Hands-on Lab - Creating an AWS Lambda

Estimated Time: 20 minutes

In this lab, you will become familiar with creating and testing AWS Lambda functions in Node js.

Important: *This lab requires use of credit card.*

Learning Objectives:

After completing this exercise, you should be able to perform the following tasks:

- Create an AWS Lambda function
- Test the output of an AWS Lambda function

Pre-requisites

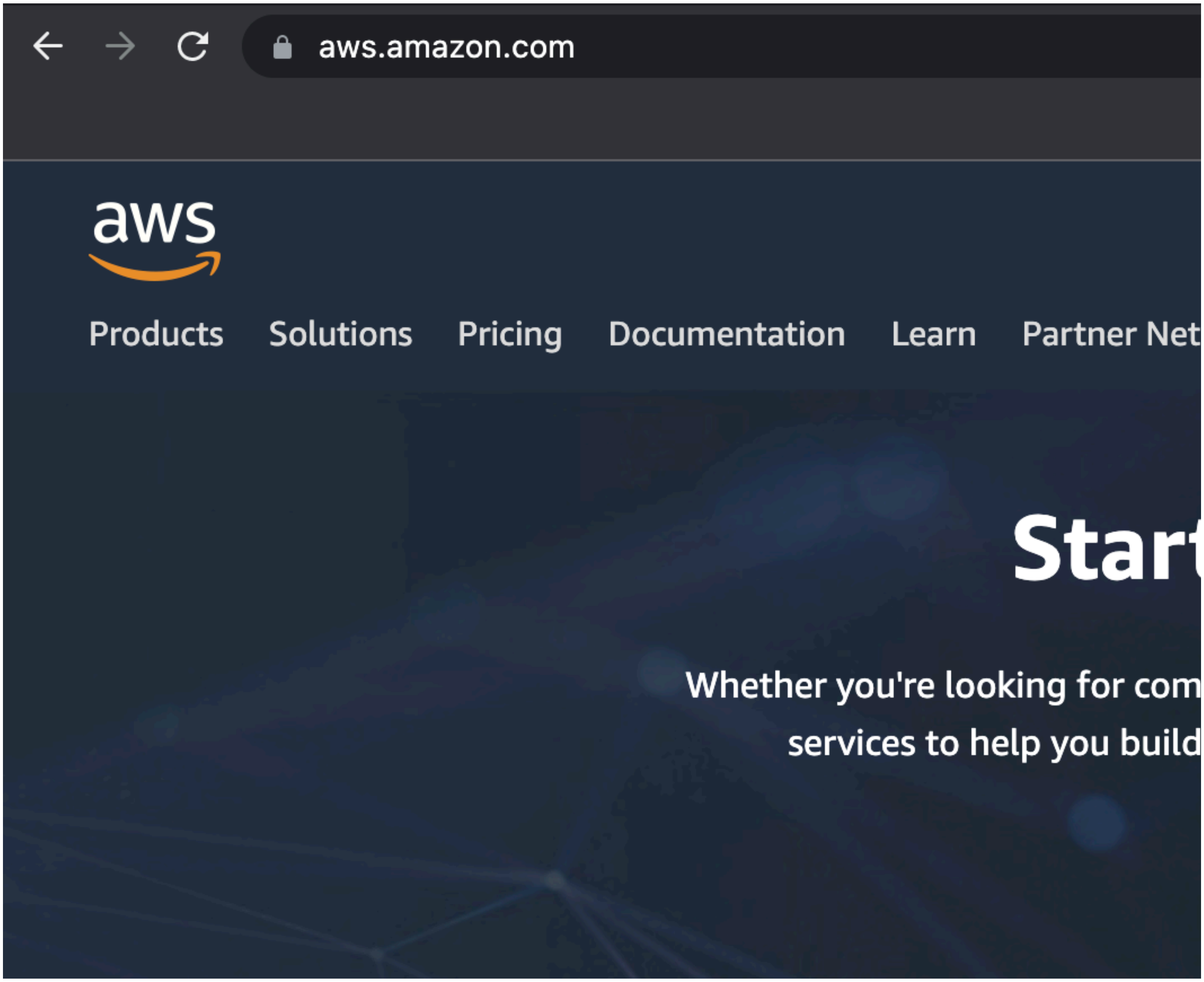
- You must have an AWS account.
- You should be familiar with Node js.

Important: *Please note that any usage beyond the free tier will be charged to the credit card you used for creating the AWS account.*

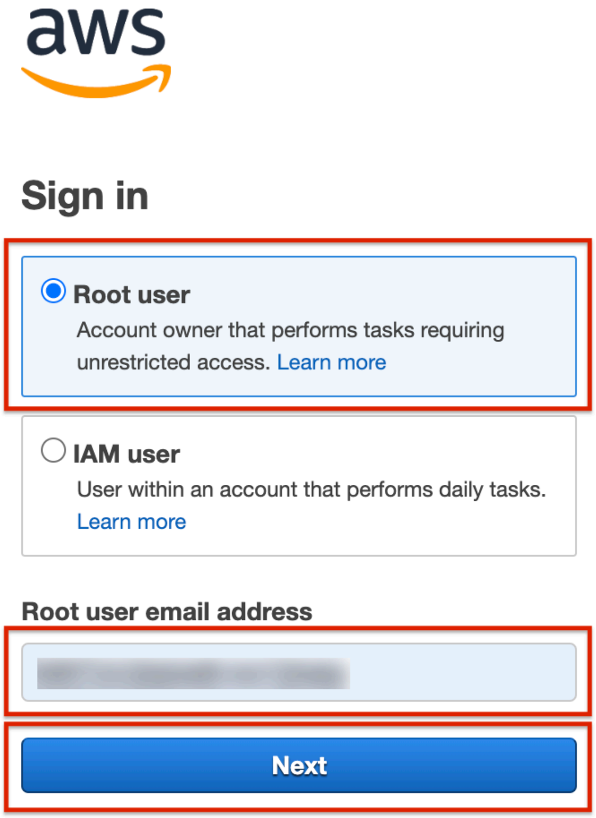
Task 1 - Sign into your AWS account

1. If you are already signed into your AWS account, you can skip this task. Go to https://aws.amazon.com.

2. Click **Sign In** to sign into your AWS account.



3. Enter the email address you registered with to sign in as root user.



Sign in

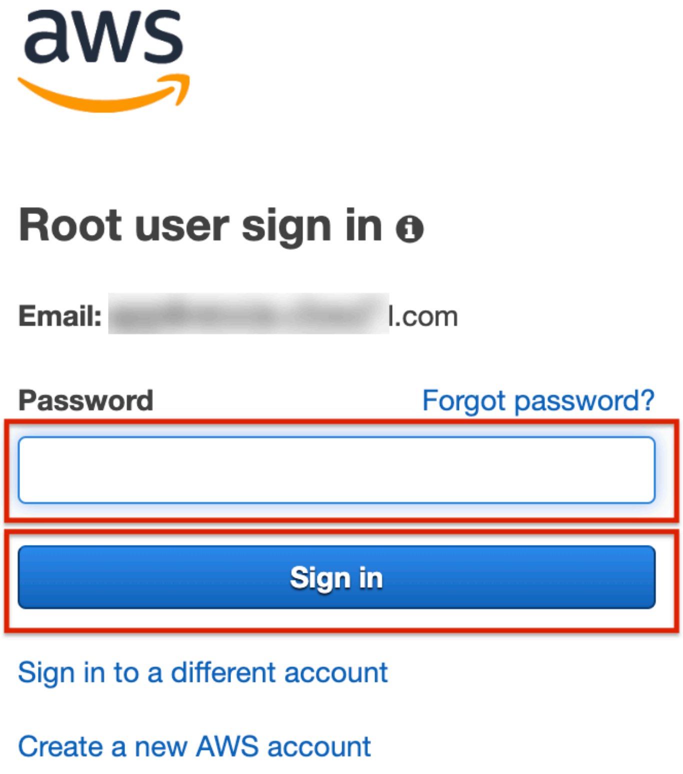
● Root user
Account owner that performs tasks requiring unrestricted access. Learn more

○ IAM user
User within an account that performs daily tasks. Learn more

Root user email address

[redacted]

4. Enter the password and click the **Sign In** button. This will take you to the **AWS Console Home**.



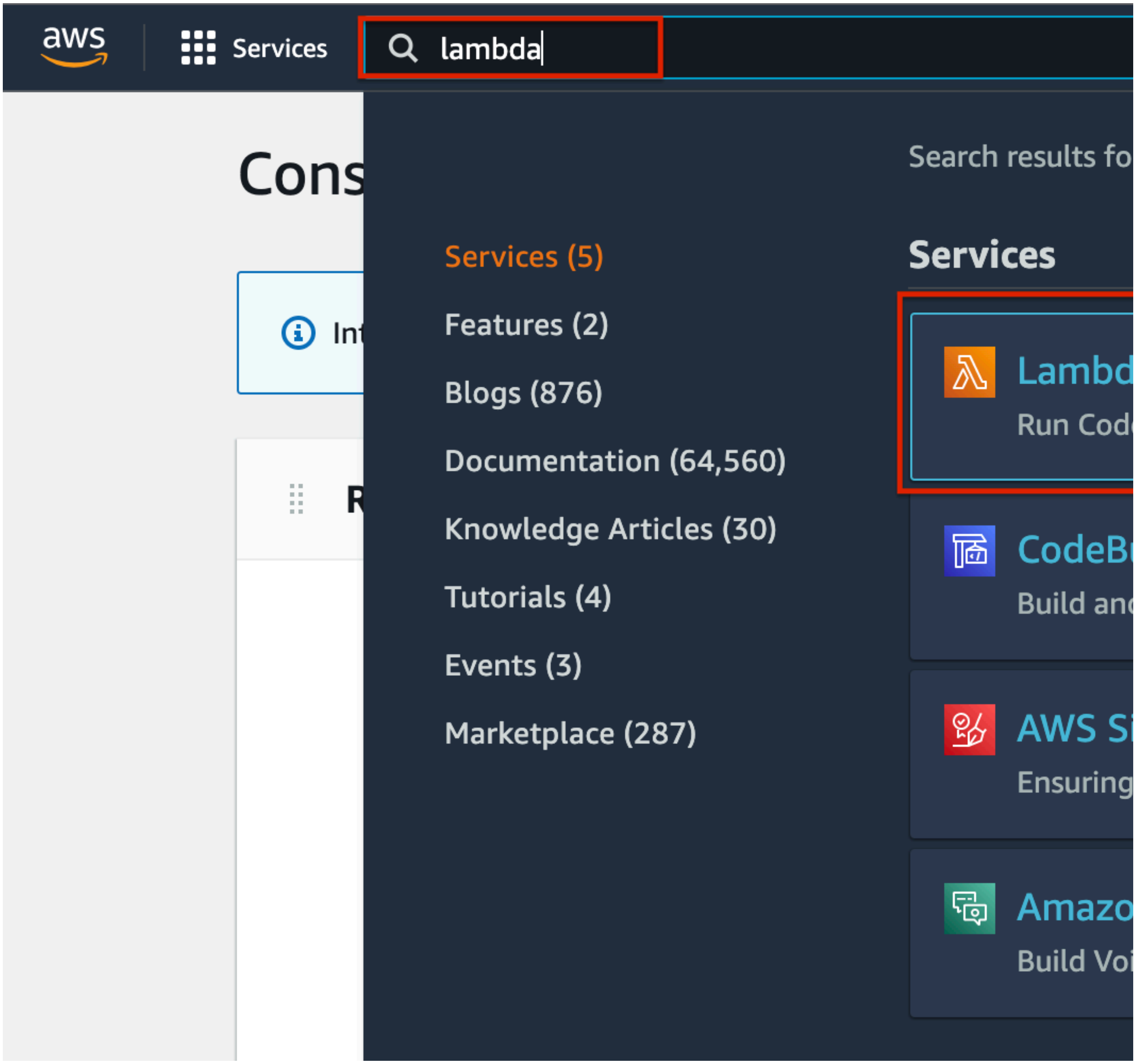
Root user sign in

Email: [redacted]l.com

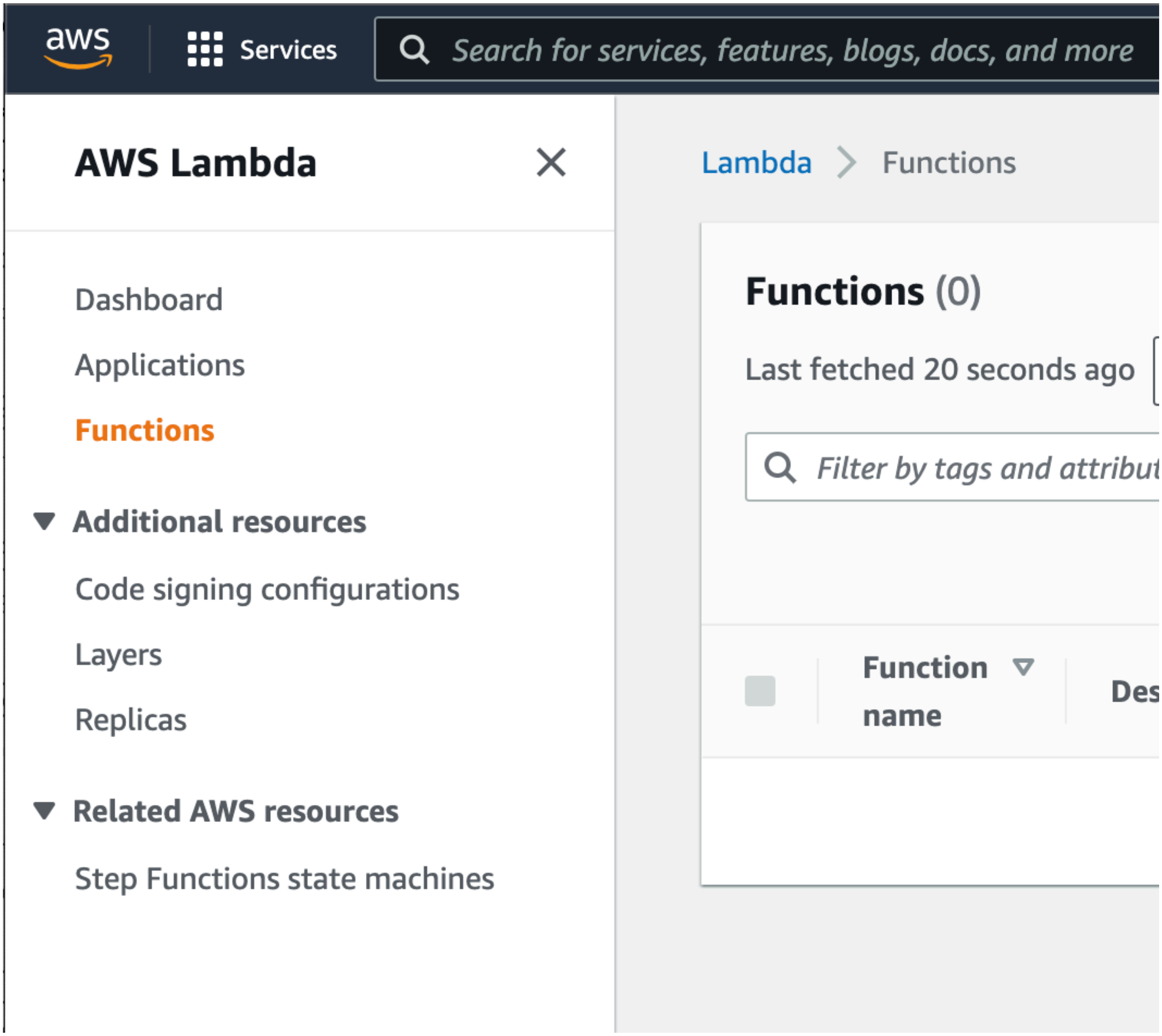
Password

Task 2 - Create AWS Lambda function

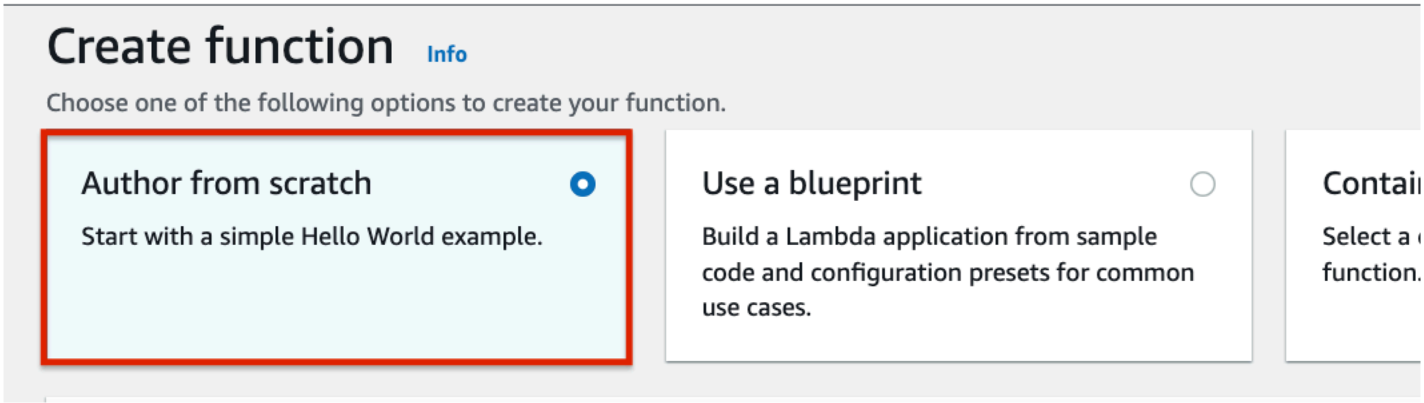
1. When the AWS **Console Home** loads up, on the top search bar, type **Lambda**, and you will see that the Lambda service is listed as the first choice. Choose **Lambda**.



2. **Action** is chosen by default. Click **Create Function** to start creating your AWS Lambda function.


aws
Services
Search for services, features, blogs, docs, and more
AWS Lambda
Lambda > Functions
Dashboard
Applications
Functions
Additional resources
Code signing configurations
Layers
Replicas
Related AWS resources
Step Functions state machines
Functions (0)
Last fetched 20 seconds ago
Filter by tags and attribut
Function name
Des

3. You can choose to **Author from Scratch** as you will be adding your own code to it.


Create function Info

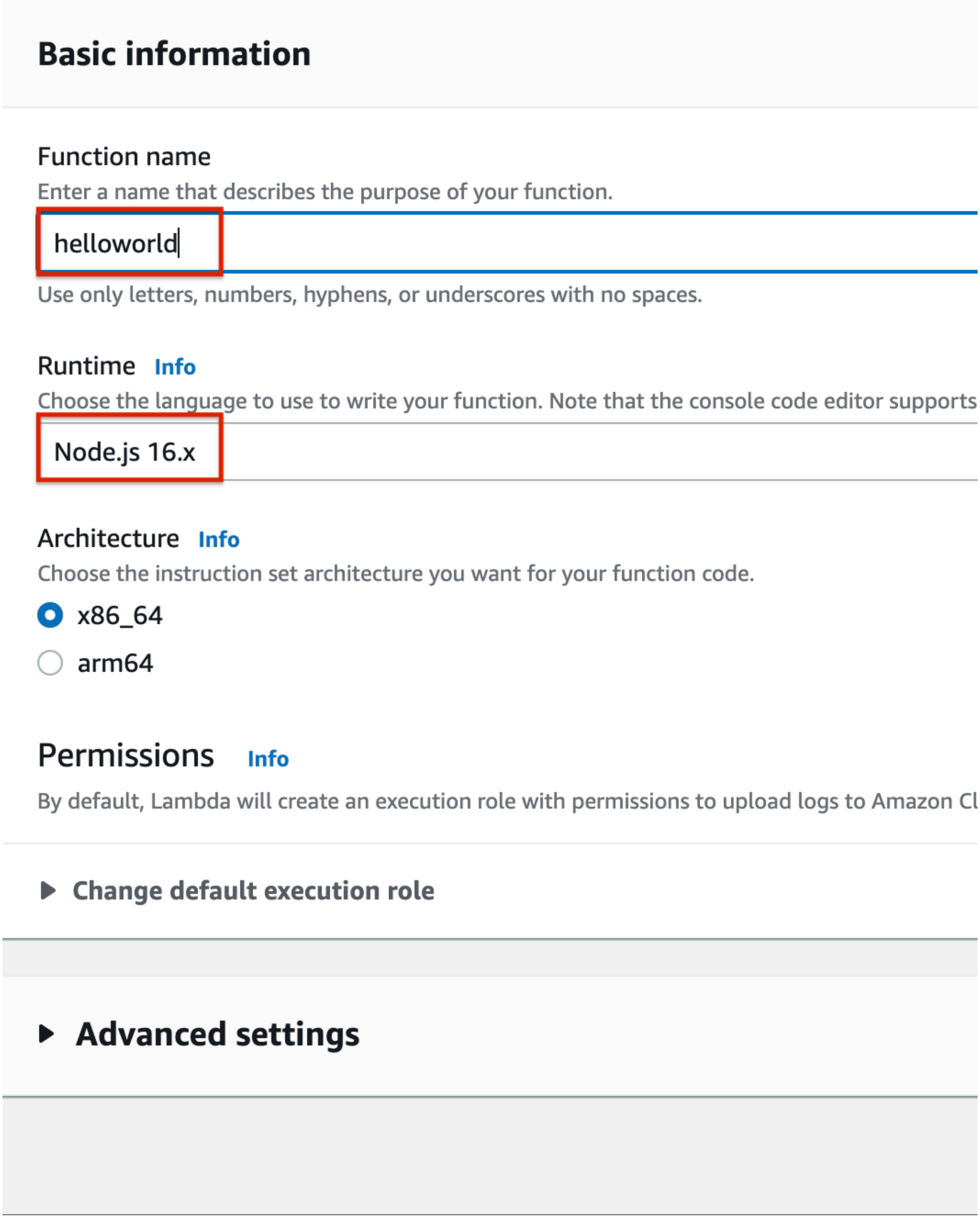
Choose one of the following options to create your function.

● 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.

Contai
Select a
function.

4. Provide basic information for your function - name of the function, runtime. You will be creating a Node.js function. So the runtime will be **Node.js 16.x**. Allow the rest to be default and click the **Create Function** button.



Basic information

Function name

Enter a name that describes the purpose of your function.

helloworld

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

Node.js 16.x

Architecture Info

Choose the instruction set architecture you want for your function code.

● x86_64

○ arm64

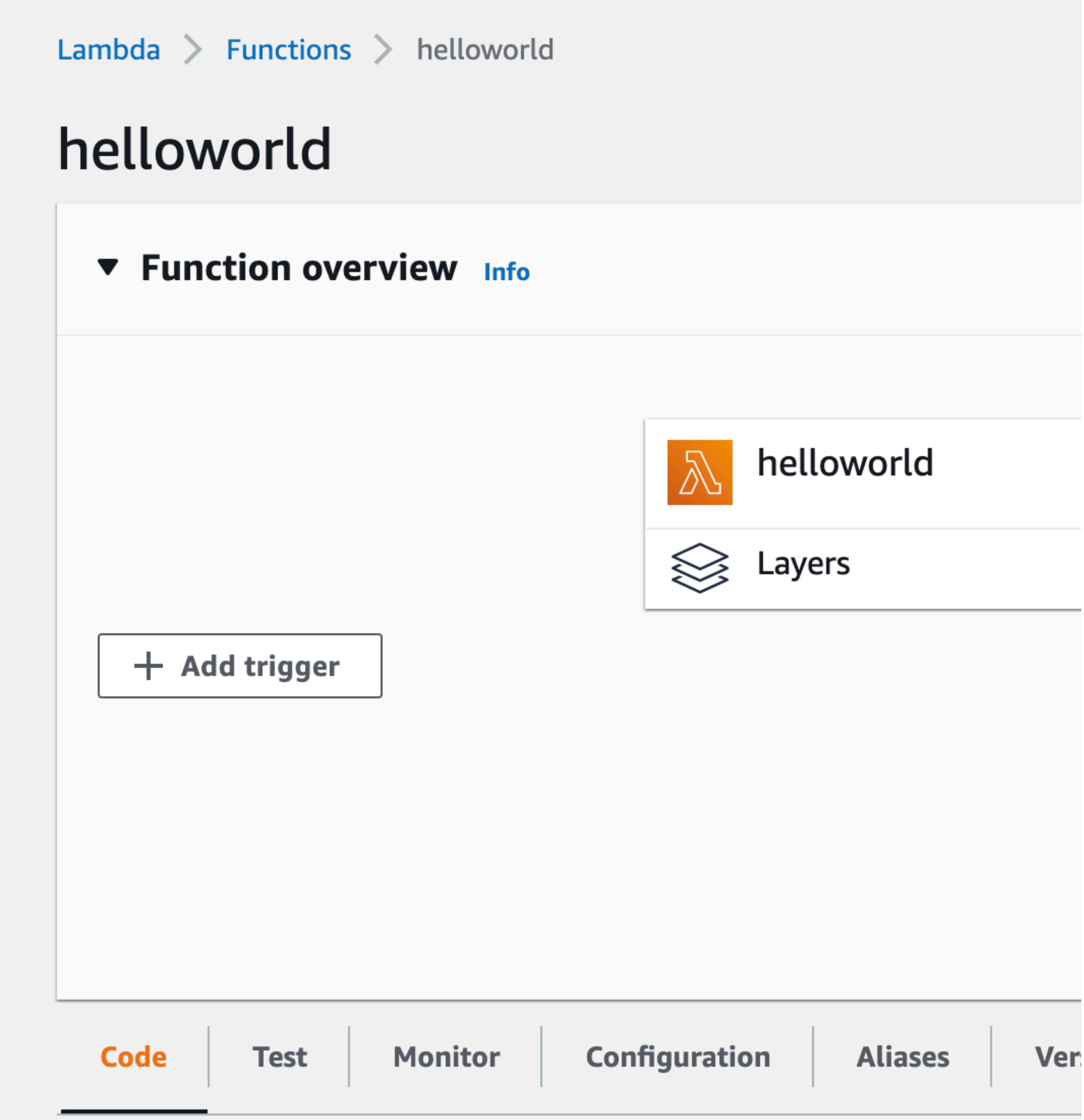
Permissions Info

By default, Lambda will create an execution role with permissions to upload logs to Amazon Cl

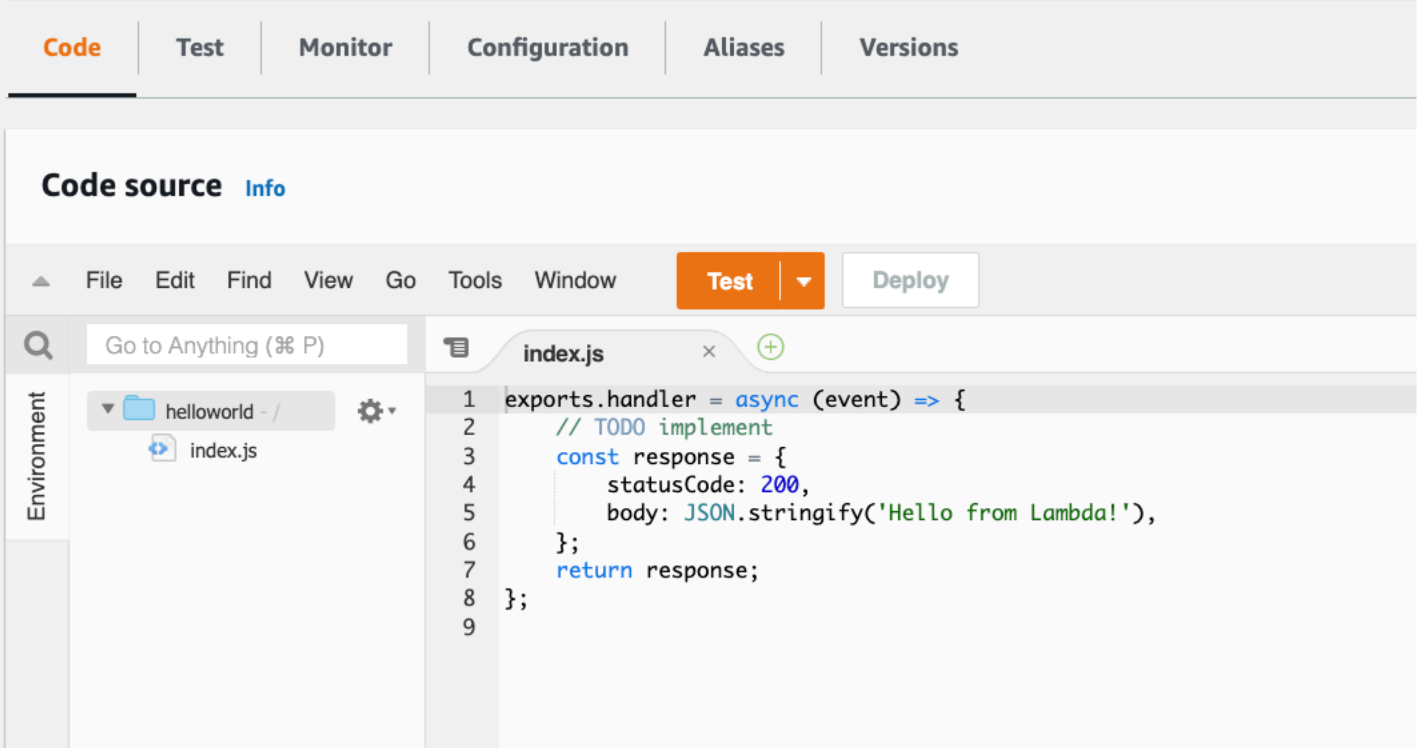
▶ Change default execution role

▶ Advanced settings

5. After a few seconds, you will see the function details page once the function is created.



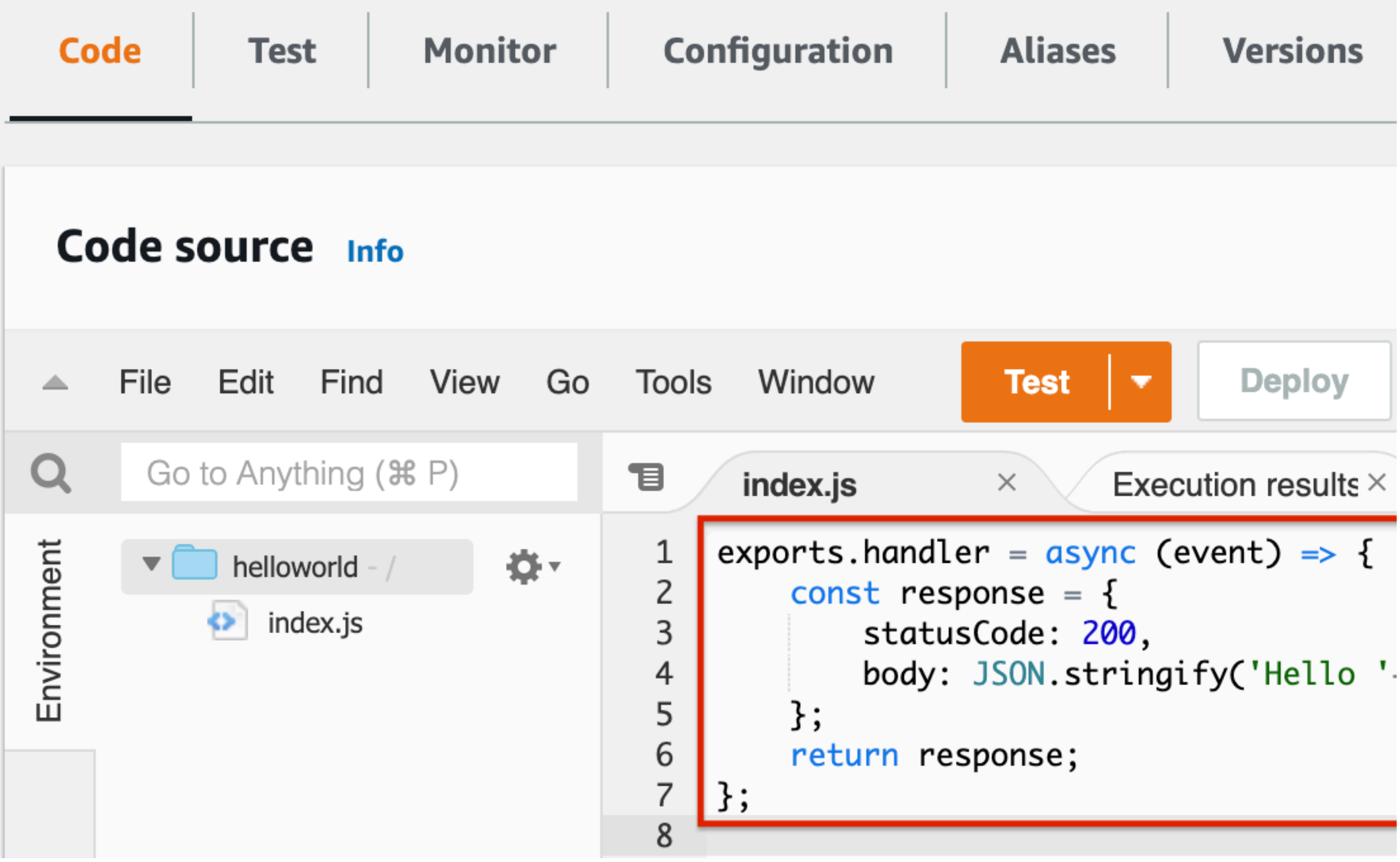
6. Scroll down on the same page to see the default **Hello Lambda** code prewritten in the **Code** tab.



```
exports.handler = async (event) => {
    // TODO implement
    const response = {
        statusCode: 200,
        body: JSON.stringify('Hello from Lambda!'),
    };
    return response;
};
```

7. Replace the code with the following custom code. This code will take the **name** parameter from the event and return a personalized **Hello**. Click **Deploy** once you add the script.

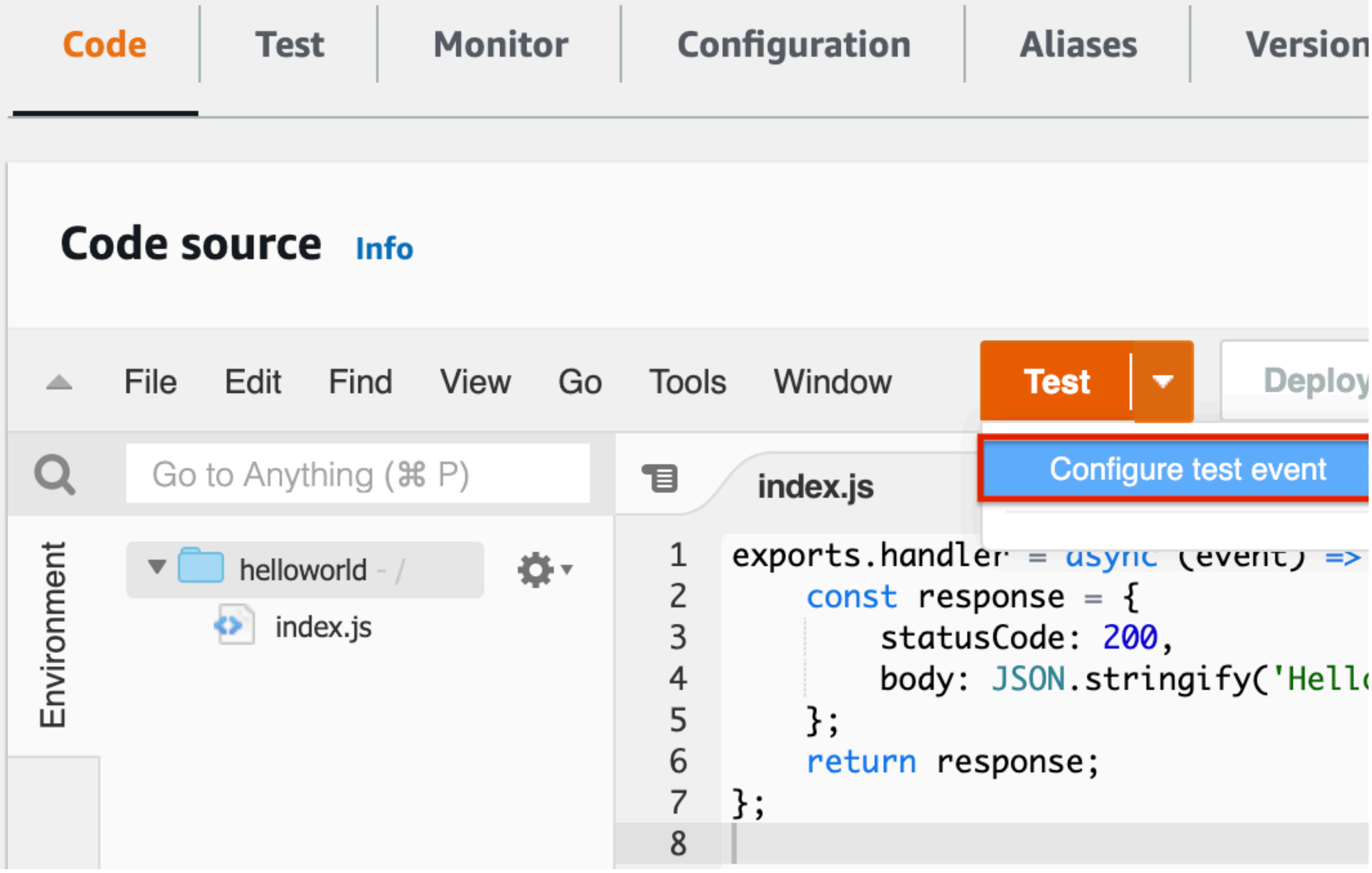
```
exports.handler = async (event) => {
    const response = {
        statusCode: 200,
        body: JSON.stringify('Hello '+event['name'] + "!")
    };
    return response;
};
```



```
exports.handler = async (event) => {
    const response = {
        statusCode: 200,
        body: JSON.stringify('Hello '
    };
    return response;
};
```

Task 3 - Test the Lambda function

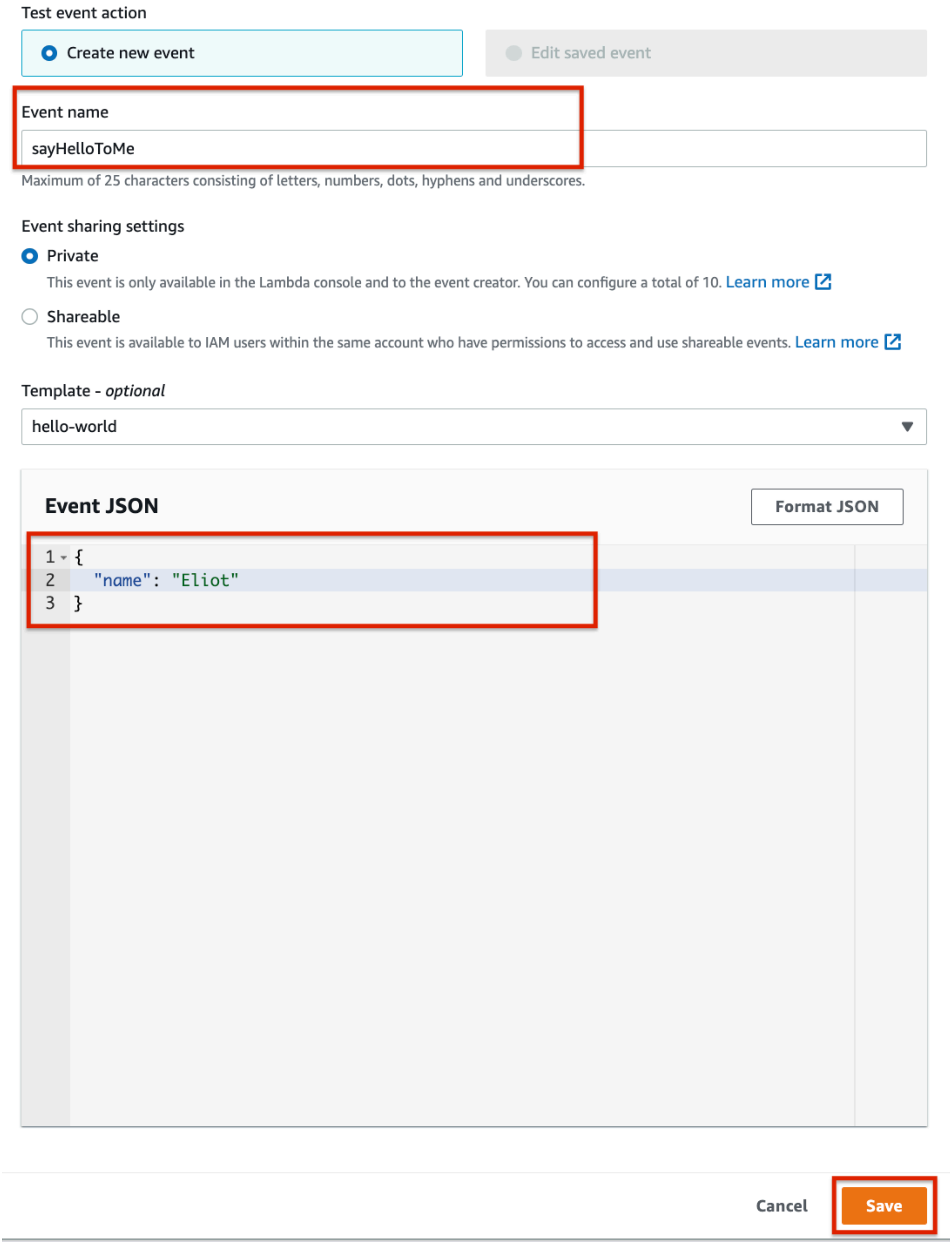
1. Once the code is deployed, you should configure an event and test the output of the Lambda function. Click the drop-down next to the **Test** button and choose **Configure test event**.



```
exports.handler = async (event) =>
    const response = {
        statusCode: 200,
        body: JSON.stringify('Hell
    };
    return response;
};
```

2. Give the event a name and then enter or copy and paste the JSON below to add the parameter you want to pass to the event. This event is triggered when you want to test your Lambda function. Add the **Event JSON** and click **Save**.

```
{
  "name":"Eliot"
}
```



Test event action

● Create new event ○ Edit saved event

Event name

sayHelloToMe

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

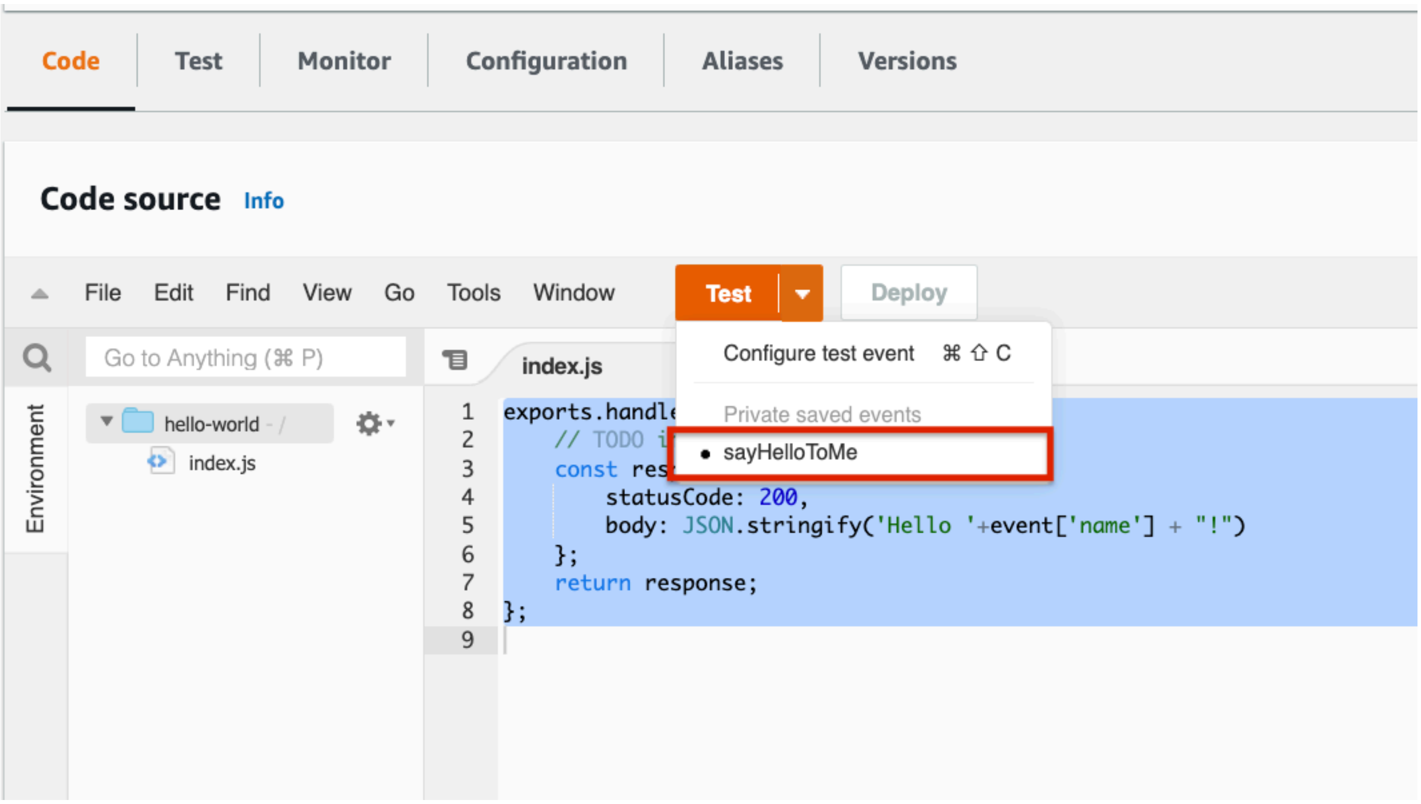
Template - *optional*

hello-world

Event JSON

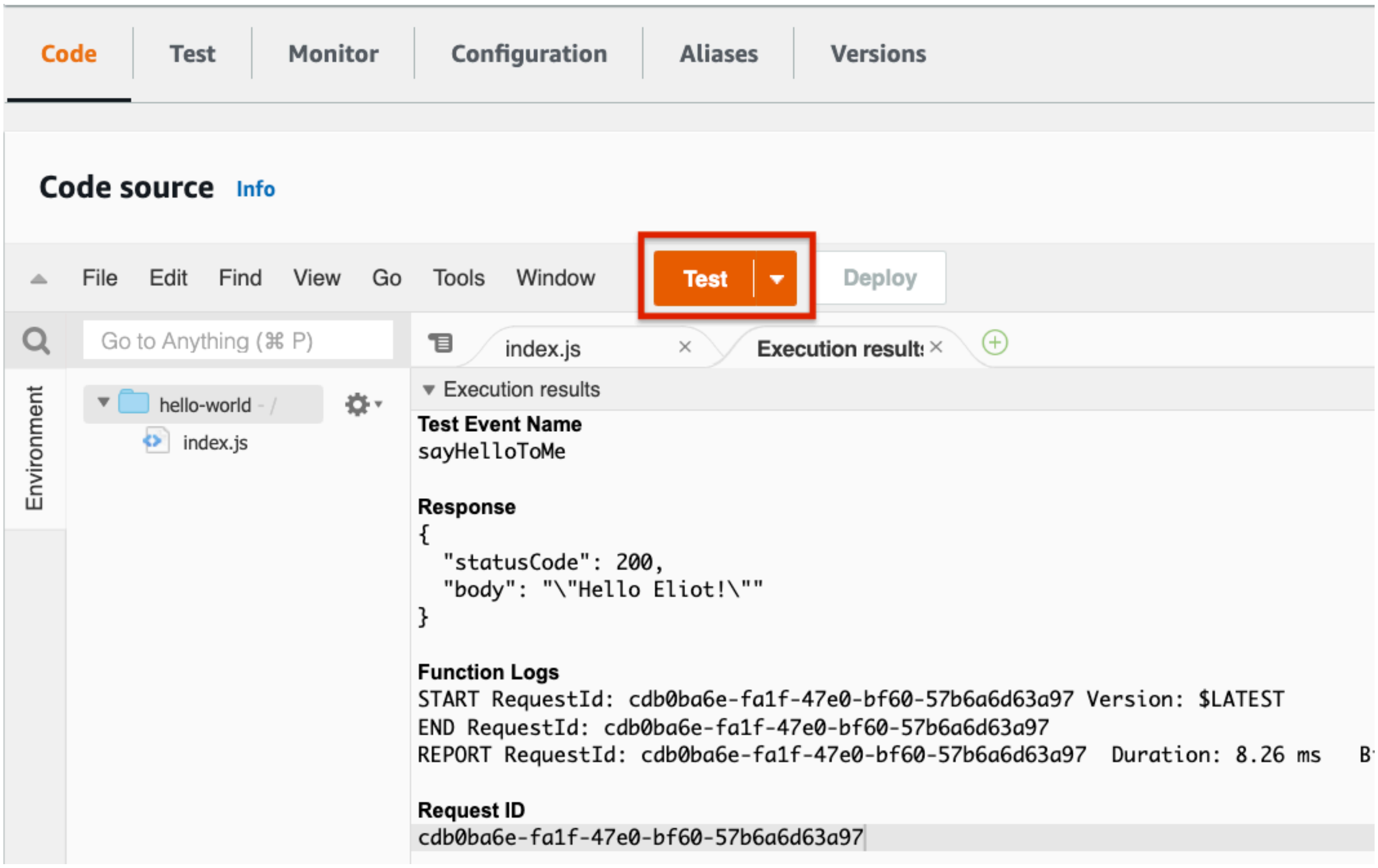
```
{
  "name": "Eliot"
}
```

3. Check if the event has been created by clicking the drop-down next to **Test** again.



```
exports.handle
    // TODO i
    const res
        statusCode: 200,
        body: JSON.stringify('Hello '+event['name'] + "!")
    };
    return response;
};
```

4. Click **Test** to invoke the Lambda function and see the response. You should see the response as shown in the image below.



Test Event Name

sayHelloToMe

Response

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

Function Logs

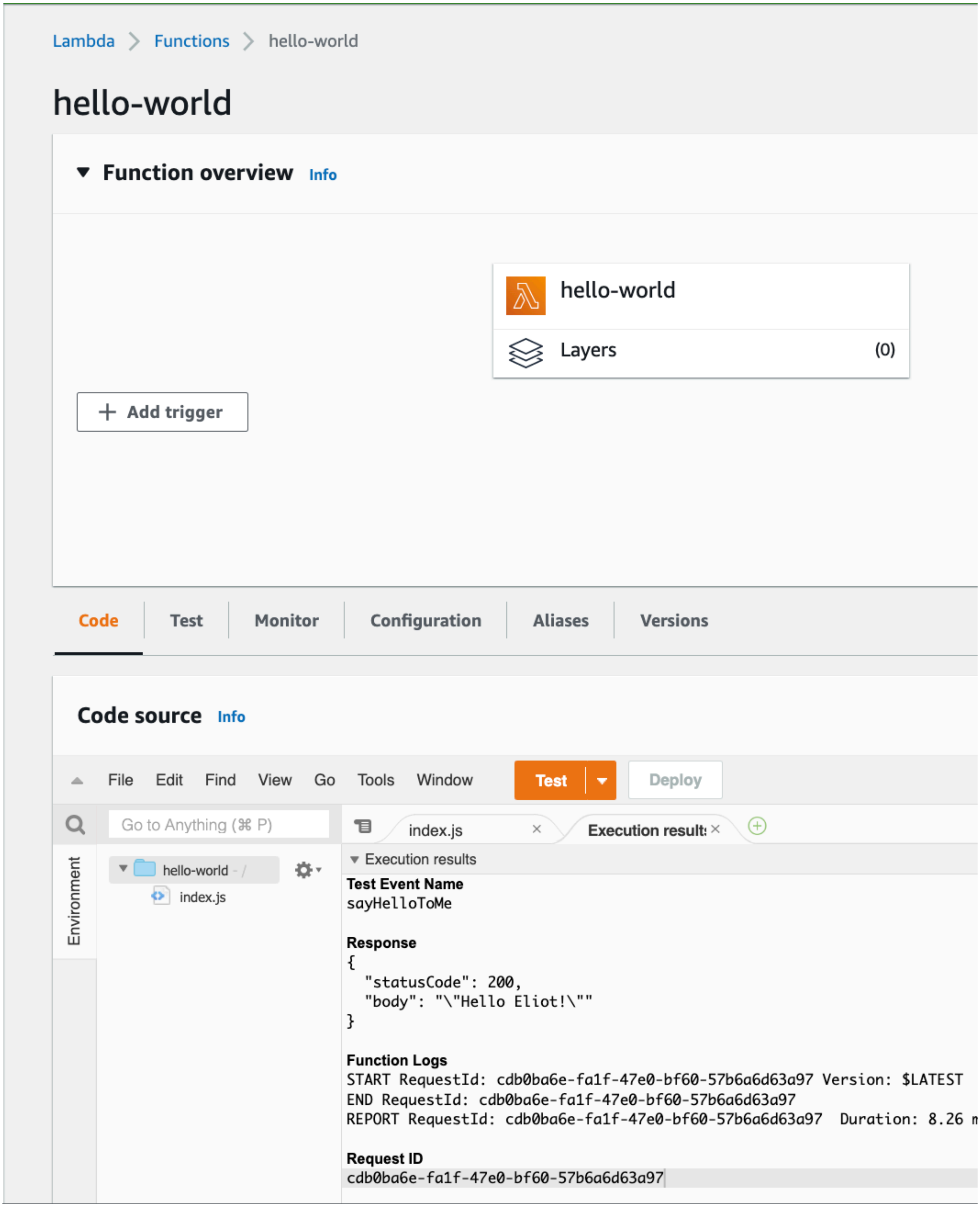
```
START RequestId: cdb0ba6e-fa1f-47e0-bf60-57b6a6d63a97 Version: $LATEST
END RequestId: cdb0ba6e-fa1f-47e0-bf60-57b6a6d63a97
REPORT RequestId: cdb0ba6e-fa1f-47e0-bf60-57b6a6d63a97  Duration: 8.26 ms   B
```

Request ID

cdb0ba6e-fa1f-47e0-bf60-57b6a6d63a97

Task 4 - Delete the Lambda function

1. Now that you have created a Lambda function and successfully tested it, you can delete it. On the top right, click the **Action** menu and choose the **delete** option.



Test Event Name

```
sayHelloToMe
```

Response

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

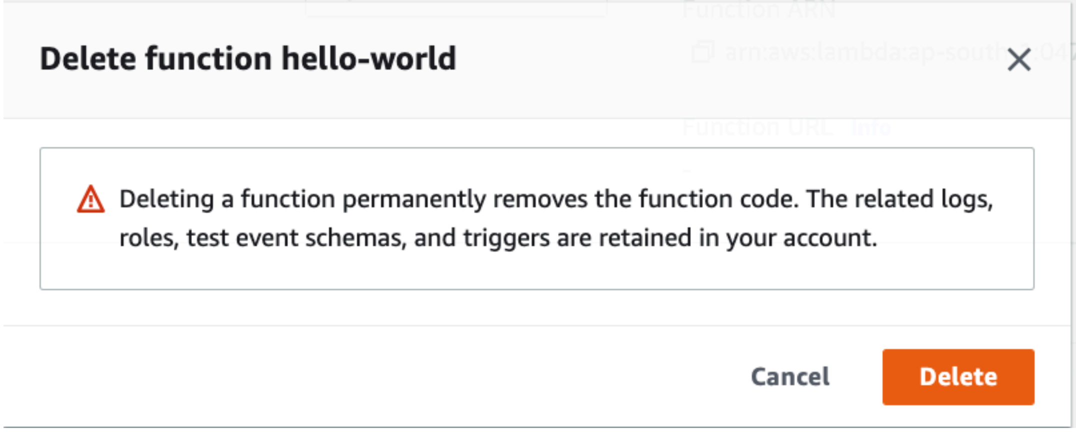
Function Logs

```
START RequestId: cdb0ba6e-fa1f-47e0-bf60-57b6a6d63a97 Version: $LATEST
END RequestId: cdb0ba6e-fa1f-47e0-bf60-57b6a6d63a97
REPORT RequestId: cdb0ba6e-fa1f-47e0-bf60-57b6a6d63a97  Duration: 8.26 m
```

Request ID

```
cdb0ba6e-fa1f-47e0-bf60-57b6a6d63a97
```

2. When it asks for confirmation, you can confirm that you want to delete the action.



Delete function hello-world

Deleting a function permanently removes the function code. The related logs, roles, test event schemas, and triggers are retained in your account.

Congratulations! You just created your first AWS Lambda function.

Tutorial details

Author: Lavanaya T S

Contributors: Pallavi Rai

Skills Network