

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

Theory:

AWS Lambda

A fully managed, serverless computing service where you run code without provisioning or managing servers. Lambda automatically scales your application based on the number of incoming requests or events, ensuring efficient resource utilization. You are only charged for the time your code is running, with no upfront cost, making it cost-effective for on-demand workloads.

Lambda Workflow

- **Create a Function:** Write the function code and define its handler (entry point). You can use the AWS Console, CLI, or upload a deployment package.
- **Set Event Sources:** Define how the function is triggered (e.g., when an object is uploaded to S3 or a DynamoDB table is updated).
- **Execution:** When triggered, Lambda runs your function, executes the logic, and automatically scales to handle the incoming event volume.
- **Scaling and Concurrency:** Lambda scales automatically by launching more instances of the function to handle simultaneous invocations. There are also options for configuring **reserved concurrency** to manage traffic.
- **Monitoring and Logging:** Lambda integrates with Amazon CloudWatch for logging and monitoring. Logs for each invocation are sent to CloudWatch, allowing you to track performance and troubleshoot errors.

AWS Lambda Functions

- **Python:** Great for quick development with its rich standard library and support for lightweight tasks.
- **Java:** Typically used for more complex, compute-intensive tasks. While it's robust, cold start times can be higher.
- **Node.js:** Excellent for I/O-bound tasks like handling APIs or streaming data, with fast startup times and efficient memory usage.

Prerequisites: AWS Personal/Academy Account

Name: Bhushan Mukund Kor

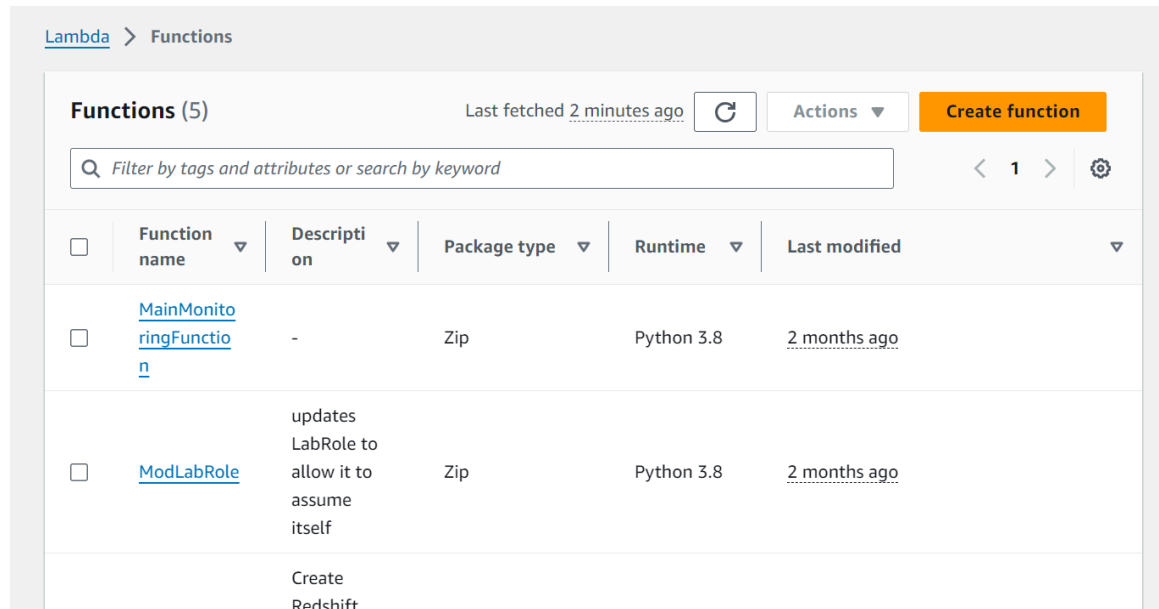
Academic Year: 2024-2025

Division: D15C

Roll No: 28

Steps To create the lambda function:

Step 1: Login to your AWS Personal/Academy Account. Open lambda and click on create function button.



Step 2: Now Give a name to your Lambda function, Select the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby. So will select Python 3.12, Architecture as x86, and Execution role to Create a new role with basic Lambda permissions.

The screenshot shows the 'Create function' wizard in the AWS Lambda console. It has three tabs: 'Author from scratch' (selected), 'Use a blueprint', and 'Container image'. Under 'Author from scratch', there's a 'Basic information' section with the following fields:

- Function name:** 'anshi-lambda' (with a note: 'Enter a name that describes the purpose of your function. Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).')
- Runtime:** 'Python 3.12' (with a note: 'Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.')
- Architecture:** 'x86_64' (with a note: 'Choose the instruction set architecture you want for your function code.')

Throttle

Copy ARN

Actions

Function overview

Info

Export to Application Composer

Download

Diagram

Template

anshi-lambda

Layers (0)

+ Add trigger

+ Add destination

Description

-

Last modified

2 seconds ago

Function ARN

arn:aws:lambda:us-east-1:708398963195:function:anshi-lambda

Function URL

Info

Code source

Info

Upload from

File Edit Find View Go Tools Window

Test

Deploy

Go to Anything (Ctrl-P)

Environment

anshi-lambda

lambda_function.py

lambda_function

Environment Variables

```

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

```

So See or Edit the basic settings go to configuration then click on edit general setting.

General configuration

Info

Edit

Description	Memory	Ephemeral storage
-	128 MB	512 MB
Timeout	SnapStart	
0 min 3 sec	None	

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.

Basic settings

Info

Description - optional

Basic settings

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](#)

512 MB

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

SnapStart

Info

Reduce startup time by having Lambda cache a snapshot of your function after the function has initialized. To evaluate whether your function code is resilient to snapshot operations, review the [SnapStart compatibility considerations](#)

None

Supported runtimes: Java 11, Java 17, Java 21.

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.

LabRole

View the LabRole role on the IAM console.

Step 3: Now Click on the Test tab then select Create a new event, give a name to the event and select Event Sharing to private, and select hello-world template.

Test event [Info](#) Save Test

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action

Create new event Edit saved event

Event name

our_event

Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.

Event sharing settings

☒ Private

This event is only available in the Lambda console and to the event creator. You can configure a total of 10. [Learn more](#)

☐ Shareable

This event is available to IAM users within the same account who have permissions to access and use shareable events. [Learn more](#)

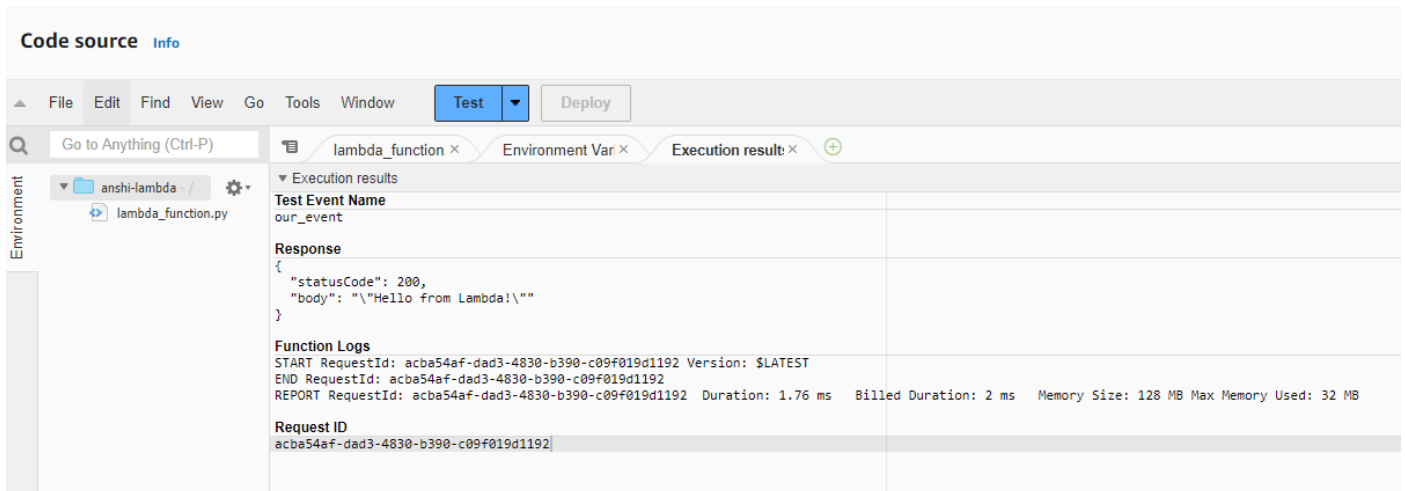
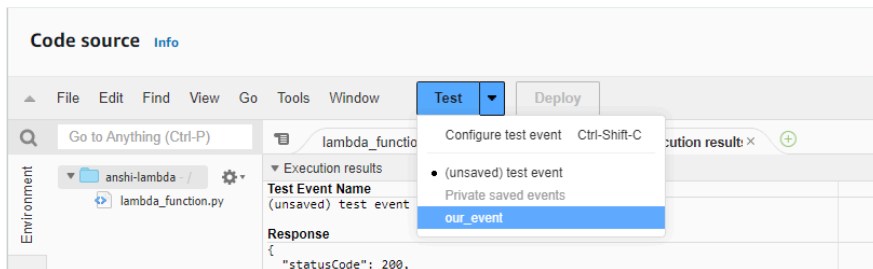
Template - optional

hello-world

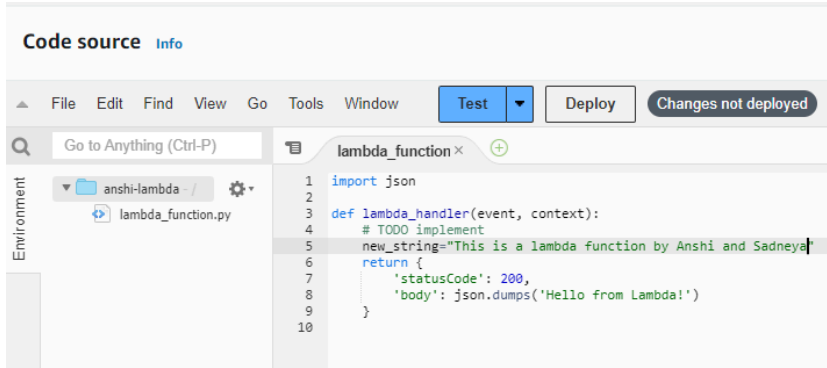
Event JSON Format JSON

```
1 {
2   "key1": "value1",
3   "key2": "value2",
4   "key3": "value3"
5 }
```

Step 4: Now In Code section select the created event from the dropdown of test then click on test . You will see the below output.

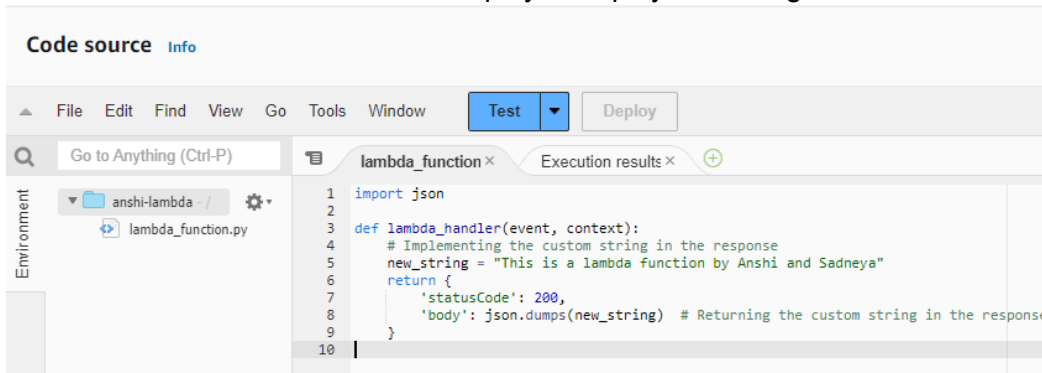


Step 5: You can edit your lambda function code. I have changed the code to display the new String.



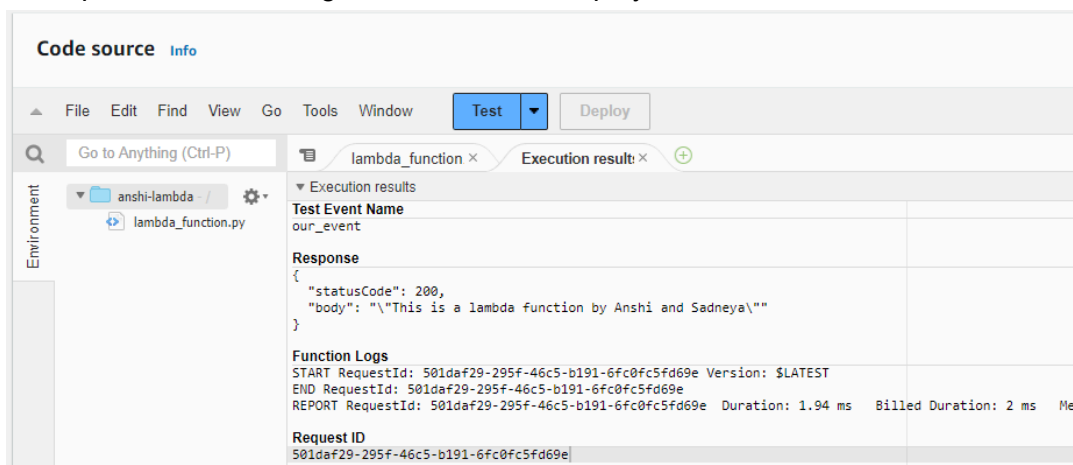
```
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     new_string="This is a lambda function by Anshi and Sadneya"
6     return {
7         'statusCode': 200,
8         'body': json.dumps('Hello from Lambda!')}
9
10
```

Now ctrl+s to save and click on deploy to deploy the changes.



```
1 import json
2
3 def lambda_handler(event, context):
4     # Implementing the custom string in the response
5     new_string = "This is a lambda function by Anshi and Sadneya"
6     return {
7         'statusCode': 200,
8         'body': json.dumps(new_string) # Returning the custom string in the response
9     }
10
```

Step 6: Now click on the test and observe the output. We can see the status code 200 and your string output and function logs. On successful deployment.



Test Event Name
our_event

Response
{ "statusCode": 200, "body": "\"This is a lambda function by Anshi and Sadneya\"" }

Function Logs
START RequestId: 501daf29-295f-46c5-b191-6fc0fc5fd69e Version: \$LATEST
END RequestId: 501daf29-295f-46c5-b191-6fc0fc5fd69e
REPORT RequestId: 501daf29-295f-46c5-b191-6fc0fc5fd69e Duration: 1.94 ms Billed Duration: 2 ms

Request ID
501daf29-295f-46c5-b191-6fc0fc5fd69e

Conclusion: In this experiment, we successfully created an AWS Lambda function and walked through its essential steps. After setting up the function with Python, we configured the basic settings, including adjusting the timeout to 1 second. We then created a test event, deployed the function, and validated the output. Additionally, we modified the Lambda function's code and redeployed it to observe the changes in real-time.

This practical experience demonstrated the simplicity and flexibility of AWS Lambda in creating serverless applications, allowing you to focus on code while AWS manages the infrastructure and scaling.