

Name: Hrishikesh Kumbhar

Div: D15A

Roll no: 32

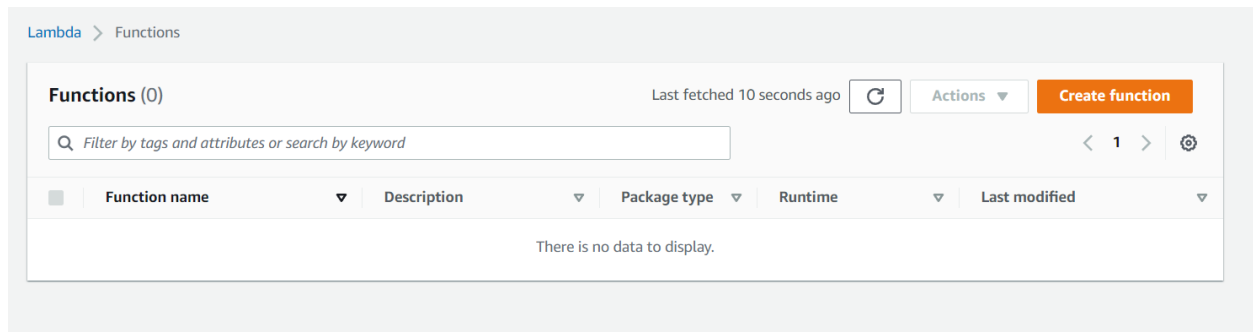
Sub: Advanced DevOps

Experiment No: 11

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

Steps:

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.

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.

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](#).

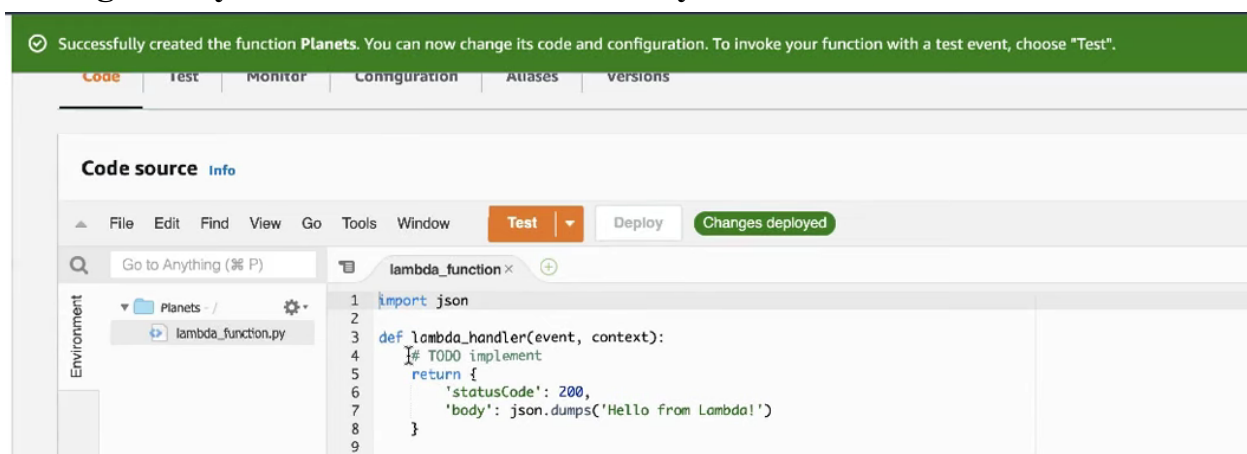
☒ Create a new role with basic Lambda permissions
☐ Use an existing role
☐ Create a new role from AWS policy templates

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

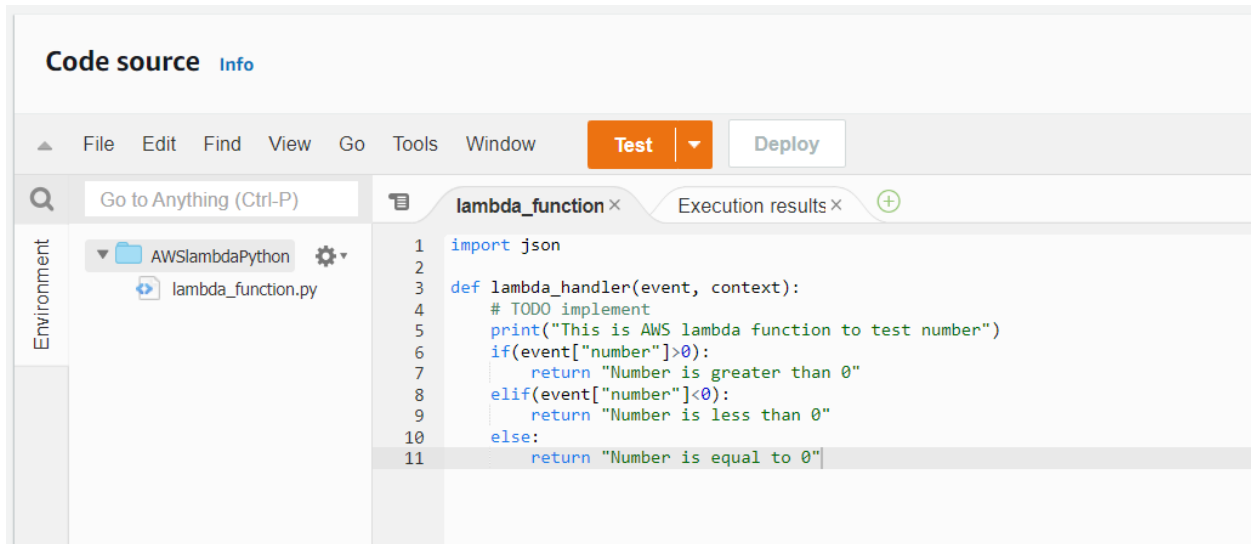
Lambda will create an execution role named <myFunctionName>-role-16k7i7iv, with permission to upload logs to Amazon CloudWatch Logs.

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.



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.



Lambda_function.py

```
import json

def lambda_handler(event, context):
    # TODO implement
    print("This is AWS lambda function to test number")
    if(event["number"]>0):
        return "Number is greater than 0"
    elif(event["number"]<0):
        return "Number is greater than 0"
    else:
        return "Number is equal to 0"
```

Step 4: Configure test cases by clicking on test:

Configure test event ✕

A test event is a JSON object that mocks the structure of requests emitted by AWS services to invoke a Lambda function. Use it to see the function's invocation result.

To invoke your function without saving an event, modify the event, then choose Test. Lambda uses the modified event to invoke your function, but does not overwrite the original event until you choose Save changes.

Test event action

☐ Create new event

☒ Edit saved event

Event name

TestingNumber ▼ ↺ Delete

Event JSON Format JSON

```
1 {
2   "number": -20
3 }
```

You can set event parameters in Event JSON. Here we have created an event TestingNumber to check whether the number is positive or negative or zero. By setting key “number”: <number> we can test the function for different inputs. “number” key is imported via event[“number”] in the lambda_function.py file.

Step 5: Now click on Test and you should be able to see the results.

Tools Window **Test** ▼ Deploy ⌵ ⌵ ⌵ ⚙

lambda_function × Execution results × +

▼ Execution results Status: Succeeded Max memory used: 36 MB Time: 1.06 ms

Test Event Name

TestingNumber

Response

"Number is less than 0"

Function Logs

START RequestId: 210e1939-23eb-41cb-96e7-5c1df98ca12c Version: \$LATEST
This is AWS lambda function to test number
END RequestId: 210e1939-23eb-41cb-96e7-5c1df98ca12c
REPORT RequestId: 210e1939-23eb-41cb-96e7-5c1df98ca12c Duration: 1.06 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 36 MB Init Duration: 92.56 ms

Request ID

210e1939-23eb-41cb-96e7-5c1df98ca12c

Test event action

☐ Create new event

☒ Edit saved event

Event name

TestingNumber ▼



Delete

Event JSON Format JSON

1 {

2 "number":45

3 }

Tools Window Test Deploy

lambda_function x Execution result: x

Execution results Status: Succeeded Max memory used: 37 MB Time: 0.86 ms

Test Event Name

TestingNumber

Response

"Number is greater than 0"

Function Logs

START RequestId: 864ba921-047a-4c50-846f-cae4c0a6acd5 Version: \$LATEST
This is AWS lambda function to test number
END RequestId: 864ba921-047a-4c50-846f-cae4c0a6acd5
REPORT RequestId: 864ba921-047a-4c50-846f-cae4c0a6acd5 Duration: 0.86 ms Billed Duration: 1 ms Memory Size: 128 MB Max Memory Used: 37 MB

Request ID

864ba921-047a-4c50-846f-cae4c0a6acd5

Test event action

☐ Create new event

☒ Edit saved event

Event name

TestingNumber ▼



Delete

Event JSON Format JSON

1 {

2 "number":0

3 }

Execution results	
Test Event Name	TestingNumber
Response	"Number is equal to 0"
Function Logs	START RequestId: ad3bbf2f-7ffe-4d4f-91e9-64c42da48e4f Version: \$LATEST This is AWS lambda function to test number END RequestId: ad3bbf2f-7ffe-4d4f-91e9-64c42da48e4f REPORT RequestId: ad3bbf2f-7ffe-4d4f-91e9-64c42da48e4f Duration: 0.89 ms Billed Duration: 1 ms Memory Size: 128 MB Max Memory Used: 37 MB
Request ID	ad3bbf2f-7ffe-4d4f-91e9-64c42da48e4f

Step 6: 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.

Lambda > Functions > AWSLambdaPython > Edit basic settings

Edit basic settings

Basic settings [Info](#)

Description - *optional*

Memory [Info](#)
Your function is allocated CPU proportional to the memory configured.

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](#)

MB

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

Timeout

min 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.

[View the AWSLambdaPython-role-16k7i7iv role on the IAM console.](#)

Cancel Save

Conclusion:

In this experiment, we learned about AWS Lambda and using it to create, deploy and test serverless functions in the Cloud.