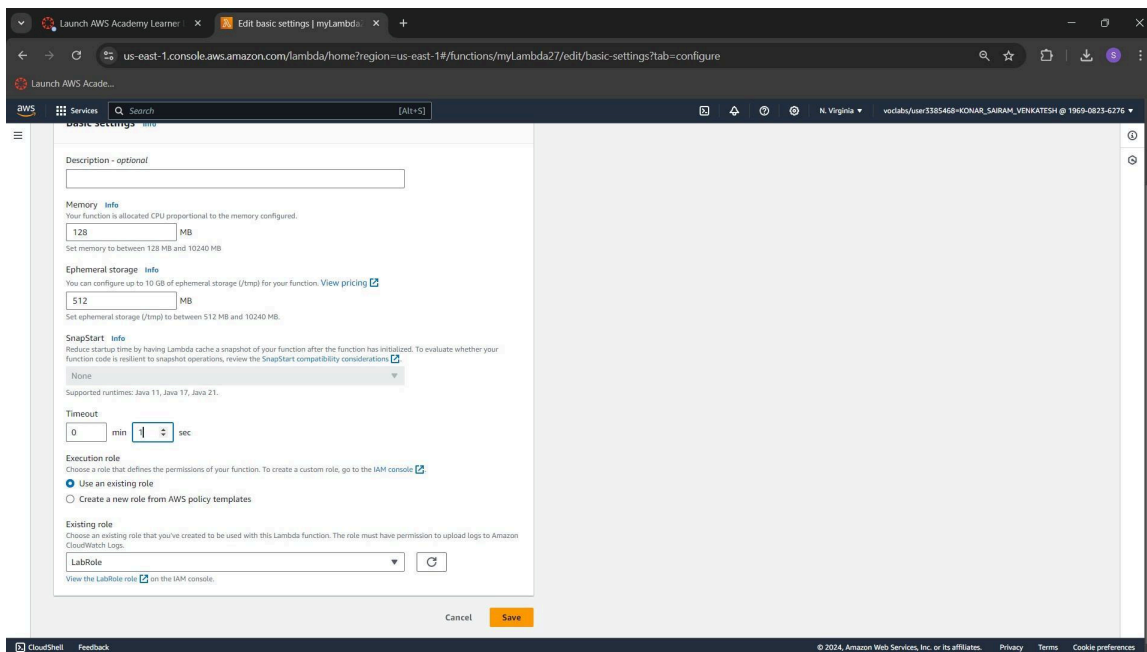


Step 3: Set up configurations and test events

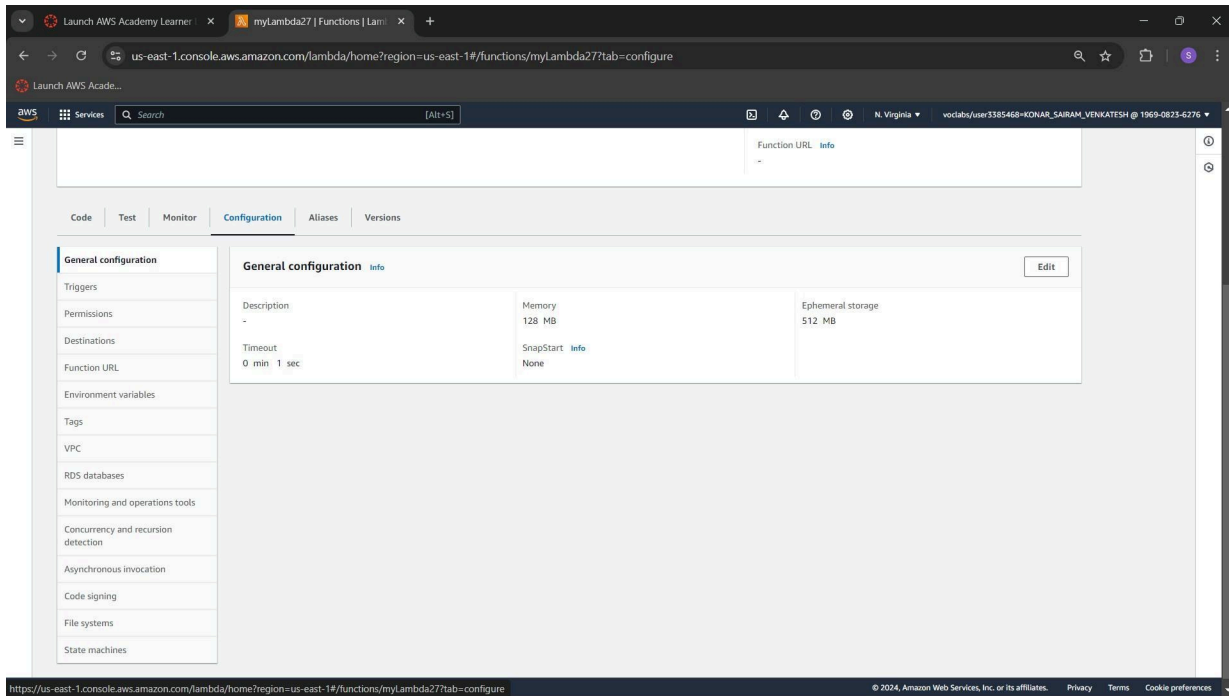
1) Just above the test code, you would find Configuration, click on it. Then click on Edit.



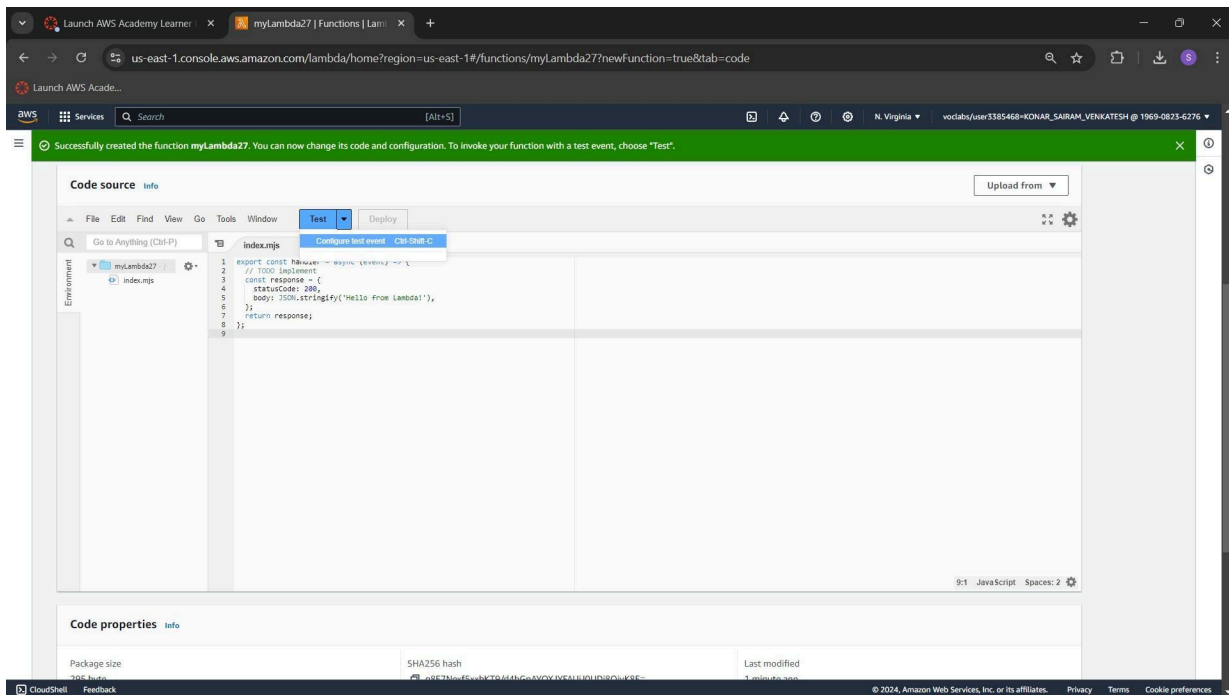
2) Here, change the Timeout to 1 sec. This is the time for which the function can be running before it is forcibly terminated.



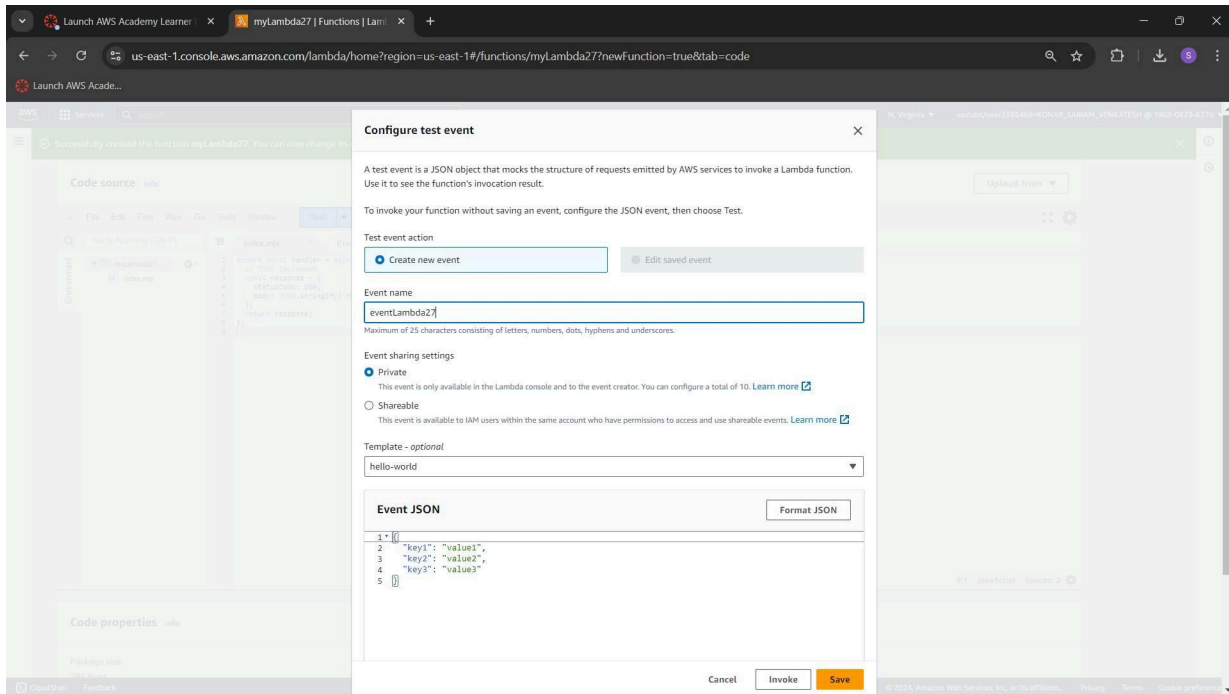
3) We can see the executed changes.



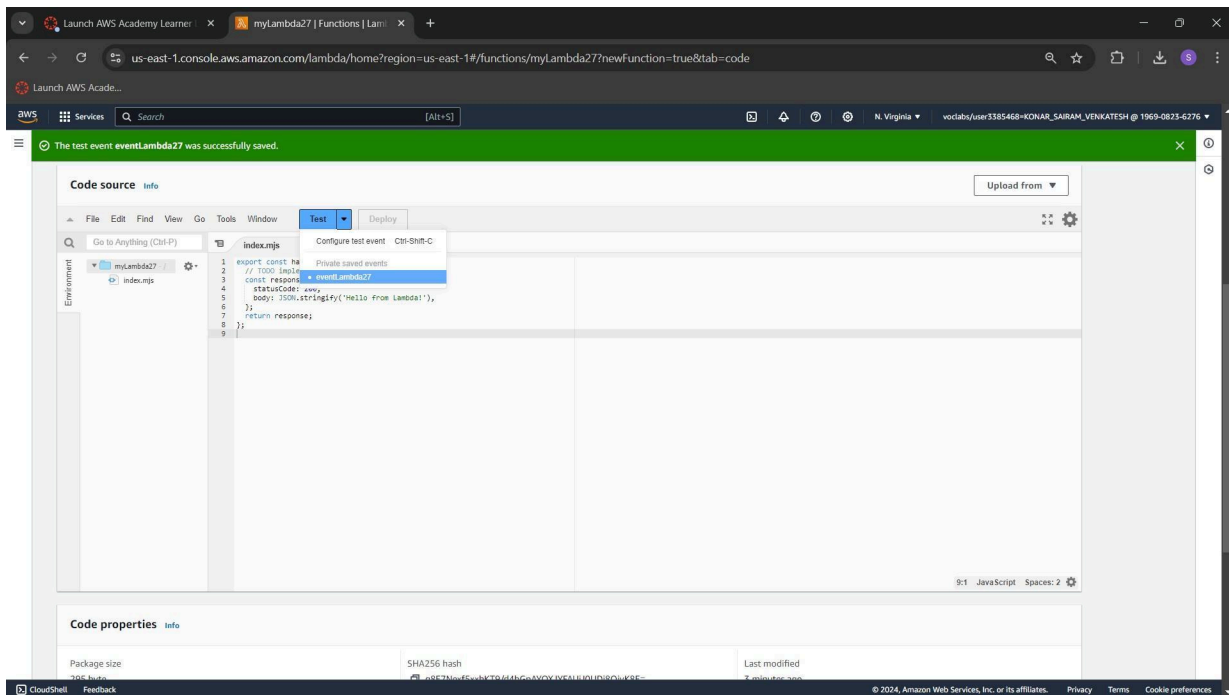
4) Switch back to the code tab. Click on the dropdown arrow near test. Then select configure test event.



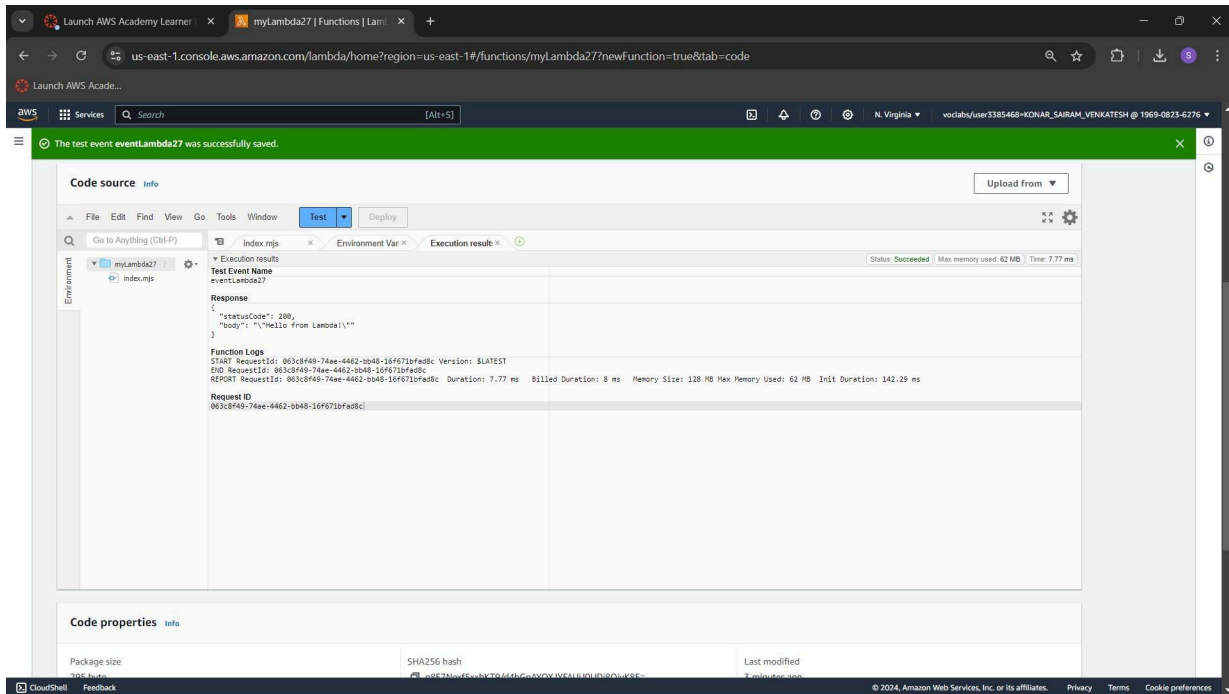
5) Here, create a new event, keep the other options default and save the event.



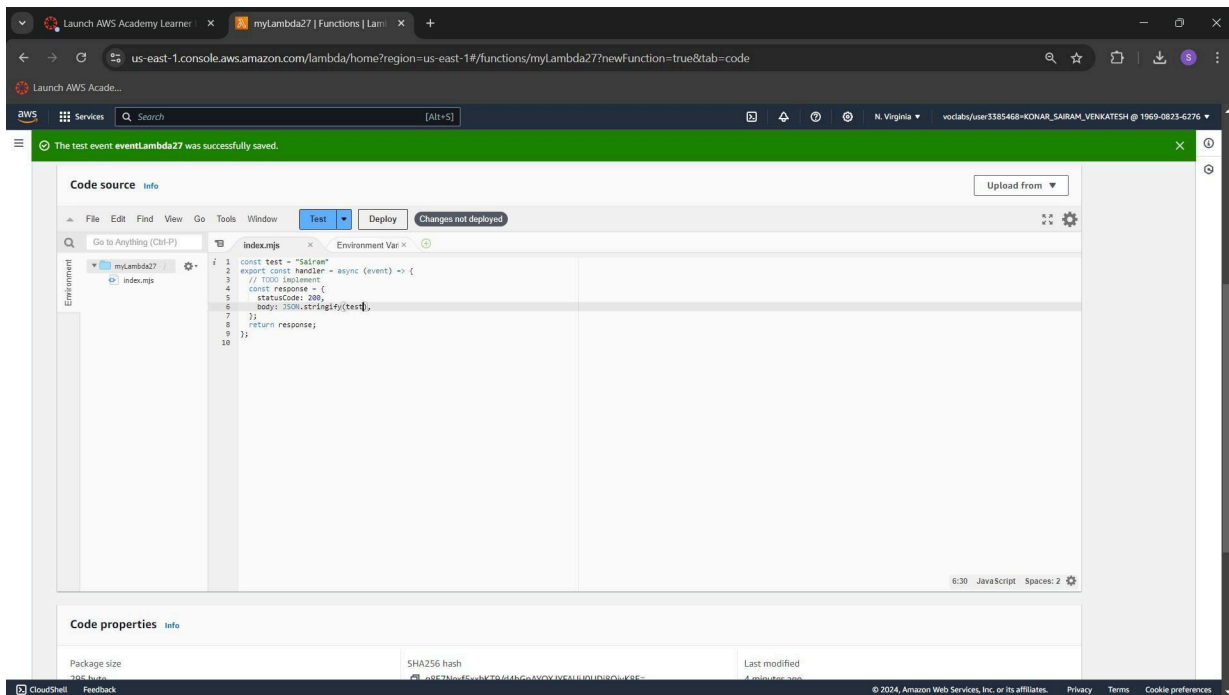
6) Now, again click on the dropdown. This time, select the event you have created. Then, click on TEST.



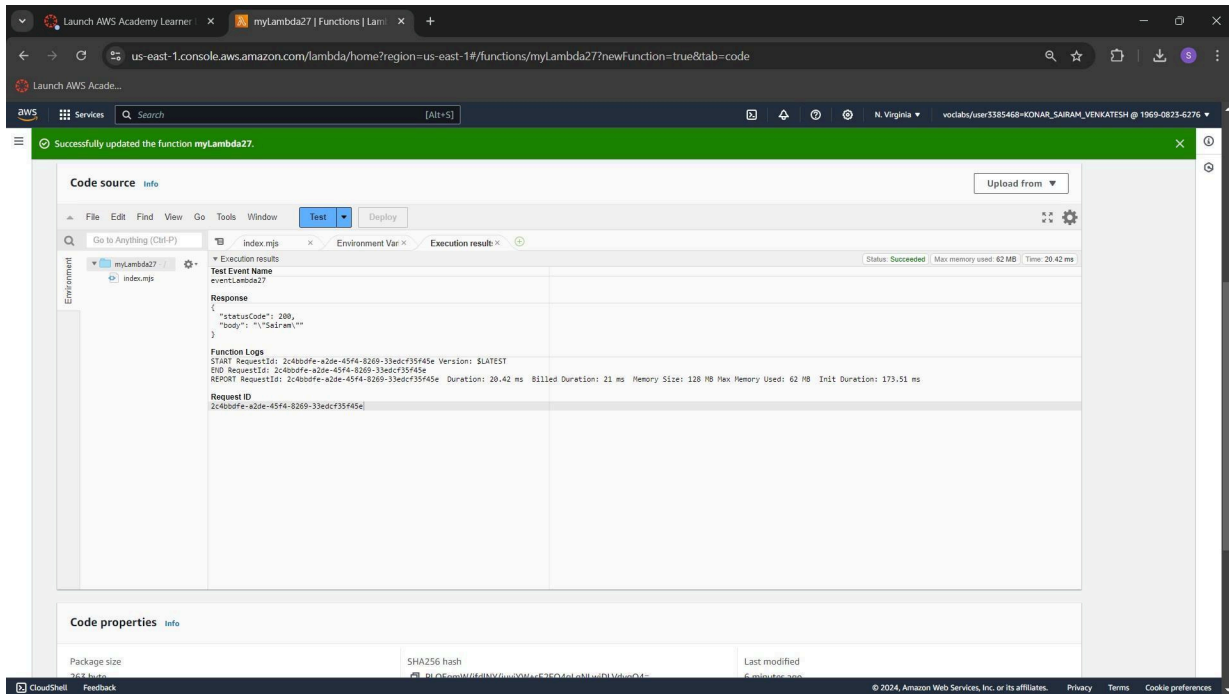
7) We can see the expected output for the sample code.



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



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



Conclusion:

In this experiment, we successfully explored the AWS Lambda service by creating and configuring Lambda functions using Python, Java, or Node.js. We learned how to set up a Lambda function, modify its configuration (such as adjusting the timeout), and test the function with custom events. Through this process, we observed how Lambda handles executions, including managing timeouts and returning expected outputs based on the code changes.