

Adv.DevOps Exp 11

Name- Ishika Devare RollNo- 14 Batch- A

	<p><u>Adv.DevOps</u></p>
	<p><u>Experiment No. 11</u></p>
	<p>* Aim - To understand AWS Lambda, its work flow, various functions and create your own lambda functions using python / java / Node JS.</p>
	<p>* Theory -</p> <p>What is AWS Lambda?</p> <p>AWS Lambda is a serverless computing service provided by Amazon web services. Users of AWS Lambda create functions, self-contained applications written in one of the supported languages and upload them to AWS Lambda, which executes those functions in an efficient and flexible manner.</p> <p>The Lambda function can perform any kind of computing task, from serving web pages and processing streams of data to calling API's and integrating with other AWS services.</p>
	<p>The concept of "serverless" computing refers to not needing to maintain your own servers to run these functions. AWS Lambda is a fully managed service that takes care of all infrastructure for you.</p>
	<p>What are components of AWS Lambda function?</p> <p>The 3 components of AWS Lambda are -</p> <p>a) function - This is the actual code that performs a task.</p>

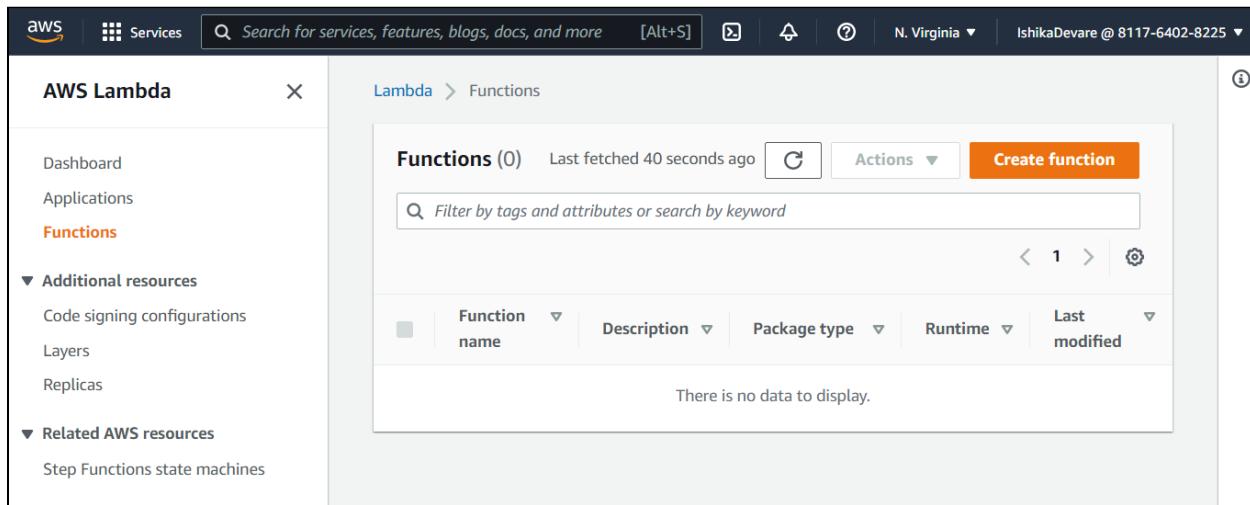
- b) A configuration - This specifies how your function is executed.
- c) An event source - This is the event that triggers the function. One can trigger with several AWS services or a third party service.

Implementation:

Prerequisites:

1. An AWS Account

Step 1: Open up the Lambda Console and click on the Create button. Be mindful of where you create your functions since Lambda is region-dependent.



Step 2: 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.

Then, choose a runtime env for your function, under the dropdown, you can see all the options AWS supports, Python, Nodejs, .NET and Java being the most popular ones.

After that, choose to create a new role with basic Lambda permissions if you don't have an existing one.

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.

Architecture [Info](#)
Choose the instruction set architecture you want for your function code.
 x86_64
 arm64

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
 Create a new role from AWS policy templates

Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

Lambda will create an execution role named Experiment11-role-51qq4j66, with permission to upload logs to Amazon CloudWatch Logs.

Click on the Create button.

Step 3: This process will take a while to finish and after that, you'll get a message that your function was successfully created.

Code source [Info](#)

File Edit Find View Go Tools Window **Test** Deploy

Go to Anything (Ctrl-P) lambda_function

Environment

```
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 
```

Step 4: To change the configuration, open up the Configuration Tab and under General Configuration, choose Edit.

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

The screenshot shows two views of the AWS Lambda function configuration interface. The top view is the 'General configuration' tab, which lists various settings like Triggers, Permissions, Destinations, and VPC. The bottom view is the 'Basic settings' tab, where specific configuration parameters are set. In the 'Basic settings' tab, the 'Memory' is set to 128 MB and the 'Timeout' is set to 0 min 1 sec. The 'Execution role' dropdown is set to 'service-role/Experiment11-role-51qq4j66'. There are 'Save' and 'Cancel' buttons at the bottom.

Successfully created the function Experiment11. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Code Test Monitor Configuration Aliases Versions

General configuration

Triggers
Permissions
Destinations
Function URL
Environment variables
Tags
VPC
Monitoring and operations tools
Concurrency
Asynchronous invocation
Code signing
Database proxies
File systems
State machines

General configuration Info

Description -
Memory 128 MB Ephemeral storage 512 MB

Timeout 0 min 1 sec

AWS Compute Optimizer Opt in to see memory recommendations for your Lambda functions. View details [View details](#)

Basic settings Info

Description - optional

Memory Info Your function is allocated CPU proportional to the memory configured.

128 MB Set memory to between 128 MB and 10240 MB

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

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

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

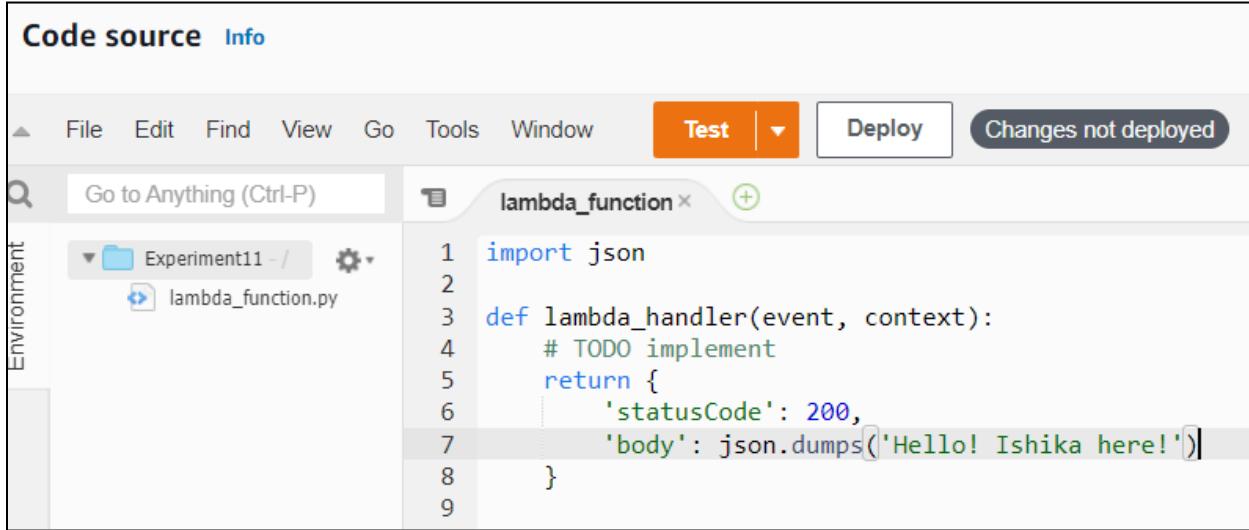
Existing role Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

service-role/Experiment11-role-51qq4j66

View the [Experiment11-role-51qq4j66](#) role on the IAM console.

Cancel Save

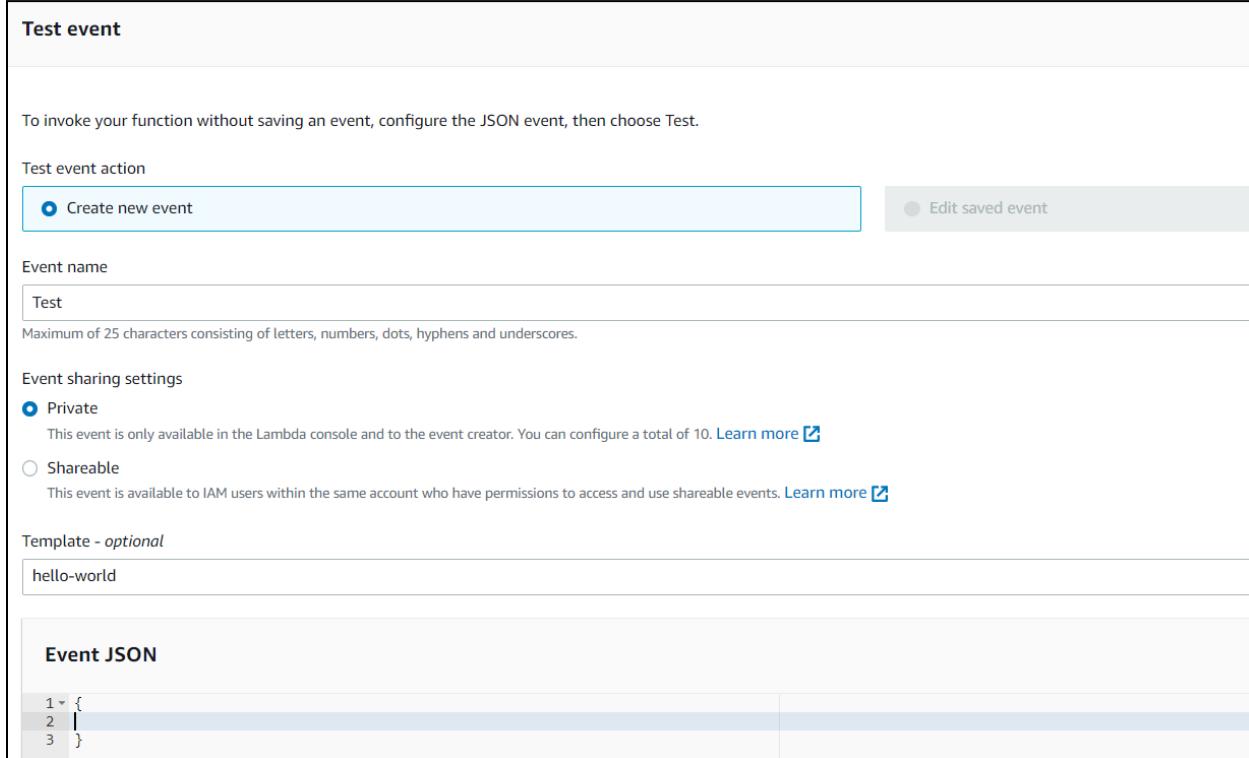
Step 5: You can make changes to your function inside the code editor. You can also upload a zip file of your function or upload one from an S3 bucket if needed. Press Ctrl + S to save the file and click Deploy to deploy the changes.



The screenshot shows the AWS Lambda code editor interface. At the top, there's a navigation bar with File, Edit, Find, View, Go, Tools, Window, a Test button, a Deploy button, and a status message 'Changes not deployed'. Below the navigation bar is a search bar labeled 'Go to Anything (Ctrl-P)'. To the left is an 'Environment' sidebar. The main area displays a Python file named 'lambda_function.py' with the following code:

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

Step 6: Click on Test and you can change the configuration, like so. If you do not have anything in the request body, it is important to specify two curly braces as valid JSON, so make sure they are there

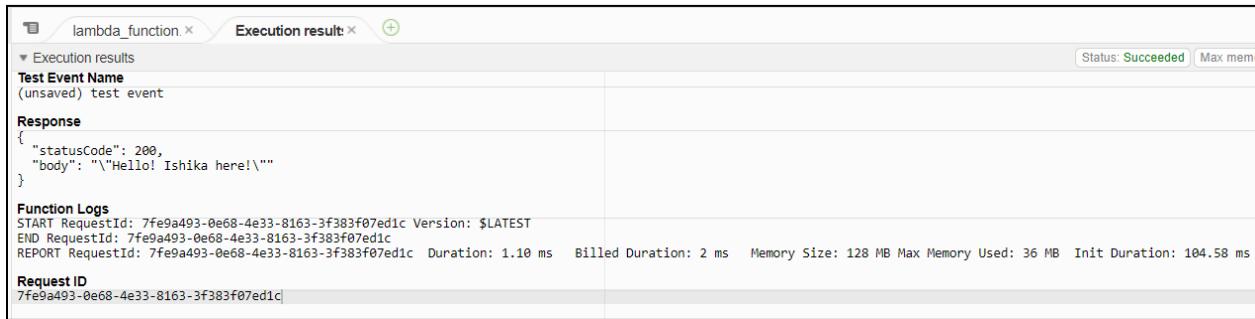


The screenshot shows the 'Test event' configuration dialog. It includes fields for creating a new event ('Create new event' selected), naming the event ('Test'), and setting sharing settings ('Private'). There's also a 'Template - optional' field containing 'hello-world' and an 'Event JSON' field which currently contains an empty object: { }.

Execution result: succeeded ([logs](#))

► Details

Step 7: Now click on Test and you should be able to see the results



The screenshot shows the AWS Lambda Test Execution Results interface. At the top, there's a green success message: "Execution result: succeeded ([logs](#))". Below it is a "Details" link. The main area contains the following information:

- Execution results**:
 - Test Event Name**: (unsaved) test event
 - Response**:

```
{  "statusCode": 200,  "body": "\"Hello! Ishika here!\""}  
}
```
 - Function Logs**:

```
START RequestId: 7fe9a493-0e68-4e33-8163-3f383f07ed1c Version: $LATEST
END RequestId: 7fe9a493-0e68-4e33-8163-3f383f07ed1c
REPORT RequestId: 7fe9a493-0e68-4e33-8163-3f383f07ed1c Duration: 1.10 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 36 MB Init Duration: 104.58 ms
```
 - Request ID**: 7fe9a493-0e68-4e33-8163-3f383f07ed1c

* Conclusion - In this experiment, we learnt about AWS lambda function, its features and how serverless system can be designed and how its functions. Also, we use it to create, deploy and test serverless functions in cloud.