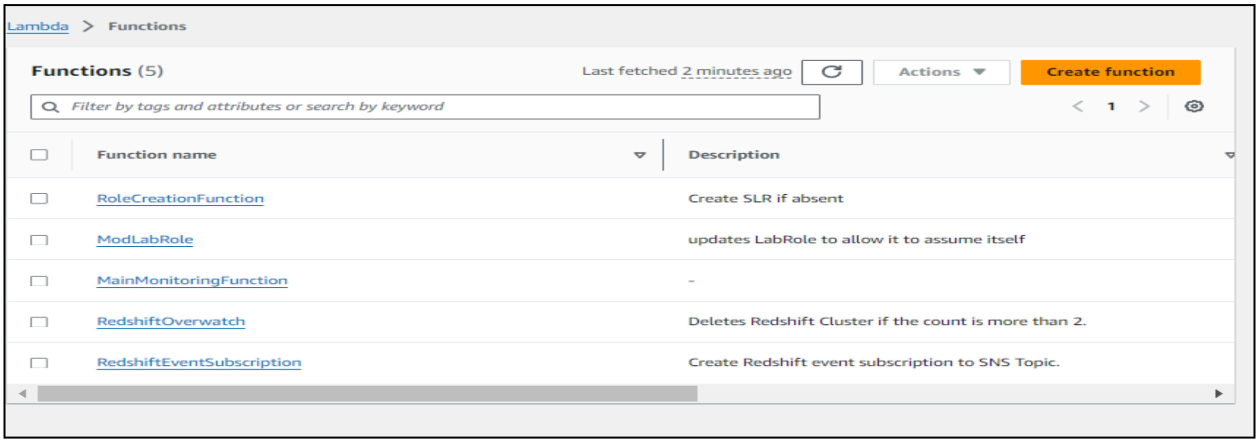
ADVANCE DEVOPS EXP-11

ANSH SARFARE **D15A/50**

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

Step 1: Login to your AWS Personal/Academy Accout.Open lambda andclick on create function button.



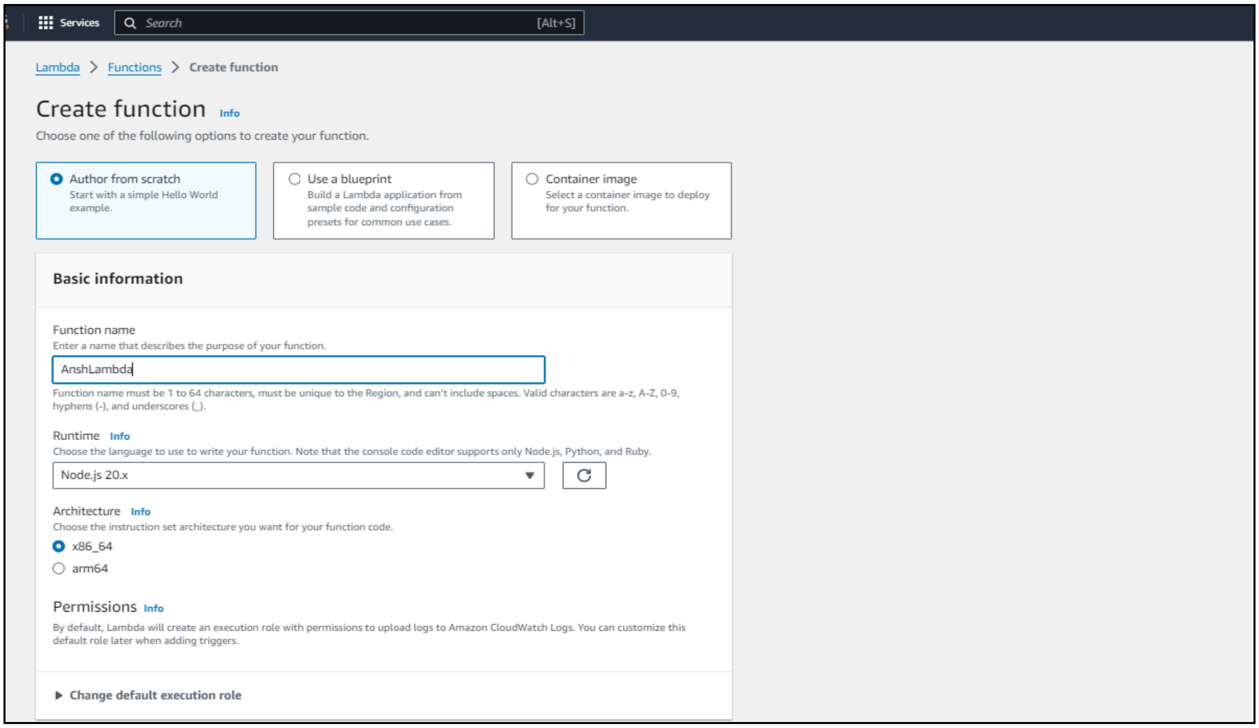
Lambda > Functions

Functions (5) Last fetched 2 minutes ago Actions ▼ Create function

Filter by tags and attributes or search by keyword

□	Function name	Description
□	RoleCreationFunction	Create SLR if absent
□	ModLabRole	updates LabRole to allow it to assume itself
□	MainMonitoringFunction	-
□	RedshiftOverwatch	Deletes Redshift Cluster if the count is more than 2.
□	RedshiftEventSubscription	Create Redshift event subscription to SNS Topic.

Step 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 will select Python 3.12 , Architecture as x86, and Exceution role to Create a new role with basic Lambda permissions.



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

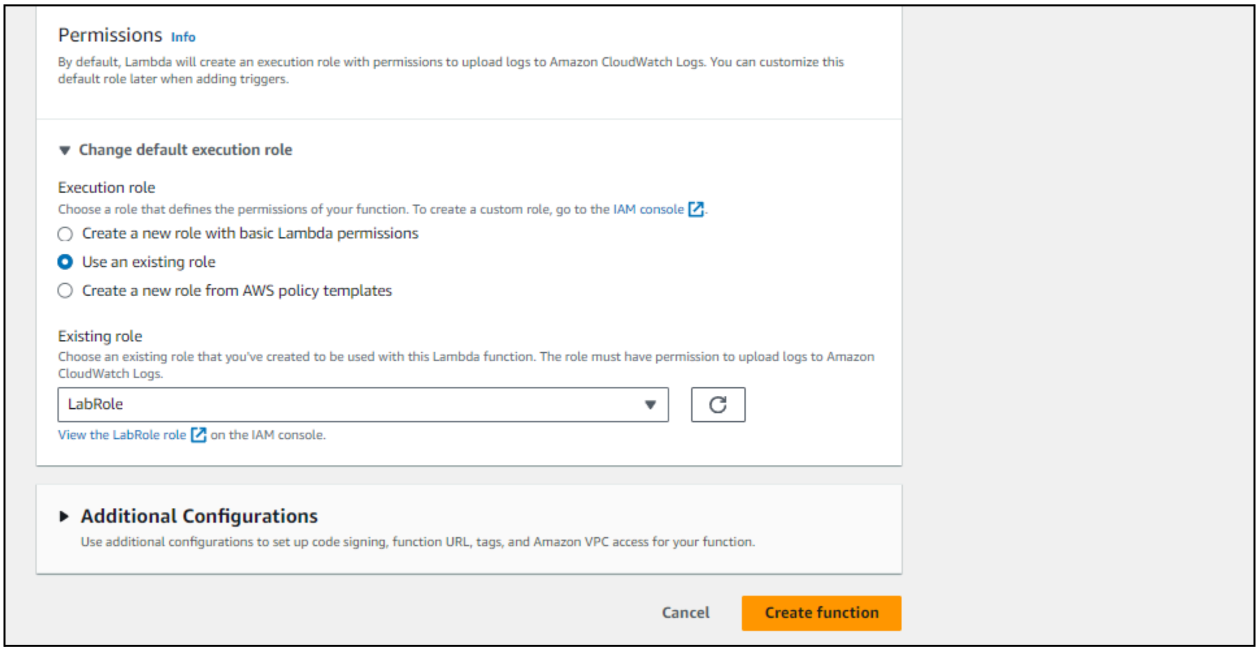
Function name
Enter a name that describes the purpose of your function.
AnshLambda
Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Node.js 20.x ▼

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



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

Existing role

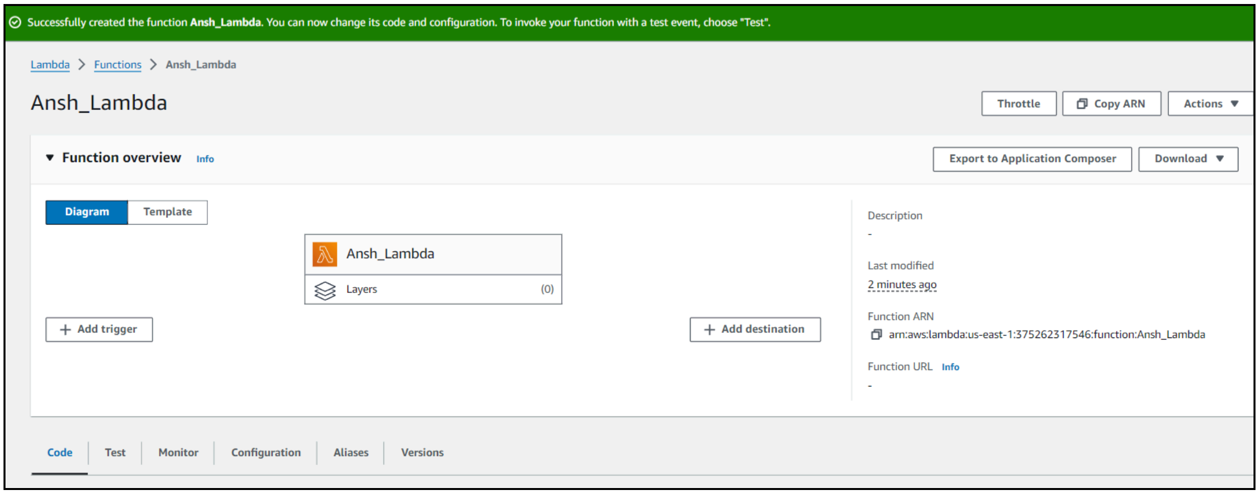
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

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.



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

Lambda > Functions > Ansh_Lambda

Ansh_Lambda

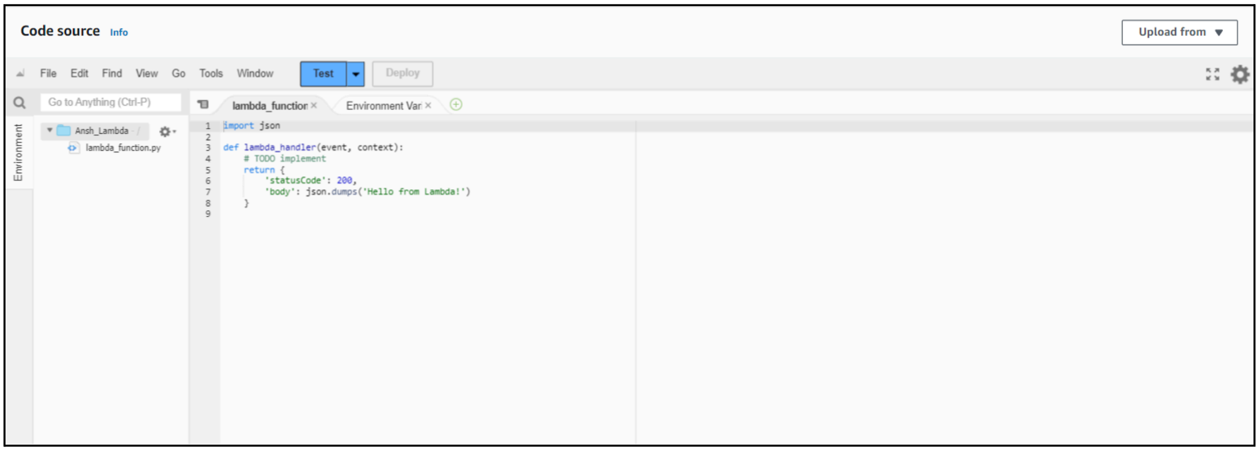
▼ Function overview Info

Description
-

Last modified
2 minutes ago

Function ARN
arn:aws:lambda:us-east-1:375262317546:function:Ansh_Lambda

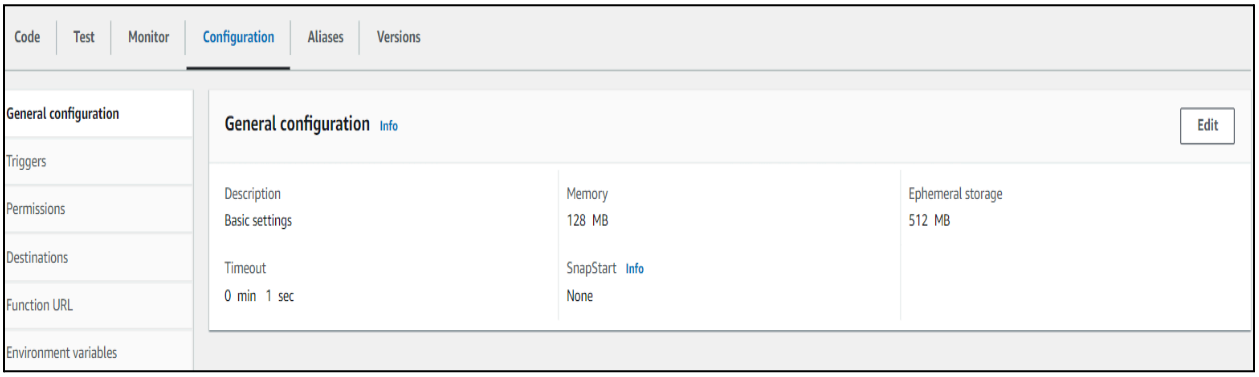
Function URL Info
-



Code source Info

```
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }

```

So See or Edit the basic settings go to configuration then click on edit general setting.



General configuration

Description
Basic settings

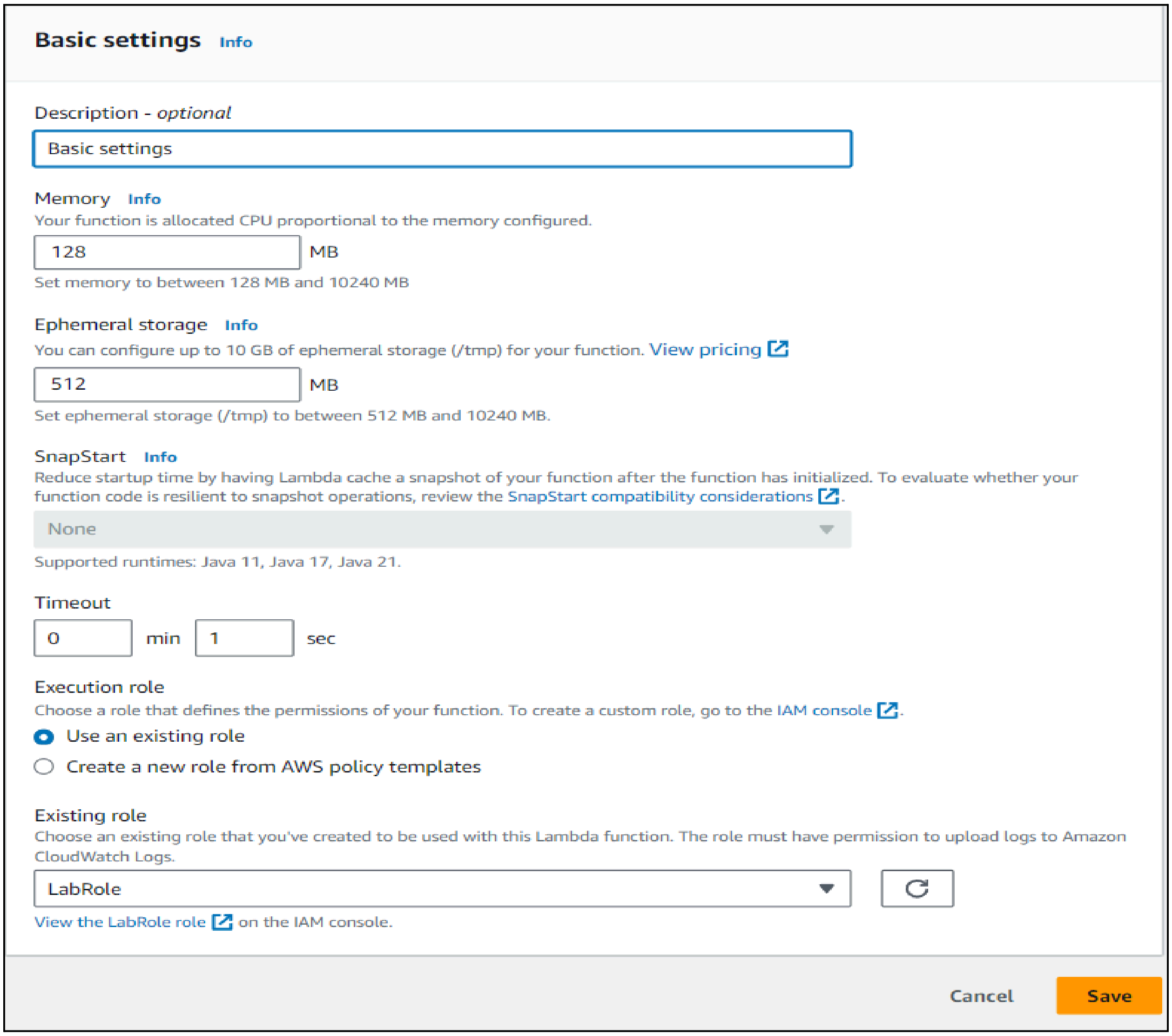
Memory
128 MB

Ephemeral storage
512 MB

Timeout
0 min 1 sec

SnapStart
None

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



Basic settings

Description - *optional*
Basic settings

Memory
Your function is allocated CPU proportional to the memory configured.
128 MB
Set memory to between 128 MB and 10240 MB

Ephemeral storage
You can configure up to 10 GB of ephemeral storage (/tmp) for your function.
512 MB
Set ephemeral storage (/tmp) to between 512 MB and 10240 MB.

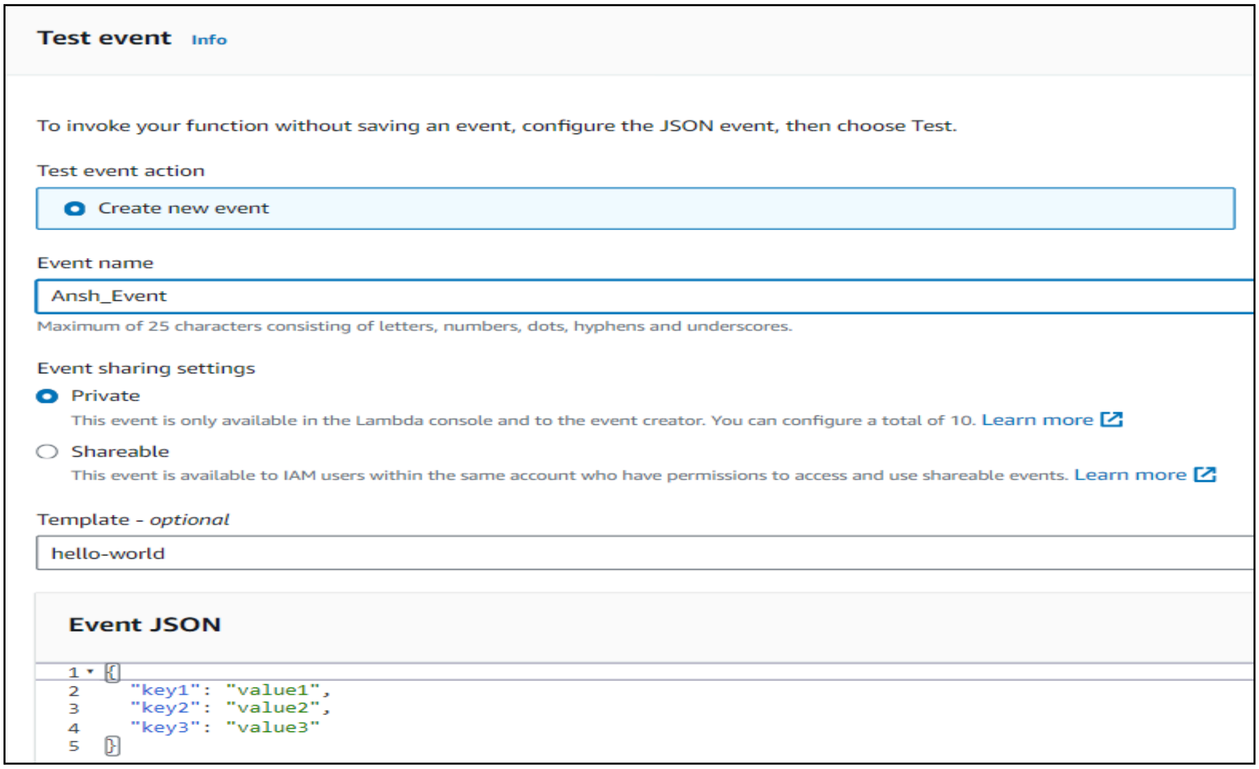
SnapStart
Reduce startup time by having Lambda cache a snapshot of your function after the function has initialized. To evaluate whether your function code is resilient to snapshot operations, review the SnapStart compatibility considerations.
None
Supported runtimes: Java 11, Java 17, Java 21.

Timeout
0 min 1 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.

Step 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.



Test event Info

To invoke your function without saving an event, configure the JSON event, then choose Test.

Test event action

● Create new event

Event name

Ansh_Event

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

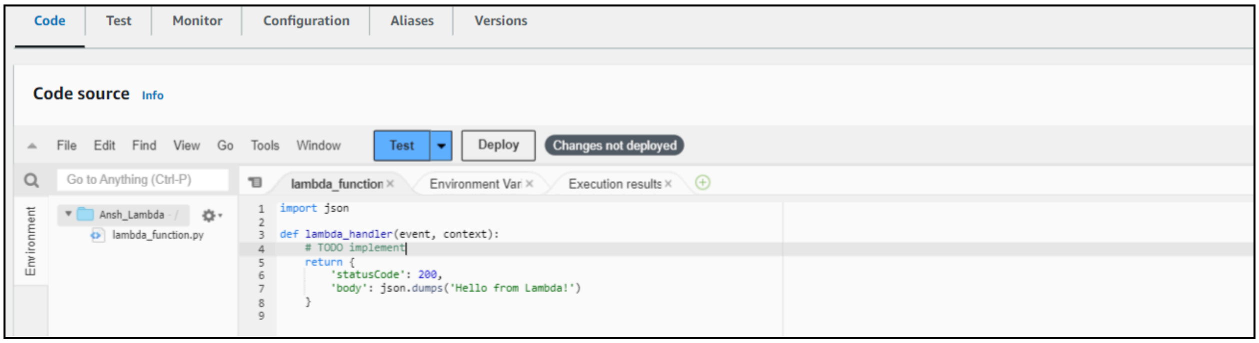
Template - *optional*

hello-world

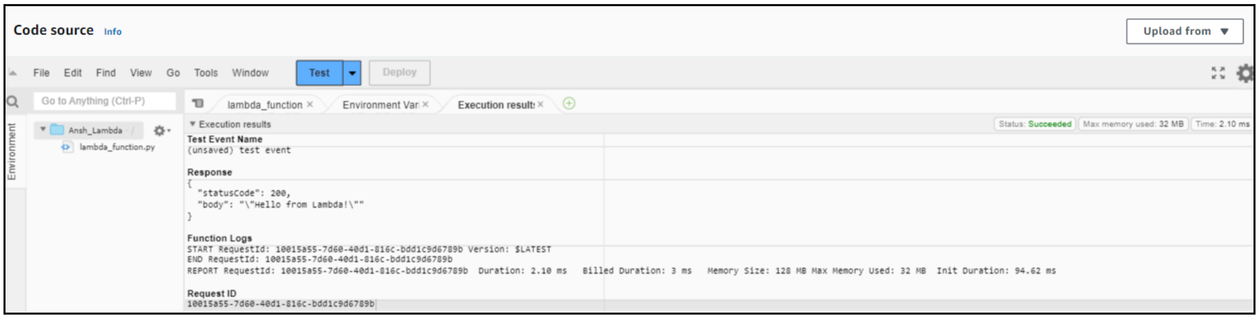
Event JSON

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

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



```
import json

def lambda_handler(event, context):
    # TODO implement
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }

```



Status: Succeeded | Max memory used: 32 MB | Time: 2.10 ms

Test Event Name

```
(unsaved) test event
```

Response

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

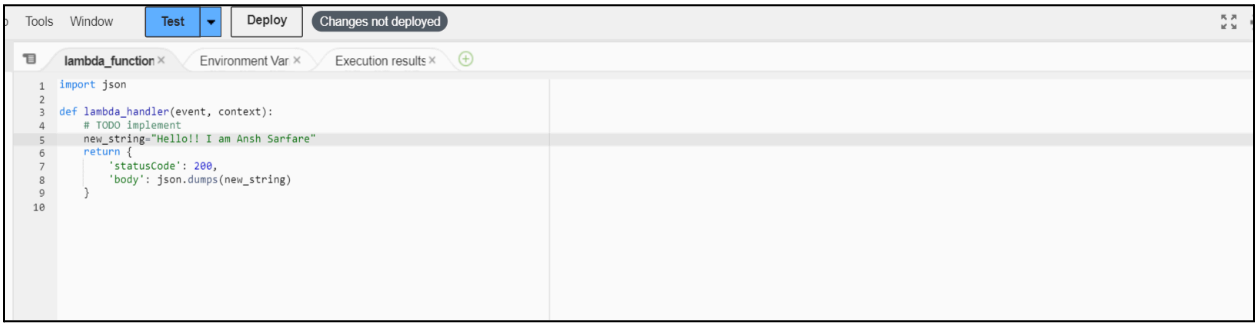
Function Logs

```
START RequestId: 10015a55-7d60-40d1-816c-bdd1c9d6789b Version: $LATEST
END RequestId: 10015a55-7d60-40d1-816c-bdd1c9d6789b
REPORT RequestId: 10015a55-7d60-40d1-816c-bdd1c9d6789b  Duration: 2.10 ms   Billed Duration: 3 ms   Memory Size: 128 MB Max Memory Used: 32 MB  Init Duration: 94.62 ms
```

Request ID

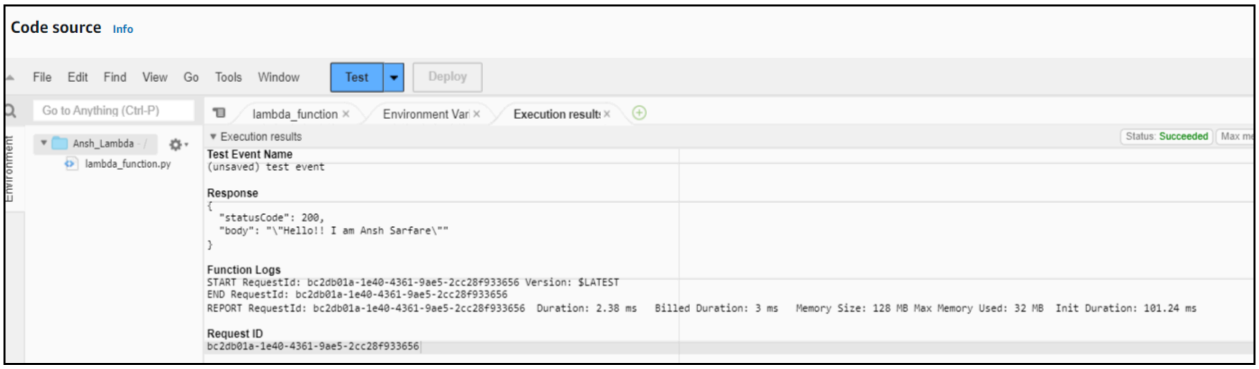
```
10015a55-7d60-40d1-816c-bdd1c9d6789b
```

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



```
import json

def lambda_handler(event, context):
    # TODO implement
    new_string="Hello!! I am Ansh Sarfare"
    return {
        'statusCode': 200,
        'body': json.dumps(new_string)
    }

```

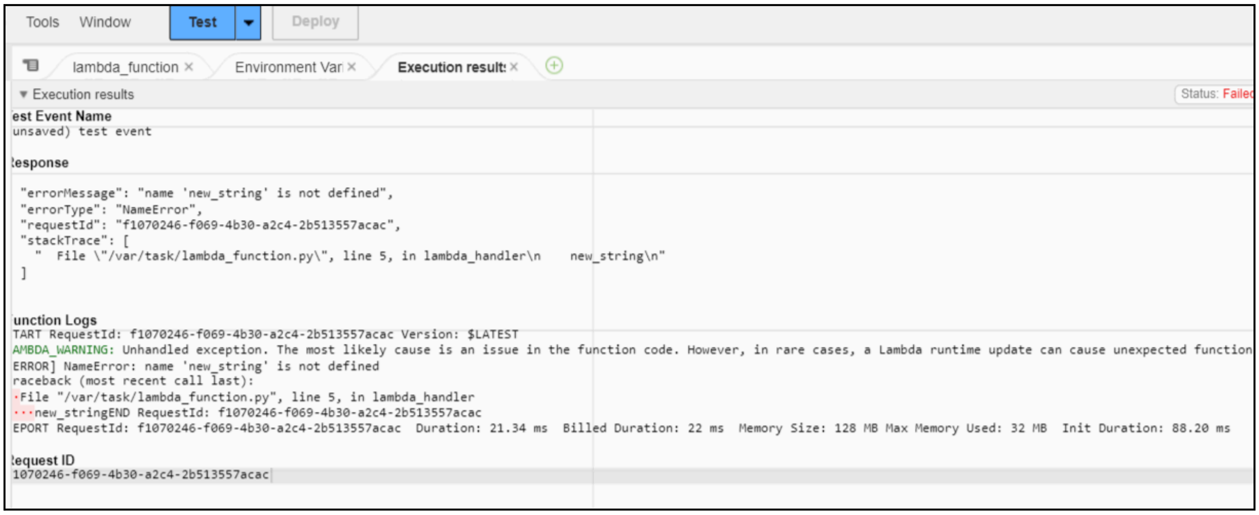
Step 6: Now click on the test and observe the output. We can see the status code 200 and your string output and function logs. On successful deployment.



```
Test Event Name
(unsaved) test event

Response
{
  "statusCode": 200,
  "body": "\"Hello!! I am Ansh Sarfare\""
}

Function Logs
START RequestId: bc2db01a-1e40-4361-9ae5-2cc28f933656 Version: $LATEST
END RequestId: bc2db01a-1e40-4361-9ae5-2cc28f933656
REPORT RequestId: bc2db01a-1e40-4361-9ae5-2cc28f933656  Duration: 2.38 ms   Billed Duration: 3 ms   Memory Size: 128 MB Max Memory Used: 32 MB  Init Duration: 101.24 ms

Request ID
bc2db01a-1e40-4361-9ae5-2cc28f933656
```

This error had occurred because the new string was not passed properly in the return statement

error:



```
est Event Name
unsaved) test event

esponse

 "errorMessage": "name 'new_string' is not defined",
 "errorType": "NameError",
 "requestId": "f1070246-f069-4b30-a2c4-2b513557acac",
 "stackTrace": [
   "  File \"/var/task/lambda_function.py\", line 5, in lambda_handler\n    new_string\n"
 ]


unction Logs
TART RequestId: f1070246-f069-4b30-a2c4-2b513557acac Version: $LATEST
AMBDA_WARNING: Unhandled exception. The most likely cause is an issue in the function code. However, in rare cases, a Lambda runtime update can cause unexpected function
ERROR] NameError: name 'new_string' is not defined
raceback (most recent call last):
 File "/var/task/lambda_function.py", line 5, in lambda_handler
   new_stringEND RequestId: f1070246-f069-4b30-a2c4-2b513557acac
EPORT RequestId: f1070246-f069-4b30-a2c4-2b513557acac  Duration: 21.34 ms  Billed Duration: 22 ms  Memory Size: 128 MB Max Memory Used: 32 MB  Init Duration: 88.20 ms

equest ID
1070246-f069-4b30-a2c4-2b513557acac
```