

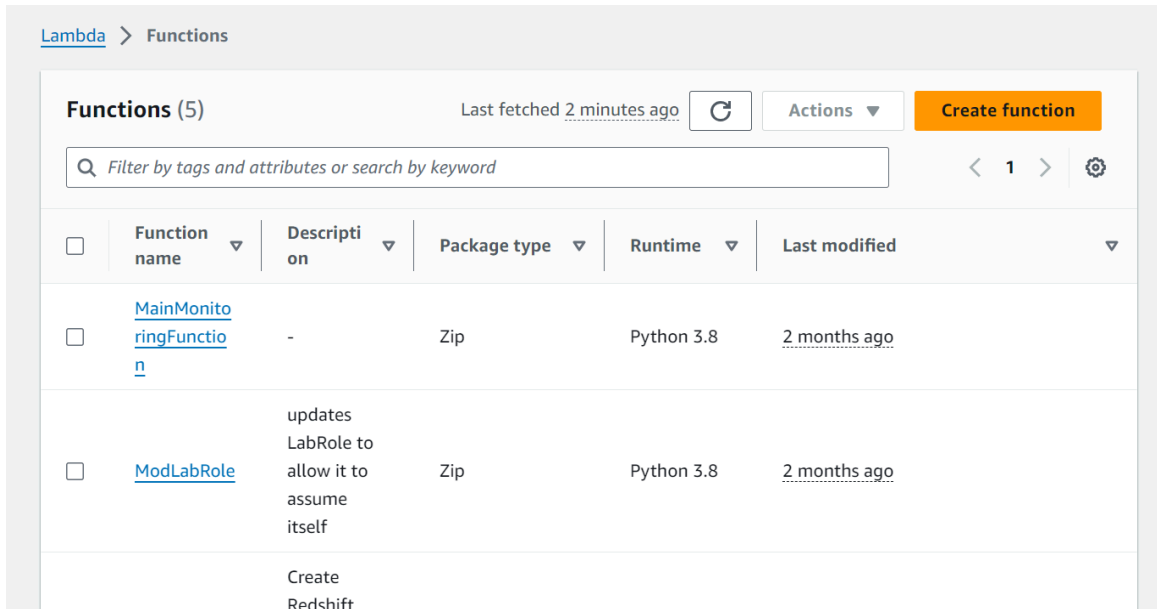
## Experiment No:11

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

### Steps:

1. Go to AWS ACADEMY.
2. **Create the lambda function:**

Firstly, Search lambda, then Open lambda and then click on create function button.




3. Now Give a name to your Lambda function,

The screenshot shows the 'Create function' wizard in the AWS Lambda console. The breadcrumb is 'Lambda > Functions > Create function'. The title is 'Create function' with an 'Info' link. Below the title, it says 'Choose one of the following options to create your function.' There are three radio button options: 'Author from scratch' (selected), 'Use a blueprint', and 'Container image'. The 'Author from scratch' option has a subtext 'Start with a simple Hello World example.' Below the options is a section titled 'Basic information'. Under 'Function name', there's a text input field containing 'anshi-lambda' and a subtext 'Enter a name that describes the purpose of your function.' Below the input field, a note states: '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 (\_).'

4. 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.

**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


Thus the Lambda function was created successfully.

**anshi-lambda** [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](#)  
-

5. Then go to code section.

**Code source** [Info](#) [Upload from](#)

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

Go to Anything (Ctrl-P)

Environment: anshi-lambda  Environment Variables: 

lambda\_function.py

```
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 to Edit the basic settings, go to configuration then click on edit.

**General configuration** [Info](#) [Edit](#)

<b>Description</b> -	<b>Memory</b> 128 MB	<b>Ephemeral storage</b> 512 MB
<b>Timeout</b> 0 min 3 sec	<b>SnapStart</b> <a href="#">Info</a> None	

Now enter a description and change Memory and Timeout. Here, I've changed the Timeout period to 1 sec.

**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](#).  

None

  
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

**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.

6. Now Click on the Test tab then select Create a new event, give a name to the event here i have given name as “our\_event” and then 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  
  
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](#)

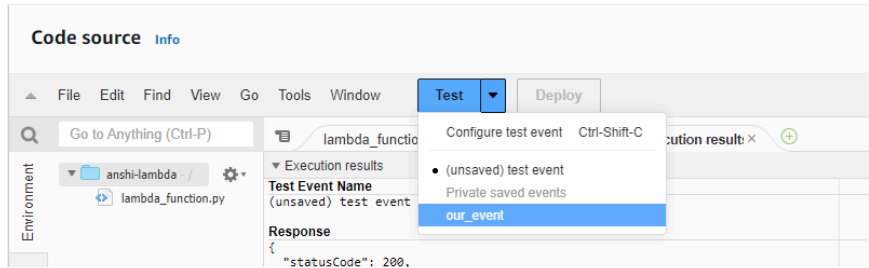
Template - *optional*  

hello-world

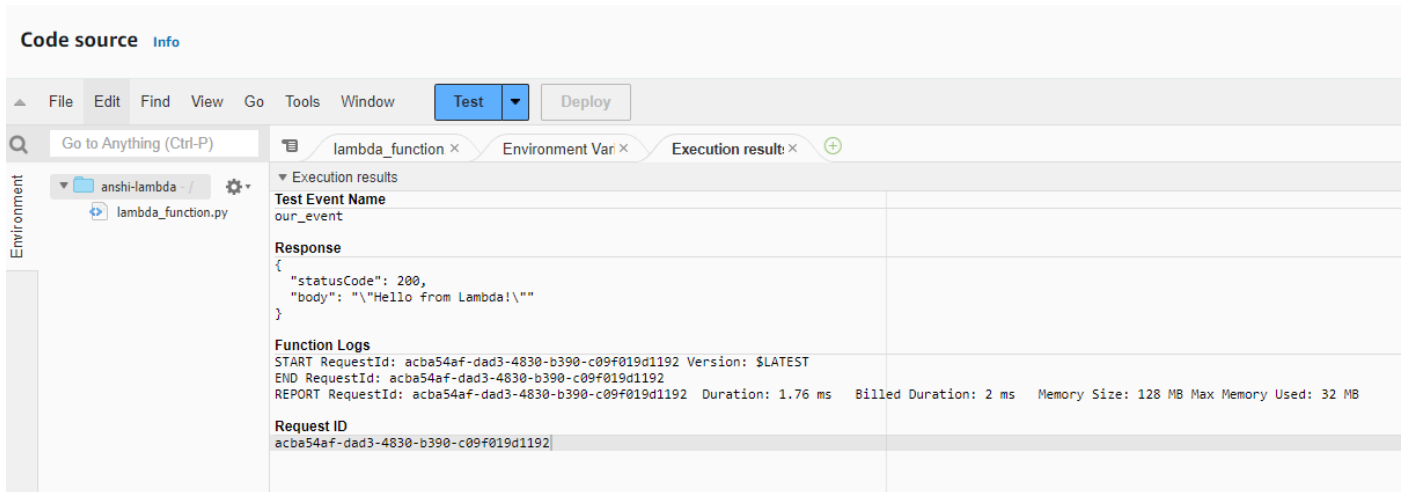
**Event JSON** Format JSON  

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

7. **Testing & Deployment:** Now In Code section select the created event (our\_event) from the dropdown of test ,then click on test .

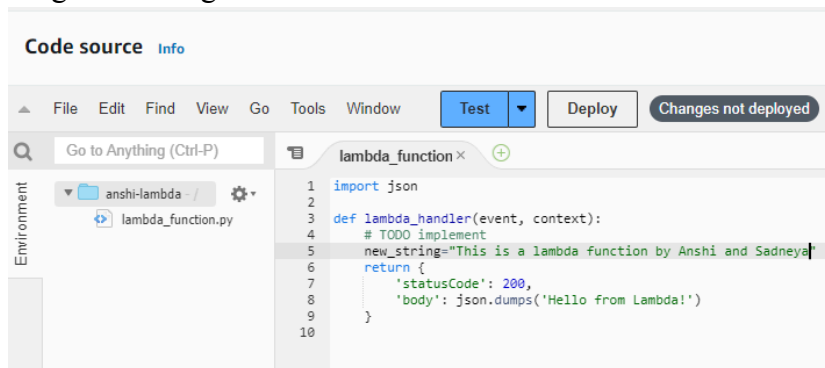


8. Now you will see the following output.

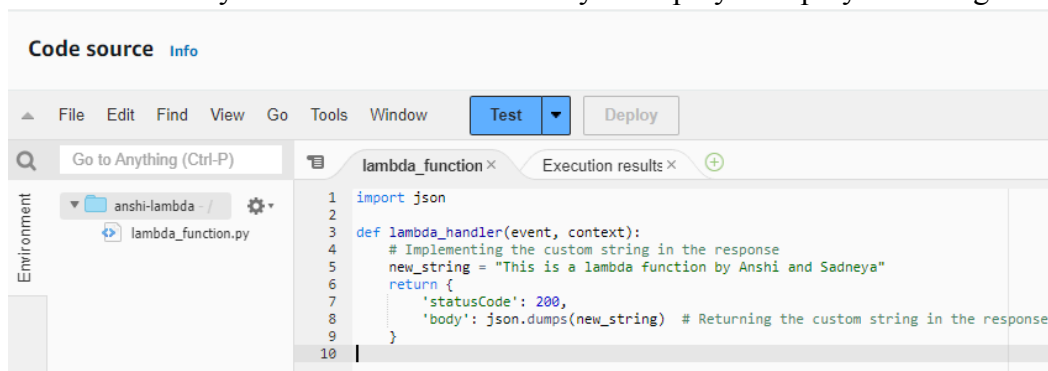


## 9. Making changes

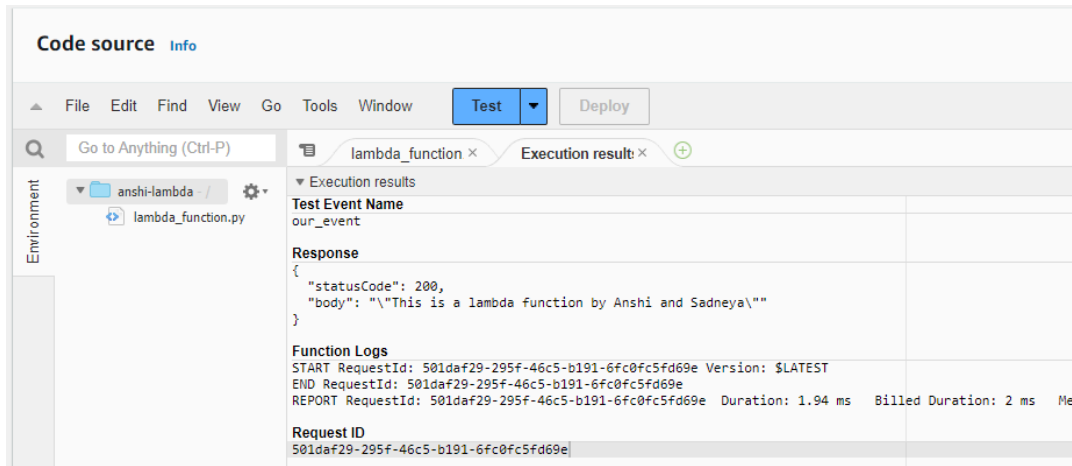
You can edit your lambda function code. Here I have created a new string name “new\_string” and assigned a string to it.



Now save it by ctrl+s and then click finally on deploy to deploy the changes.



**10. Testing and redeploying changes** Now click on the test and observe the output. Thus Output gives status code 200. Thus deployment is done successfully.



**Conclusion:** In this experiment, we have successfully created an AWS Lambda function. After we have chosen the python language for writing the function. Then we edited the basic settings, including adjusting the timeout to 1 second. Then we tested it and finally deployed it. Thus it got deployed successfully. Additionally, we modified the Lambda function's code and redeployed it to observe the changes in real-time. This provided information about simplicity of AWS Lambda in creating serverless application.