

## AWS Lambda

**AWS Lambda** is a serverless computing service that lets you run code without managing servers. You write and deploy code, and Lambda handles the *execution, scaling, and infrastructure management*. You only pay for the compute time used.

### Advantages of Lambda

1. Serverless Architecture: No need to manage or provision servers.
2. Cost-Effective: Pay only for the compute time your code runs.
3. Scalability: Automatically scales to handle changes in workload.
4. Event-Driven: Can be triggered by various AWS services or HTTP requests.
5. Flexibility: Supports multiple programming languages.
6. Easy Integration: Seamlessly integrates with other AWS services.
7. Reduced Administrative Overhead: No server maintenance or patching required.

## Lab Steps

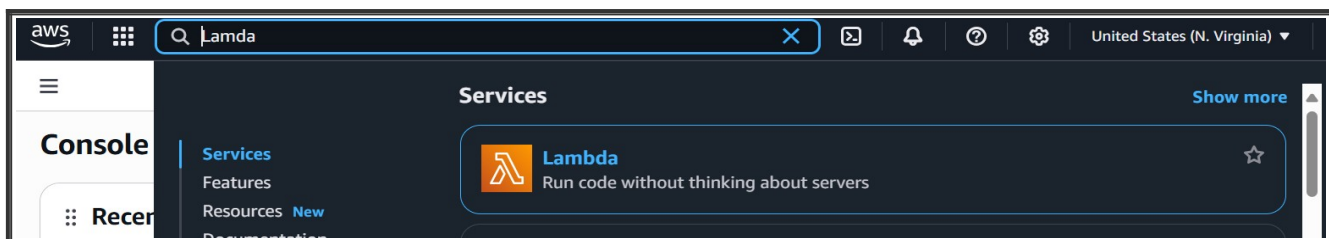
### Task 1 :- Sign in to AWS Management Console

1. Click on the **Open Console** button, and you will get redirected to AWS Console in a new browser tab.
2. On the AWS sign-in page,
  - Leave the Account ID as default. Never edit/remove the 12 digit Account ID present in the AWS Console. otherwise, you cannot proceed with the lab.
  - Now copy your **User Name** and **Password** in the Lab Console to the **IAM Username and Password** in AWS Console and click on the **Sign in** button.
3. Once Signed In to the AWS Management Console, Make the default AWS Region as **US East (N. Virginia) us-east-1**.

**Note:-** If you face any issues, please go through [FAQs and Troubleshooting for Labs](#).

### Task 2 :- Create a function

1. Search for Lambda.



2. Click on [Lambda](#).
3. Click on [Create function](#).
4. Choose :- **Author from scratch**

## 5. Write your function name.

**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.  

▼

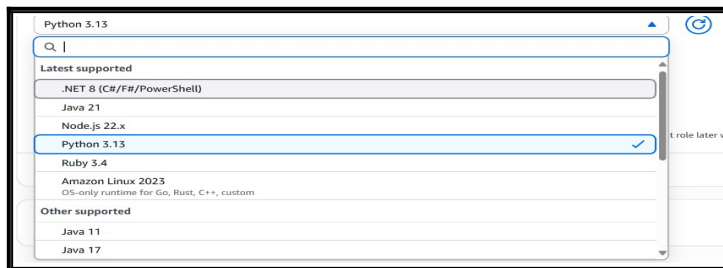


**Example:** \_ DemoFunction

## 6. Choose the language to use to write your function in **Runtime**.

**Example :-** we are going for **Python** here

**Note:** There are many language options available here, so you can choose any of them as per your convenience.

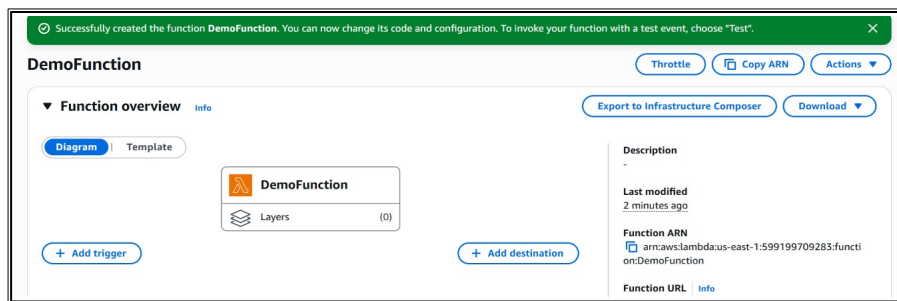


7. Choose **x84\_64** in **Architecture**

8. In **Execution role**:- Create a new role with basic Lambda permissions

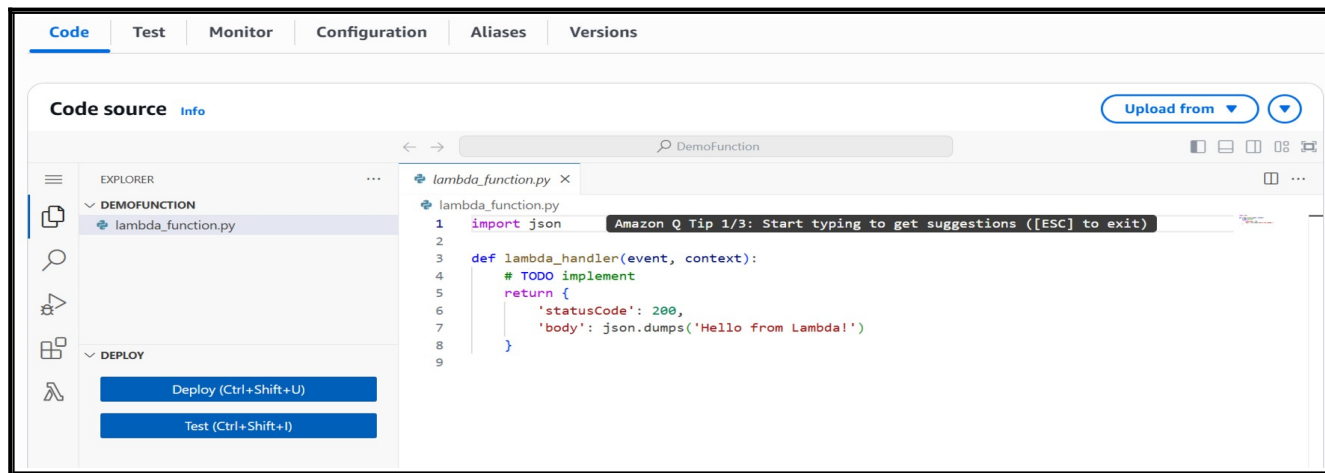
9. Leave the rest as **default** and click on **create function**.

10. **Function is successfully created.**



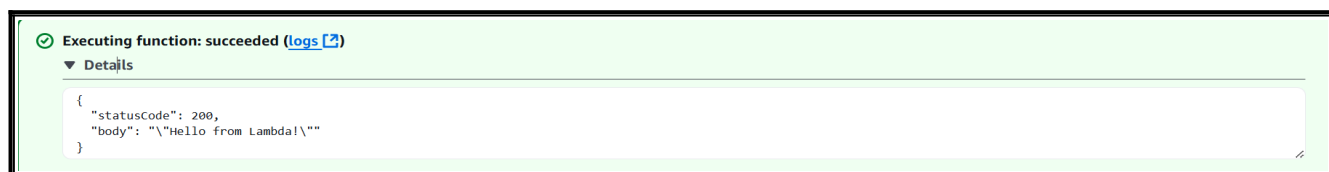
11. Go to **code** tab.

12. Here the default code will be written which will print **Hello from lambda!**



13. Now go to the **Test** tab to see the output of the written code.

14. Click on **Test**.



**What if we need to return something other than “Hello from Lambda!” ?**

1. Change Hello from Lambda! To Hello everyone! .

2. Doing this will not give any output, we will have to **deploy** it after this.

3. Click on **Deploy**.

✓ Successfully updated the function **DemoFunction**.

## Code source [Info](#)

EXPLORER

▼ DEMOFUNCTION

lambda\_function.py

▼ DEPLOY

Deploy (Ctrl+Shift+U)

Test (Ctrl+Shift+I)

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 everyone!')
8     }
9
```

4. This will be output :-

✓ Executing function: succeeded ([logs](#))

▼ Details

```
{
  "statusCode": 200,
  "body": "\"Hello everyone!\""
}
```