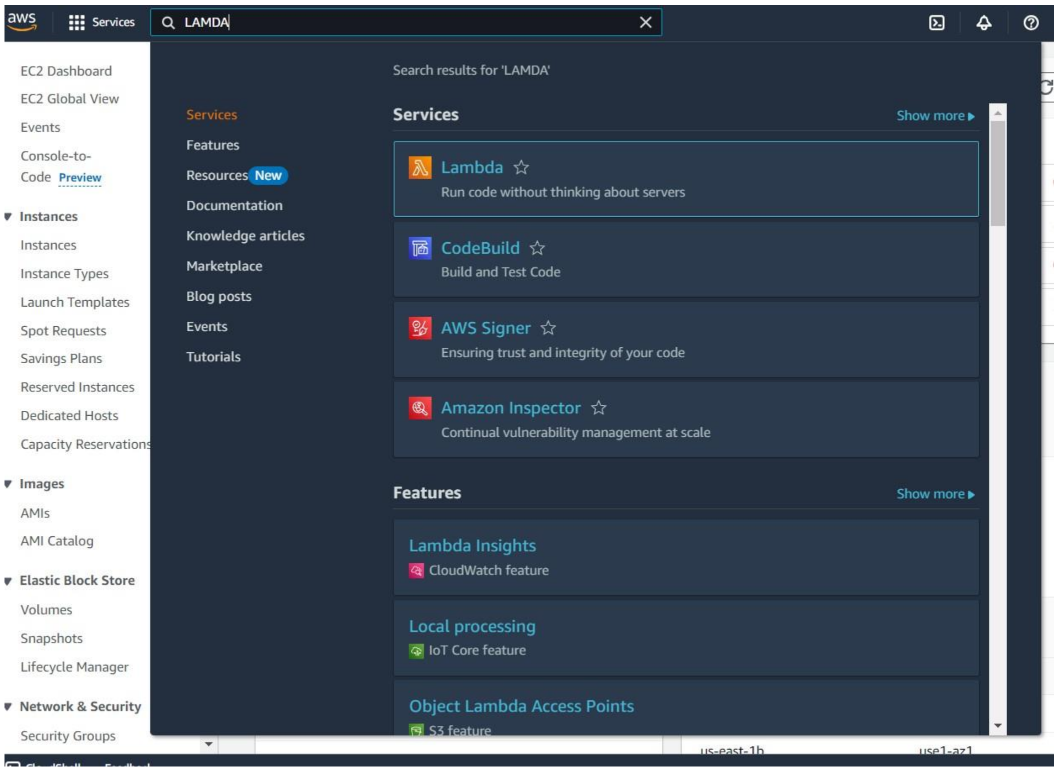
Name:Aditya Kirtane **Std:D15C** **ROLL:26**

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

Step 1: Accessing AWS

Log in to your AWS Personal/Academy account. Navigate to the Lambda service by searching for "Lambda" in the AWS Management Console.



Search results for 'LAMDA'

Services

Lambda — Run code without thinking about servers

CodeBuild — Build and Test Code

AWS Signer — Ensuring trust and integrity of your code

Amazon Inspector — Continual vulnerability management at scale

Features

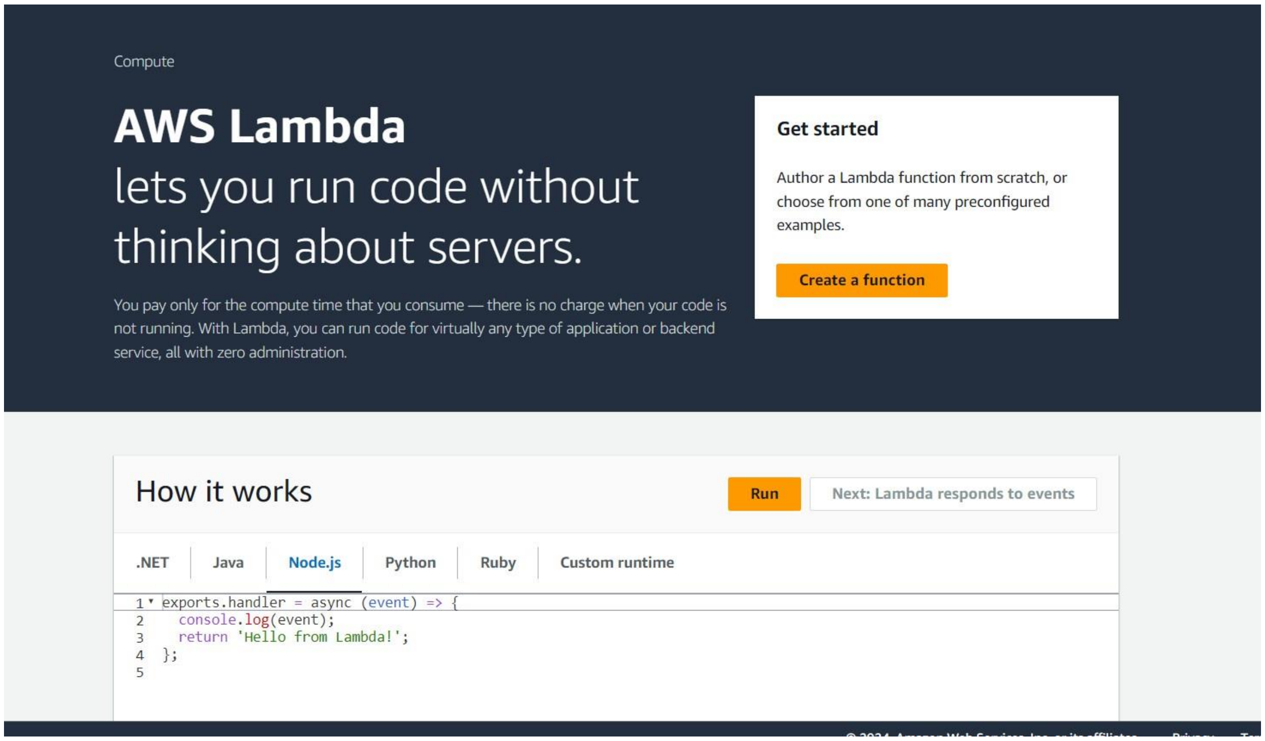
Lambda Insights — CloudWatch feature

Local processing — IoT Core feature

Object Lambda Access Points — S3 feature

Step 2: Creating a New Lambda Function

Click on the "Create function" button. Provide a name for your Lambda function and select the language you wish to use, such as Python 3.12. For architecture, choose x86, and for execution role, opt to create a new role with basic Lambda g permissions.


Compute

AWS Lambda
lets you run code without thinking about servers.

You pay only for the compute time that you consume — there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service, all with zero administration.

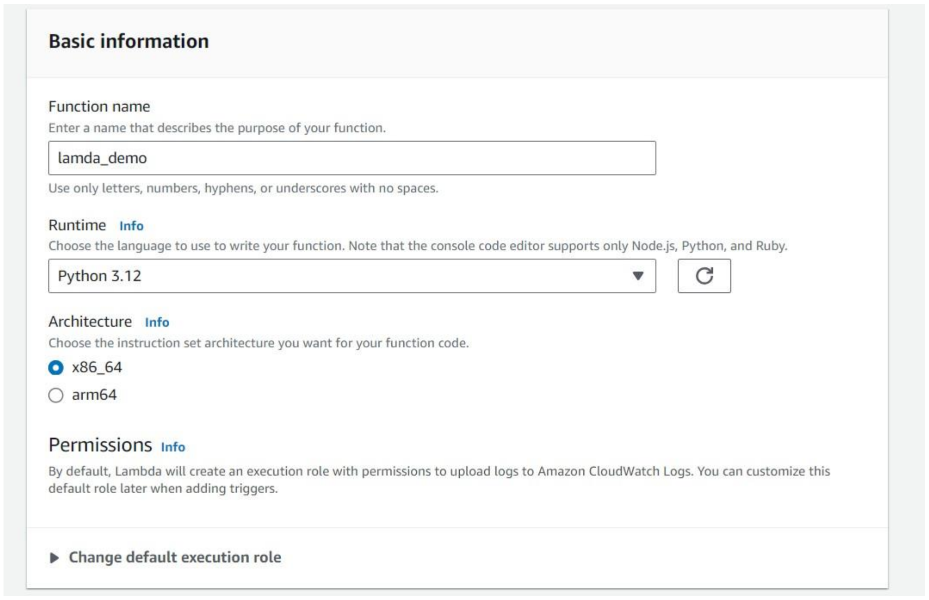
Get started

Author a Lambda function from scratch, or choose from one of many preconfigured examples.

How it works

```
exports.handler = async (event) => {
  console.log(event);
  return 'Hello from Lambda!';
};
```

Step 3: Configuring Basic Settings
To modify the basic settings, navigate to the "Configuration" tab and click on "Edit" under General Settings. Here, you can add a description and adjust the memory and timeout settings. For this experiment, I set the timeout to 1 second, which is sufficient for testing.


Basic information

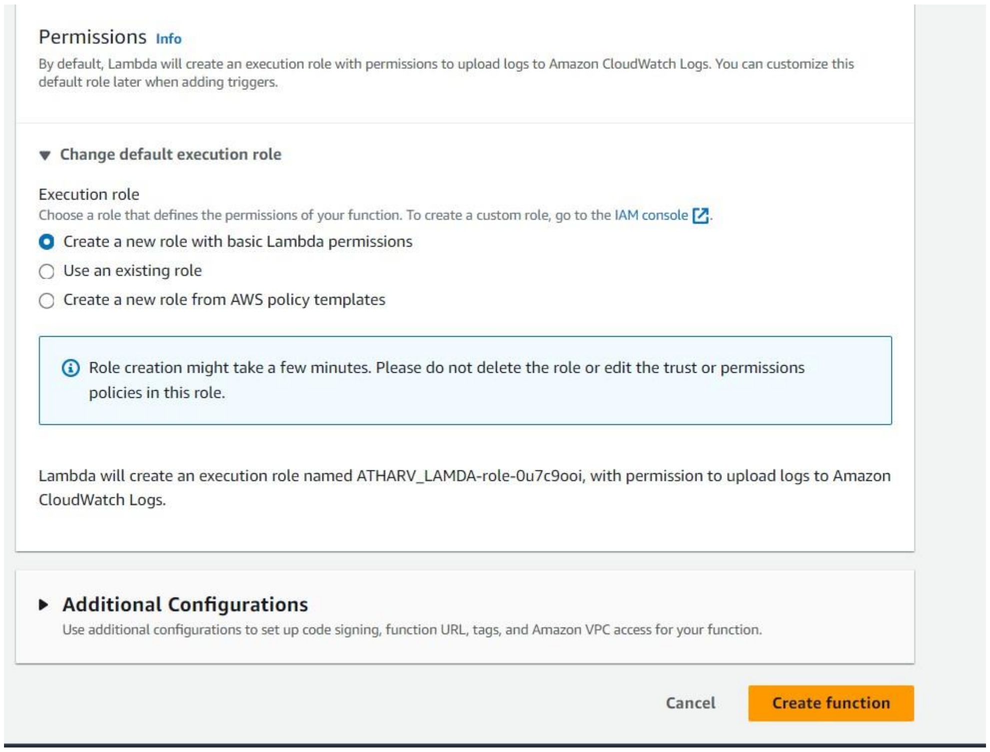
Function name
Enter a name that describes the purpose of your function.
lamda_demo
Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.
Python 3.12

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.

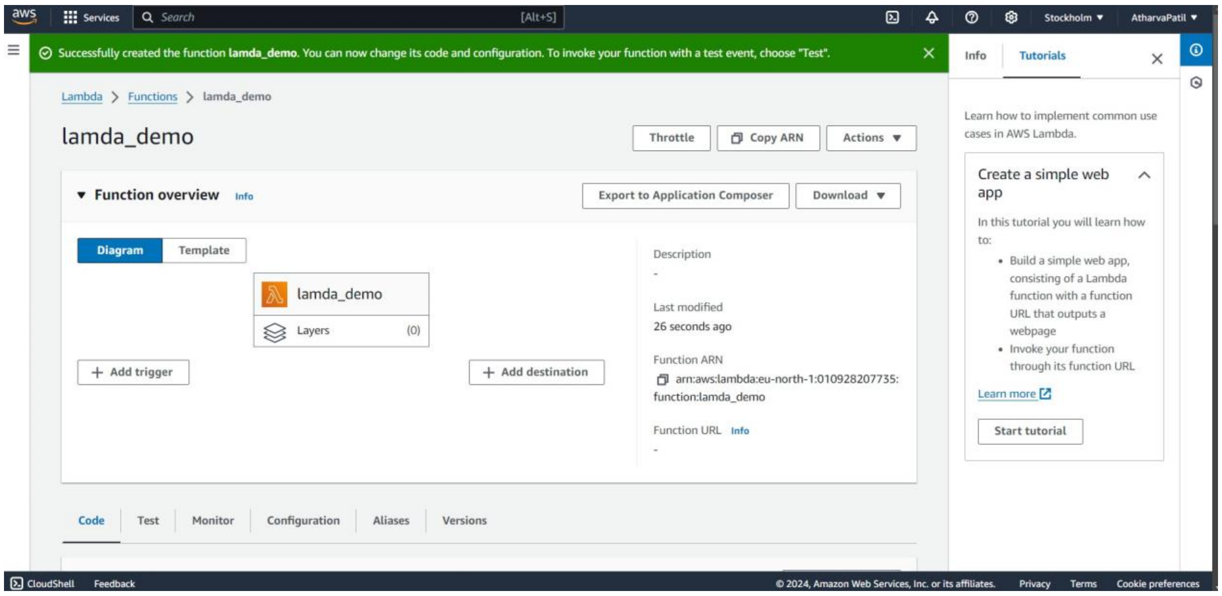
- [✓ Create a new role with basic Lambda permissions]
- Use an existing role
- Create a new role from AWS policy templates

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 ATHARV_LAMDA-role-0u7c9ooi, with permission to upload logs to Amazon CloudWatch Logs.

▶ Additional Configurations

Use additional configurations to set up code signing, function URL, tags, and Amazon VPC access for your function.



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

Lambda > Functions > lamda_demo

lamda_demo

Description
-

Last modified
26 seconds ago

Function ARN
arn:aws:lambda:eu-north-1:010928207735:function:lamda_demo

Function URL Info
-

Info | Tutorials

Learn how to implement common use cases in AWS Lambda.

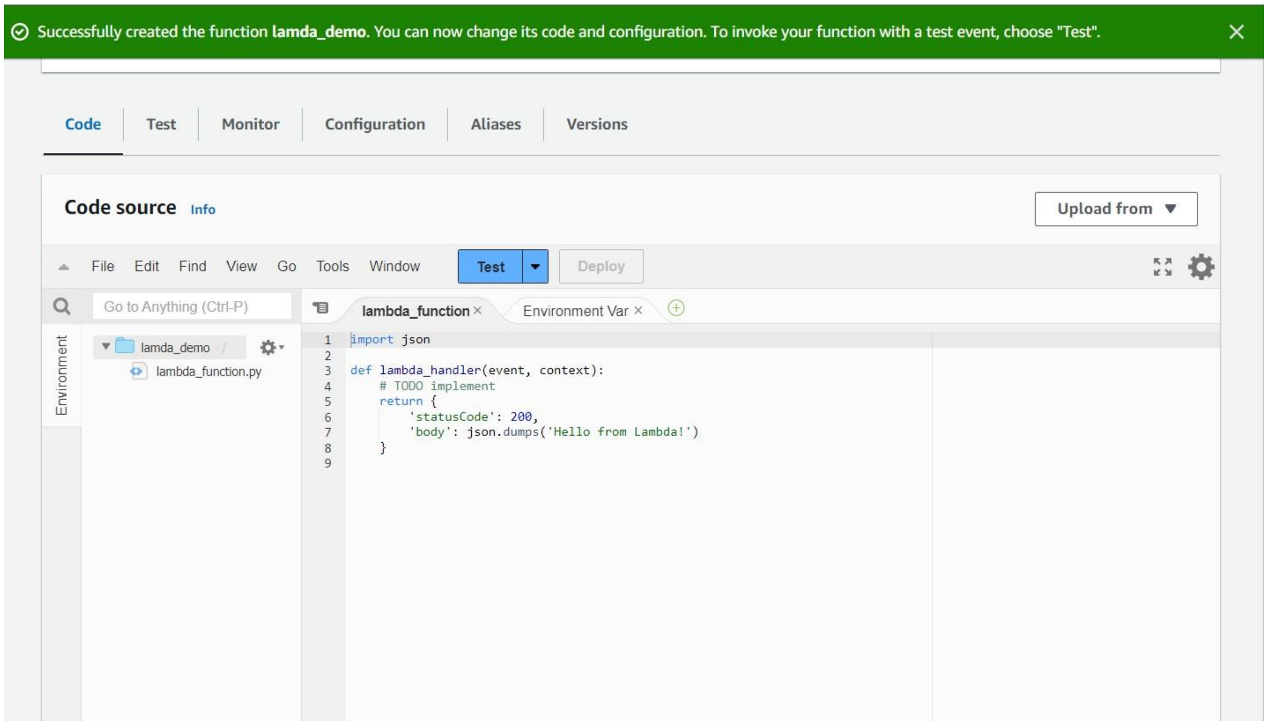
Create a simple web app

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

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences


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

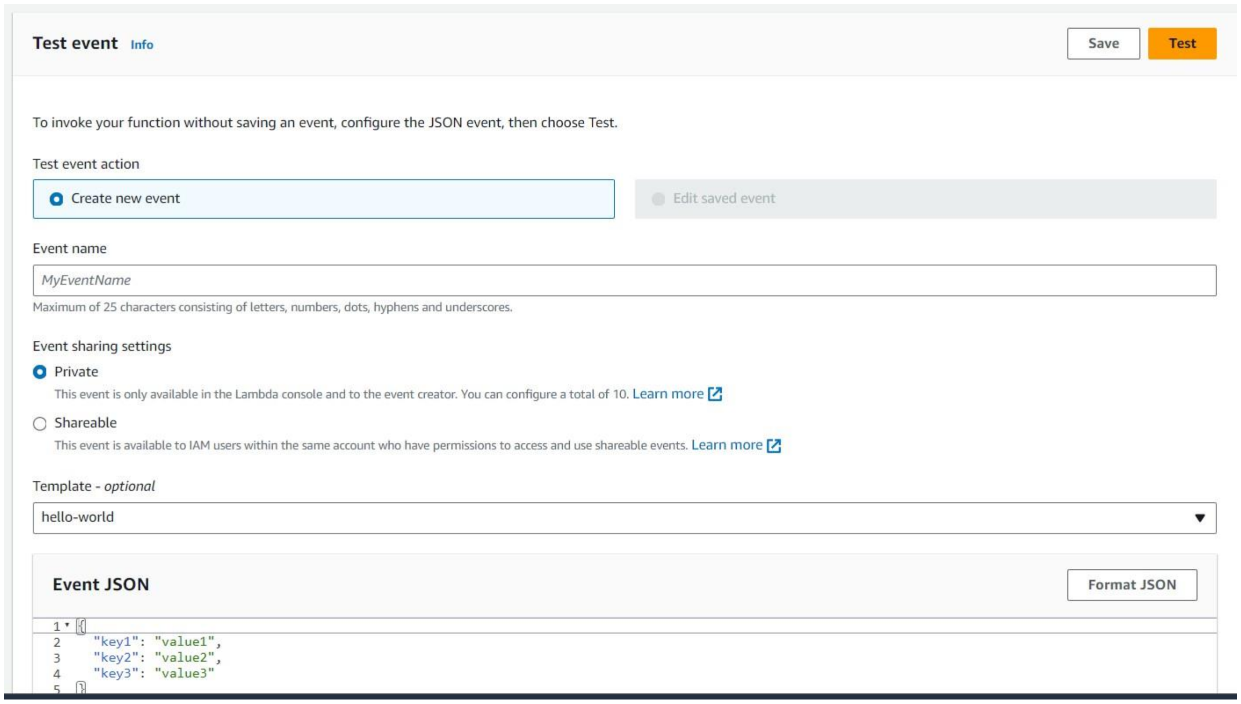
```
import json

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

```

Step 4: Testing the Function

Click on the "Test" tab and select "Create a new event." Name your event, set the event sharing to private, and choose the "hello-world" template.



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

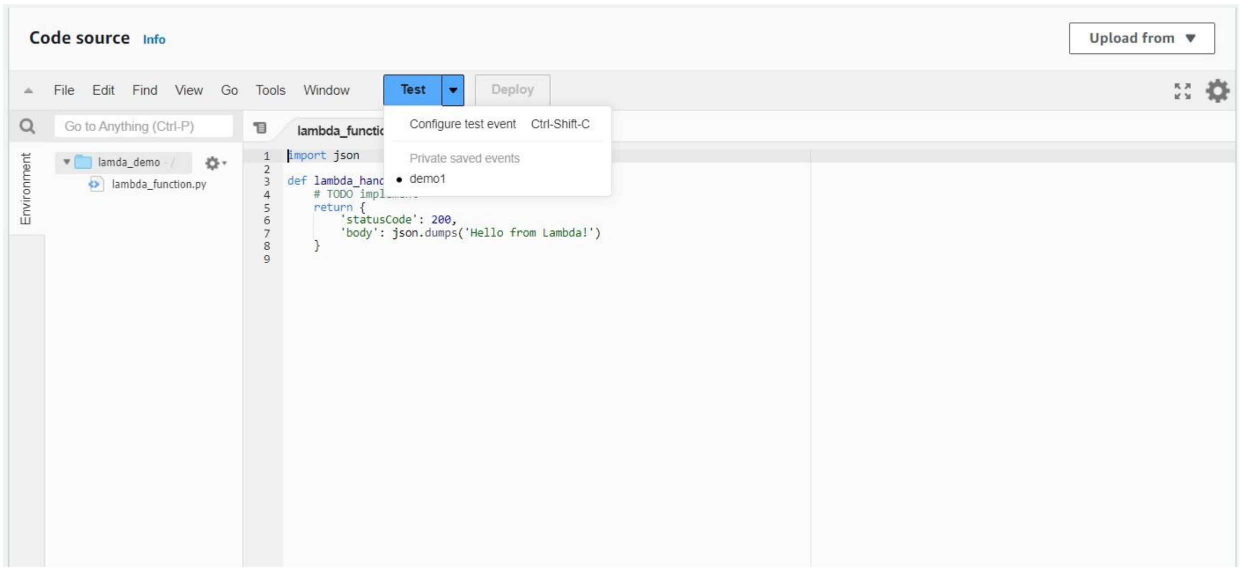
Shareable

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

Template - *optional*

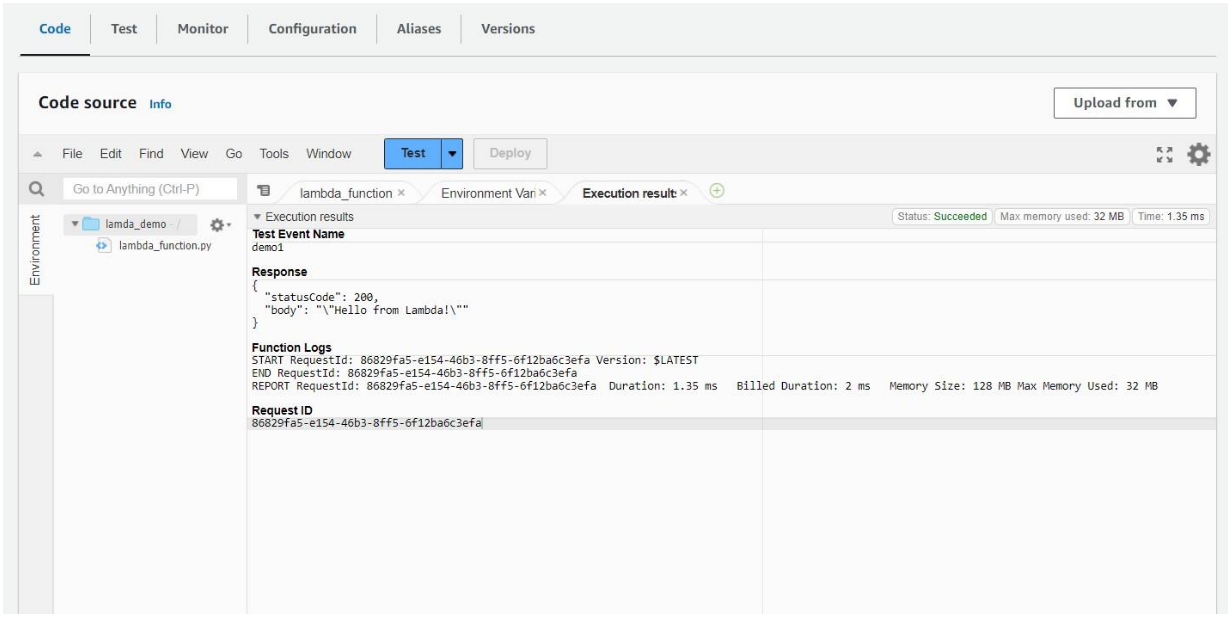
hello-world

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



```
import json

def lambda_hanc
    # TODO impl
    return {
        'statusCode': 200,
        'body': json.dumps('Hello from Lambda!')
    }
```

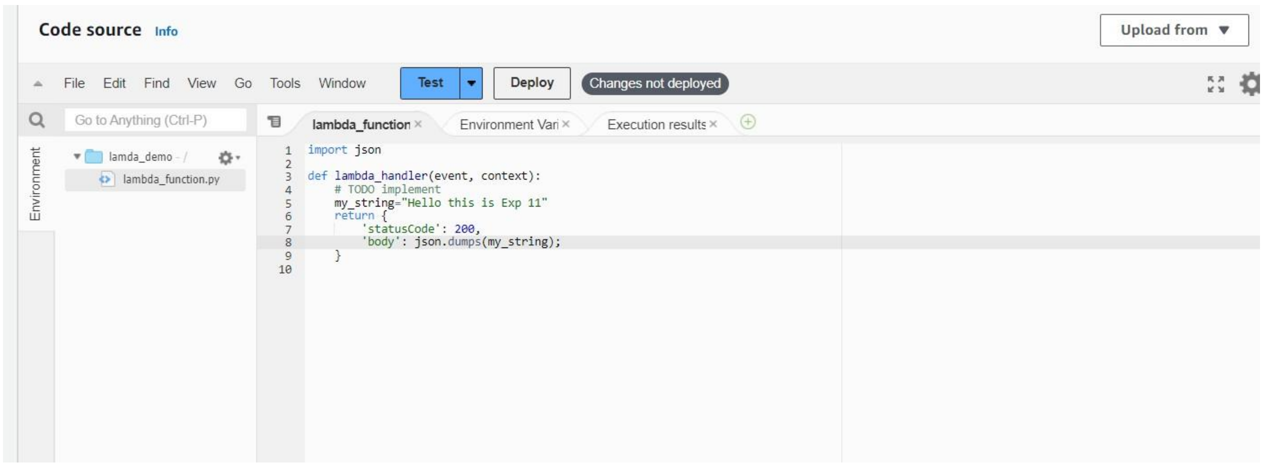


Test Event Name
```
demo1
```

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

Function Logs
```
START RequestId: 86829fa5-e154-46b3-8ff5-6f12ba6c3efa Version: $LATEST
END RequestId: 86829fa5-e154-46b3-8ff5-6f12ba6c3efa
REPORT RequestId: 86829fa5-e154-46b3-8ff5-6f12ba6c3efa  Duration: 1.35 ms   Billed Duration: 2 ms   Memory Size: 128 MB Max Memory Used: 32 MB
```

Request ID
```
86829fa5-e154-46b3-8ff5-6f12ba6c3efa
```

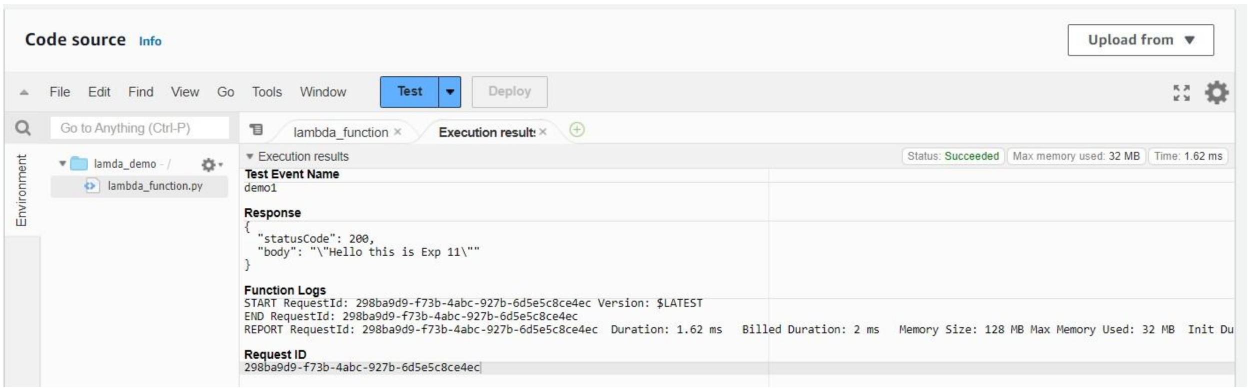


```python
import json

def lambda_handler(event, context):
    # TODO implement
    my_string="Hello this is Exp 11"
    return {
        'statusCode': 200,
        'body': json.dumps(my_string);
    }
```

Step 5: Running the Test

In the Code section, select the newly created event from the dropdown menu and click on "Test." You should see the output displayed below.



Test Event Name
```
demo1
```

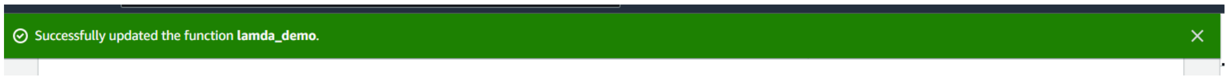
Response
```
{
  "statusCode": 200,
  "body": "\"Hello this is Exp 11\""
}
```

Function Logs
```
START RequestId: 298ba9d9-f73b-4abc-927b-6d5e5c8ce4ec Version: $LATEST
END RequestId: 298ba9d9-f73b-4abc-927b-6d5e5c8ce4ec
REPORT RequestId: 298ba9d9-f73b-4abc-927b-6d5e5c8ce4ec  Duration: 1.62 ms   Billed Duration: 2 ms   Memory Size: 128 MB Max Memory Used: 32 MB  Init Du
```

Request ID
```
298ba9d9-f73b-4abc-927b-6d5e5c8ce4ec
```

Step 6: Editing and Deploying the Code

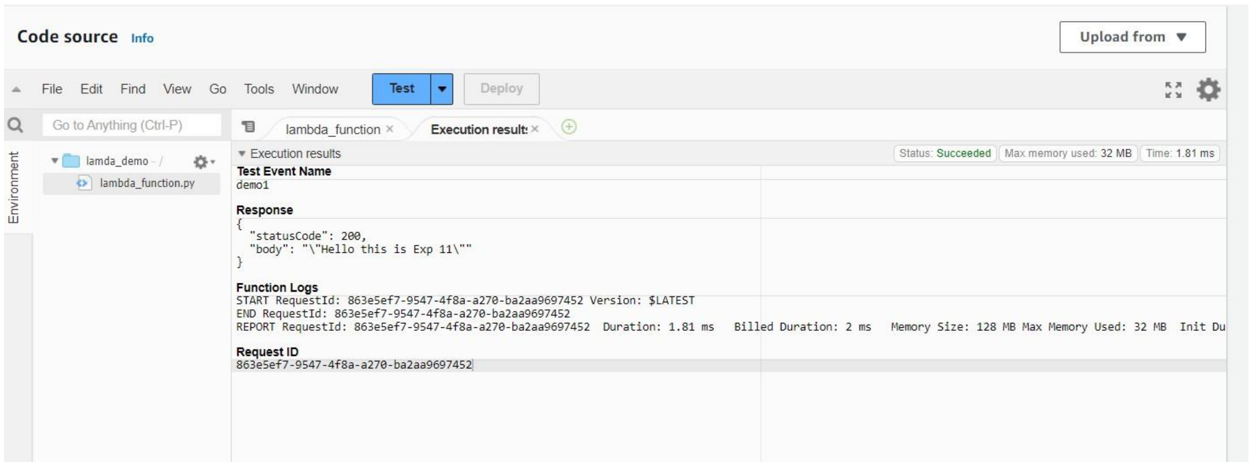
You can modify your Lambda function's code as needed. I updated the code to display a new string. After making changes, press `Ctrl + S` to save and then click on "Deploy" to apply the updates.



Successfully updated the function **lamda_demo**.

Step 7: Final Testing

Return to the "Test" tab and execute the test again to observe the output. You should see a status code of 200 along with your string output and function logs confirming a successful deployment.



Test Event Name

```
demo1
```

Response

```
{
  "statusCode": 200,
  "body": "\"Hello this is Exp 11\""
}
```

Function Logs

```
START RequestId: 863e5ef7-9547-4f8a-a270-ba2aa9697452 Version: $LATEST
END RequestId: 863e5ef7-9547-4f8a-a270-ba2aa9697452
REPORT RequestId: 863e5ef7-9547-4f8a-a270-ba2aa9697452  Duration: 1.81 ms   Billed Duration: 2 ms   Memory Size: 128 MB Max Memory Used: 32 MB  Init Du
```

Request ID

```
863e5ef7-9547-4f8a-a270-ba2aa9697452
```

Conclusion

In this experiment, I successfully navigated the process of creating an AWS Lambda function. After configuring the function with Python, I adjusted the settings to optimize its performance. I created a test event, deployed the function, and verified the output, which confirmed the expected behavior. This hands-on experience highlighted the user-friendly nature of AWS Lambda, illustrating how it enables developers to focus on coding while AWS efficiently handles the underlying infrastructure and scaling. This project not only deepened my understanding of serverless computing but also reinforced the practical application of cloud services in modern software development.