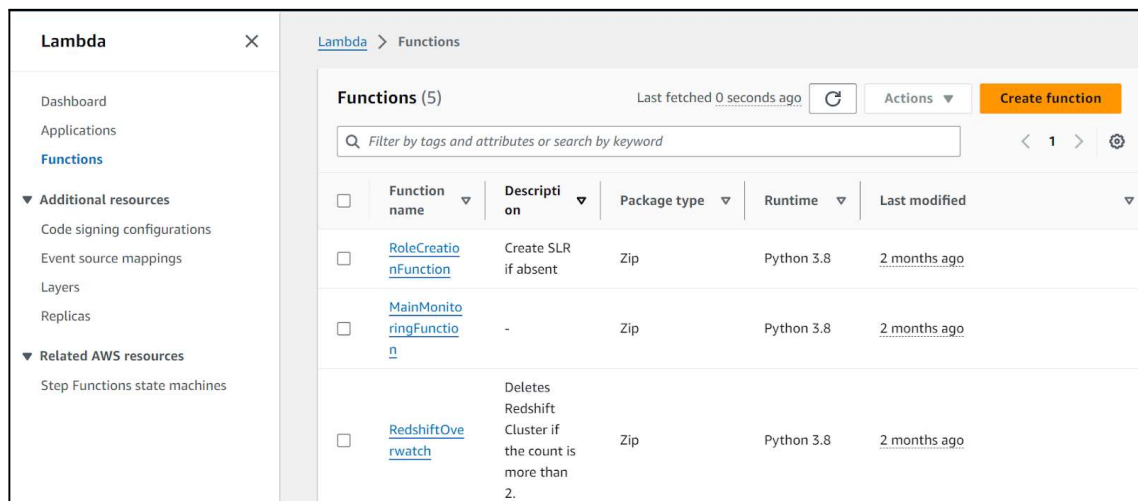
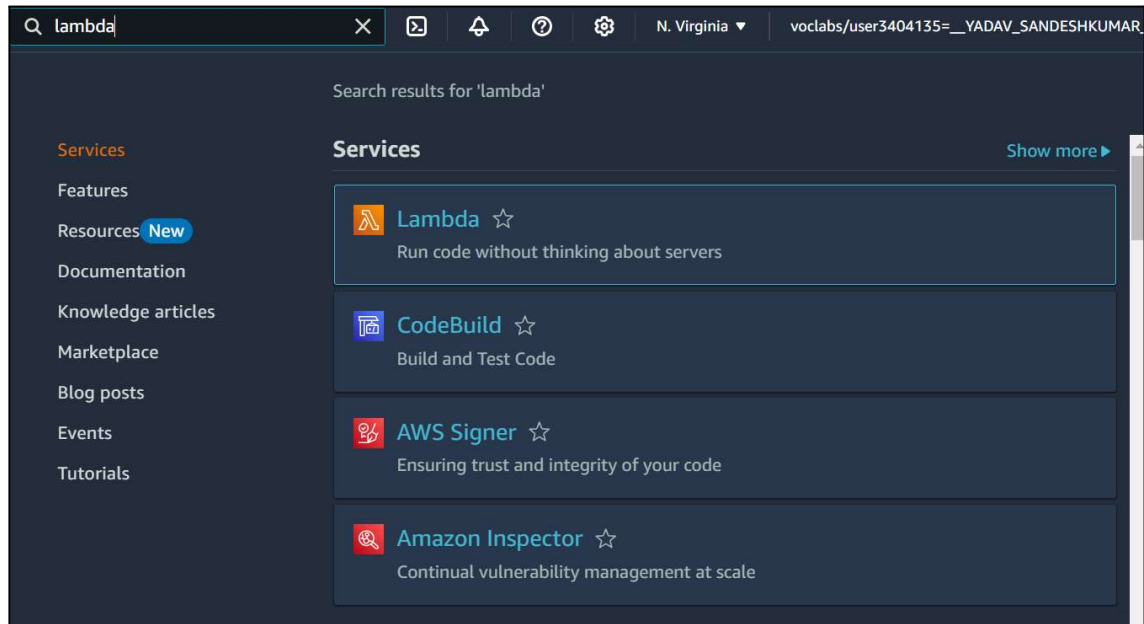


Experiment No: 11

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

CREATION OF LAMBDA FUNCTION:

Step1 : Log in to your AWS Personal or Academy account. Navigate to Lambda, then select the 'Create Function' button



Step 2 : Give your Lambda function a name and choose a programming language. The code editor only supports Node.js, Python, and Ruby, so in my case I have chosen Python 3.12. Set the architecture to x86. For the execution role, select 'Use an existing role,' then pick 'Lab role' from the dropdown menu under existing roles .

(This is because the Lab role already has the permissions needed for Lambda to run properly, so you don't need to create a new role from scratch. It's a quicker and more convenient option)

[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.

Basic information

Function name
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 [Info](#)
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Python 3.12 ▼

↻

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

☒ x86_64

☐ arm64

▼ 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

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.

► Additional Configurations

Use additional configurations to set up code signing, function URL, tags, and Amazon VPC access for your function.

Cancel

Create function

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

Lambda > Functions > Sandesh_Lambda

Sandesh_Lambda

Throttle Copy ARN Actions

▼ Function overview Info

Export to Application Composer Download

Diagram Template

Sandesh_Lambda

Layers (0)

+ Add trigger

+ Add destination

Description

-

Last modified

in 2 seconds

Function ARN

arn:aws:lambda:us-east-1:073011525842:function:Sandesh_Lambda

Function URL [Info](#)

-

Successfully created Lambda function

Step 3 : To view or change the basic settings, go to the 'Configuration' tab and click 'Edit' under 'General settings.' (THIS STEP IS OPTIONAL)

The screenshot shows the AWS Lambda console's Configuration tab. On the left, a sidebar lists configuration options: General configuration, Triggers, Permissions, Destinations, Function URL, and Environment variables. The main area displays the 'General configuration' section with an 'Info' link and an 'Edit' button. The configuration details are as follows:

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

You can add a description and adjust the memory and timeout settings. I've changed the timeout to 1 second, as that's enough for now.

The screenshot shows the 'Edit basic settings' page in the AWS Lambda console. The breadcrumb trail is 'Lambda > Functions > Sandesh_Lambda > Edit basic settings'. The page title is 'Edit basic settings'. Below the title, there's a 'Basic settings' section with an 'Info' link. The settings are as follows:

- Description - optional:** A text input field containing 'Basic Settings of Sandesh_Lambda'.
- Memory:** A text input field containing '128' followed by 'MB'. Below it, a note says 'Your function is allocated CPU proportional to the memory configured.' and 'Set memory to between 128 MB and 10240 MB'.
- Ephemeral storage:** A text input field containing '512' followed by 'MB'. Below it, a note says 'You can configure up to 10 GB of ephemeral storage (/tmp) for your function. View pricing' with a link icon. Below that, it says 'Set ephemeral storage (/tmp) to between 512 MB and 10240 MB'.
- SnapStart:** A dropdown menu currently showing 'None'. Below it, a note says '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' with a link icon. Below that, it says 'Supported runtimes: Java 11, Java 17, Java 21'.
- Timeout:** Two text input fields, one containing '0' followed by 'min' and another containing '1' followed by 'sec'.

The screenshot shows the AWS Lambda console's Configuration tab after editing the settings. The sidebar and tabs are the same as in the first screenshot. The 'General configuration' section now shows the updated settings:

Property	Value
Description	Basic Settings of Sandesh_Lambda
Memory	128 MB
Ephemeral storage	512 MB
Timeout	0 min 1 sec
SnapStart	None

Step 4: Go to the 'Test' tab and click 'Create a new event.' Give the event a name, set 'Event Sharing' to private, and choose the 'hello-world' template.

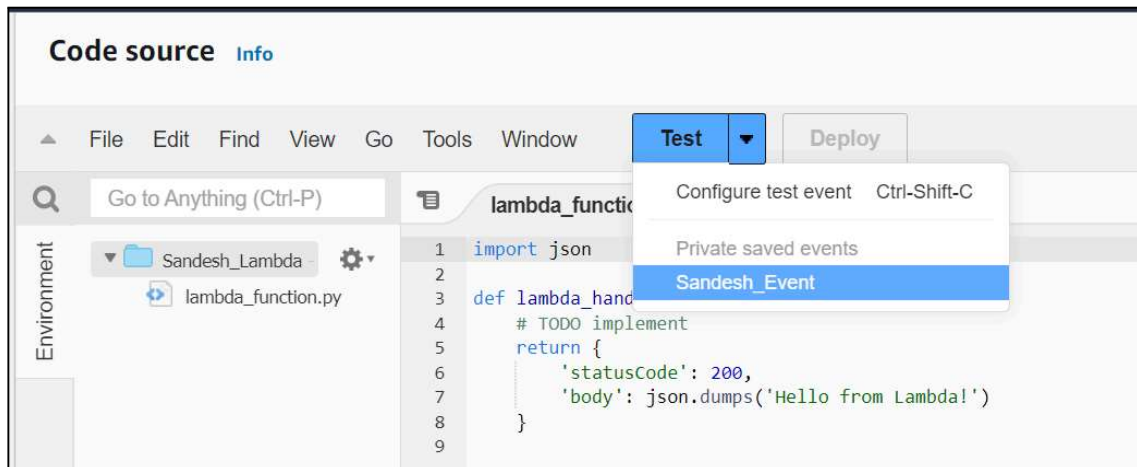
We basically create a new event to test and verify your Lambda function; setting Event Sharing to private keeps it secure and choosing the "hello-world" template provides a simple structure for testing without complex inputs.

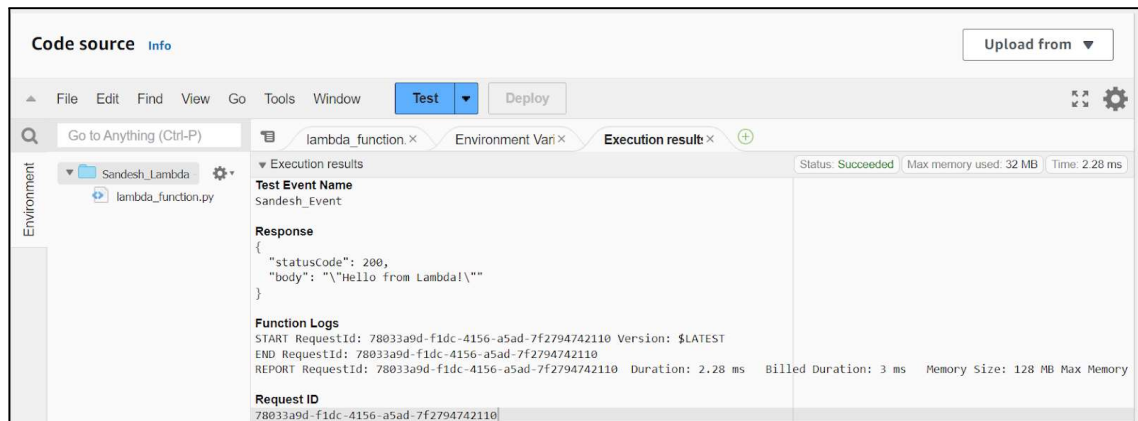
The screenshot shows the 'Test event' configuration page in the AWS Lambda console. At the top, there are 'Save' and 'Test' buttons. Below, a message states: 'To invoke your function without saving an event, configure the JSON event, then choose Test.' The 'Test event action' section has two options: 'Create new event' (selected) and 'Edit saved event'. The 'Event name' field contains 'Sandesh_Event' with a note: 'Maximum of 25 characters consisting of letters, numbers, dots, hyphens and underscores.' The 'Event sharing settings' section has two radio buttons: 'Private' (selected) and 'Shareable'. The 'Template - optional' dropdown is set to 'hello-world'. The 'Event JSON' section shows a pre-filled JSON object:

```
{  "key1": "value1",  "key2": "value2",  "key3": "value3"}
```

 with a 'Format JSON' button.

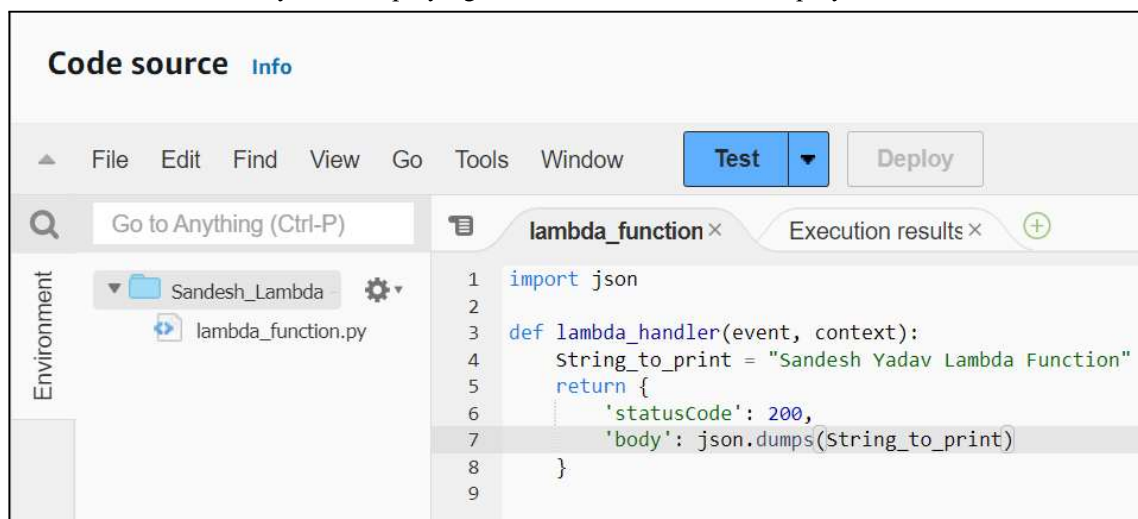
Step 5: In the Code section, select the event you created from the dropdown menu under 'Test,' then click 'Test.' You should see the output below."

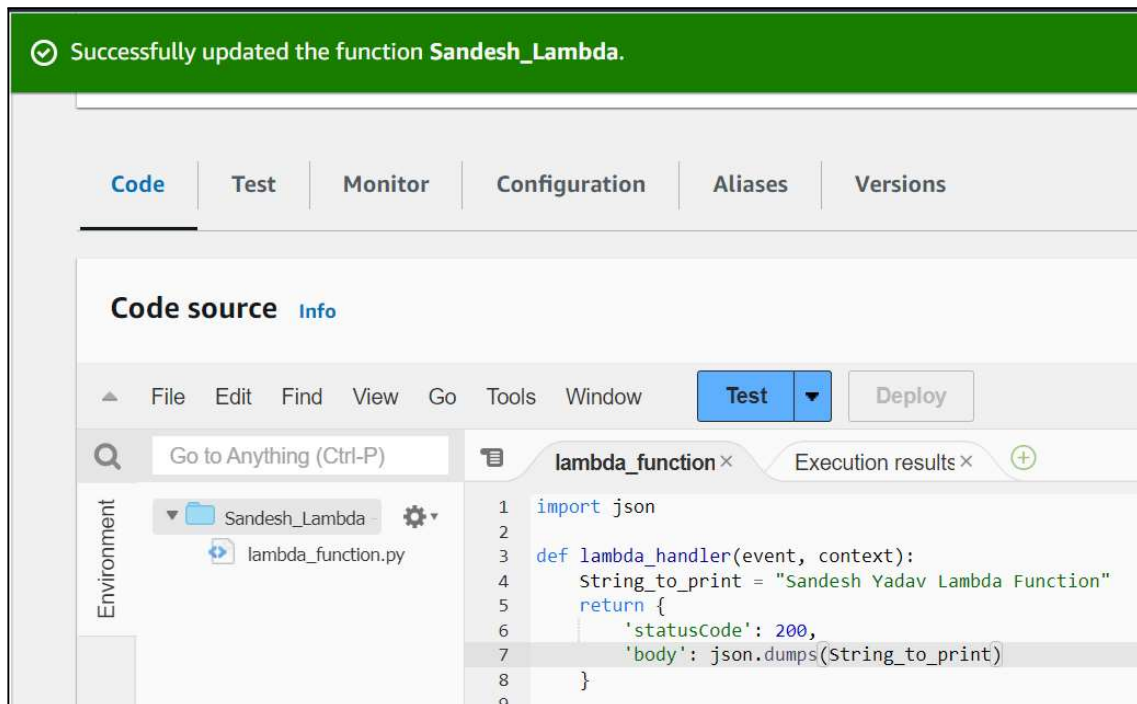




You select the created event to run the specific test you set up, and clicking 'Test' executes your Lambda function to check if it works as expected and produces the desired output

Step 6: You can edit your lambda function code. I have changed the code to display the new String. After Changing save it by Control + S and click on Deploy . Make sure you have internet connectivity while deploying or else it will show failed deployment





Step 7: Click on 'Test' to see the output. You'll get a status code of 200 which means "OK" and indicates that the request was successful, your string output, and the function logs, showing that it was deployed successfully.



CONCLUSION:

In this experiment, we created and deployed an AWS Lambda function using Python. Key steps included setting up the function, adjusting the timeout, creating a test event, and modifying the code. AWS Lambda simplifies development by handling infrastructure, allowing developers to focus on code, while automatically scaling and managing servers.