

ADVANCED DEVOPS EXP 11

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

Theory:

AWS Lambda

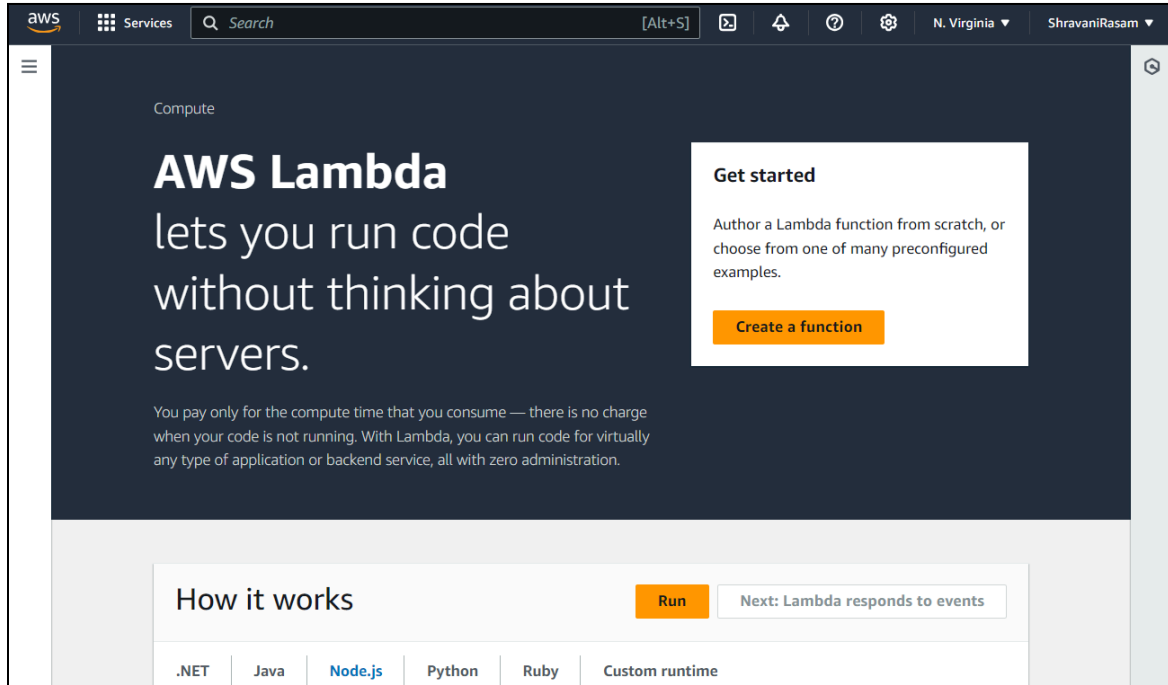
- AWS Lambda is a serverless computing service provided by Amazon Web Services (AWS). Users of AWS Lambda create functions, self-contained applications written in one of the supported languages and runtimes, and upload them to AWS Lambda, which executes those functions in an efficient and flexible manner.
- The Lambda functions can perform any kind of computing task, from serving web pages and processing streams of data to calling APIs and integrating with other AWS services. The concept of “serverless” computing refers to not needing to maintain your own servers to run these functions.
- AWS Lambda is a fully managed service that takes care of all the infrastructure for you. And so “serverless” doesn’t mean that there are no servers involved: it just means that the servers, the operating systems, the network layer and the rest of the infrastructure have already been taken care of so that you can focus on writing application code.

Features of AWS Lambda

- AWS Lambda easily scales the infrastructure without any additional configuration. It reduces the operational work involved.
- It offers multiple options like AWS S3, CloudWatch, DynamoDB, API Gateway, Kinesis, CodeCommit, and many more to trigger an event.
- You don’t need to invest upfront. You pay only for the memory used by the lambda function and minimal cost on the number of requests hence cost-efficient.
- AWS Lambda is secure. It uses AWS IAM to define all the roles and security policies.
- It offers fault tolerance for both services running the code and the function. You do not have to worry about the application down

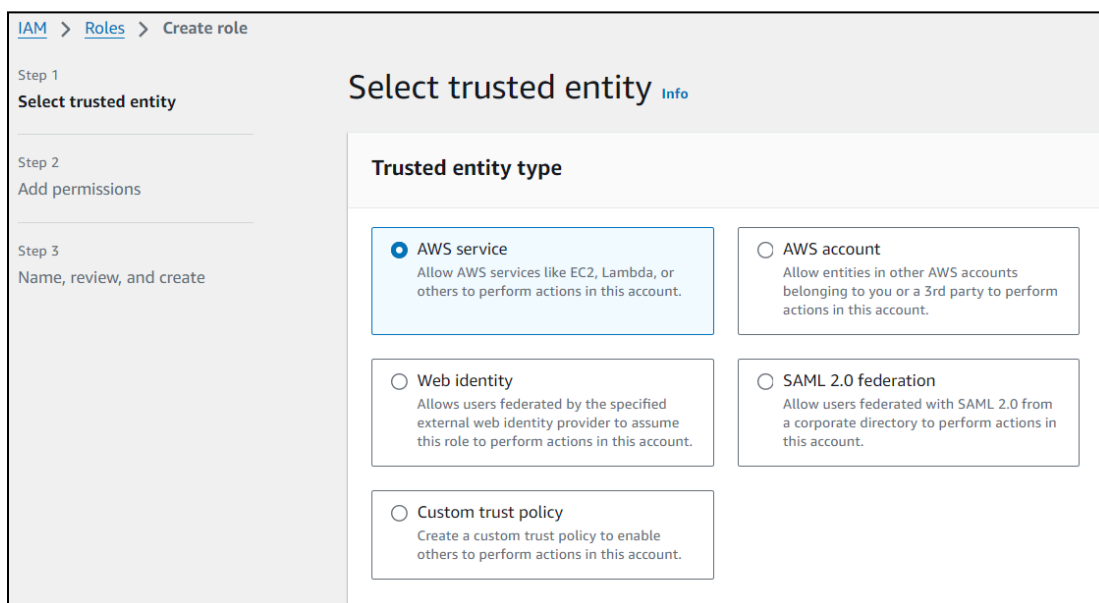
Steps to create an AWS Lambda function

1. Open up the Lambda Console and click on the Create button. Be mindful of where you create your functions since Lambda is region-dependent.



2. Choose the Lambda service:

- Under "Trusted entity type," select **AWS service**.
- Choose **Lambda** from the list of services, and click **Next**.



3. Attach CloudWatch Logs permissions:

- In the "Permissions" step, search for the policy called **AWSLambdaBasicExecutionRole**.
- Select this policy, which gives your Lambda function permission to write logs to CloudWatch.
- Click **Next**.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Service or use case

Lambda ▼

Choose a use case for the specified service.

Use case

☒ **Lambda**
Allows Lambda functions to call AWS services on your behalf.

[IAM](#) > [Roles](#) > Create role

Step 1
[Select trusted entity](#)

Step 2
[Add permissions](#)

Step 3
Name, review, and create

Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.

lambda-user

Maximum 64 characters. Use alphanumeric and '+=, @-_' characters.

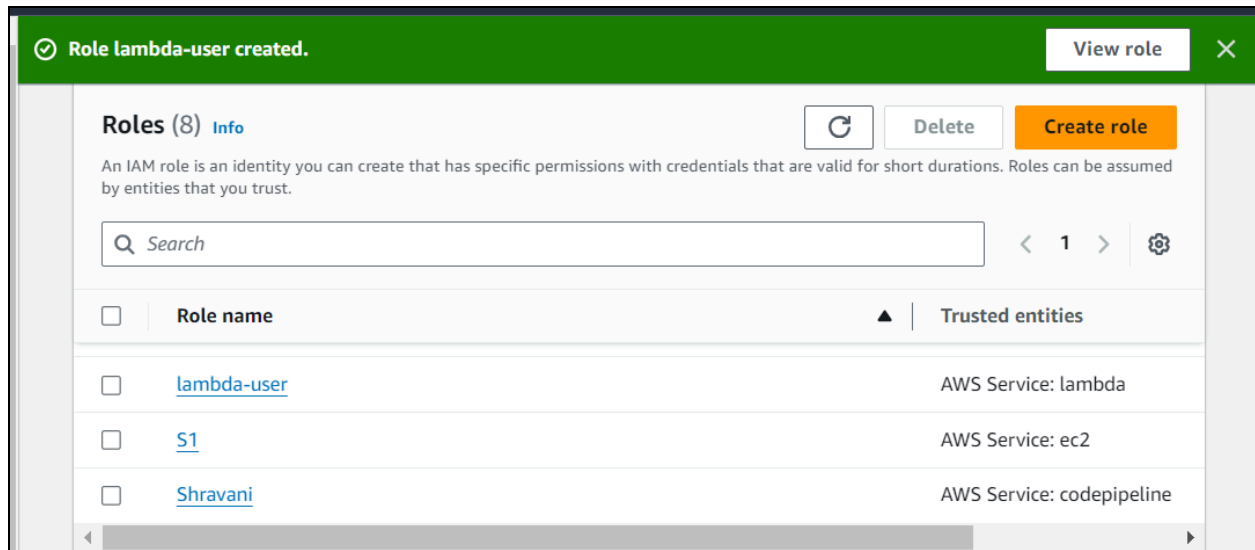
Description
Add a short explanation for this role.

Allows Lambda functions to call AWS services on your behalf.

Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=, @-/[]!#\$%^&*()~:~'`

Step 1: Select trusted entities

Edit



2. Choose to create a function from scratch or use a blueprint, i.e templates defined by AWS for you with all configuration presets required for the most common use cases.

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. After that, choose to create a new role with basic Lambda permissions if you don't have an existing one.

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.

lambdafunction-1

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.

Python 3.11

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

☒ x86_64

☐ arm64

[Lambda](#) > [Functions](#) > [lambdafunction-1](#) > Edit basic settings

Edit basic settings

Basic settings [Info](#)

Description - optional

Memory [Info](#)

Your function is allocated CPU proportional to the memory configured.

128 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](#)

512 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.

✓ Successfully updated the function **lambdafunction-1**.

Code source [Info](#)

Upload from ▼

File Edit Find View Go Tools Window

Test

Deploy

Go to Anything (Ctrl-P)

lambda_function x

Environment Vari x

Environment
▼ lambdafunction-1
 ▶ 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
```

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.

lambda-user ▼



[View the lambda-user 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.

Cancel

Create function

✓ Successfully created the function **lambdafunction-1**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

[Lambda](#) > [Functions](#) > lambdafunction-1

lambdafunction-1

Throttle

Copy ARN

Actions ▼

▼ Function overview [Info](#)

Export to Application Composer

Download ▼

Diagram

Template



lambdafunctio
n-1



Layers (0)

+ Add trigger

+ Add destination

Description

-

Last modified

14 seconds ago

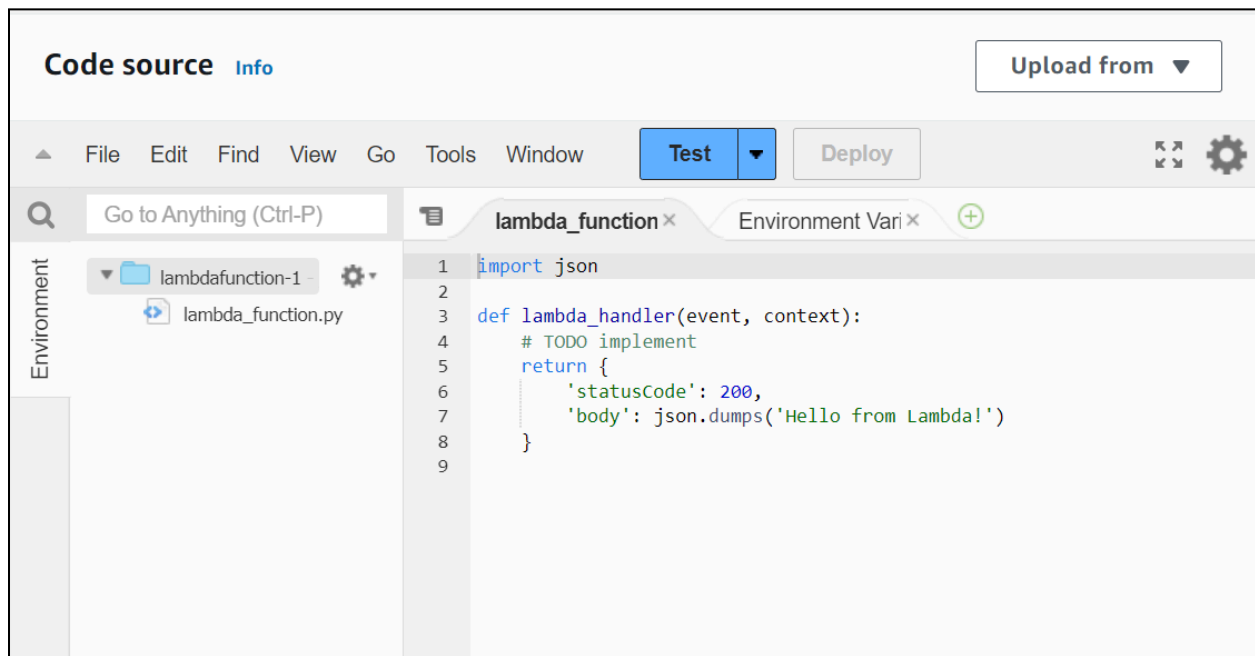
Function ARN

arn:aws:lambda:us-east-1:361769589277:function:lambdafunction-1

Function URL [Info](#)

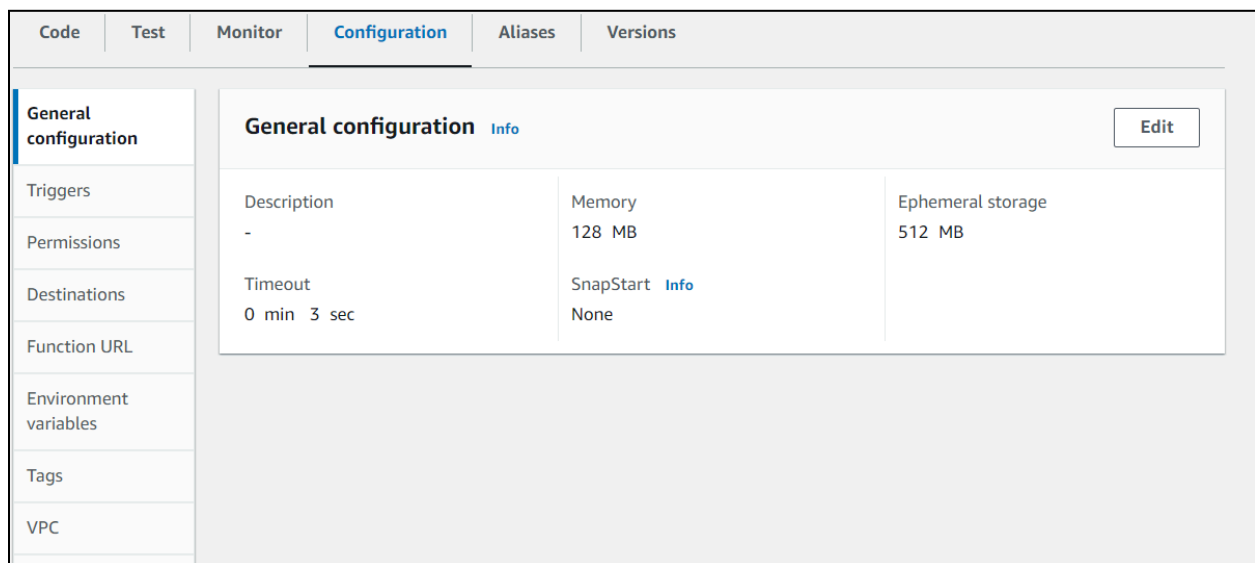
-

3. This process will take a while to finish and after that, you'll get a message that your function was successfully created



Edit Basic Settings:

- On the function's **Configuration** tab, locate the **Basic settings** section



Configuring test event which triggers when the function is tested

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

event1

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

Template - optional

hello-world

Event JSON

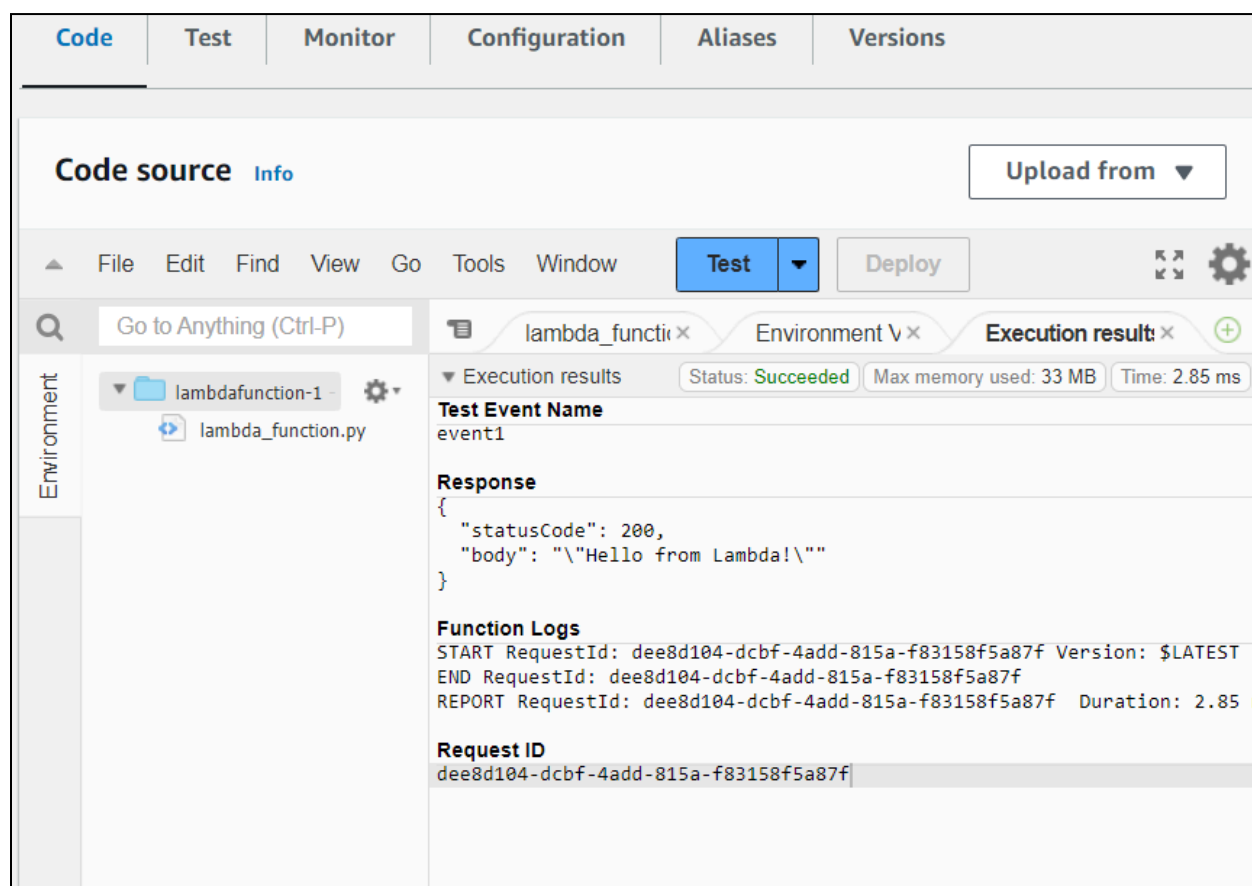
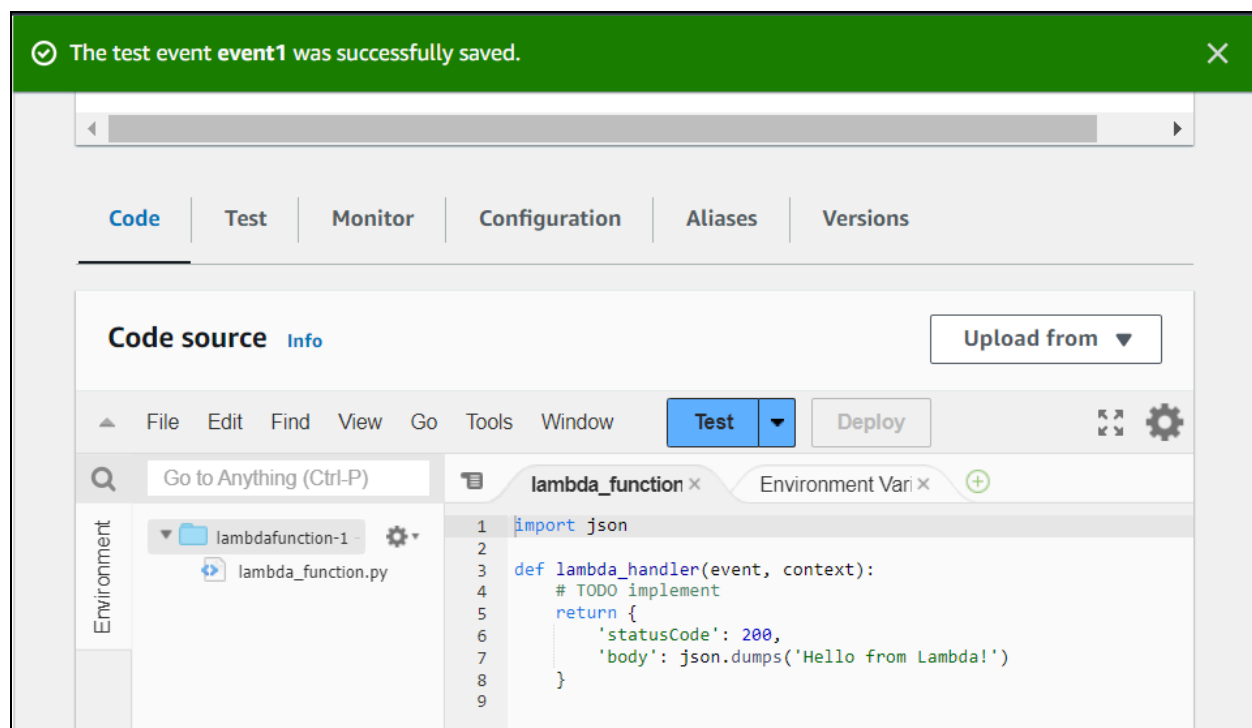
Format JSON

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

Cancel

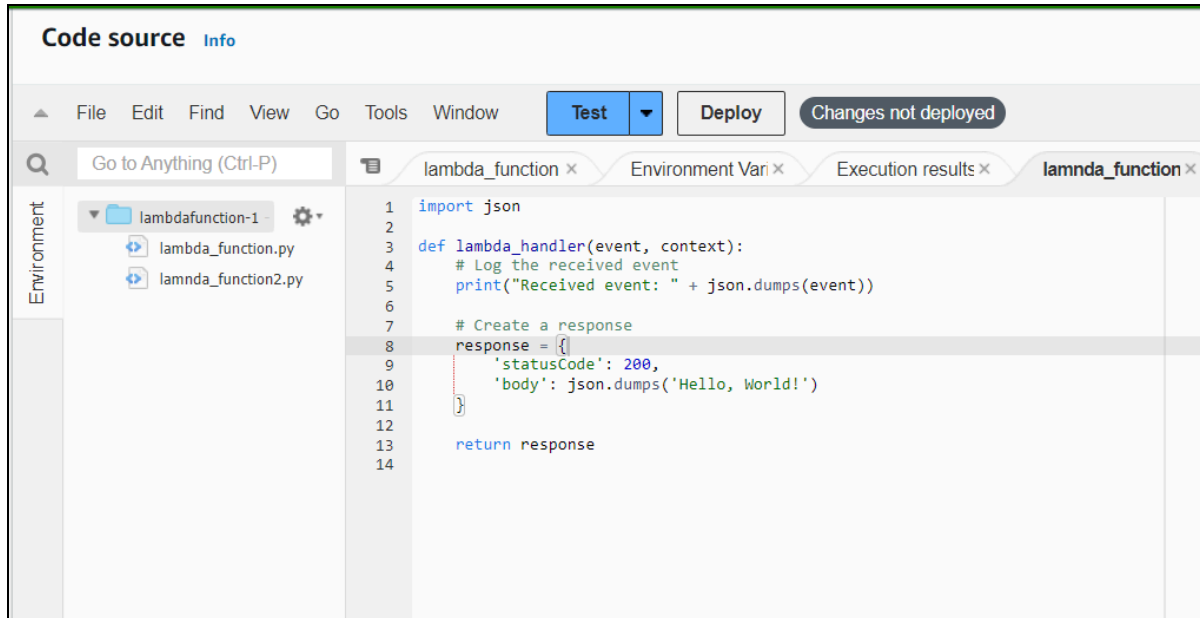
Invoke

Save



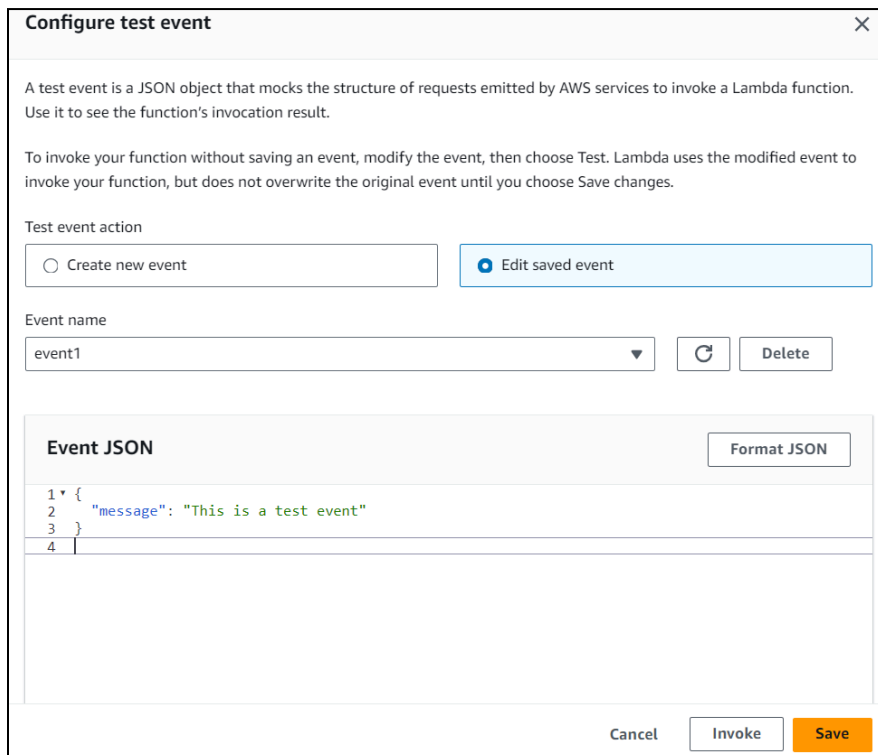
Basic "Hello, World!" Lambda Function (Python)

1. Open the AWS Lambda Console.
2. Create a new function.
3. Use the following code to test the Lambda function.



The screenshot shows the AWS Lambda console's code editor for a function named 'lambda_function'. The interface includes a menu bar (File, Edit, Find, View, Go, Tools, Window), buttons for 'Test' and 'Deploy', and a status bar indicating 'Changes not deployed'. The left sidebar shows the project structure with 'lambdafunction-1' and two files: 'lambda_function.py' and 'lamnda_function2.py'. The main editor displays the following Python code:

```
1 import json
2
3 def lambda_handler(event, context):
4     # Log the received event
5     print("Received event: " + json.dumps(event))
6
7     # Create a response
8     response = {
9         'statusCode': 200,
10        'body': json.dumps('Hello, World!')}
11
12
13    return response
14
```



The 'Configure test event' dialog box provides instructions and options for testing a Lambda function. It explains that a test event is a JSON object that mocks requests from AWS services. It offers two actions: 'Create new event' and 'Edit saved event' (which is selected). Below this, the 'Event name' is set to 'event1'. The 'Event JSON' section shows a sample JSON object: `{ "message": "This is a test event" }`. At the bottom, there are buttons for 'Cancel', 'Invoke', and 'Save'.

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, modify the event, then choose Test. Lambda uses the modified event to invoke your function, but does not overwrite the original event until you choose Save changes.

Test event action

☐ Create new event ☒ Edit saved event

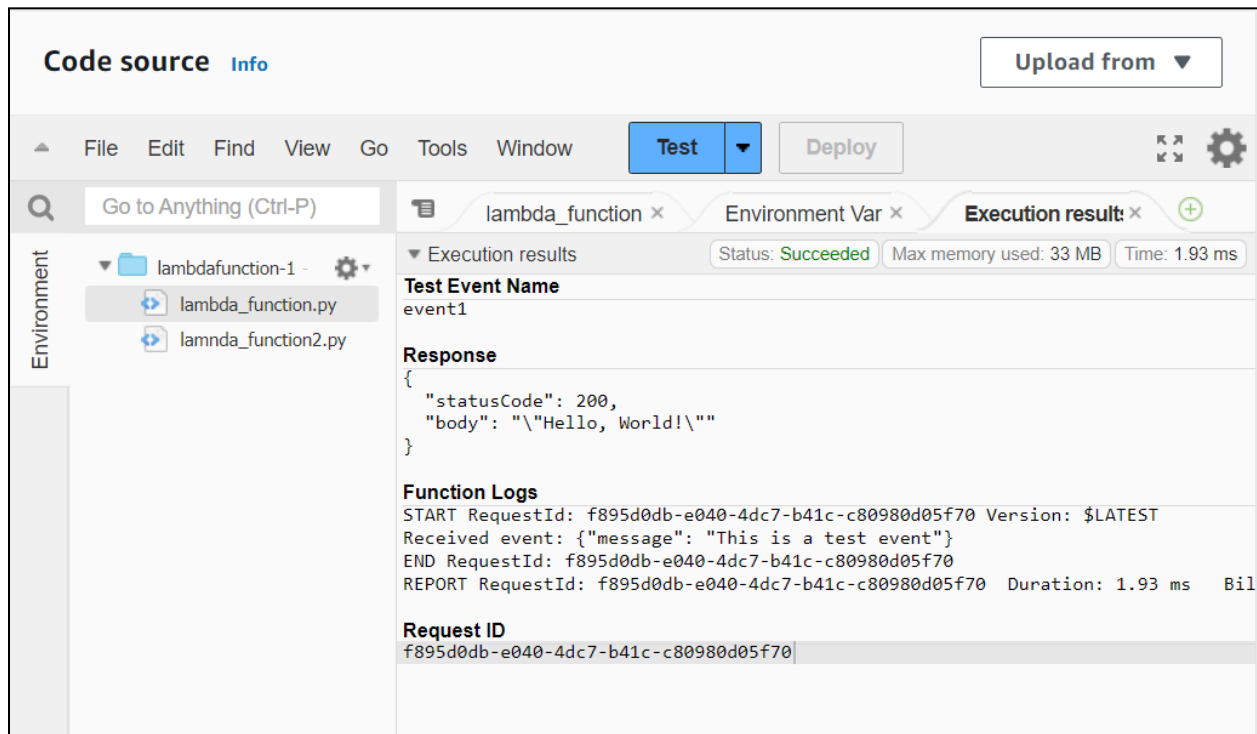
Event name

event1

Event JSON

```
1 {
2   "message": "This is a test event"
3 }
4
```

- The `lambda_handler` function is the entry point for Lambda execution.
- The function logs the received event, processes it (in this case, just returns "Hello, World!"), and sends a response with HTTP status code 200.



Conclusion:

AWS Lambda is a serverless computing service that allows you to run code without managing servers, making it highly scalable, cost-effective, and easy to use. It automatically manages the compute resources, executes your code in response to specific events such as API calls, file uploads, or database updates, and scales based on the demand.