

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

### **Theory:**

#### **AWS Lambda**

A fully managed, serverless computing service where you run code without provisioning or managing servers. Lambda automatically scales your application based on the number of incoming requests or events, ensuring efficient resource utilization. You are only charged for the time your code is running, with no upfront cost, making it cost-effective for on-demand workloads.

#### **Lambda Workflow**

- **Create a Function:** Write the function code and define its handler (entry point). You can use the AWS Console, CLI, or upload a deployment package.
- **Set Event Sources:** Define how the function is triggered (e.g., when an object is uploaded to S3 or a DynamoDB table is updated).
- **Execution:** When triggered, Lambda runs your function, executes the logic, and automatically scales to handle the incoming event volume.
- **Scaling and Concurrency:** Lambda scales automatically by launching more instances of the function to handle simultaneous invocations. There are also options for configuring **reserved concurrency** to manage traffic.
- **Monitoring and Logging:** Lambda integrates with Amazon CloudWatch for logging and monitoring. Logs for each invocation are sent to CloudWatch, allowing you to track performance and troubleshoot errors.

#### **AWS Lambda Functions**

- **Python:** Great for quick development with its rich standard library and support for lightweight tasks.
- **Java:** Typically used for more complex, compute-intensive tasks. While it's robust, cold start times can be higher.
- **Node.js:** Excellent for I/O-bound tasks like handling APIs or streaming data, with fast startup times and efficient memory usage.

**Prerequisites:** AWS Personal/Academy Account

Name: Bhushan Mukund Kor

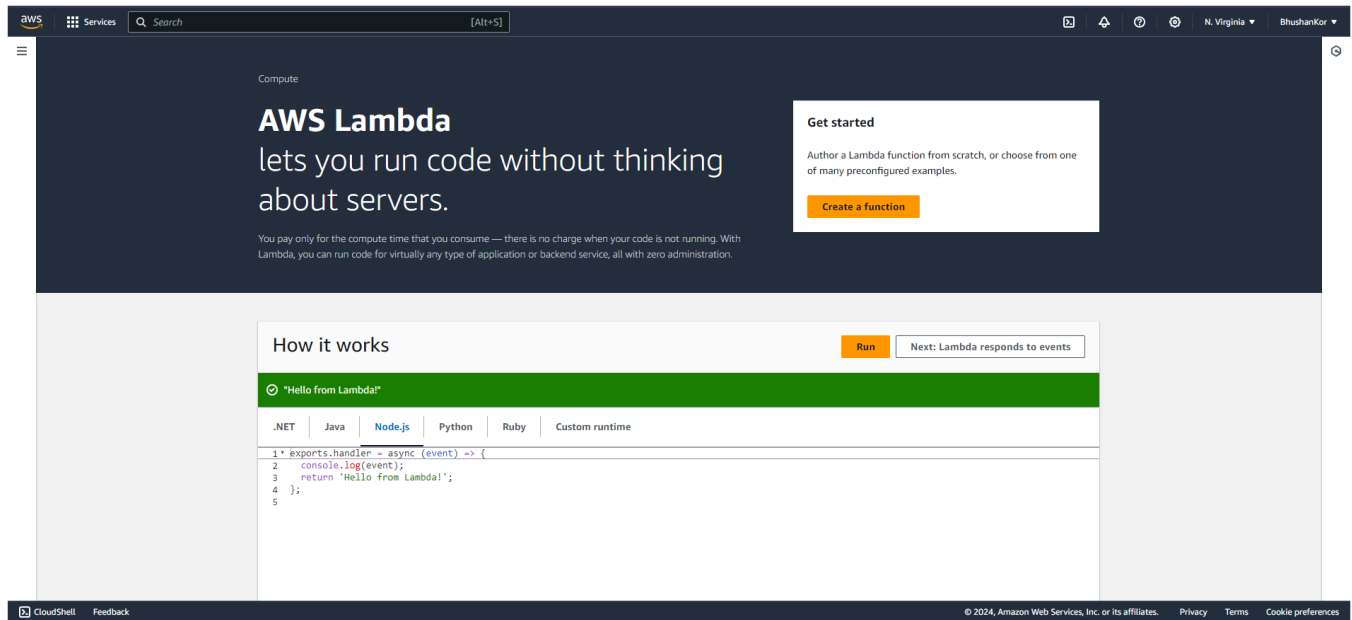
Academic Year: 2024-2025

Division: D15C

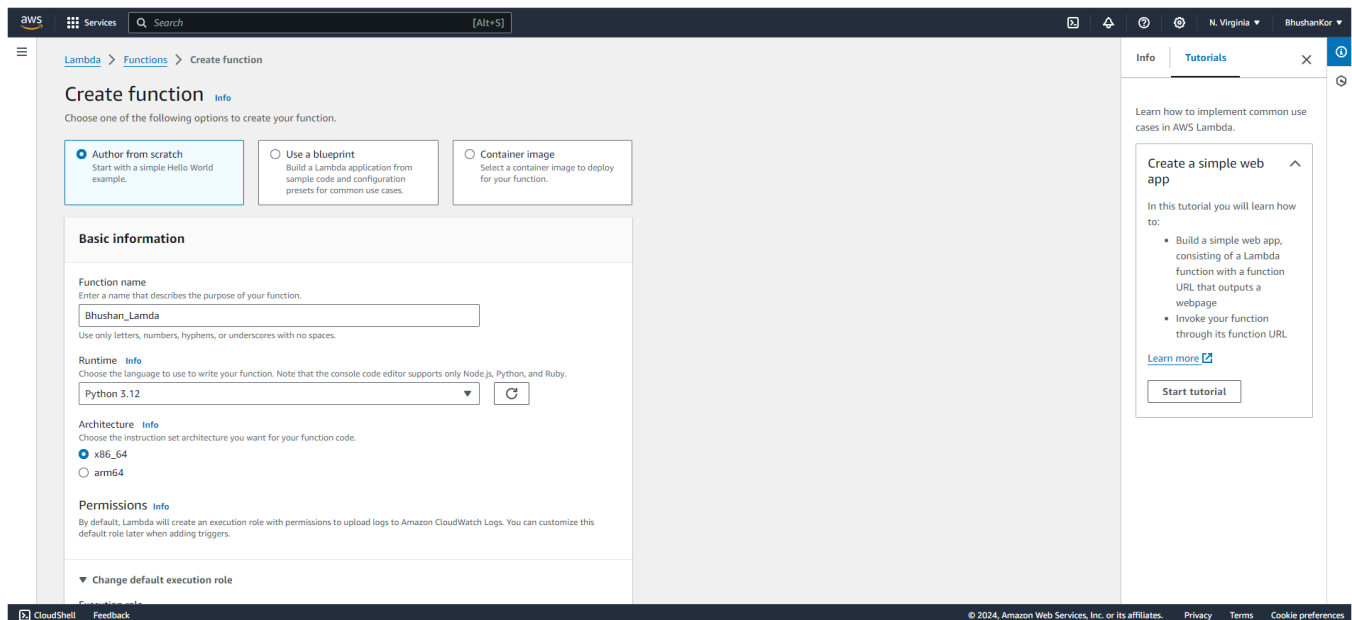
Roll No: 28

## Steps To create the lambda function:

**Step 1:** Login to your AWS Personal/Academy Account. Open lambda and click on create function button.



**Step 2:** Now Give a name to your Lambda function, Select the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby. So will select Python 3.12, Architecture as x86, and Execution role to Create a new role with basic Lambda permissions.



**Academic Year:2024-2025**

**Roll No: 28**

The image is a vertical stack of three screenshots from the AWS Lambda console, illustrating the process of creating and configuring a new function.   
  
The top screenshot shows the 'Create function' wizard. The 'Architecture' is set to 'x86\_64'. Under 'Permissions', the option 'Create a new role with basic Lambda permissions' is selected. A note states: 'Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.' The 'Create function' button is highlighted in orange.   
  
The middle screenshot shows the 'Function overview' for 'Bhushan\_Lambda'. It includes a diagram with a single layer 'Bhushan\_Lambda'. The 'Description' tab is active, showing details like 'Last modified 2 seconds ago' and 'Function ARN: arn:aws:lambda:us-east-1:1010928205712:function:Bhushan\_Lambda'.   
  
The bottom screenshot shows the 'Code source' tab. The code is a Python lambda function: 

```
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 
```

 The 'Test' button is visible above the code editor.   
  
On the right side of each screenshot, there is a 'Tutorials' sidebar. It contains a section titled 'Create a simple web app' with the text: 'In this tutorial you will learn how to: Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage. Invoke your function through its function URL.' It includes a 'Start tutorial' button and a 'Learn more' link.

Name: Bhushan Mukund Kor

Academic Year: 2024-2025

Division: D15C

Roll No: 28

So See or Edit the basic settings go to configuration then click on edit general setting.

General configuration		
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.

**Edit basic settings**

**Basic settings**

Description - optional

Basic Settings

**Memory**

Your function is allocated CPU proportional to the memory configured.

128 MB

Set memory to between 128 MB and 10240 MB

**Ephemeral storage**

You can configure up to 10 GB of ephemeral storage (/tmp) for your function. [View pricing](#)

512 MB

Set ephemeral storage (/tmp) to between 512 MB and 10240 MB.

**SnapStart**

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](#)

None

Supported runtimes: Java 11, Java 17, Java 21.

**Timeout**

0 min 1 sec

**Execution role**

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#)

☒ Use an existing role

☐ Create a new role from AWS policy templates

**Tutorials**

Learn how to implement common use cases in AWS Lambda.

**Create a simple web app**

In this tutorial you will learn how to:

- Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
- Invoke your function through its function URL.

[Learn more](#)

[Start tutorial](#)

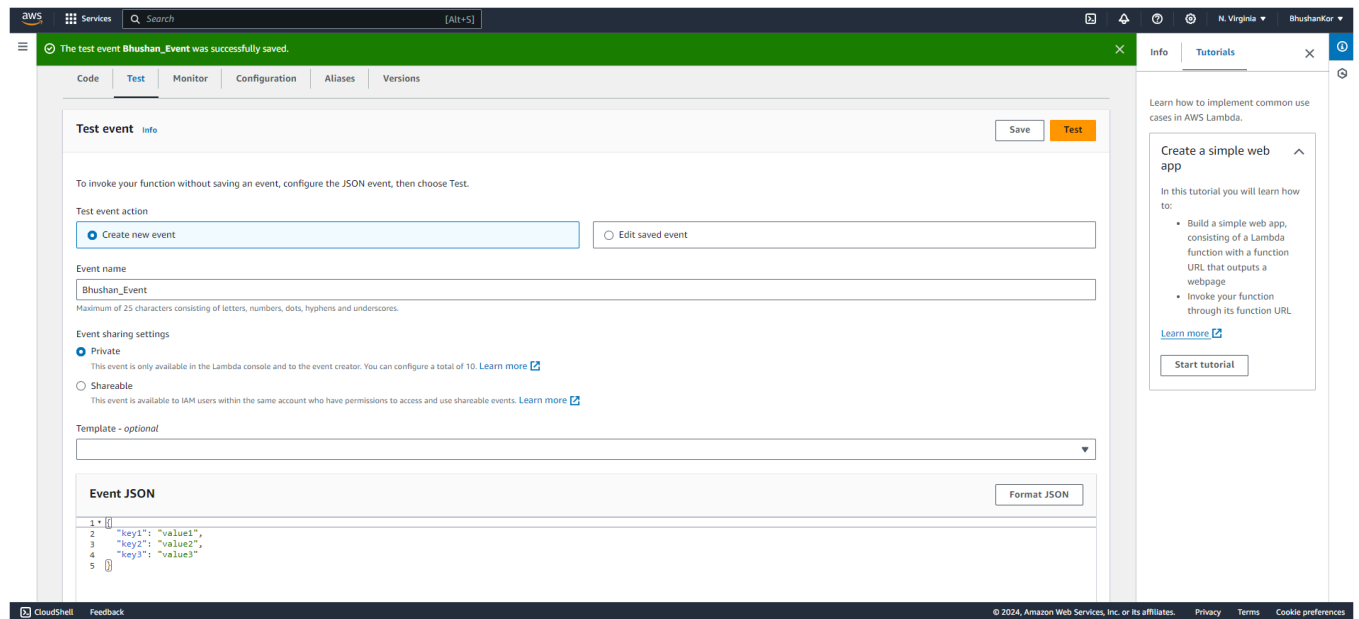
Name: Bhushan Mukund Kor

Academic Year: 2024-2025

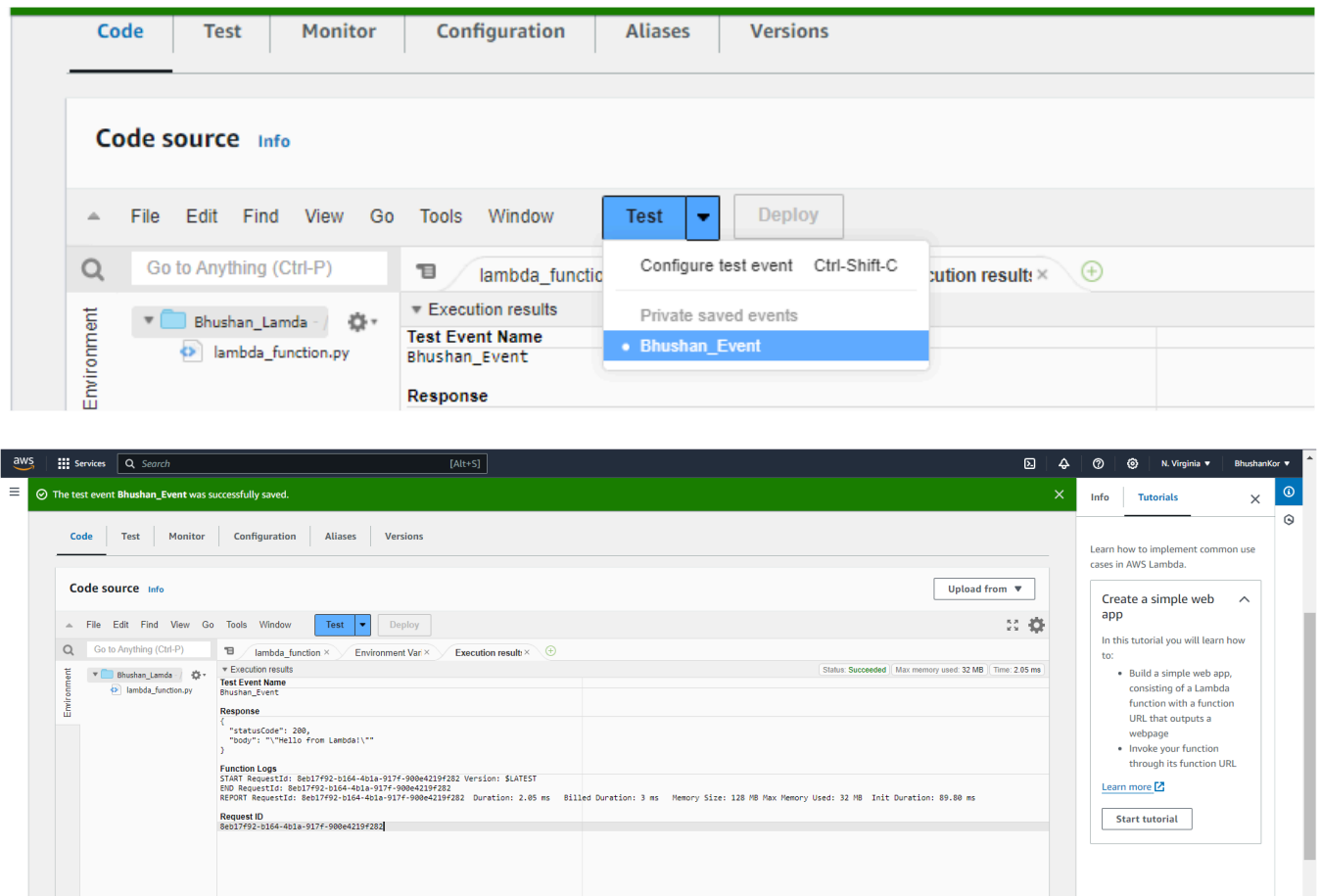
Division: D15C

Roll No: 28

**Step 3:** Now Click on the Test tab then select Create a new event, give a name to the event and select Event Sharing to private, and select hello-world template.



**Step 4:** Now In Code section select the created event from the dropdown of test then click on test . You will see the below output.



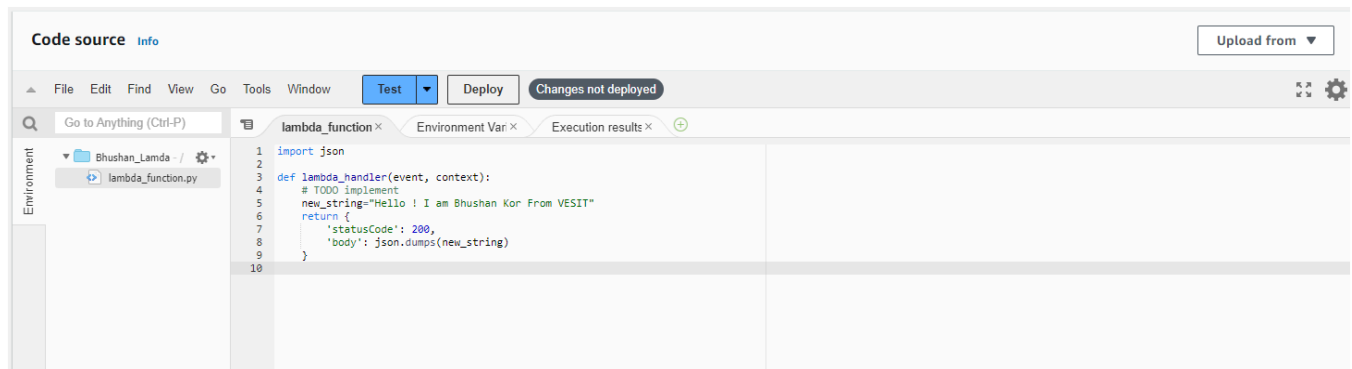
Name: Bhushan Mukund Kor

Academic Year: 2024-2025

Division: D15C

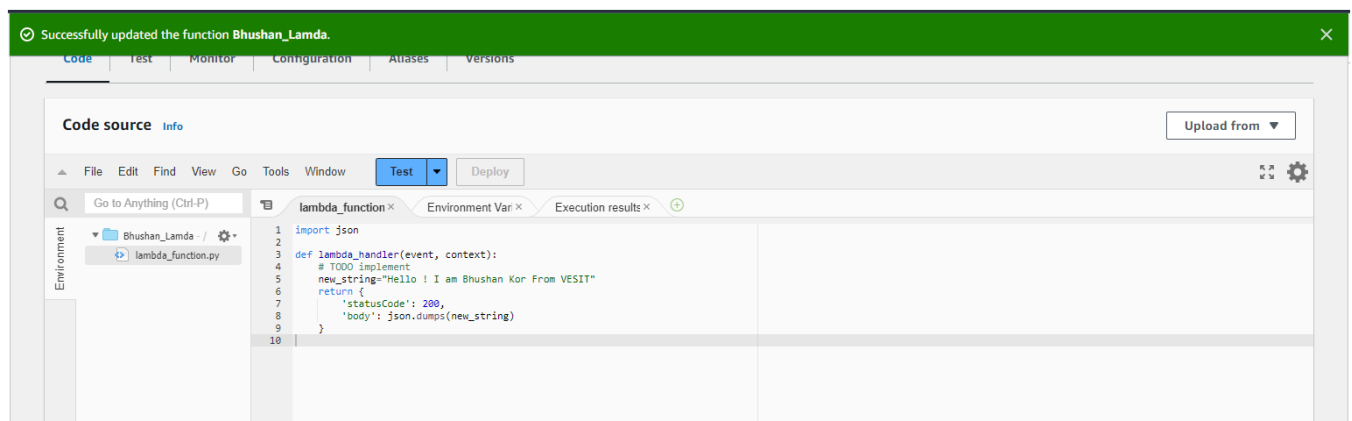
Roll No: 28

**Step 5:** You can edit your lambda function code. I have changed the code to display the new String.

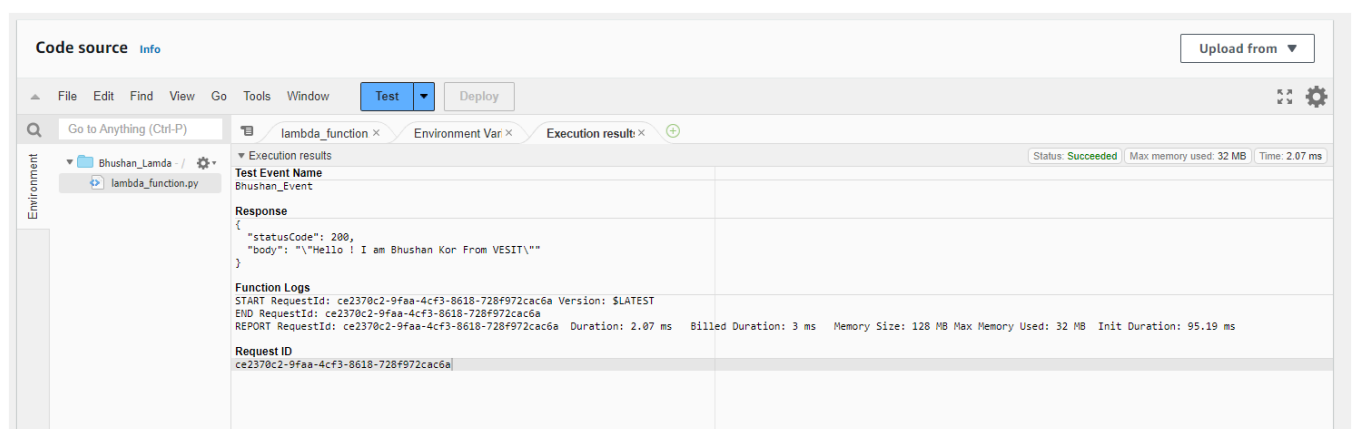


```
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     new_string="Hello ! I am Bhushan Kor From VESIT"
6     return {
7         'statusCode': 200,
8         'body': json.dumps(new_string)
9     }
10
```

Now ctrl+s to save and click on deploy to deploy the changes.



**Step 6:** Now click on the test and observe the output. We can see the status code 200 and your string output and function logs. On successful deployment.



Test Event Name	Response	Function Logs
Bhushan_Event	<pre>{   "statusCode": 200,   "body": "\"Hello ! I am Bhushan Kor From VESIT\"" }</pre>	<pre>START RequestId: ce2370c2-9faa-4cf3-8618-728f972cac6a Version: \$LATEST END RequestId: ce2370c2-9faa-4cf3-8618-728f972cac6a REPORT RequestId: ce2370c2-9faa-4cf3-8618-728f972cac6a  Duration: 2.07 ms   Billed Duration: 3 ms   Memory Size: 128 MB Max Memory Used: 32 MB Init Duration: 95.19 ms</pre>

**Conclusion:** In this experiment, we successfully created an AWS Lambda function and walked through its essential steps. After setting up the function with Python, we configured the basic settings, including adjusting the timeout to 1 second. We then created a test event, deployed the function, and validated the output. Additionally, we modified the Lambda function's code and redeployed it to observe the changes in real-time.

This practical experience demonstrated the simplicity and flexibility of AWS Lambda in creating serverless applications, allowing you to focus on code while AWS manages the infrastructure and scaling.