



# Vivekanand Education Society's

## Institute of Technology

An Autonomous Institute Affiliated to University of Mumbai,, Approved by AICTE & Recognized by Govt. of Maharashtra  
Hashu Advani Memorial Complex, Collector Colony, Chembur East, Mumbai - 400074.

### Department of Information Technology

A.Y. 2024-25

## Advance DevOps Lab

### Experiment 11

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

|           |  |
|-----------|--|
| Roll No.  | 53   |
| Name      | Aryan Deepak Saraf   |
| Class     | D15B   |
| Subject   | Advance DevOps Lab   |
| LO Mapped | LO1: To understand the fundamentals of Cloud Computing and be fully proficient with Cloud based DevOps solution deployment options to meet your business requirements.<br><br>LO6: To engineer a composition of nano services using AWS Lambda and Step Functions with the Serverless Framework. |
| Grade:    |  |

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

## **THEORY :**

**AWS Lambda** is a serverless computing service by AWS that allows you to run code without provisioning or managing servers. You create functions in supported languages like Python, Java, and Node.js, and these functions are executed in response to specific events such as API calls, file uploads to S3, or data changes in DynamoDB.

### **Key Features**

- **Automatic Scaling:** Lambda automatically scales the infrastructure to handle incoming requests, reducing operational complexity.
- **Cost-Efficiency:** You only pay for the compute time you consume, with no upfront costs or server management fees.
- **Security:** Lambda integrates with AWS Identity and Access Management (IAM) to define roles and policies, ensuring secure execution.
- **Fault Tolerance:** AWS Lambda is designed to provide high availability and fault tolerance, handling server failures and maintaining continuous operation.

### **Execution Model**

Lambda functions run in stateless containers fully managed by AWS. When an event triggers a function, AWS initiates a container to execute the function. If subsequent requests come in, additional containers are spun up to handle them. AWS may keep containers warm for a short period to reduce cold start latency.

### **Stateless Functions**

Due to the stateless nature of Lambda, each function invocation is independent, running in a fresh environment. Code outside the main handler function runs once per container lifecycle, while the handler itself runs on every invocation.

### **Common Use Cases**

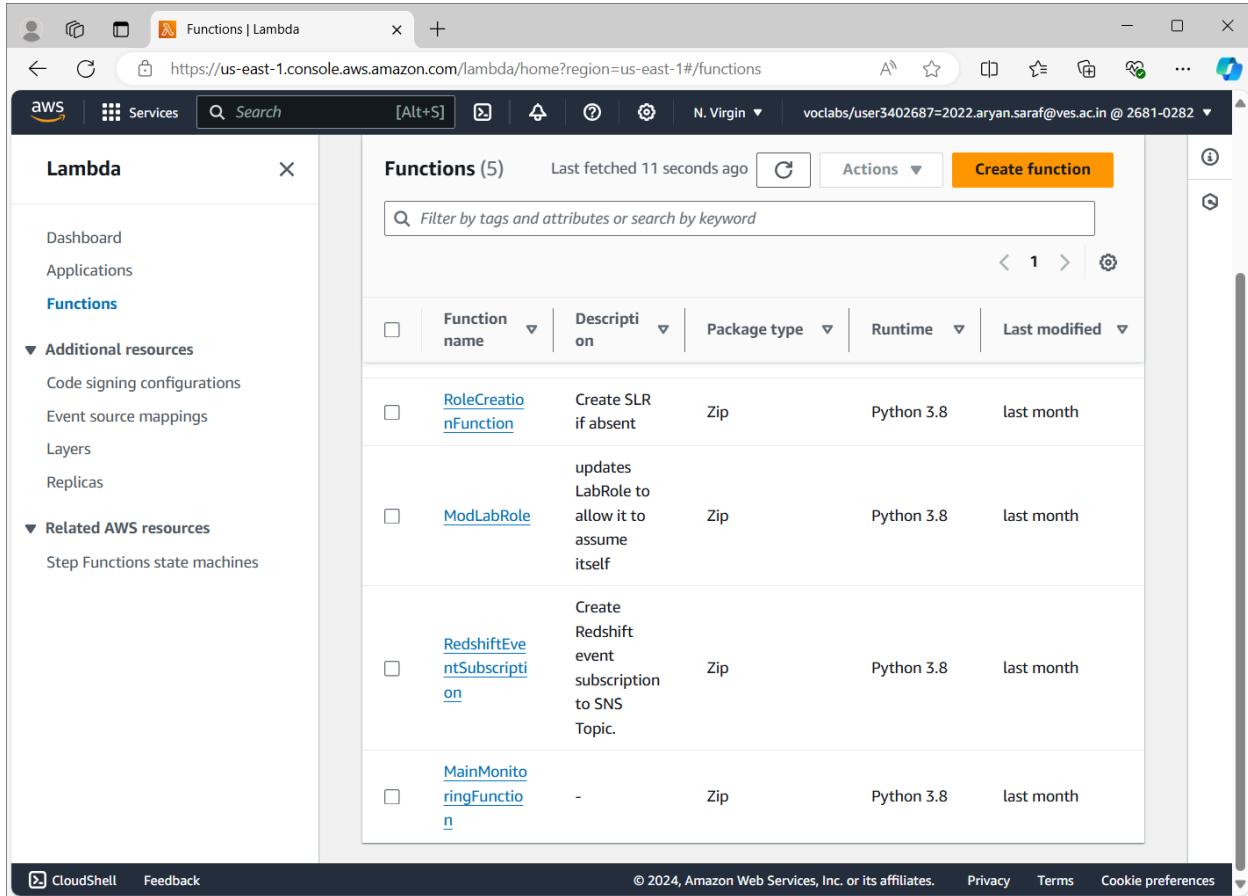
- **Scalable APIs:** Lambda is ideal for building APIs that need to scale according to demand. Each API request can be routed to a specific Lambda function, and the service automatically adjusts to handle varying workloads.
- **Event-Driven Data Processing:** Lambda excels in scenarios like real-time data processing, where functions are triggered by events from sources like S3 or DynamoDB, making it suitable for tasks like data transformation, analytics, and notifications.

### **Packaging and Deployment**

Lambda functions, along with their dependencies, are packaged and uploaded to AWS, often using an S3 bucket. AWS Lambda then uses this package to execute the function when an event occurs. Tools like the Serverless Stack Framework (SST) can simplify this process.

## Steps to create an AWS Lambda function

Open up the Lambda Console and click on the Create button.



The screenshot shows the AWS Lambda console interface. The left sidebar contains navigation links: Dashboard, Applications, Functions (selected), Additional resources (Code signing configurations, Event source mappings, Layers, Replicas), and Related AWS resources (Step Functions state machines). The main content area displays a list of functions under the heading 'Functions (5)'. A search bar is present with the placeholder text 'Filter by tags and attributes or search by keyword'. The function list has columns for checkboxes, Function name, Description, Package type, Runtime, and Last modified. The functions listed are:

|                          | Function name                             | Description                                      | Package type | Runtime    | Last modified |
|--------------------------|---|--|--------------|------------|---------------|
| <input type="checkbox"/> | <a href="#">RoleCreationFunction</a>      | Create SLR if absent                             | Zip          | Python 3.8 | last month    |
| <input type="checkbox"/> | <a href="#">ModLabRole</a>                | updates LabRole to allow it to assume itself     | Zip          | Python 3.8 | last month    |
| <input type="checkbox"/> | <a href="#">RedshiftEventSubscription</a> | Create Redshift event subscription to SNS Topic. | Zip          | Python 3.8 | last month    |
| <input type="checkbox"/> | <a href="#">MainMonitoringFunction</a>    | -  | Zip          | Python 3.8 | last month    |

The bottom of the console shows the footer with 'CloudShell', 'Feedback', and copyright information for Amazon Web Services, Inc. or its affiliates, along with links for Privacy, Terms, and Cookie preferences.

Choose to create a function from scratch or use a blueprint, i.e templates defined by AWS for you with all configuration presets required for the most common use cases.

**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.  
  
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.  
 [Refresh](#)

**Architecture** [Info](#)  
Choose the instruction set architecture you want for your function code.  
☒ x86\_64  
☐ arm64

CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

**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

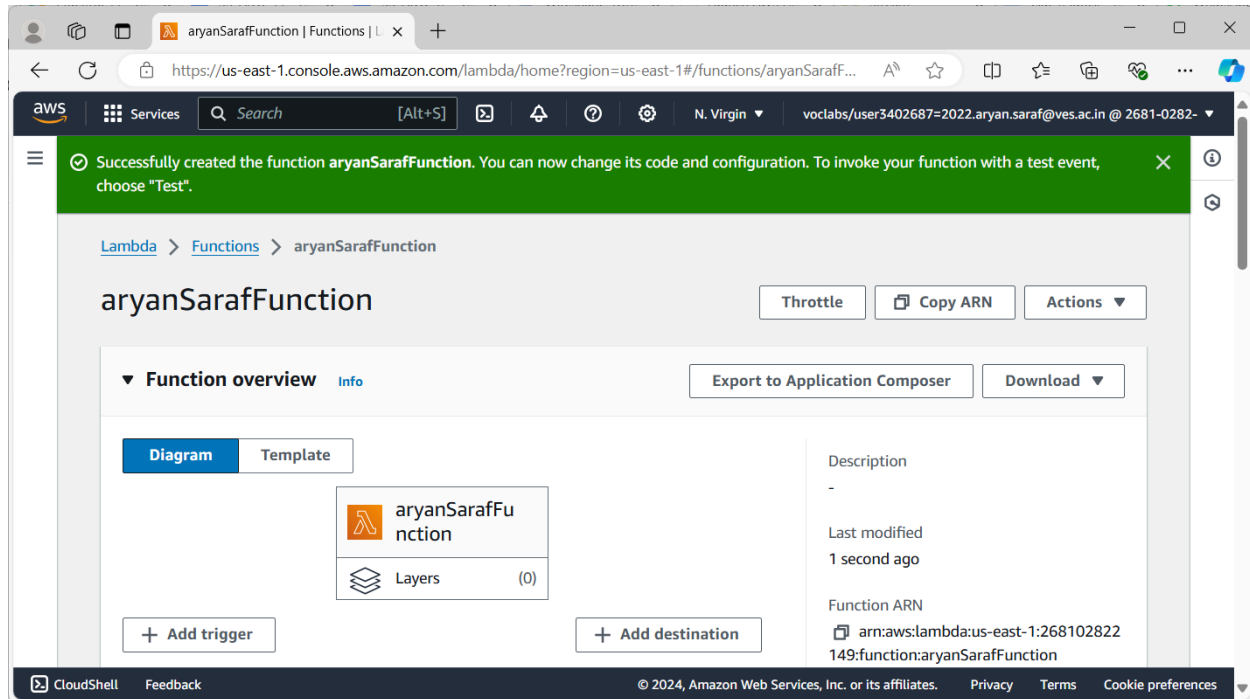
**Existing role**  
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.  
 [Refresh](#)  
[View the LabRole role](#) on the IAM console.

**Advanced settings**

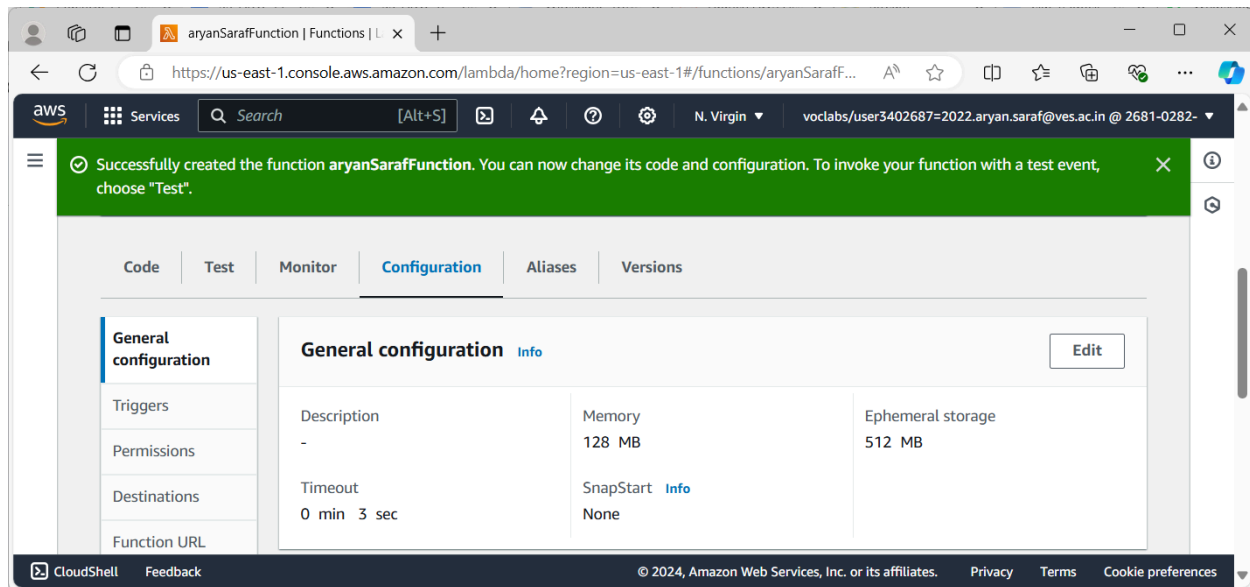
Cancel **Create function**

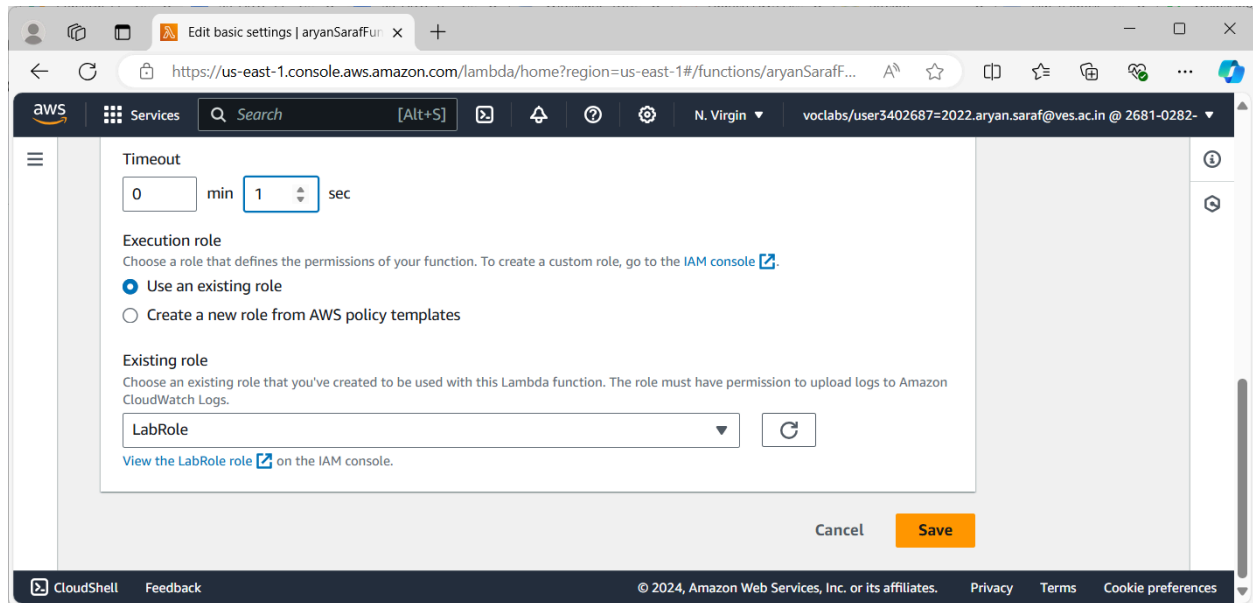
CloudShell Feedback © 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

This process will take a while to finish and after that, you'll get a message that your function was successfully created.

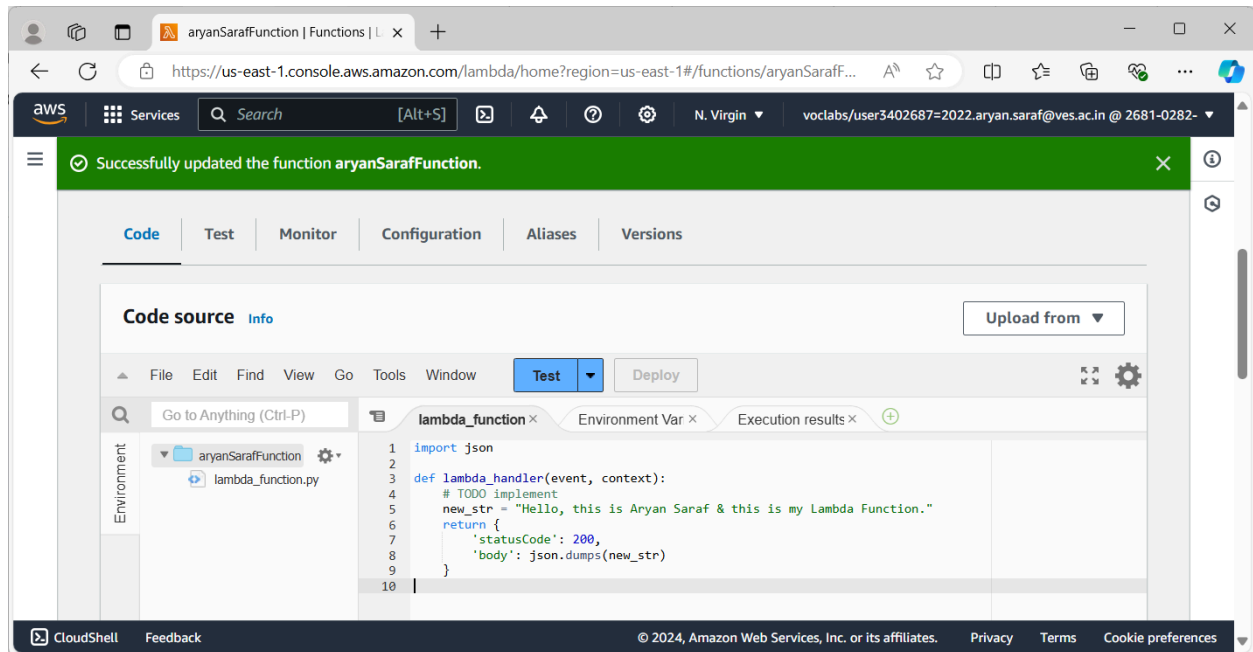


To change the configuration, open up the Configuration tab and under General Configuration, choose Edit.





Press Ctrl + S to save the file and click Deploy to deploy the changes.



Click on Test and you can change the configuration, like so.

The screenshot shows the 'Configure test event' dialog in the AWS Lambda console. The dialog has a title bar and a close button. It contains the following sections:

- Test event action:** Two buttons, 'Create new event' (selected) and 'Edit saved event'.
- Event name:** A text input field containing 'SarafEventet'. Below it, a note states: 'Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.'
- Event sharing settings:** Two radio buttons. '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 unselected, 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](#)'.
- Template - optional:** A dropdown menu showing 'hello-world'.
- Event JSON:** A text area with a 'Format JSON' button. The JSON content is: 

```
{ 1: { 2: { "key1": "value1",
```
- Buttons:** 'Cancel', 'Invoke', and 'Save' at the bottom right.

Now click on Test and you should be able to see the results.

The screenshot shows the AWS Lambda console with the 'Test' tab selected. A green banner at the top indicates 'Successfully updated the function aryanSaraFunction.' The 'Code source' tab is active, showing the 'lambda\_function.py' file. The 'Test' button is highlighted. The 'Execution result' tab is open, displaying the following information:

- Execution results:** Status: Succeeded | Max memory used: 30 MB | Time: 2.38 ms
- Test Event Name:** SarafEventet
- Response:**

```
{  "statusCode": 200,  "body": "\"Hello, this is Aryan Saraf & this is my Lambda Function.\""} 
```
- Function Logs:**

```
START RequestId: 3c2e927f-441e-43aa-abb5-afa9da31531d Version: $LATEST  END RequestId: 3c2e927f-441e-43aa-abb5-afa9da31531d  REPORT RequestId: 3c2e927f-441e-43aa-abb5-afa9da31531d Duration: 2.38 ms Billed Duration: 3 ms Memory Si
```
- Request ID:** 3c2e927f-441e-43aa-abb5-afa9da31531d

**CONCLUSION :**

AWS Lambda simplifies the process of running code in the cloud by handling server management, scaling, and security for you. Its flexibility and cost-efficiency make it an ideal choice for a wide range of applications, from scalable APIs to real-time data processing. By leveraging AWS Lambda, you can focus on writing code while AWS takes care of the rest.