Guided Lab: Creating a NodeJS Function in AWS Lambda

Description

AWS Lambda is a serverless computing service provided by Amazon Web Services (AWS) that allows you to run code without having to manage or provision servers. With AWS Lambda, you can execute code in response to events, such as changes to data in an Amazon S3 bucket or an update to a DynamoDB table, without the need for server management. AWS Lambda automatically scales your application by running code in response to each trigger and only charges you for the compute time you consume.

Node.js is a runtime environment that allows you to execute JavaScript code on the server side, outside of a web browser. It is built on the V8 JavaScript engine, which is used in Google Chrome, and it provides an event-driven, non-blocking I/O model, making it lightweight and efficient for building scalable network applications.

In this lab, you will learn how to create, test, and modify a NodeJS function in AWS Lambda using a pre-built template. AWS Lambda is a serverless compute service that lets you run code without provisioning or managing servers. This lab focuses on using the “Hello world function” blueprint, which provides a basic starting point for NodeJS functions.

Prerequisites

This lab assume you have basic understanding of AWSLambda and Node.JS programming language.

If you find any gaps in your knowledge, consider taking the following lab:

- Creating an AWS Lambda function

Objectives

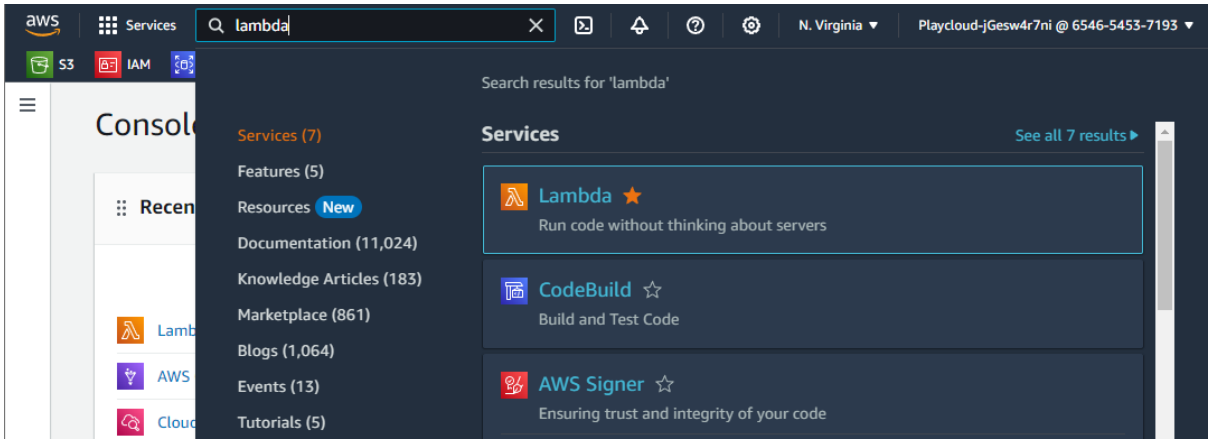
By the end of this lab, you will be able to:

- Create a serverless function using AWS Lambda and a pre-built NodeJS template.
- Configure and run test events with custom data inputs.
- Understand how to modify and deploy code changes in AWS Lambda.

Lab Steps

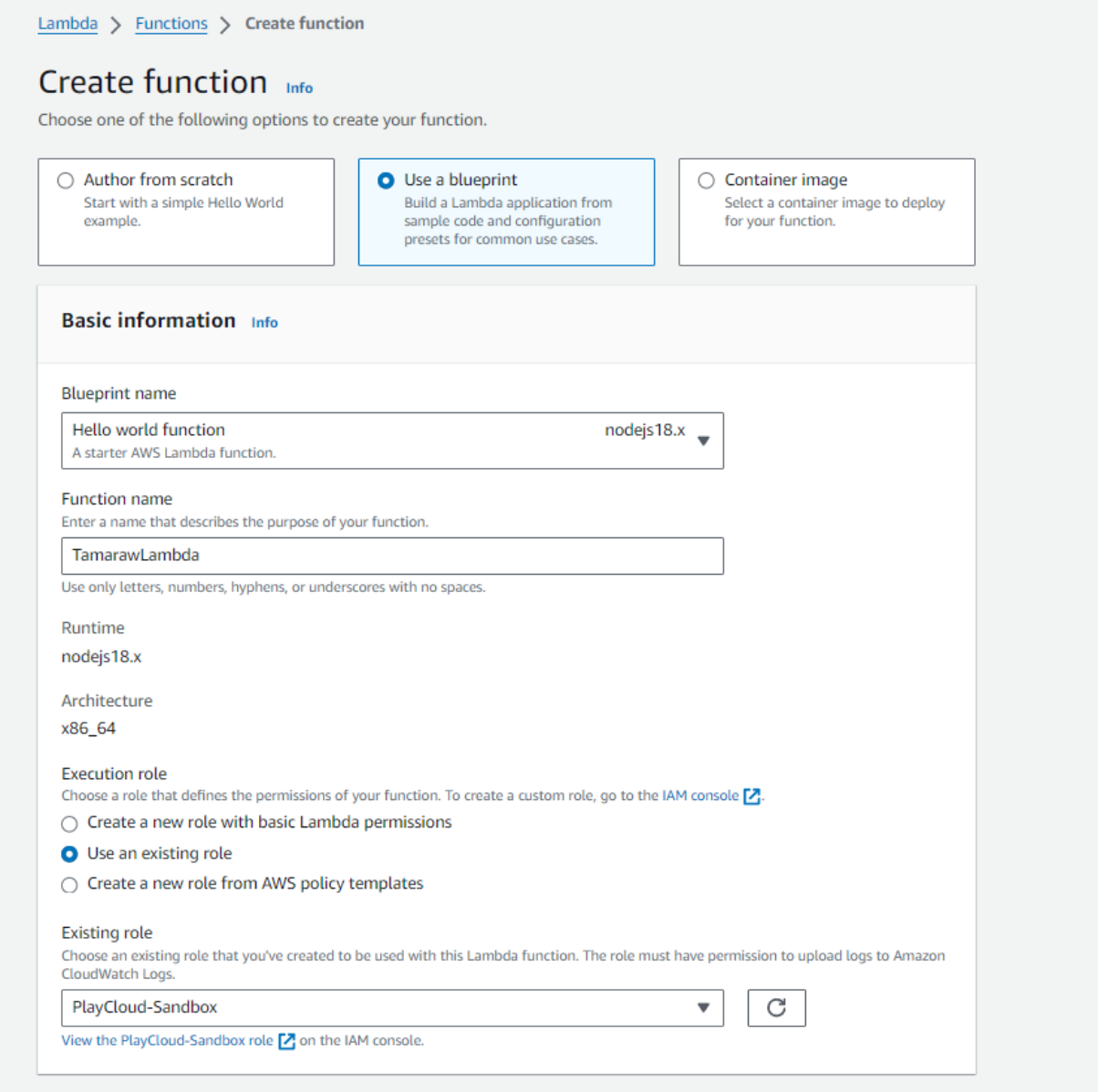
CREATE AN AWS LAMBDA FUNCITON

1. Navigate to AWS Lambda Console



2. Create Function using the following confgurations:

- Select **Use a blueprint**
- Blueprint name : **Hello world function nodejs18.x**
- Function name: **TamarawLambda**
- **Execution role:**
    - Select Use an Existing Role: **PlayCloud-Sanbox**

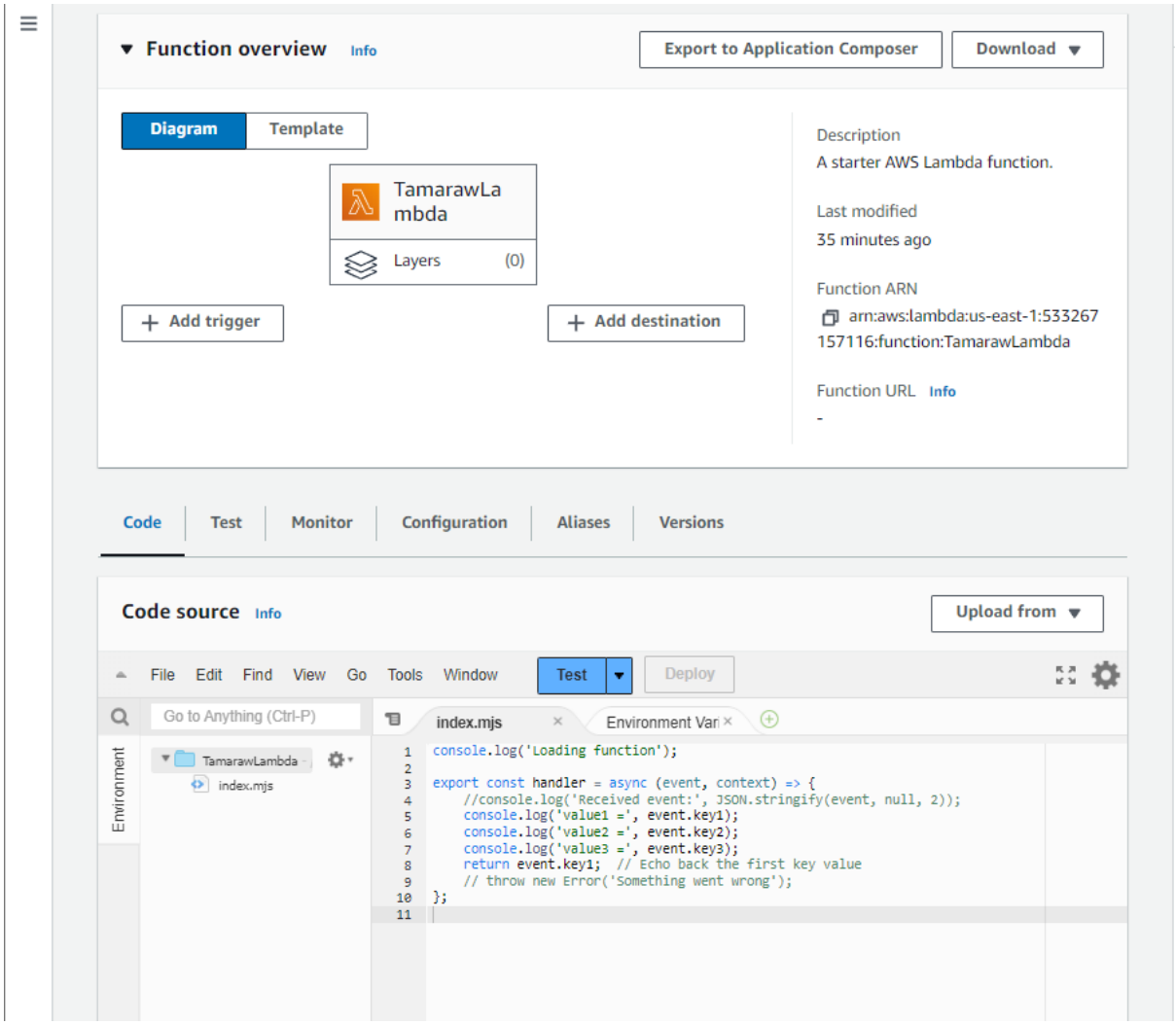


- Click on **Create Function**

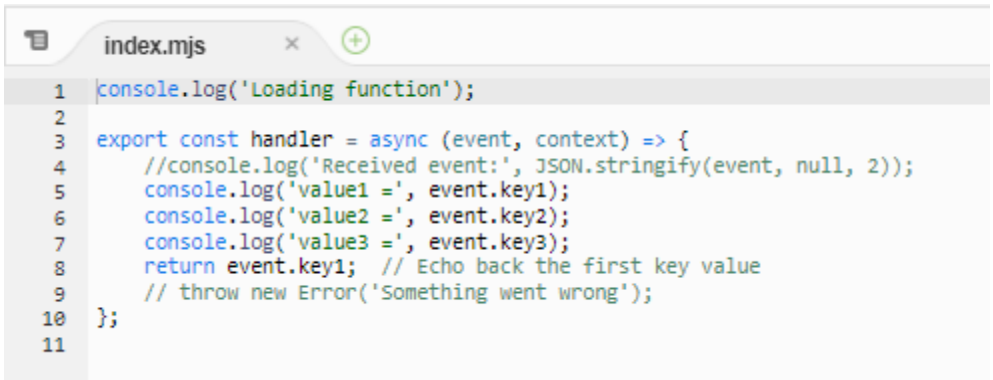
nvoke the Lambda Function Using the Test Button

1. Once your function is created, you'll be directed to the function's dashboard

***Note**: We are using the **old console editor** for this lab. You can switch to the **new or old editor** as you desire; the process remains the same, but the interface may look slightly different.*



2. Take your time to review the code.



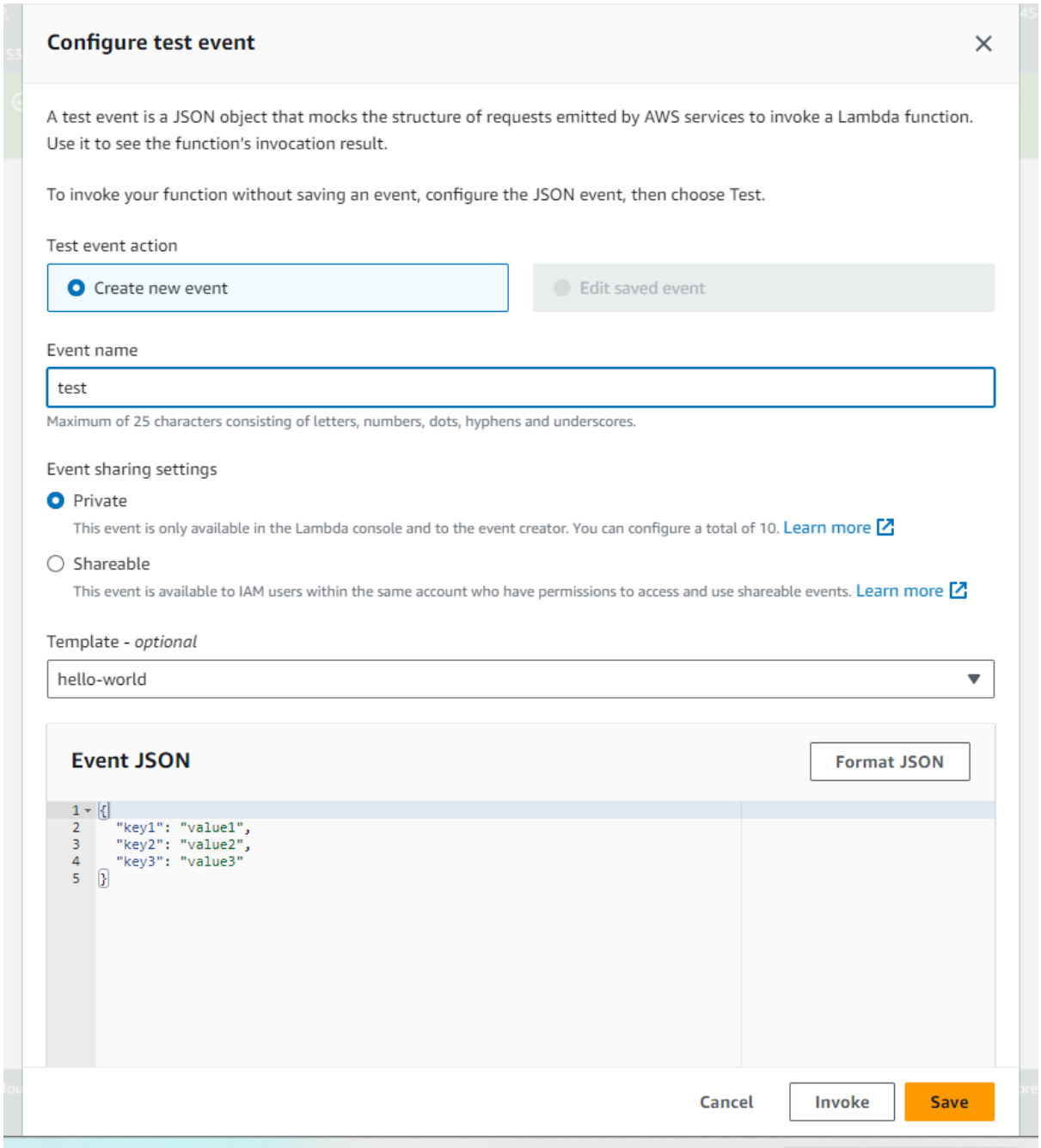
- **console.log('Loading function');** – This logs the message "Loading function" to indicate that the function has started executing.
- **export const handler = async (event, context) => { … }** – This defines the main function (called the handler) that AWS Lambda will run when triggered. It uses async syntax, which means it can handle asynchronous operations.

- **console.log('value1 =', event.key1);** – This logs the value of key1 from the event data passed to the function.
- **console.log('value2 =', event.key2);** – This logs the value of key2 from the event data.
- **console.log('value3 =', event.key3);** – This logs the value of key3 from the event data.
- **return event.key1;** – This returns the value of key1 from the event, effectively "echoing" it back as the function's result

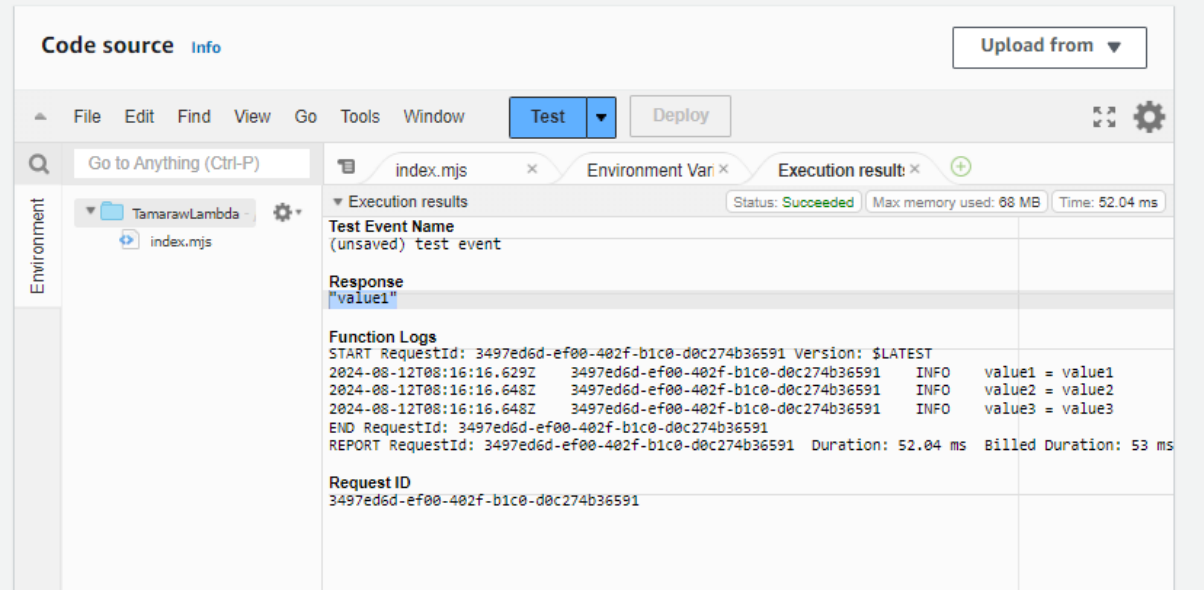
---

3. Click the blue **Test** button at the top-right of the page.

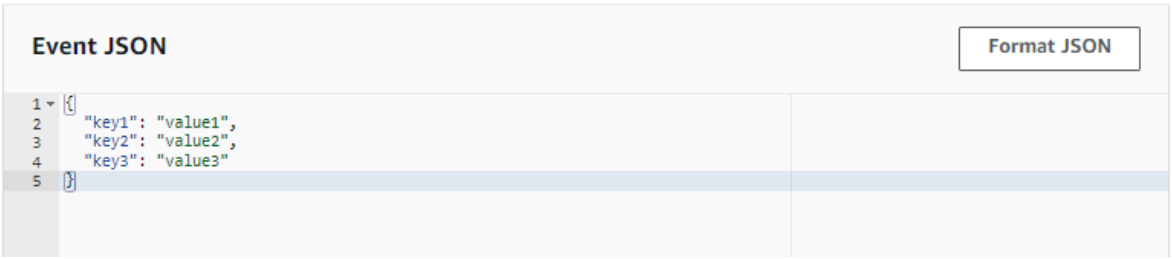
- A Configure test event dialog will pop up, give your test event a name (e.g. test)
- Then, for this initial test, leave the rest as default, save it and…
- Click on **Test** again.



4. Review the response:



Notice the response? This is becaue the default Test Event we triggered is this Event JSON:

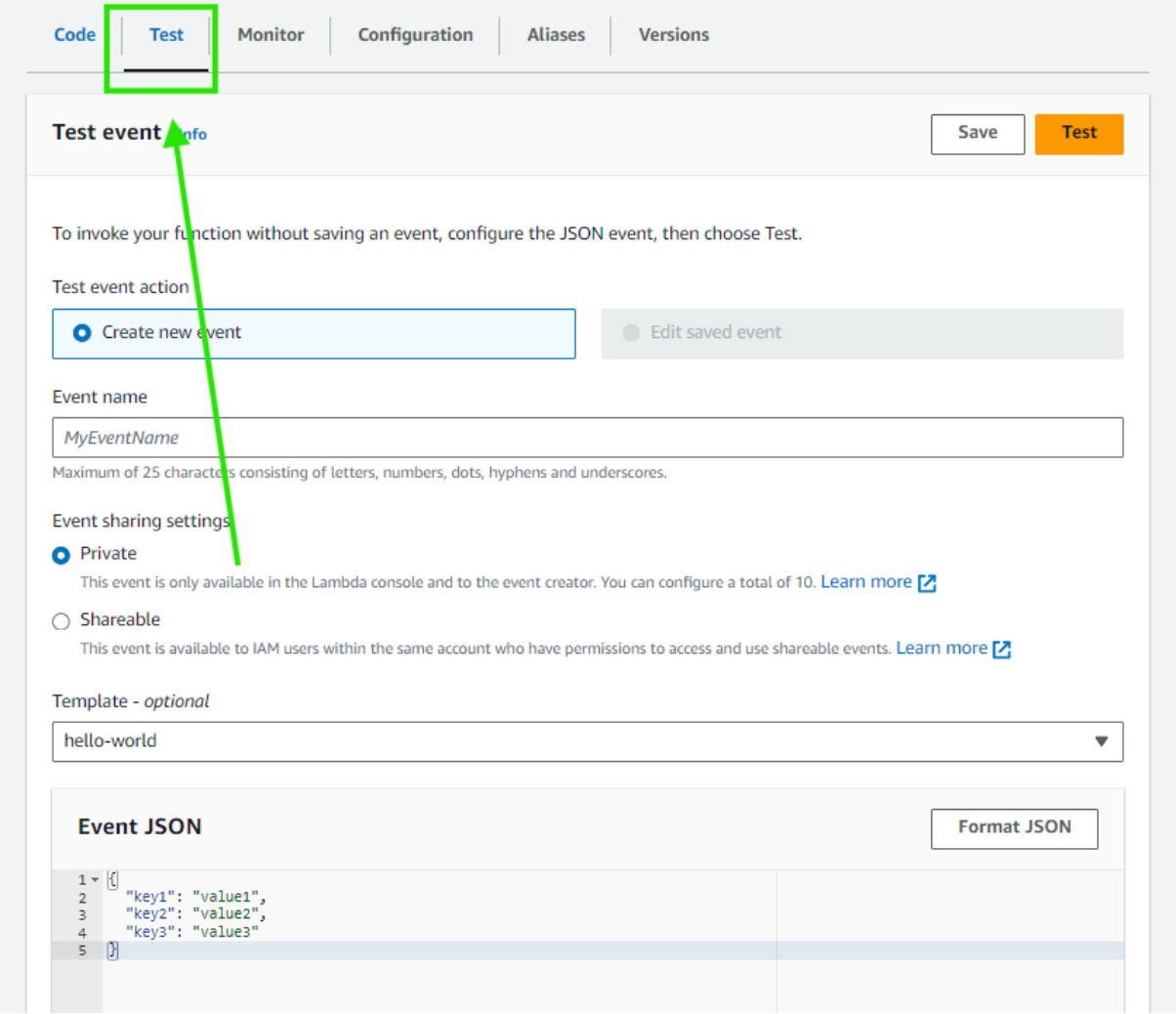


An **event** is the input that your Lambda function processes. You can create and save up to 10 test events per function. Saved events are stored in Lambda, meaning they'll be available even if you switch browsers or devices. Unsaved events will only be available during your current session and will be lost if the session ends.

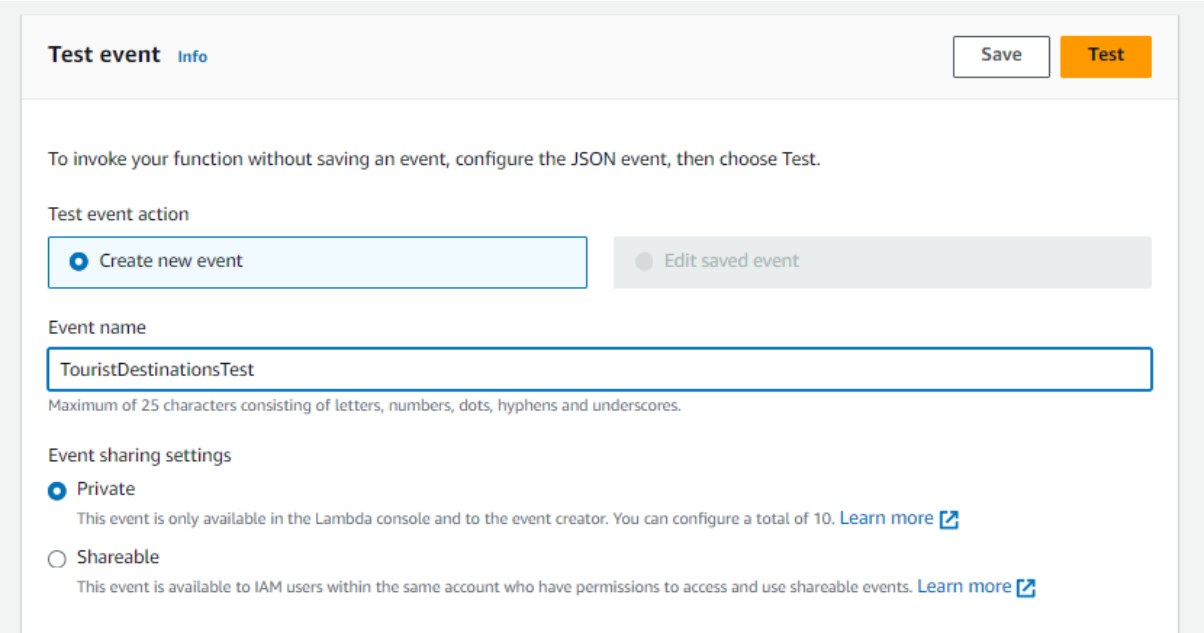
Running a test event in the console triggers your function synchronously. Lambda takes the event (in JSON format), converts it into an object, and passes it to your function's handler method for processing.

---

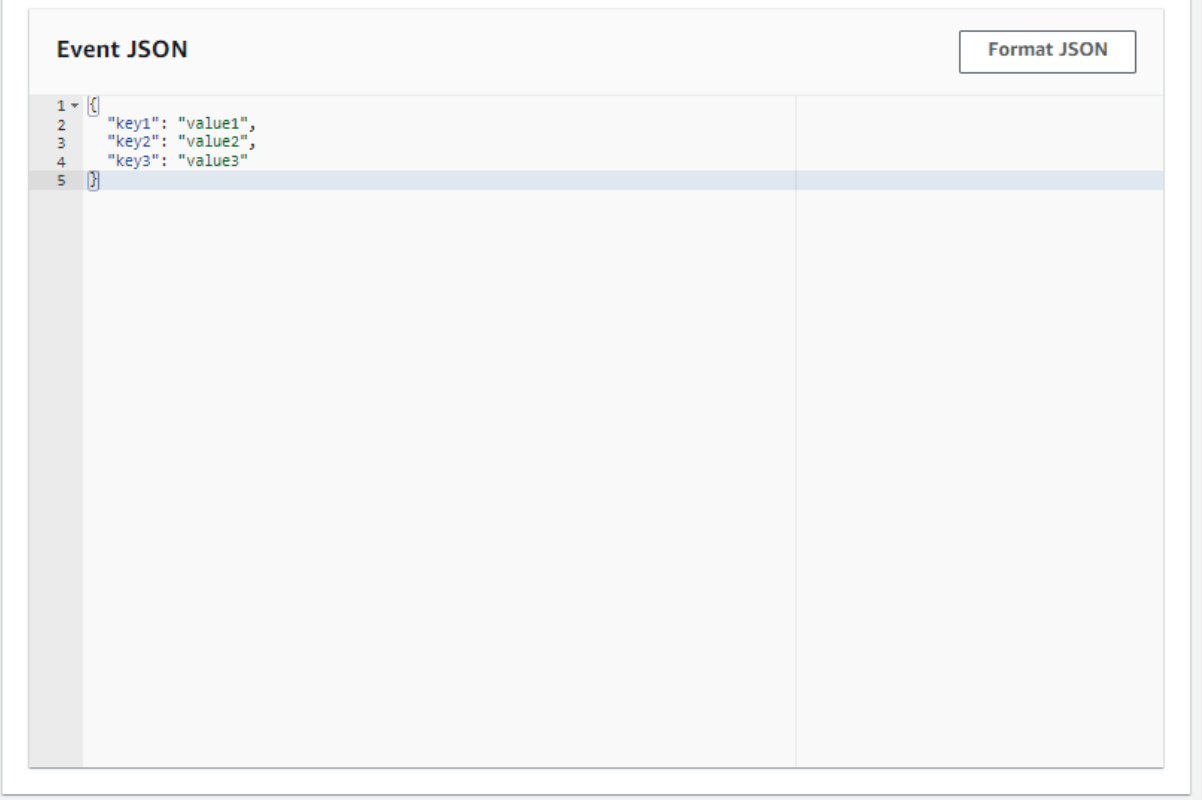
5. To modify this Even JSON, naviagte to the **Test Tab** beside **Code tab**



6. **Enter TouristDestinationsTest for the Event name**



7. Scroll down to the Event JSON



8. Replace the default values with the following:

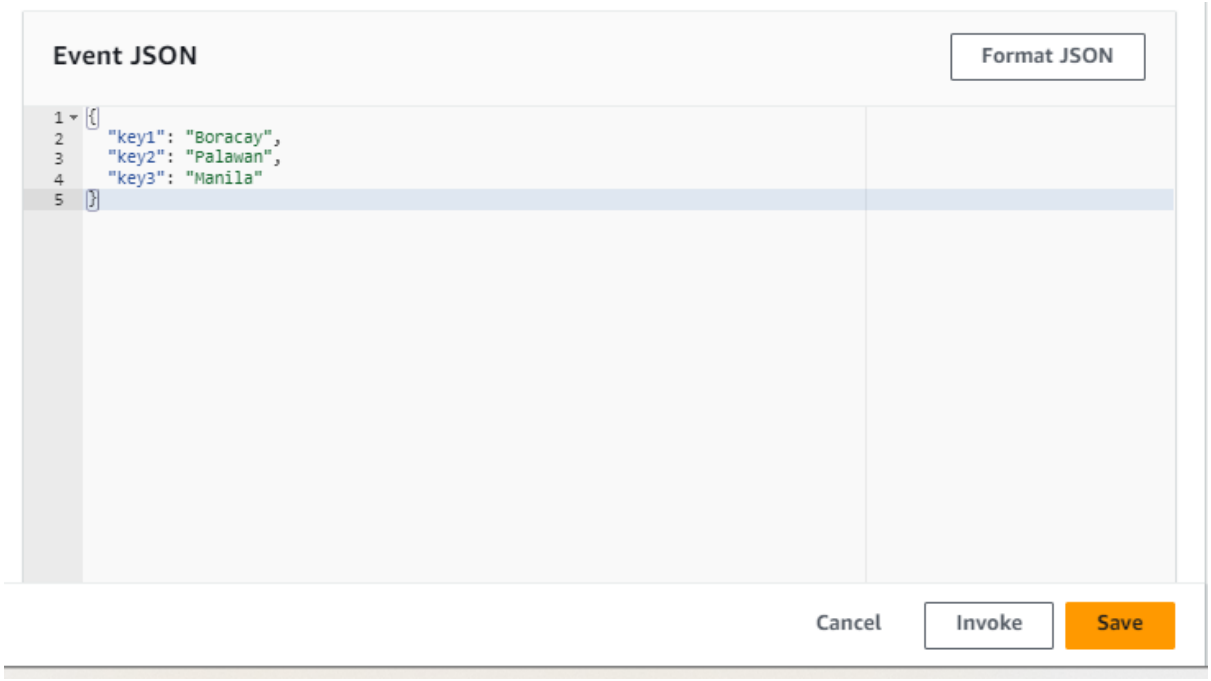
{

"key1": "Boracay",

"key2": "Palawan",

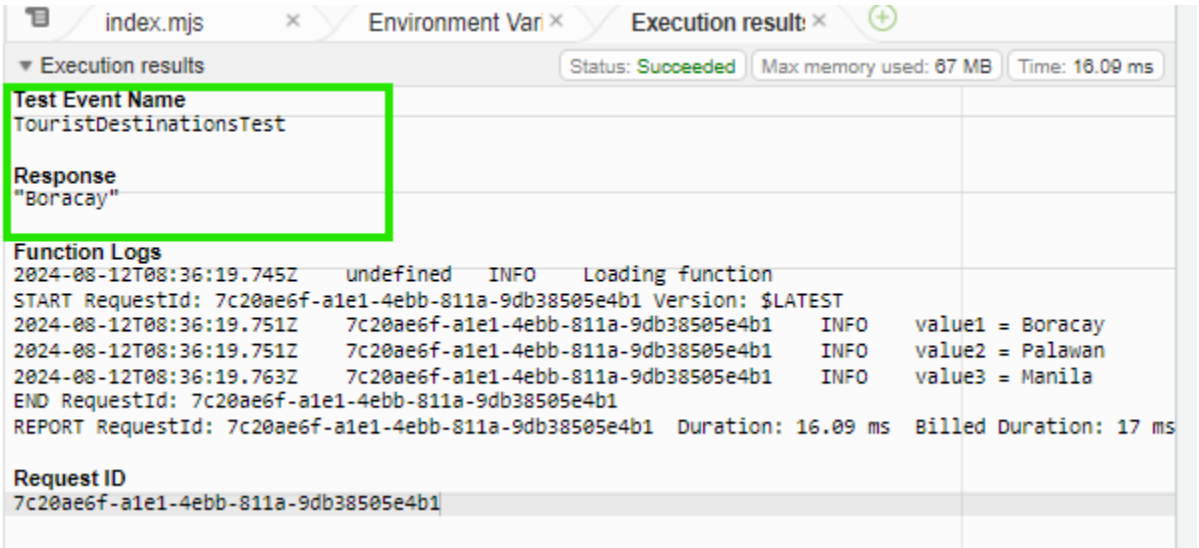
"key3": "Manila"

}



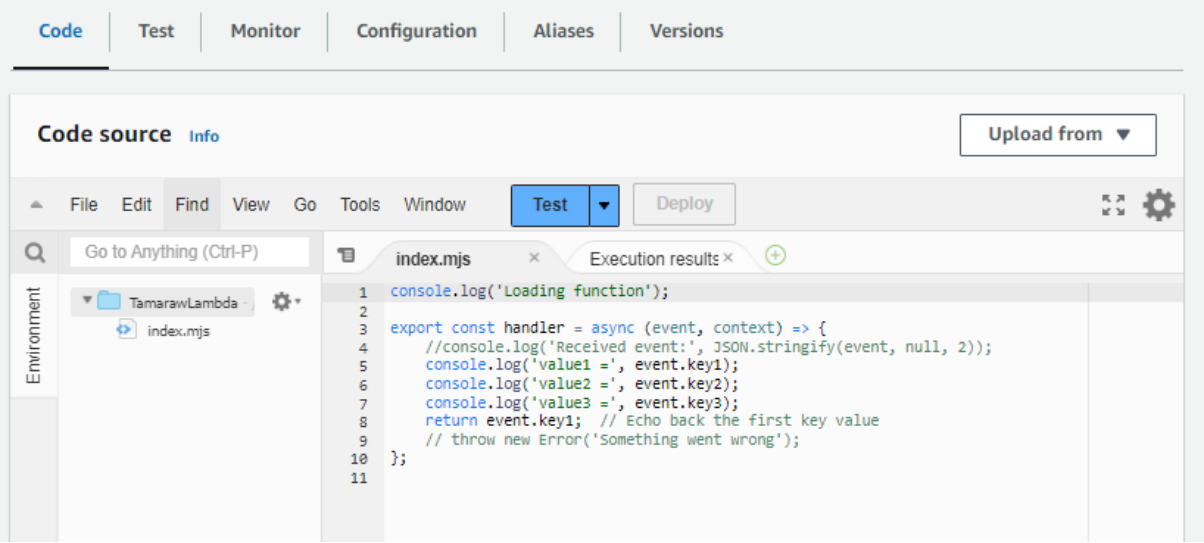
- Click on Save

9. Now, click on Test again. Observe the output. The function should return "Boracay" since the original code returns event.key1.

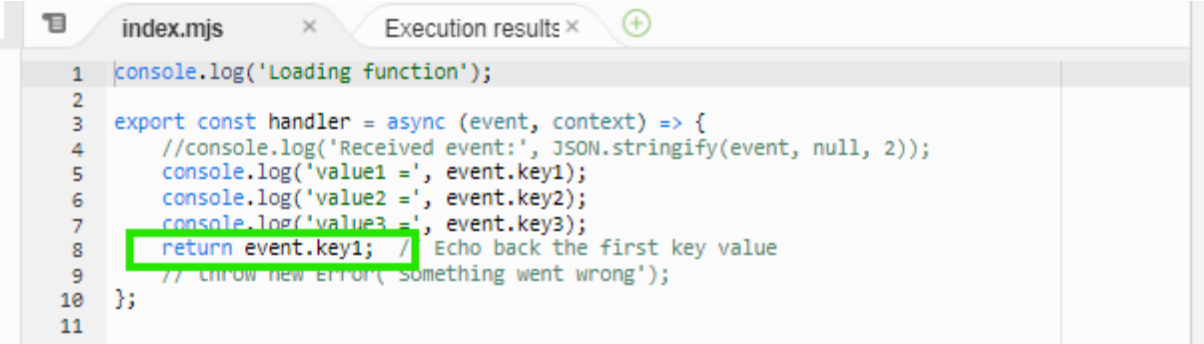


Modify the Function Code

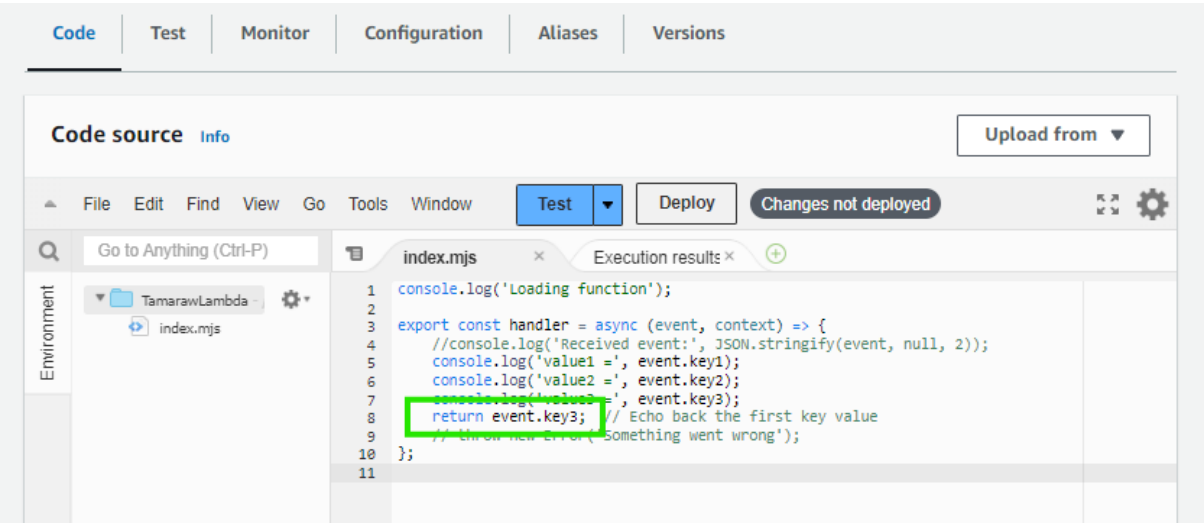
1. Navigate to the **Code tab** or **Code source** section.



2. Locate **Line 8** in the code, which currently reads:

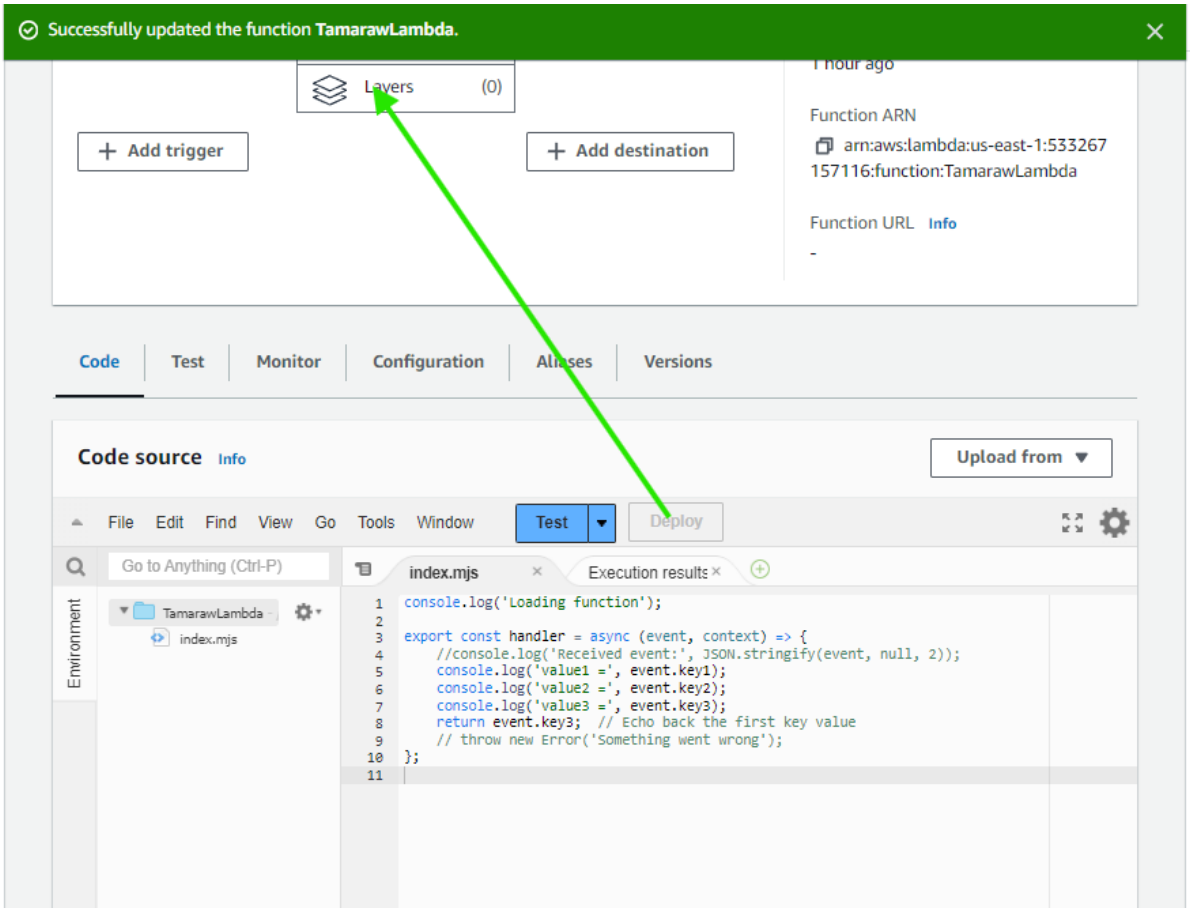


3. Modify this line to return a different key (e.g., event.key3)

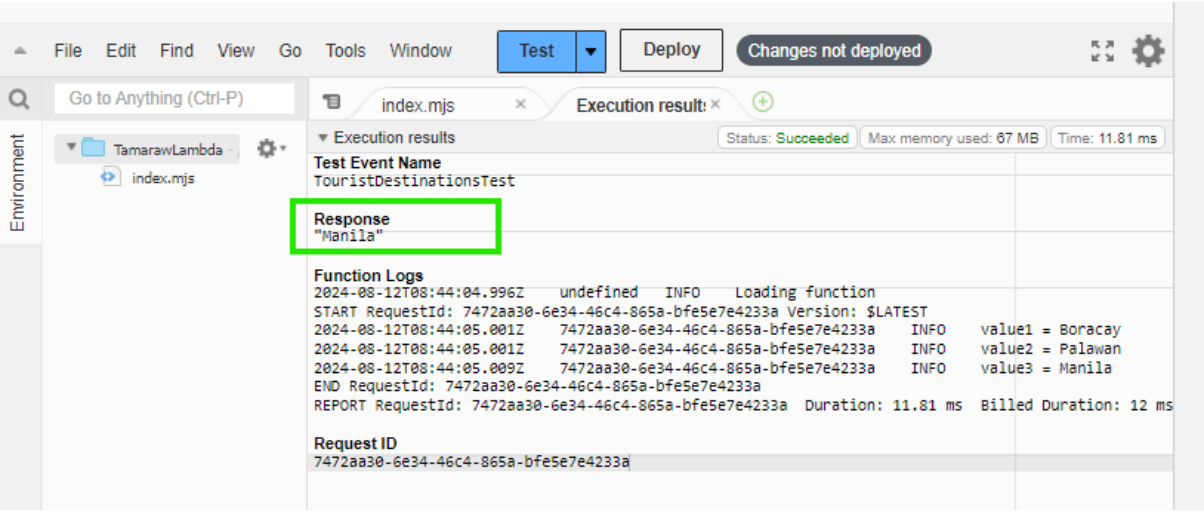


4. Click on **Deploy**

When testing your Lambda function, it's important to remember that any code changes must be deployed before you can test and see the changes in action.



5. Click on Test again. Now, the function should return "Manila" as the output, reflecting the change made to event.key3.



That’s It! congratulations! You’ve successfully created a NodeJS function in AWS Lambda using a blueprint, configured test events with Filipino tourist destinations, and modified the function’s code to demonstrate the impact of code changes after deployment.

This lab serves as an introductory guide to using NodeJS in AWS Lambda, providing you with the foundational skills needed to build and deploy serverless functions. As you progress, you can explore more advanced features and capabilities of AWS Lambda to further enhance your serverless applications. Happy learning!