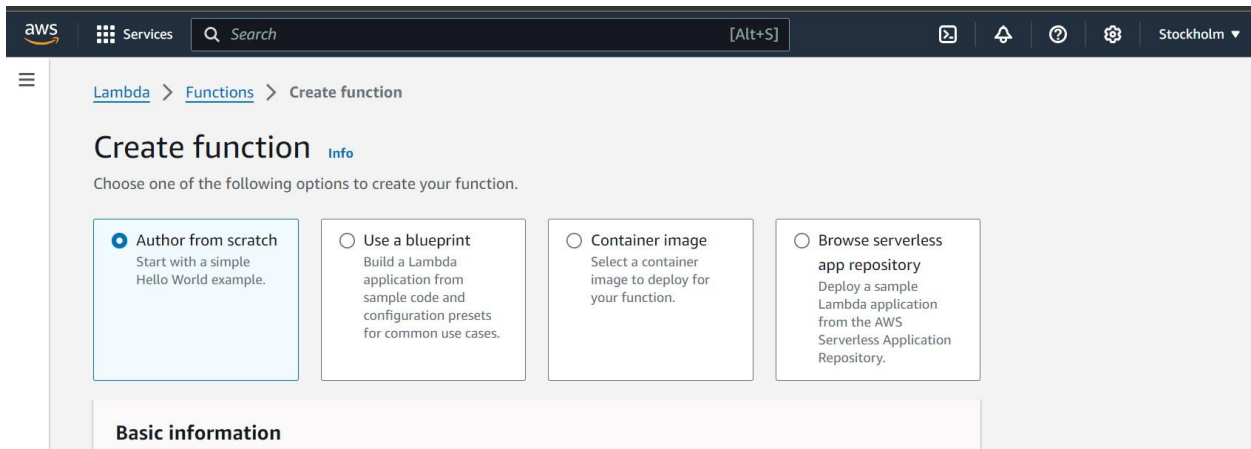


AIM: To understand AWS Lambda, its workflow, various functions and create your first Lambda functions using Python / Java / Nodejs.

STEP1: Go on your AWS console account and search for lambda and then go on create function. Select the author from scratch, add function name and then, choose a runtime env for your function, under the dropdown, you can see all the options AWS supports, Python, Nodejs, .NET and Java being the most popular ones.



Basic information

Function name
Enter a name that describes the purpose of your function.

Use only letters, numbers, hyphens, or underscores with no spaces.

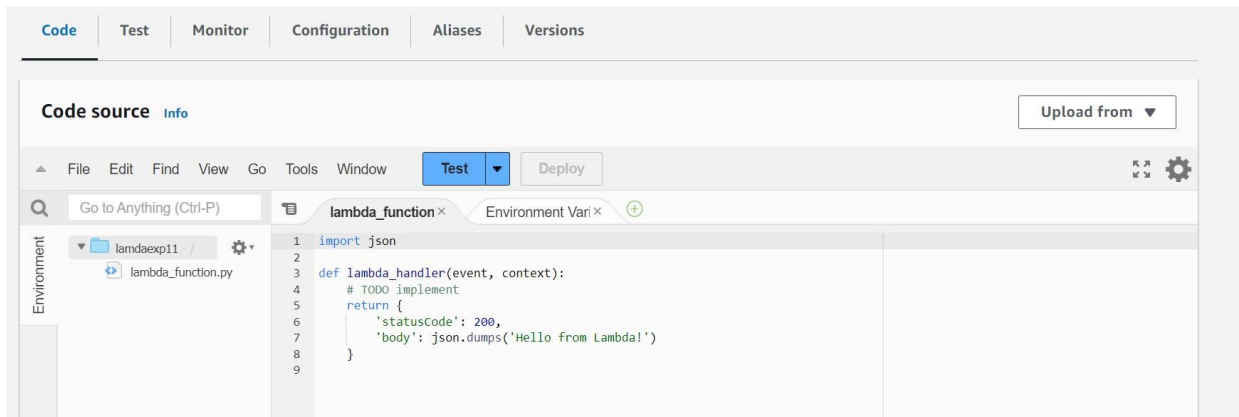
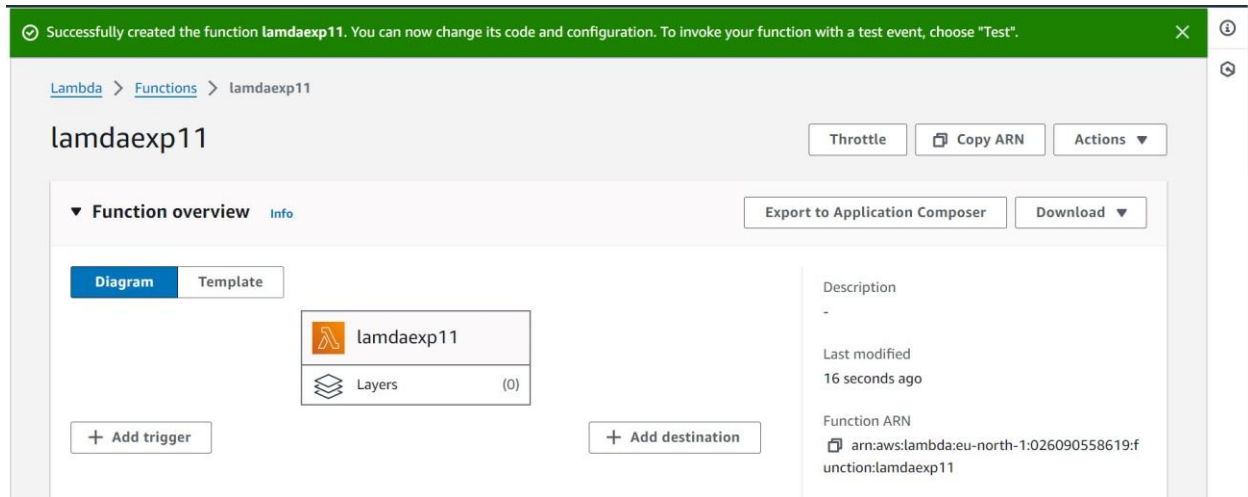
Runtime [Info](#)
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Architecture [Info](#)
Choose the instruction set architecture you want for your function code.
☒ x86_64
☐ arm64

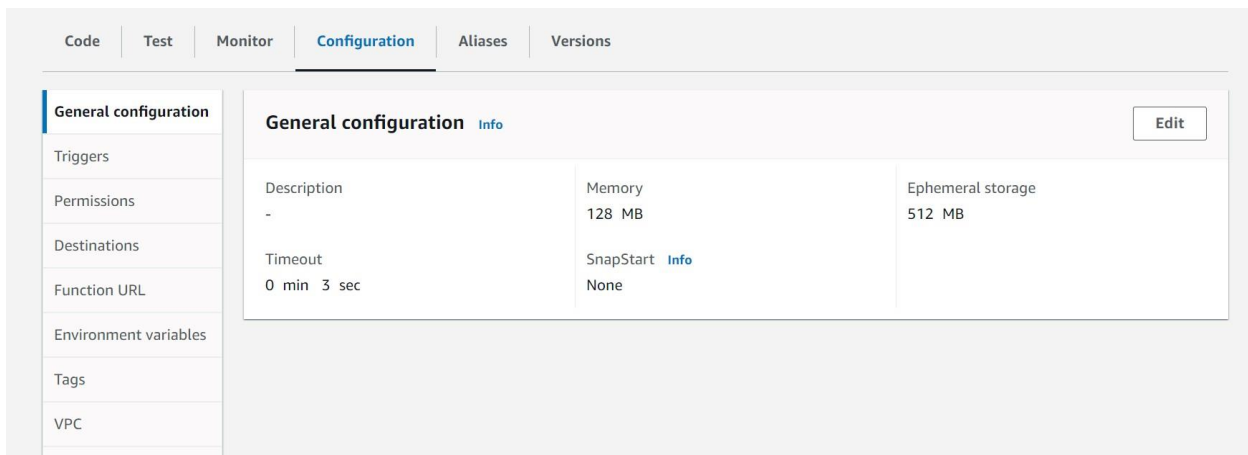
Permissions [Info](#)
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

► **Change default execution role**

STEP 2: After the function is created successfully go on code write the default code and then configure them.



STEP 3: Then go on edit basic settings and add the description and then save it .



[Lambda](#) > [Functions](#) > [lamdaexp11](#) > Edit basic settings

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](#).

STEP 4: Click on “use an existing role” option and then ahead add the role and save it.

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.

▼

[View the lamdaexp11-role-vj5j9g95 role](#) on the IAM console.

STEP 5: Go on configure test event click on “create new event” edit the event sharing accordingly and select hello world template for template option and then save it.

Configure test event

A test event is a JSON object that mocks the structure of requests emitted by AWS services to invoke a Lambda function. Use it to see the function's invocation result.

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

MyEventName

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

Cancel Invoke Save

STEP 6: Click on the test and test the code.

STEP 7: The function is successfully added .

✓ The test event **testevent** was successfully saved.

File Edit Find View Go Tools Window Test Deploy

Go to Anything (Ctrl-P)

Environment

- lamdaexp11
 - lambda_function.py

Execution results

Status: Succeeded Max memory used: 32 MB Time: 3.10 ms

Test Event Name

testevent

Response

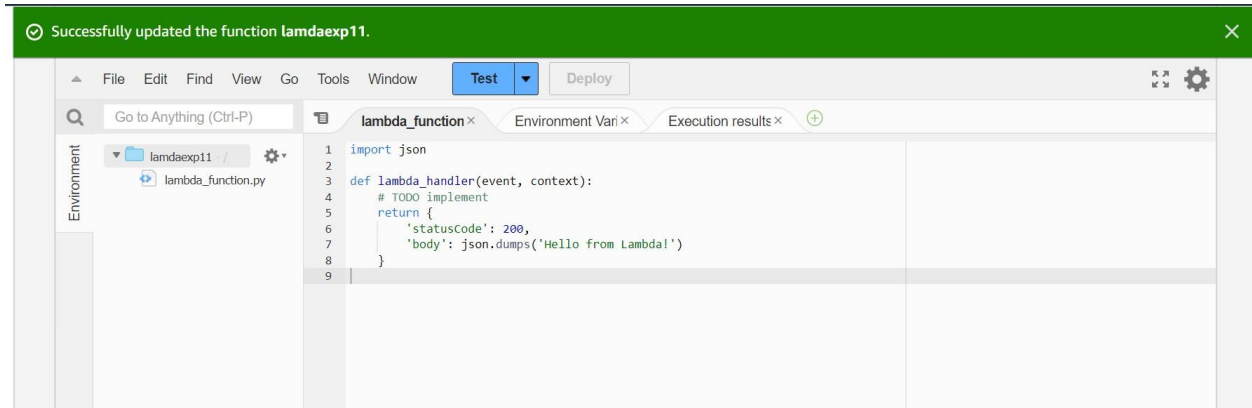
```
{
  "statusCode": 200,
  "body": "\"Hello from Lambda!\""
}
```

Function Logs

START RequestId: 3390556e-ac6a-452c-97db-d874e239e76a Version: \$LATEST
END RequestId: 3390556e-ac6a-452c-97db-d874e239e76a
REPORT RequestId: 3390556e-ac6a-452c-97db-d874e239e76a Duration: 3.10 ms Billed Duration: 4 ms Memory Size: 128 MB Max Memory

Request ID

3390556e-ac6a-452c-97db-d874e239e76a



Conclusion: In conclusion, the experiment successfully involved the creation, coding, and deployment of AWS Lambda function. By writing and refining the source code, we demonstrated the ability to implement specific functionality within the Lambda environment. The successful testing of the function confirmed its operational integrity and effectiveness in executing the desired tasks.