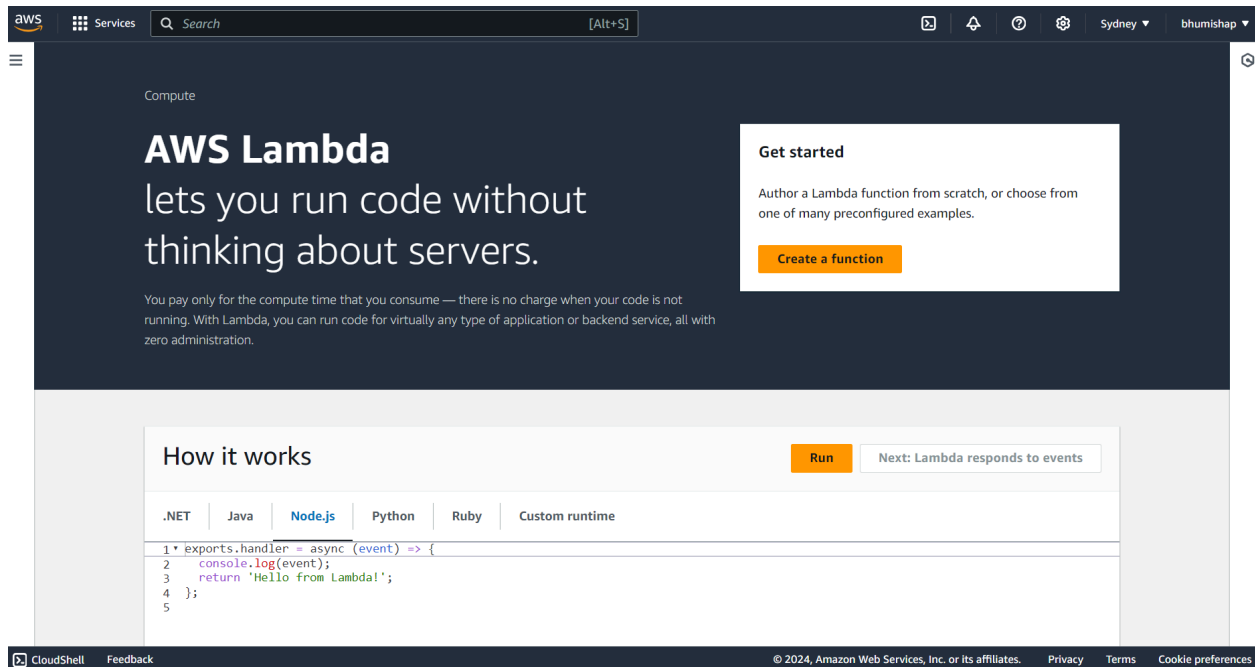


EXPERIMENT NO. 11

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

Steps To create the lambda function:

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



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 we 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'. The 'Basic information' section contains a 'Function name' field with the value 'bhumi-lambda' and a 'Runtime' dropdown menu set to 'Python 3.12'. The footer of the form includes a 'C' icon for copying the code.

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 `bhumi-lambda-role-hzeu5hul`, with permission to upload logs to Amazon CloudWatch Logs.

► Advanced settings

Cancel

Create function

Thus, we have successfully created a lambda function named `bhumi-lambda`.

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

Lambda > Functions > bhumi-lambda

bhumi-lambda

Throttle

Copy ARN

Actions

▼ Function overview [Info](#)

Export to Application Composer

Download

Diagram

Template

bhumi-lambda

Layers (0)

+ Add trigger

+ Add destination

Description

-

Last modified

7 seconds ago

Function ARN

`arn:aws:lambda:ap-southeast-2:010928192223:function:bhumi-lambda`

Function URL

[Info](#)

-

Info

Tutorials

Learn how to implement common use cases in AWS Lambda.

Create a simple web app

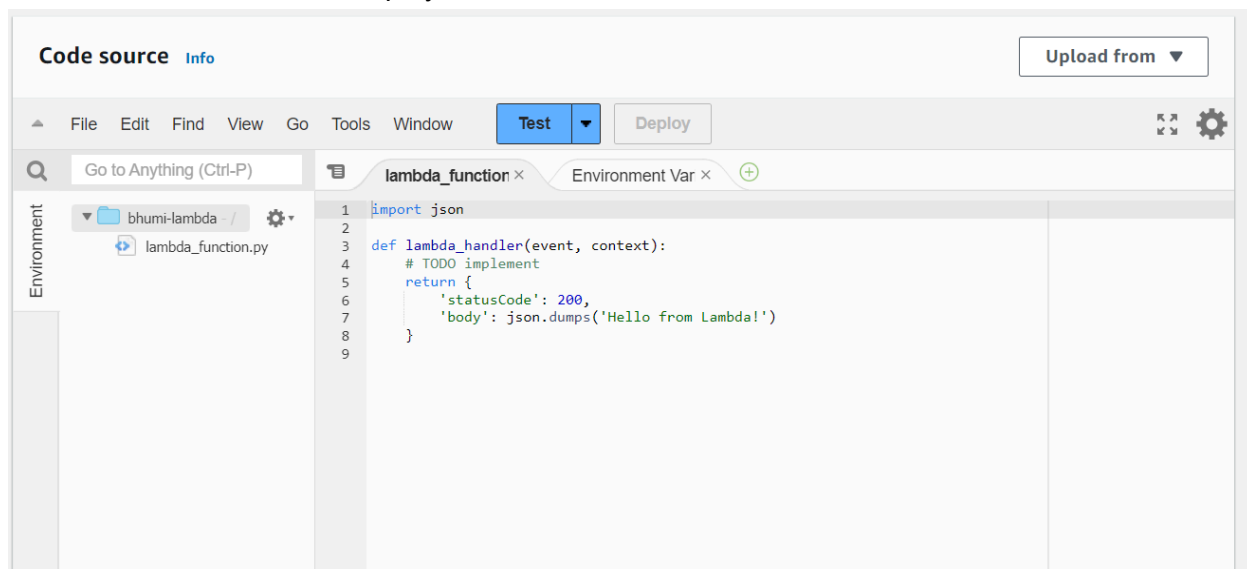
In this tutorial you will learn how to:

- Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage
- Invoke your function through its function URL

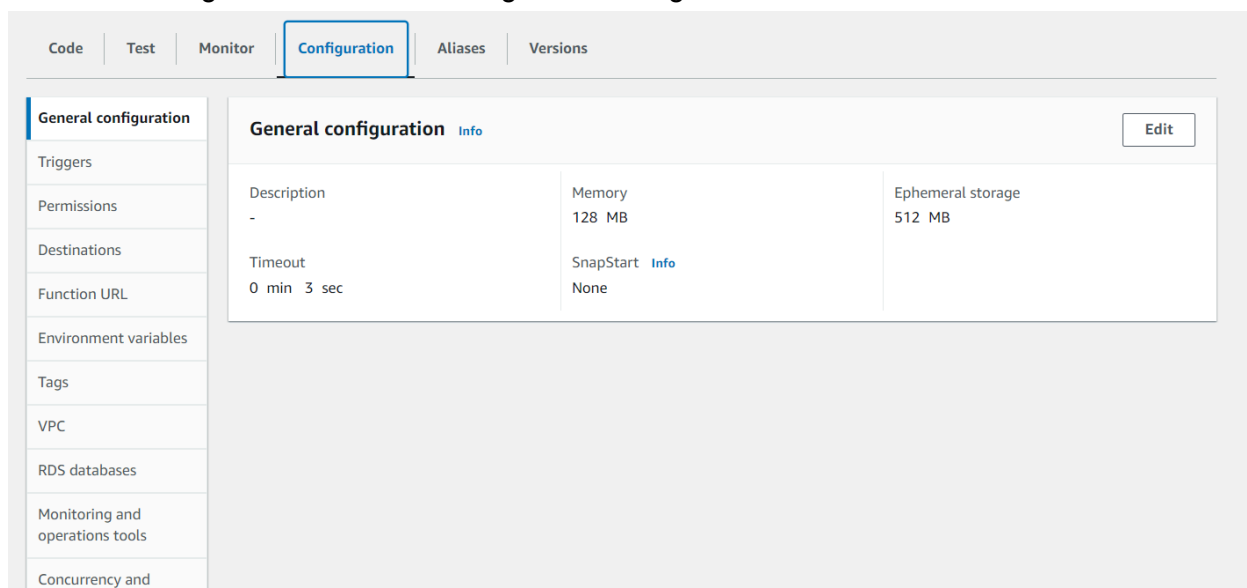
[Learn more](#)

Start tutorial

This is how the function is displayed.



Go to the Configuration tab to see the general configuration of our function.



We want to edit the timeout time and the rest can be kept the default.

Here, we can enter a description and change Memory and Timeout. I've changed the Timeout period to 1 sec since that is sufficient for now.

To edit, we go to the Edit button seen in the above screenshot.

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.

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](#).

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

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

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.

[Code](#) | [Test](#) | [Monitor](#) | [Configuration](#) | [Aliases](#) | [Versions](#)

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

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

Event JSON

[Format JSON](#)

```
1 {  
2   "key1": "value1",  
3   "key2": "value2",  
}
```

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

bhumi-event

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

Event sharing settings

☒ Private ☐ Shareable

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

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

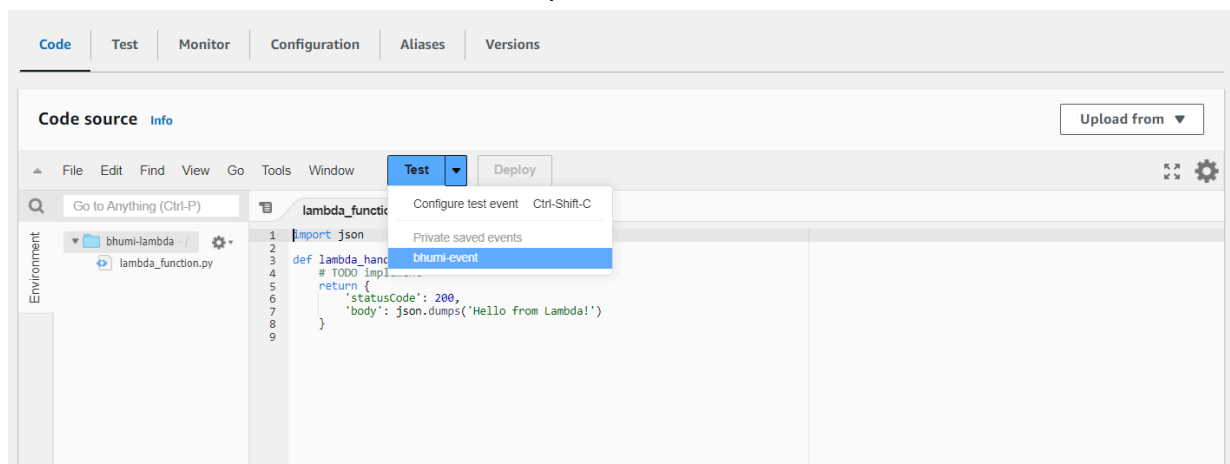
Template - optional

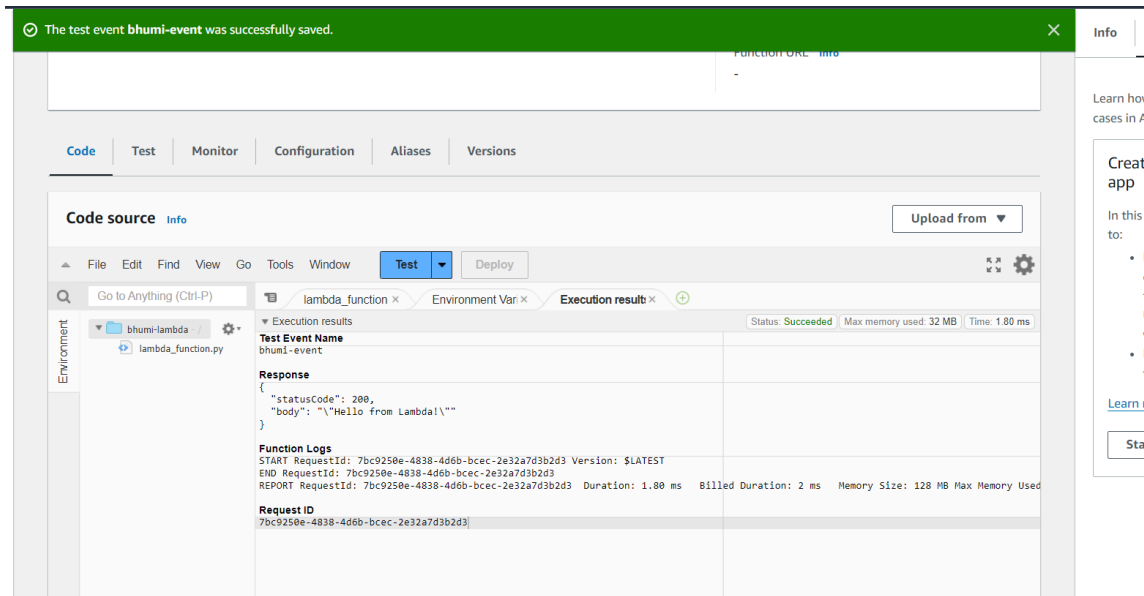
hello-world

Event JSON Format JSON

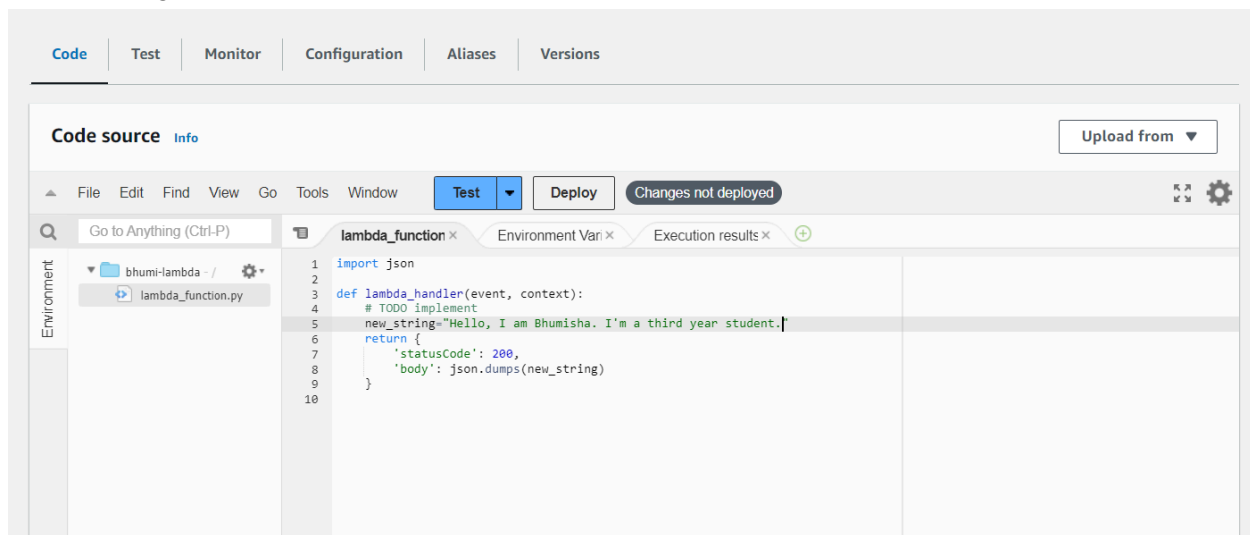
```
1 {
2   "key1": "value1",
3   "key2": "value2",
4   "key3": "value3"
5 }
```

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

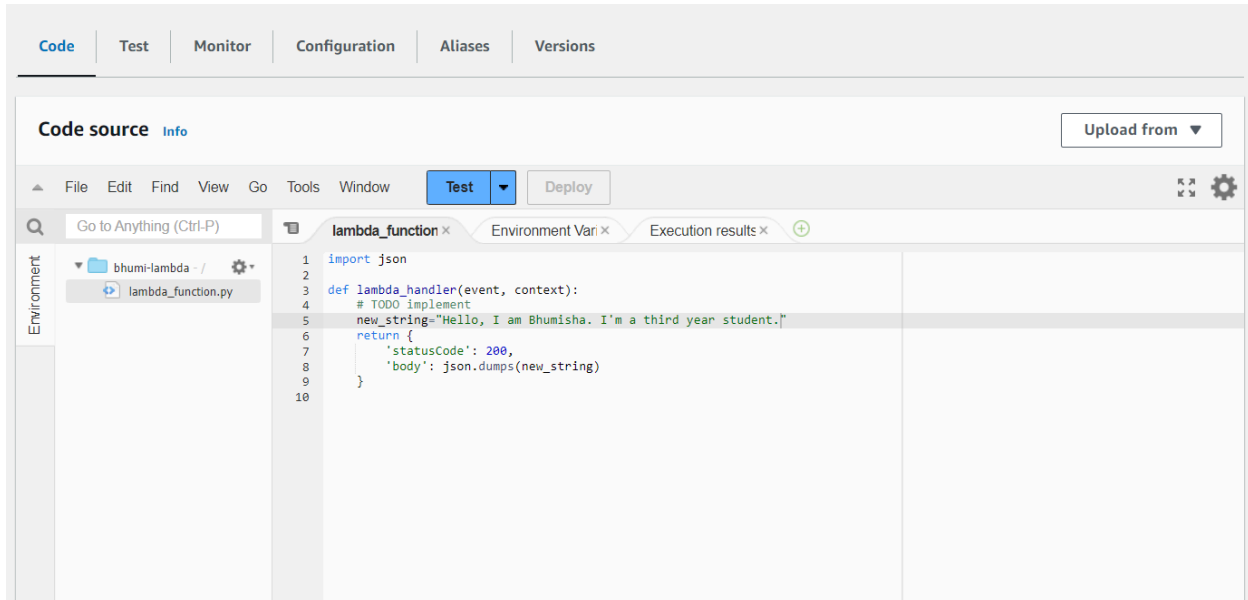




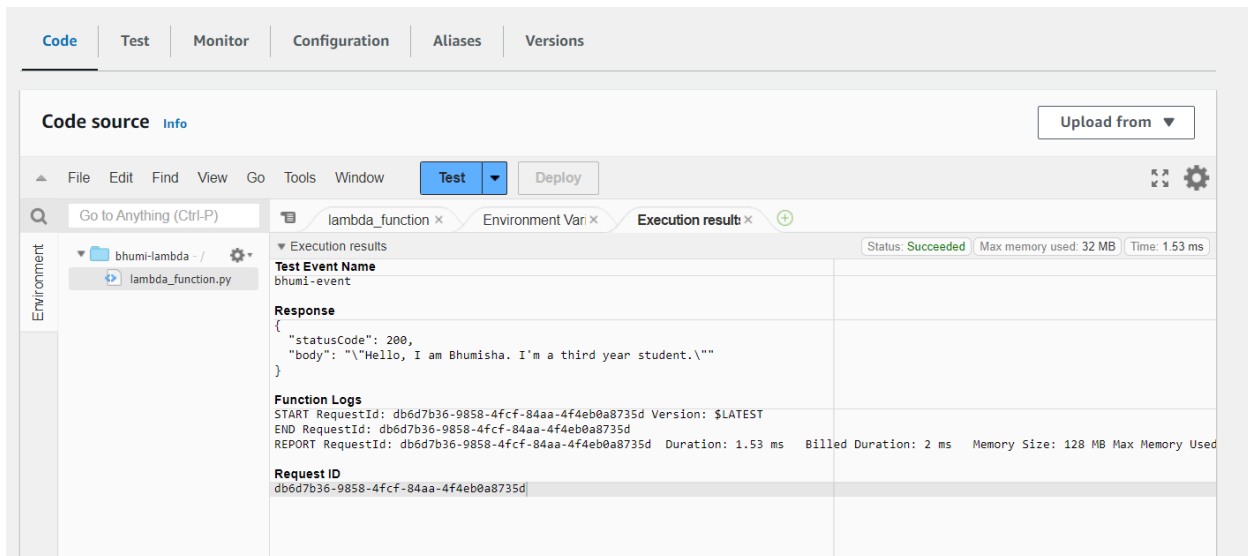
5. You can edit your lambda function code. I have changed the code to display the new string.



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



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.



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.