



PARSHVANATH CHARITABLE TRUST'S

A. P. SHAH INSTITUTE OF TECHNOLOGY

Department of Information Technology

(NBA Accredited)



Academic Year: 2023-24

Semester: V

Class / Branch: TE IT

Subject: Advanced Devops Lab (ADL)

Subject Lab Incharge: Prof. Manjusha Kashilkar

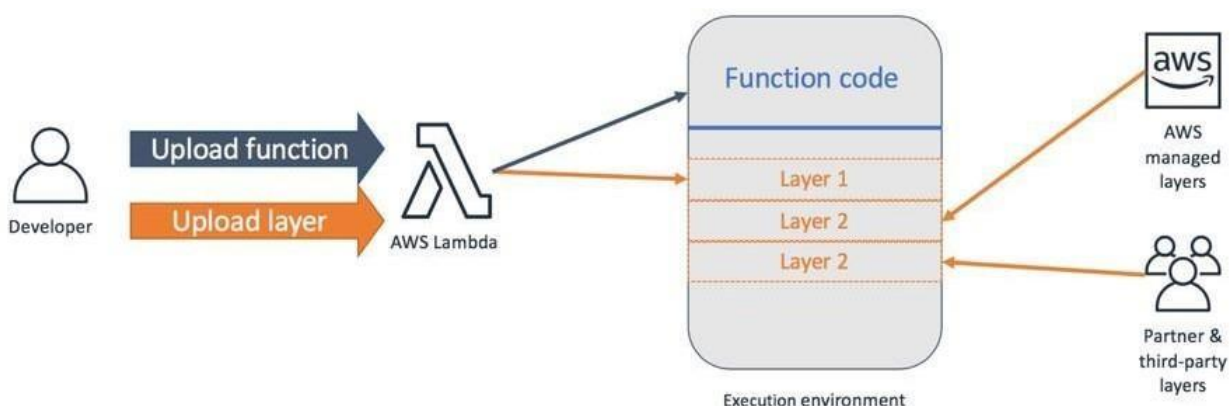
EXPERIMENT NO. 11

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

Theory:

Lambda is a compute service that lets you run code without provisioning or managing servers. Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, code monitoring and logging. With Lambda, you can run code for virtually any type of application or backend service. All you need to do is supply your code in one of the languages that Lambda supports.

You organize your code into Lambda functions. Lambda runs your function only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running.





PARSHVANATH CHARITABLE TRUST'S

A. P. SHAH INSTITUTE OF TECHNOLOGY

Department of Information Technology

(NBA Accredited)



You can invoke your Lambda functions using the Lambda API, or Lambda can run your functions in response to events from other AWS services. For example, you can use Lambda to:

- Build data-processing triggers for AWS services such as Amazon Simple Storage Service (Amazon S3) and Amazon DynamoDB.
- Process streaming data stored in Amazon Kinesis.
- Create your own backend that operates at AWS scale, performance, and security.

What is a Lambda function?

The code you run on AWS Lambda is called a “Lambda function.” After you create your Lambda function, it is always ready to run as soon as it is triggered, similar to a formula in a spreadsheet. Each function includes your code as well as some associated configuration information, including the function name and resource requirements. Lambda functions are “stateless”, with no affinity to the underlying infrastructure, so that Lambda can rapidly launch as many copies of the function as needed to scale to the rate of incoming events.

After you upload your code to AWS Lambda, you can associate your function with specific AWS resources, such as a particular Amazon S3 bucket, Amazon DynamoDB table, Amazon Kinesis stream, or Amazon SNS notification. Then, when the resource changes, Lambda will execute your function and manage the compute resources as needed to keep up with incoming requests.

AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers, creating workload-aware cluster scaling logic, maintaining event integrations, or managing runtimes. With Lambda, you can run code for virtually any type of application or backend service - all with zero administration. Just upload your code as a ZIP file or container image, and Lambda automatically and precisely allocates compute execution power and runs your code based on the incoming request or event, for any scale of traffic. You can set up your code to automatically trigger from over 200 AWS services and SaaS applications or call it directly from any web or mobile app. You can write Lambda functions in your favorite language (Node.js, Python,



PARSHVANATH CHARITABLE TRUST'S

A. P. SHAH INSTITUTE OF TECHNOLOGY

Department of Information Technology

(NBA Accredited)



Go, Java, and more) and use both serverless and container tools, such as AWS SAM or Docker CLI, to build, test, and deploy your functions.

Benefits

1. No servers to manage

AWS Lambda automatically runs your code without requiring you to provision or manage infrastructure. Just write the code and upload it to Lambda either as a ZIP file or container image.

2. Continuous scaling

AWS Lambda automatically scales your application by running code in response to each event. Your code runs in parallel and processes each trigger individually, scaling precisely with the size of the workload, from a few requests per day, to hundreds of thousands per second.

3. Cost optimized with millisecond metering

With AWS Lambda, you only pay for the compute time you consume, so you're never paying for over-provisioned infrastructure. You are charged for every millisecond your code executes and the number of times your code is triggered. With Compute Savings Plan, you can additionally save up to 17%.

4. Consistent performance at any scale

With AWS Lambda, you can optimize your code execution time by choosing the right memory size for your function. You can also keep your functions initialized and hyper-ready to respond within double digit milliseconds by enabling Provisioned Concurrency.

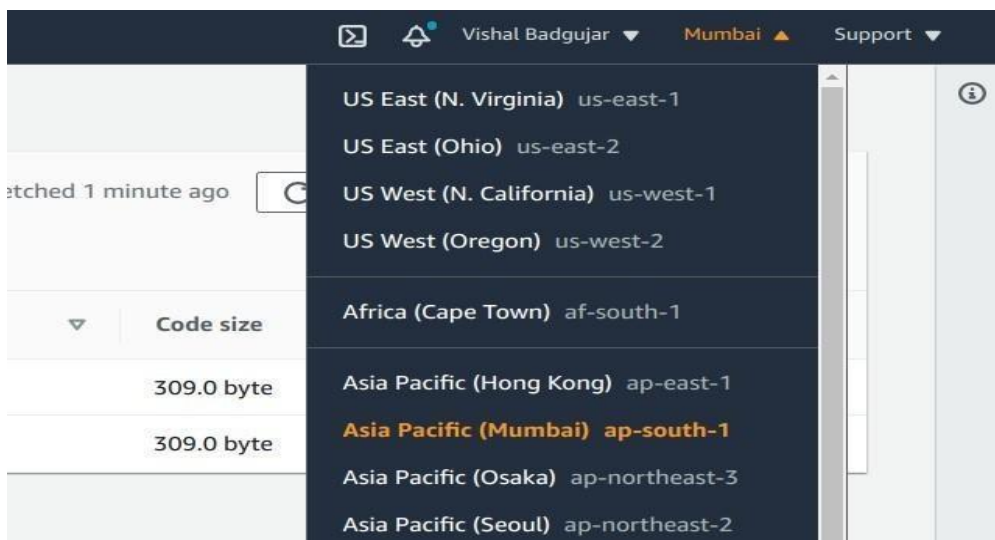


Steps: First Lambda functions using Python

1. Open Aws Console and search for Lambda Service and open home screen of Lambda.



2. Choose region in which you need to create Lambda function as it is region specific.





PARSHVANATH CHARITABLE TRUST'S
A. P. SHAH INSTITUTE OF TECHNOLOGY
Department of Information Technology
(NBA Accredited)



3. Create sum as a Lambda Function in Python Language so select latest version of Python and choose role with basic Lambda Permission to allow cloudwatch for monitoring.

Lambda > Functions > Create function

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.

Browse serverless app repository ☐
Deploy a sample Lambda application from the AWS Serverless Application Repository.

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.

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

4. Lambda sum function is created successfully

aws Services Search for services, features, marketplace products, and docs [Alt+S]


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


Lambda > Functions > sum

sum

Throttle Copy ARN Actions

Function overview [Info](#)

**sum**

**Layers** (0)

+ Add trigger + Add destination

Description -

Last modified 33 seconds ago

Function ARN [arn:aws:lambda:ap-south-1:229296960472:function:sum](#)

Code Test Monitor Configuration Aliases Versions

Code source [Info](#) Upload from

File Edit Find View Go Tools Window Test Deploy Changes deployed

Go to Anything (Ctrl-P)

Environment

sum lambda_function.py

```
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
```



5. Write a sample python code for sum of two numbers:

The screenshot shows a code editor interface with a menu bar (File, Edit, Find, View, Go, Tools, Window) and buttons for 'Test' and 'Deploy'. The 'Test' button is highlighted. Below the menu bar is a search bar labeled 'Go to Anything (Ctrl-P)'. On the left, there is an 'Environment' panel showing a folder 'sum' and a file 'lambda_function.py'. The main editor area shows the following Python code:

```
1 import json
2
3 def lambda_handler(event, context):
4     first_number = 100
5     second_number = 200
6     sum = first_number + second_number
7     return sum
```

6. Configure Test Event in Json Format

The screenshot shows a 'Configure test event' dialog box. It contains the following elements:

- A title bar with a close button (X).
- A paragraph: "A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events."
- Two radio buttons: "Create new test event" (selected) and "Edit saved test events".
- An "Event template" dropdown menu with "hello-world" selected.
- An "Event name" text input field with "mytest1" entered.
- A code editor area showing a JSON structure:

```
1 {
2
3 }
```




PARSHVANATH CHARITABLE TRUST'S
A. P. SHAH INSTITUTE OF TECHNOLOGY
Department of Information Technology
(NBA Accredited)



Code source Info

File Edit Find View Go Tools Window Test Deploy Changes deployed

Go to Anything (Ctrl-P)

sum1.py Execution results

Environment

sum /

sum1.py

Execution results

Test Event Name

sum

Response

300

Function Logs

START RequestId: bc4cb31a-e6ff-42e3-8804-0e3d78afa7d6 Version: \$LATEST

END RequestId: bc4cb31a-e6ff-42e3-8804-0e3d78afa7d6

REPORT RequestId: bc4cb31a-e6ff-42e3-8804-0e3d78afa7d6 Duration: 1.07 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 51 MB Init Duration: 125.07 ms

Request ID

bc4cb31a-e6ff-42e3-8804-0e3d78afa7d6

Write a sample Second sample python Code:

Code source Info

File Edit Find View Go Tools Window Test Deploy Changes deployed

Go to Anything (Ctrl-P)

lambda_function.py Execution results

Environment

sum /

lambda_function.py

```
1 def lambda_handler(event, context):
2     if event["name"] == "vishal":
3         return "apsit"
```



Configure Test Event

Configure test event

A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events.

- ☐ Create new test event
☒ Edit saved test events

Saved Test Events

vishal1

```
1 {  
2   "name": "vishal"  
3 }
```

If condition met returns a value as apsit

| Execution results | | Status: Succeeded |
|-------------------|--|-------------------|
| Test Event Name | vishal1 | |
| Response | "apsit" | |
| Function Logs | START RequestId: cddd9b6e-59b3-4457-be2b-04e4784e5c3e Version: \$LATEST END RequestId: cddd9b6e-59b3-4457-be2b-04e4784e5c3e REPORT RequestId: cddd9b6e-59b3-4457-be2b-04e4784e5c3e Duration: 1.12 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 50 MB Init Duration: 124.74 ms | |
| Request ID | cddd9b6e-59b3-4457-be2b-04e4784e5c3e | |

Conclusion: Write your own findings.