

St. Francis Institute of Technology, Mumbai-400 103  
**Department Of Information Technology**

A.Y. 2021-2022  
Class: TE-ITA/B, Semester: V

Subject: **Advanced DevOps Lab**

**Experiment – 11: To understand AWS Lambda, its workflow, and to create the first Lambda function using Python/Java.**

**1. Aim:** To write the first Lambda function using Python/Java/Node.js.

**2. Objectives:** Aim of this experiment is that, the students will learn:

- Serverless cloud concept and how to create Lambda function in various languages
- Invoke Lambda function
- Monitoring AWS Lambda

**3. Outcomes:** After study of this experiment, the students will learn following:

- Introduction to Serverless computing
- Creating Lambda function
- Testing Lambda function
- Monitoring and Logging Lambda function

**4. Prerequisite:** Knowledge of Python/Java/Node.js and AWS console.

**5. Requirements:** AWS account, browser, Personal Computer, Windows operating system, Internet Connection, Google Doc.

**6. Pre-Experiment Exercise:**

**Brief Theory:** Refer shared material

**7. Laboratory Exercise**

**A. Procedure:**

**a. Answer the following:**

- Explain Serverless concept?

→ Serverless computing is a method of providing backend services on an as-needed basis. A serverless provider allows users to write and deploy code without the hassle of worrying about the underlying infrastructure. A company that gets backend services from a serverless vendor is charged based on their computation and do not have to reserve and pay for a fixed amount of bandwidth or number of servers, as the service is auto-scaling. Note that despite the name serverless, physical servers are still used but developers do not need to be aware of them.

Serverless computing allows developers to purchase backend services on a flexible ‘pay-as-you-go’ basis, meaning that developers only have to pay for the services they use. This is like switching from a cell phone data plan with a monthly fixed limit, to one that only charges for each byte of data that actually gets used.

The term ‘serverless’ is somewhat misleading, as there are still servers providing these backend services, but all of the server space and infrastructure concerns are handled by the vendor. Serverless means that the developers can do their work without having to worry about servers at all.

Advantages are:

1. Lower costs - Serverless computing is generally very cost-effective, as traditional cloud providers of backend services (server allocation) often result in the user paying for unused space or idle CPU time.
2. Simplified scalability - Developers using serverless architecture don't have to worry about policies to scale up their code. The serverless vendor handles all of the scaling on demand.
3. Simplified backend code - With FaaS, developers can create simple functions that independently perform a single purpose, like making an API call.
4. Quicker turnaround - Serverless architecture can significantly cut time to market. Instead of needing a complicated deploy process to roll out bug fixes and new features, developers can add and modify code on a piecemeal basis.

● What are the applications of AWS Lambda?

→ An AWS Lambda application is a combination of Lambda functions, event sources, and other resources that work together to perform tasks. You can use AWS CloudFormation and other tools to collect your application's components into a single package that can be deployed and managed as one resource. Applications make your Lambda projects portable and enable you to integrate with additional developer tools, such as AWS CodePipeline, AWS CodeBuild, and the AWS Serverless Application Model command line interface (SAM CLI).

**Use cases:**

1. Data processing

You can use AWS Lambda to execute code in response to triggers such as changes in data, shifts in system state, or actions by users. Lambda can be directly triggered by AWS services such as S3, DynamoDB, Kinesis, SNS, and CloudWatch. It can also connect to existing EFS file systems or be orchestrated into workflows by AWS Step Functions. This allows you to build a variety of real-time serverless data processing systems.

2. Real-time file processing

You can use Amazon S3 to trigger AWS Lambda to process data immediately after an upload. You can also connect to an existing Amazon EFS file system directly, which enables massively parallel shared access for large scale file processing. For example, you can use Lambda to thumbnail images, transcode videos, index files, process logs, validate content, and aggregate and filter data in real-time.

3. It Connects to API Gateway and Other Connection Points:

In a manner of speaking, the API Gateway is the portal to your microcosm within AWS. It'll connect to many of the cloud services directly, but when you need flexibility, Lambda is your go-to. See, it's where you put custom code to process requests that come in through the API Gateway. Use a Lambda when you need to access several services or do custom processing.

4. You Only Pay for What You Use

As mentioned in the introduction, you'll only pay for what you use. Billing for functions is based on 100 ms intervals.

Interestingly, this creates an incentive to use more efficient code if you're considering the costs. Also, you can tune the maximum resource usage of each Lambda, which affects the base price for your function executions.

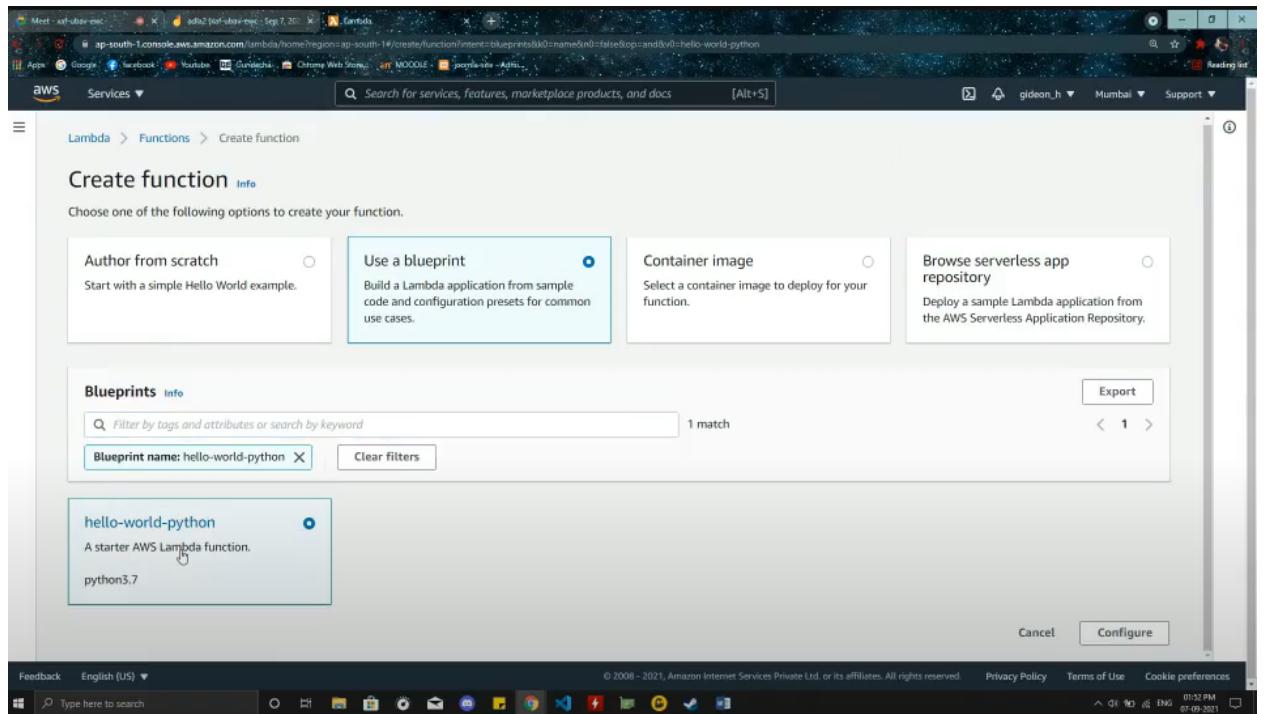
**b. Execute following and attach screenshots:**

- Enter Lambda service from console

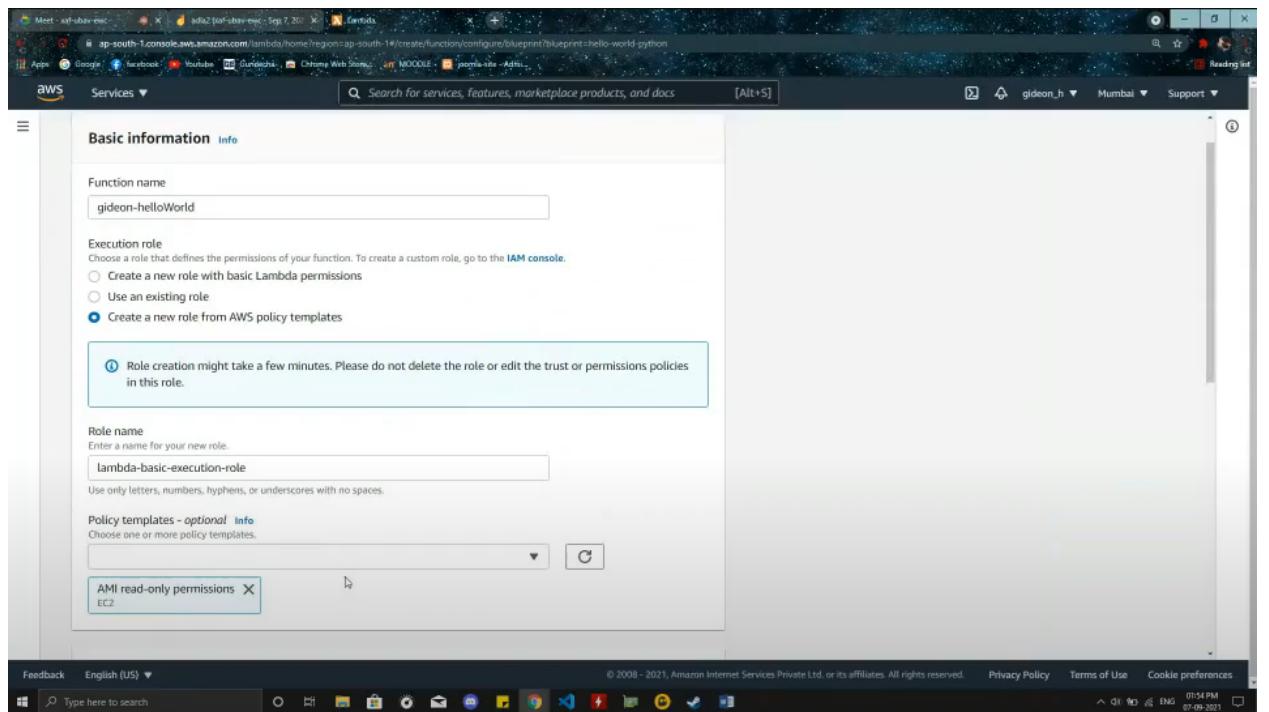
The screenshot shows the AWS Management Console with the URL <https://ap-south-1.console.aws.amazon.com/console/home?region=ap-south-1>. The left sidebar has sections like Favorites, Recently visited, and All services. Under All services, the Compute section is expanded, showing Lambda as the selected service. Other services listed under Compute include EC2, Lightsail, and Batch.

The screenshot shows the AWS Lambda service page with the URL <https://ap-south-1.console.aws.amazon.com/lambda/home?region=ap-south-1#functions>. The left sidebar shows options like Dashboard, Applications, Functions, Additional resources, and Related AWS resources. The main area is titled "Functions (0)" and displays a table with columns: Function name, Description, Package type, Runtime, Code size, and Last modified. A message at the bottom of the table says "There is no data to display."

## ● Select a Lambda Blueprint



## ● Configure and create Lambda function



The screenshot shows the AWS Lambda console interface. At the top, a green banner displays the message: "Successfully created the function gideon-helloWorld. You can now change its code and configuration. To invoke your function with a test event, choose 'Test'." Below this, the function name "gideon-helloWorld" is displayed. A "Function overview" section is open, showing the following details:

- Icon:** A yellow Lambda icon.
- Name:** gideon-helloWorld
- Layers:** (0)
- Add trigger:** A button to add triggers.
- Add destination:** A button to add destinations.
- Description:** A starter AWS Lambda function.
- Last modified:** in 0 seconds
- Function ARN:** arn:aws:lambda:ap-south-1:191048:gideon-helloWorld

The navigation bar at the bottom includes tabs for Code, Test, Monitor, Configuration, Aliases, and Versions. The "Code" tab is currently selected.

The screenshot shows the "Edit basic settings" page for the "gideon-helloWorld" function. The page has the following sections:

- Basic settings:** Includes a description field containing "A starter AWS Lambda function.", a memory field set to 128 MB, and a timeout field set to 3 seconds.
- Execution role:** Shows the "Use an existing role" option selected, with the role "service-role/lambda-basic-execution-role" chosen. A link to "View the lambda-basic-execution-role role on the IAM console" is provided.

The browser's address bar shows the URL: ap-south-1.console.aws.amazon.com/lambda/home?region=ap-south-1#/functions/gideon-helloWorld/edit/basic-settings#tab=configure.

The screenshot shows the 'Edit runtime settings' page for a Lambda function named 'gideon-helloWorld'. The 'Runtime' dropdown is set to 'Python 2.7'. A modal window titled 'New runtime available' informs the user that a new runtime is available for their function's language: Python 3.9. The 'Handler' field is set to 'lambda\_function.lambda\_handler'. At the bottom right are 'Cancel' and 'Save' buttons.

Feedback English (US) ▾

© 2008–2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

Type here to search

aws Services ▾

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

gideon\_h Mumbai Support

Edit runtime settings

Runtime settings Info

Runtime

Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Python 2.7

New runtime available

A new runtime is available for your function's language: Python 3.9

Handler Info

lambda\_function.lambda\_handler

Cancel Save

The screenshot shows the 'Code source' tab of the Lambda function configuration. The code editor displays the following Python script:

```
import json

print('Loading function')

def lambda_handler(event, context):
    print("Event = " + json.dumps(event, indent=2))
    print("Value1 = " + event['key1'])
    print("Value2 = " + event['key2'])
    print("Value3 = " + event['key3'])
    return event['key1'] # Echo back the first key value
    #raise Exception('Something went wrong')
```

The interface includes tabs for Code, Test, Monitor, Configuration, Aliases, and Versions. The 'Test' tab is currently selected. A 'Changes not deployed' message is visible. The bottom status bar shows the date and time: 02:01 PM 07-09-2021.

Feedback English (US) ▾

© 2008–2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

Type here to search

aws Services ▾

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

gideon\_h Mumbai Support

Code source Info

Code Test Monitor Configuration Aliases Versions

File Edit Find View Go Tools Window Test Deploy Changes not deployed

Upload from ▼

Environment

lambda\_function

gideon-helloWorld

lambda\_function.py

```
import json

print('Loading function')

def lambda_handler(event, context):
    print("Event = " + json.dumps(event, indent=2))
    print("Value1 = " + event['key1'])
    print("Value2 = " + event['key2'])
    print("Value3 = " + event['key3'])
    return event['key1'] # Echo back the first key value
    #raise Exception('Something went wrong')
```

Feedback English (US) ▾

© 2008–2021, Amazon Internet Services Private Ltd. or its affiliates. All rights reserved. Privacy Policy Terms of Use Cookie preferences

Type here to search

aws Services ▾

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

gideon\_h Mumbai Support

The screenshot shows the AWS Lambda console interface. A modal window titled "Configure test event" is open, prompting the user to create a new test event or edit saved ones. The "Event template" dropdown is set to "hello-world". The "Event name" field contains "test". The code editor displays the following JSON test event:

```
1: {  
2:   "Key1": "value1",  
3:   "Key2": "value2",  
4:   "key3": "Gideon says HI!!"  
5: }
```

Below the modal, the main Lambda function page is visible, showing the "Code source" tab selected. The code editor displays the function's code:

```
1: import json  
2:  
3: print('Loading function')  
4:  
5:  
6: def lambda_handler(event, context):  
7:     #print("Received event: " + json.dumps(event, indent=2))  
8:     print("Value1 = " + event['Key1'])  
9:     print("Value2 = " + event['key2'])  
10:    print("Value3 = " + event['key3'])  
11:    return event['Key1'] # Echo back the first key value  
#raise Exception('Something went wrong')
```

A green banner at the top of the main page indicates "Your changes have been saved." The browser status bar shows the date and time as 02:03 PM 07-09-2021.

● Invoke Lambda function and verify results

The screenshot displays two windows from the AWS Lambda console.

**Top Window (Execution Results):**

- Panel: thing (Ctrl-P)
- Function: lambda\_function.x
- Execution result:
- Test Event Name:** test1
- Response:** "value1"
- Function Logs:**

```
START RequestId: 30fb6dc-37e4-4c5a-9431-80da5382640b Version: $LATEST
Loading function
value1 = value1
value2 = value2
value3 = Gideon says HI!
END RequestId: 30fb6dc-37e4-4c5a-9431-80da5382640b
REPORT RequestId: 30fb6dc-37e4-4c5a-9431-80da5382640b Duration: 0.36 ms Billed Duration: 0 ms冷启动: yes
```
- Request ID:** 30fb6dc-37e4-4c5a-9431-80da5382640b

**Bottom Window (CloudWatch Search):**

- Search term: cloud watch
- Services (42) list:
  - AWS Cloud Map
  - CloudWatch
  - CloudTrail
  - Cloud9
- Features (38) list:
  - CloudWatch Metrics

The screenshot shows two separate instances of the AWS CloudWatch Management Console. Both instances are for the same log group: `/aws/lambda/gideon-helloWorld`.

**Top Window (Log Groups View):**

- The left sidebar shows the navigation menu with "Logs" selected.
- The main area displays "Log groups (1)".
- The single log group listed is `/aws/lambda/gideon-helloWorld`, which was selected by the cursor.
- Details for this log group include:
  - Retention: Never expire
  - Metric filters: -
  - Contributor Insights: -
  - Subscription filters: -

**Bottom Window (Log Stream Details View):**

- The left sidebar shows the navigation menu with "Logs" selected.
- The main area displays the log group `/aws/lambda/gideon-helloWorld`.
- The "Log streams (1)" section shows one log stream:
  - Log stream name: `2021/09/07/[LATEST]2dbe18ef3fdc4176916ddcdba23edc26`
  - Last event time: 2021-09-07 14:04:27 (UTC+05:30)

The screenshot shows the CloudWatch Management Console interface. At the top, there's a search bar labeled "Filter events" and a "Clear" button. Below it, a table has two columns: "Timestamp" and "Message". The "Timestamp" column is sorted in descending order. The "Message" column contains log entries from a Lambda function. The entries are:

- No older events at this moment. *Retry*
- 2021-09-07T14:04:26.998+05:30 START RequestId: 30fb6dc-37e4-4c5a-9431-80da5382640b Version: \$LATEST
- 2021-09-07T14:04:26.998+05:30 Loading function
- 2021-09-07T14:04:26.999+05:30 value1 = value1  
↳
- 2021-09-07T14:04:26.999+05:30 value2 = value2
- 2021-09-07T14:04:26.999+05:30 value3 = Gideon says HI!
- 2021-09-07T14:04:27.001+05:30 END RequestId: 30fb6dc-37e4-4c5a-9431-80da5382640b
- 2021-09-07T14:04:27.001+05:30 REPORT RequestId: 30fb6dc-37e4-4c5a-9431-80da5382640b Duration: 0.36 ms

At the bottom, it says "No newer events at this moment. Auto retry paused. *Resume*".

This screenshot shows the AWS CloudWatch service navigation menu. On the left, there's a sidebar with various options like "Alarms", "Logs", "Metrics", "Events", "Application monitoring", "Insights", and "Getting Started". The "Logs" section is currently selected and highlighted in orange. The main area displays the same log events as the previous screenshot, with the "Logs" tab active. The log entries are identical to those shown above.

- Clean up the lambda Function and free all the resources.

(lambda function cleared)

The screenshot shows the AWS Lambda console interface. On the left, there's a sidebar with 'AWS Lambda' selected. The main area is titled 'Functions (0)' and contains a search bar and a table header with columns: Function name, Description, Package type, Runtime, Code size, and Last modified. Below the header, it says 'There is no data to display.' At the top right, there are buttons for 'Actions' and 'Create function'. The bottom of the screen shows standard AWS footer links like Feedback, English (US), and various legal notices.

## 8. Post-Experiments Exercise

### A. Extended Theory:

Create the Lambda function using Node.js

#### → To create an execution role

1. Open the roles page in the IAM console.
2. Choose Create role.
3. Create a role with the following properties.

- Trusted entity – Lambda.
- Permissions – AWSLambdaBasicExecutionRole.
- Role name – lambda-role.

The AWSLambdaBasicExecutionRole policy has the permissions that the function needs to write logs to CloudWatch Logs.

You can add permissions to the role later, or swap it out for a different role that's specific to a single function.

#### To create a Node.js function

1. Open the Lambda console.
2. Choose Create function.
3. Configure the following settings:
  - Name – my-function.
  - Runtime – Node.js 14.x.
  - Role – Choose an existing role.
  - Existing role – lambda-role.

4. Choose Create function.
5. To configure a test event, choose Test.
6. For Event name, enter test.
7. Choose Save changes.
8. To invoke the function, choose Test.

The console creates a Lambda function with a single source file named index.js. You can edit this file and add more files in the built-in code editor. To save your changes, choose Save. Then, to run your code, choose Test.

The index.js file exports a function named handler that takes an event object and a context object. This is the handler function that Lambda calls when the function is invoked. The Node.js function runtime gets invocation events from Lambda and passes them to the handler. In the function configuration, the handler value is index.handler.

When you save your function code, the Lambda console creates a .zip file archive deployment package. When you develop your function code outside of the console (using an SDE) you need to create a deployment package to upload your code to the Lambda function.

The function runtime passes a context object to the handler, in addition to the invocation event. The context object contains additional information about the invocation, the function, and the execution environment. More information is available from environment variables.

Your Lambda function comes with a CloudWatch Logs log group. The function runtime sends details about each invocation to CloudWatch Logs. It relays any logs that your function outputs during invocation. If your function returns an error, Lambda formats the error and returns it to the invoker

B. Questions:

GIDEON HARPANAHALLI

TEITA [23]

ADL EXP 11

2015

Page No.

Date

/ /

B] Questions

- What is AWS Lambda Function

- AWS Lambda is a serverless compute service that runs code in response to a trigger.
- automatically manages the underlying compute resources.
- AWS Lambda can extend other AWS services with custom logic or create your own backend services that operate at AWS scale, performance.
- AWS Lambda can run code to response to HTTP requests etc. via Amazon API gateway.
- modifications to objects in Amazon S3 buckets, table updates in Amazon Dynamo DB & state transitions in AWS Step Functions.

- EC2 Lambda vs. Elastic Beanstalk

- Elastic Beanstalk Service is fast in deployment & managing APPS in cloud.
- Beanstalk automatically handles the deployment details & capacity, load balancing, auto-scaling & application health monitoring.
- Lambda service directly starts running code responding to modifications that are made to objects, updates or messages.
- AWS Lambda goes on its own in managing the underlying infrastructure for your compute resources.
- EC2 we need to operate with virtual machines (VMs) known as EC2 instances.
- Needs pre-configuration of image with an installed OS and create customized AMI.

### C. Conclusion:

GIDEON HARPANAHALLI  
TEITA [33] AOL EXP. II  
Date / / Page No. / /

C CONCLUSIONS

(1)

- This experiment dealt with AWS Lambda functions and its workings, understood serverless compute.
- We created a lambda function using Python 2, tested it.
- We made changes in the default code and displayed result.
- CloudWatch Log ~~session~~ section showed the result, along with time stamps.

(2)

- Lambda function is one of the most effective way to deploy and run code in cloud, in terms of cost and ease of deployment.
- Lambda function automatically manages and provisions the infrastructure for the developer, thus reducing the total administrative load on the developer.
- Highly flexible and integrates well with other AWS services.

### D. References:

- <https://aws.amazon.com/getting-started/hands-on>
- <https://www.scalyr.com/blog/aws-lambda-tutorial/>