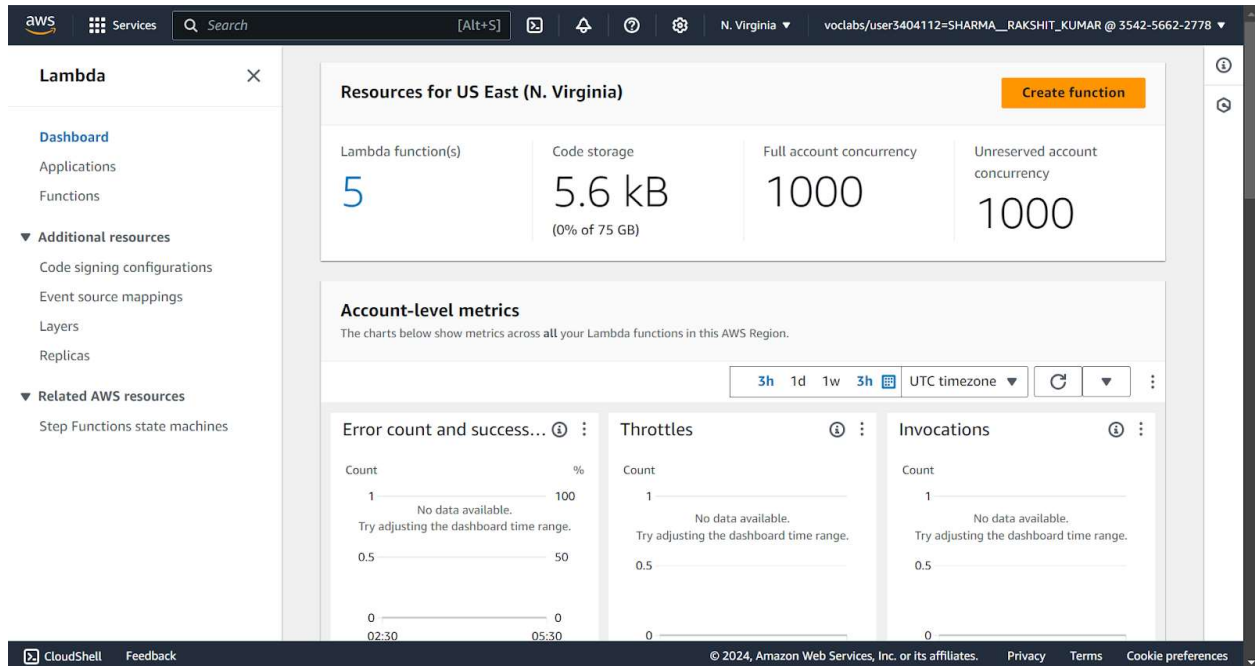


**Aim:** To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.

1. Log into your AWS academy account and search for Lambda service. Click on create function to create a new lambda function.



2. Give a name to your lambda function. Select the language you want to use to write the functions. We will use Python 3.12, Architecture x86. Select Execution role to Create a new role with basic Lambda permissions.

The screenshot shows the 'Create function' wizard. It starts with three options: 'Author from scratch' (selected), 'Use a blueprint', and 'Container image'. The 'Basic information' section is visible, showing the 'Function name' field with 'Rakshit\_Lambda' entered. Below this, the 'Runtime' dropdown is set to 'Python 3.12'. The 'Architecture' dropdown is set to 'x86\_64'. The 'Execution role' section is partially visible at the bottom.

You will be able to see the created function in Functions tab

The image shows two screenshots related to an AWS Lambda function named 'Rakshit\_Lambda'.

The top screenshot is from the AWS Lambda console. It displays the 'Function overview' tab for 'Rakshit\_Lambda'. The function is in the 'Diagram' view, showing a single step 'Rakshit\_Lambda' with 0 layers. The 'Description' tab shows the function's details: Description is '-', Last modified is 'in 1 second', Function ARN is 'arn:aws:lambda:us-east-1:354256622778:function:Rakshit\_Lambda', and Function URL is 'Info'. The console also has buttons for 'Throttle', 'Copy ARN', 'Actions', 'Export to Application Composer', and 'Download'.

The bottom screenshot is from the VS Code editor, showing the 'Test' tab. The code editor displays the following Python code for the 'lambda\_function.py' file:

```
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     return {
6         'statusCode': 200,
7         'body': json.dumps('Hello from Lambda!')
8     }
9
```

3. Scroll down and go to the configuration tab. In General configuration click on edit to change the configuration.

The screenshot shows the AWS Lambda console's Configuration tab. On the left is a sidebar with links: General configuration, Triggers, Permissions, Destinations, Function URL, Environment variables, Tags, VPC, and RDS databases. The main area is titled 'General configuration' with an 'Info' link and an 'Edit' button. It contains a table with the following details:

Description	Memory	Ephemeral storage
-	128 MB	512 MB
Timeout	SnapStart	
0 min 3 sec	None	

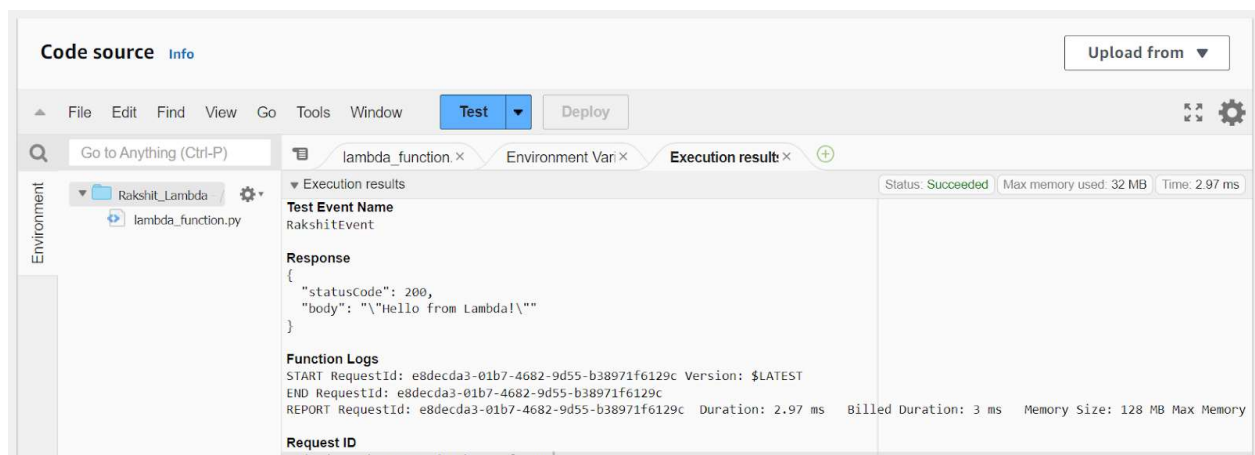
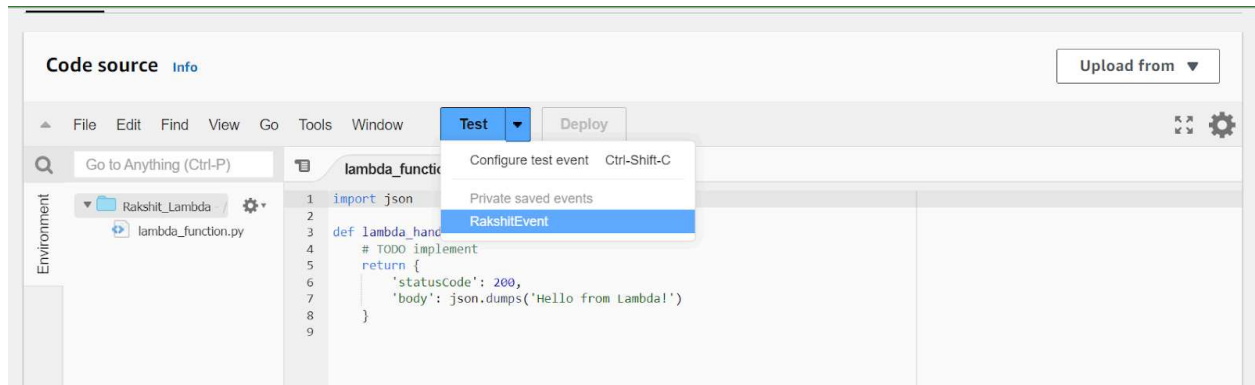
Here, you can enter a description and change Memory and Timeout. I've changed the Timeout period to 1 sec since that is sufficient for now

4. Now go to the test tab and select Create new event. Give the event an appropriate name, keep the sharing settings as private and template as hello world.

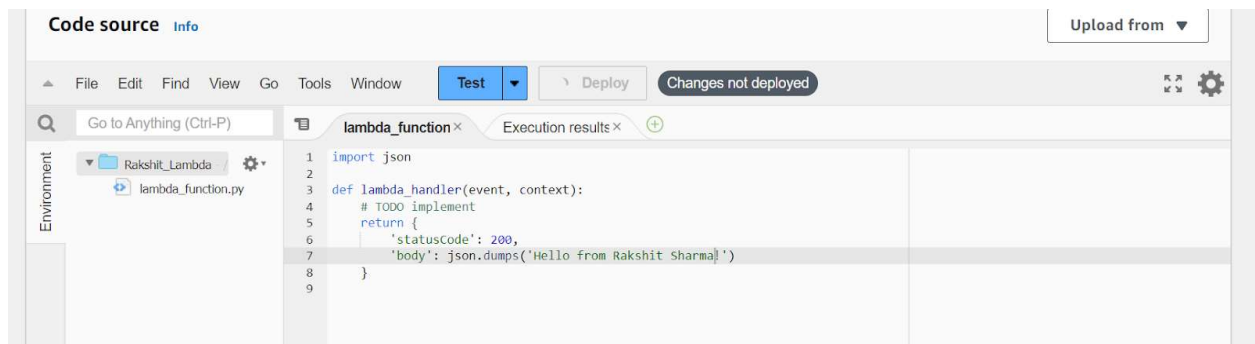
The screenshot shows the AWS Lambda console's Test tab. At the top, it says: 'To invoke your function without saving an event, configure the JSON event, then choose Test.' Below this are two buttons: 'Create new event' (selected) and 'Edit saved event'. The 'Event name' field contains 'RakshitEvent' with a note: 'Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.' Under 'Event sharing settings', 'Private' is selected with a note: 'This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)'. 'Shareable' is also an option with a note: 'This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)'. The 'Template - optional' dropdown is set to 'hello-world'. At the bottom, the 'Event JSON' section has a 'Format JSON' button and a text area with the following JSON:

```
1 {
2   "key1": "value1",
3   "key2": "value2"
}
```

5. Now In Code section select the created event from the dropdown of test then click on test . Observe the output.

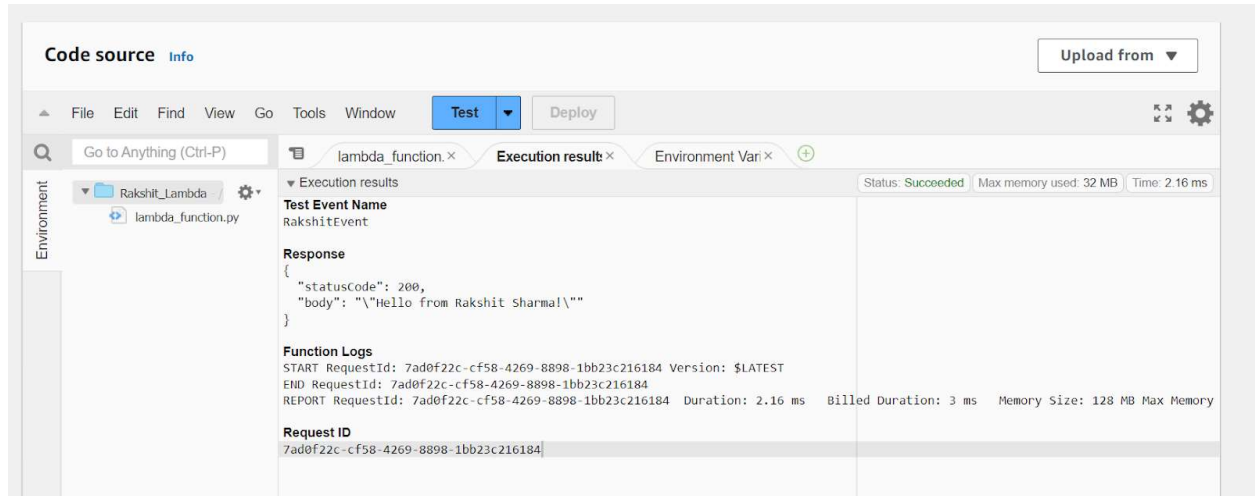


6. We can also edit the lambda function code. I have added my name in the print statement.



To save the changes click on deploy.  
Check the output displayed.

And again select the event and click on Test.



## Conclusion:

In this experiment, we explored AWS Lambda, created a basic Lambda function, and tested it using Python 3.12. We walked through creating a Lambda function, configuring its settings, and testing it with a predefined event. We also demonstrated how to modify the function's code, deployed the changes, and observed the output. This experiment gave us insight into AWS Lambda's workflow, showing how serverless functions can be easily created, tested, and deployed, making it a powerful tool for event-driven applications.