Guided Lab: Using Environment Variables in AWS Lambda

Description

Environment variables in AWS Lambda are key-value pairs that allow you to pass configuration settings to your function without hardcoding them in your code. This lab will demonstrate how to create and use environment variables in an AWS Lambda function.

Prerequisites

This lab assumes you have a basic understanding of AWS Lambda and Python programming.

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

- Creating an AWS Lambda function

Objectives

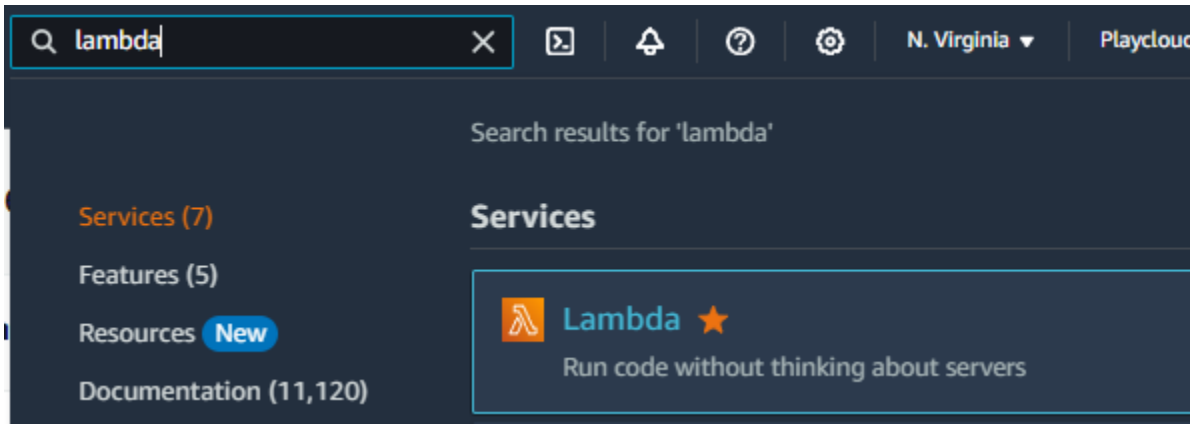
In this lab, you will:

- Create an AWS Lambda function using the console.
- Define environment variables and retrieve them within the Lambda function.
- Update environment variables without changing the function code.
- Test the Lambda function using environment variables.

Lab Steps

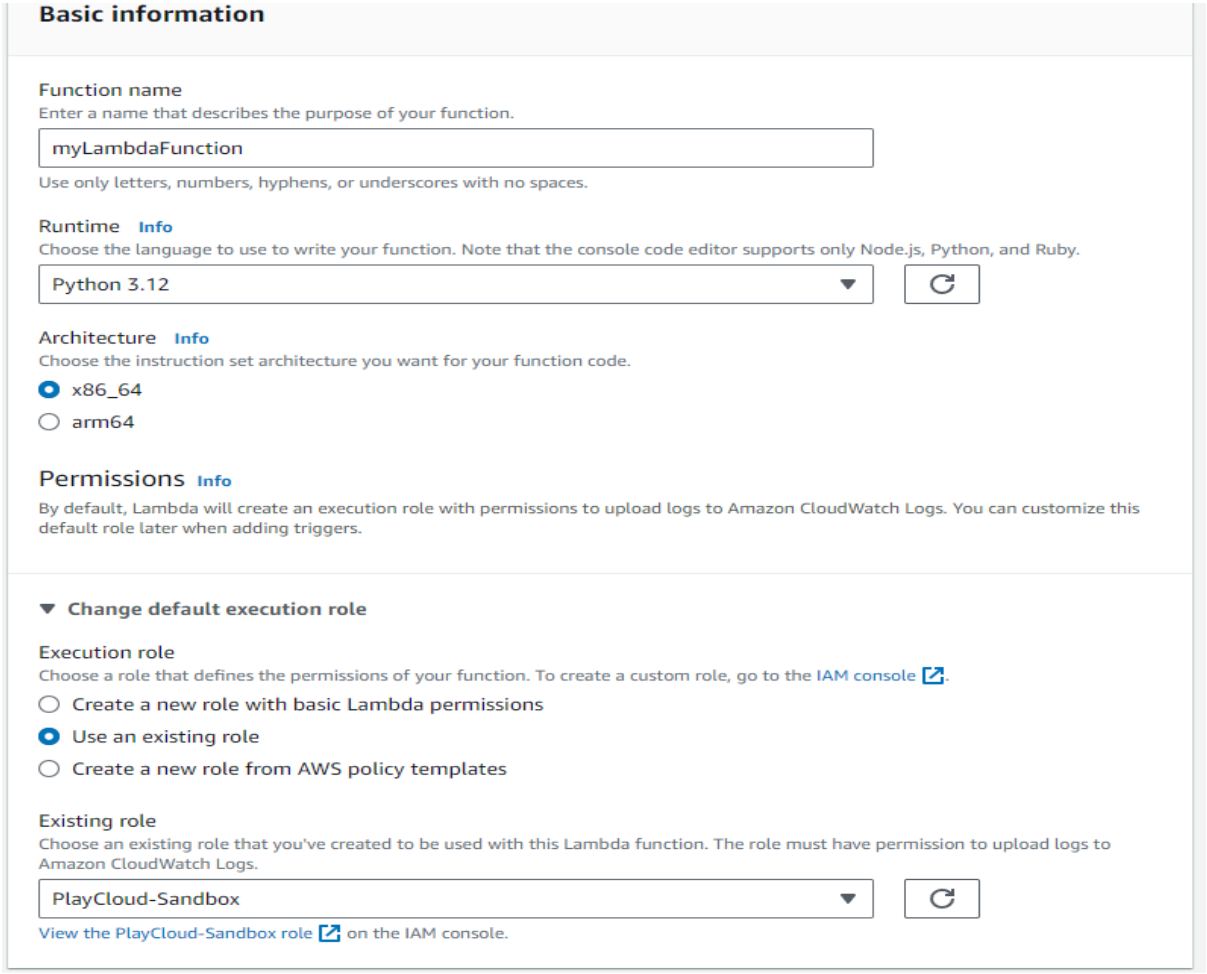
Create a Simple Lambda Function

1. Navigate to the AWS Lambda Console



2. Create a new Lambda function using the following configurations:

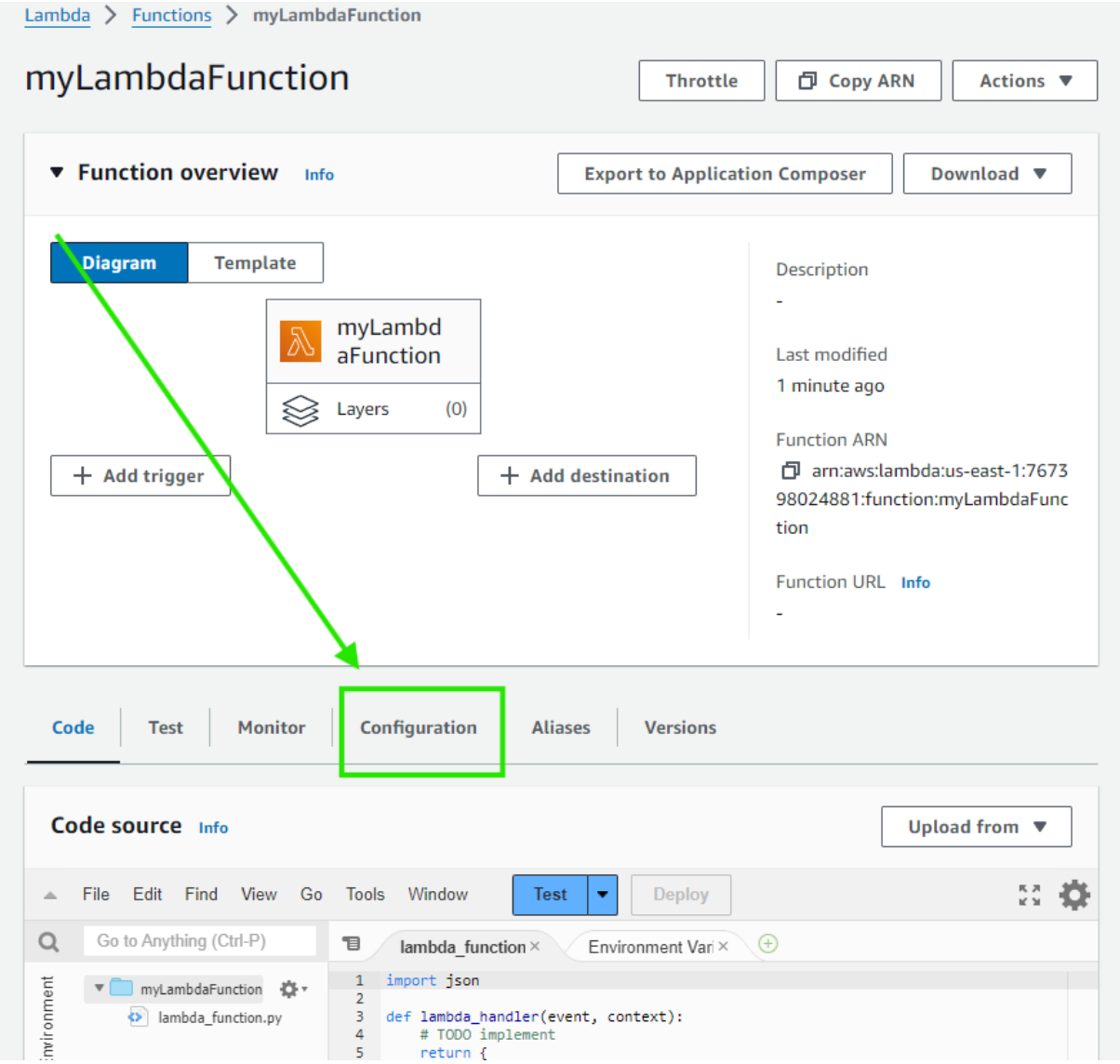
- Choose **Author from scratch.**
- Function name: myLambdaFunction
- Select Python 3.12 as the runtime
- .**Execution role:**
    - Select Use an Existing Role: PlayCloud-Sanbox



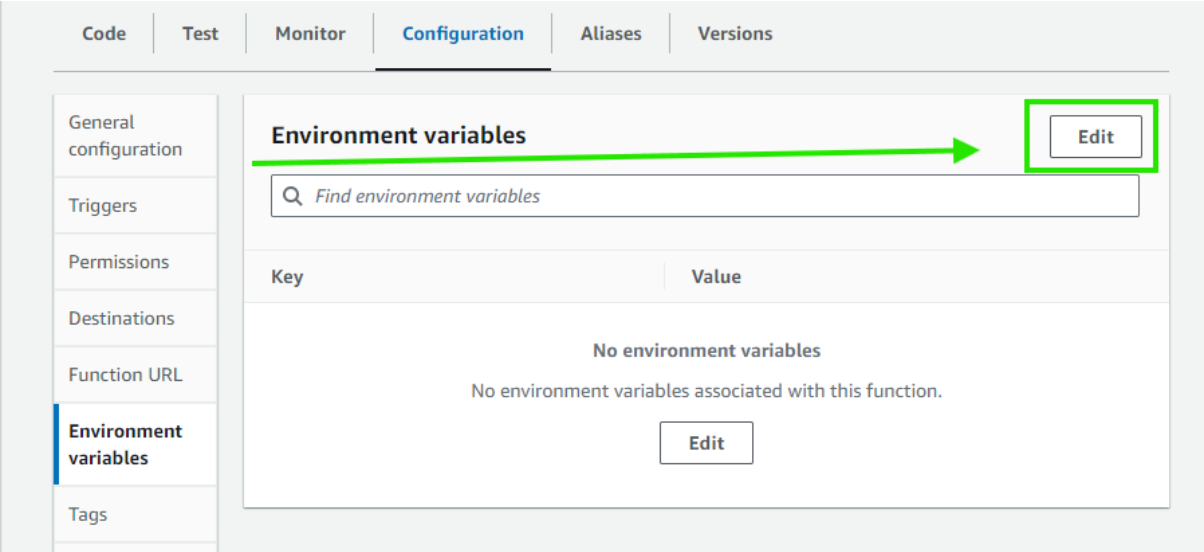
- Click **Create function**

Adding Environment Variables

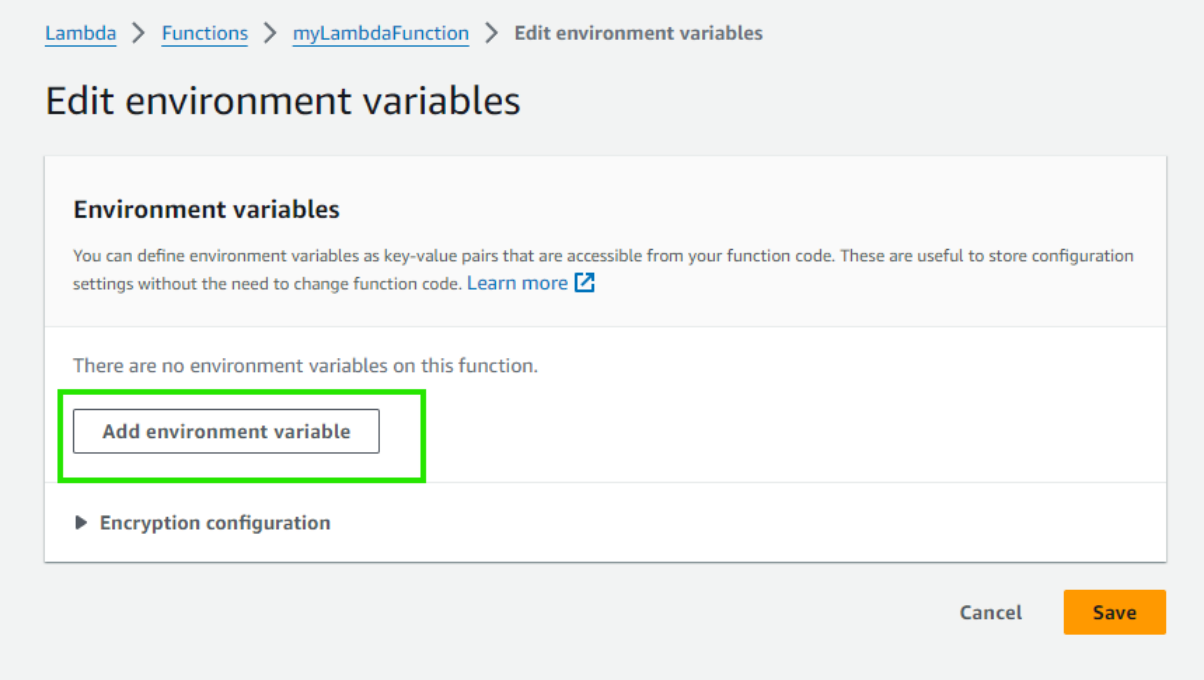
1. Navigate to the **"Configuration"** tab of the Lambda function.



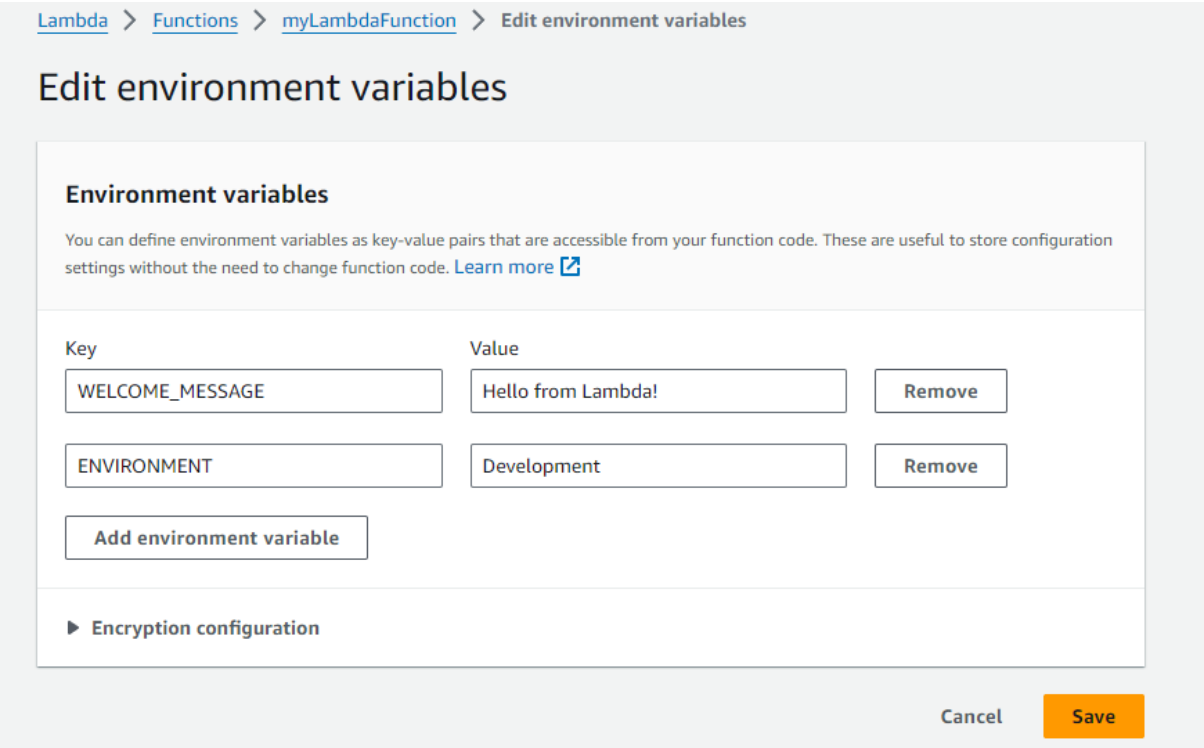
2. Under **"Environment Variables,"** click **Edit.**



3. Click the **Add environment variable.**



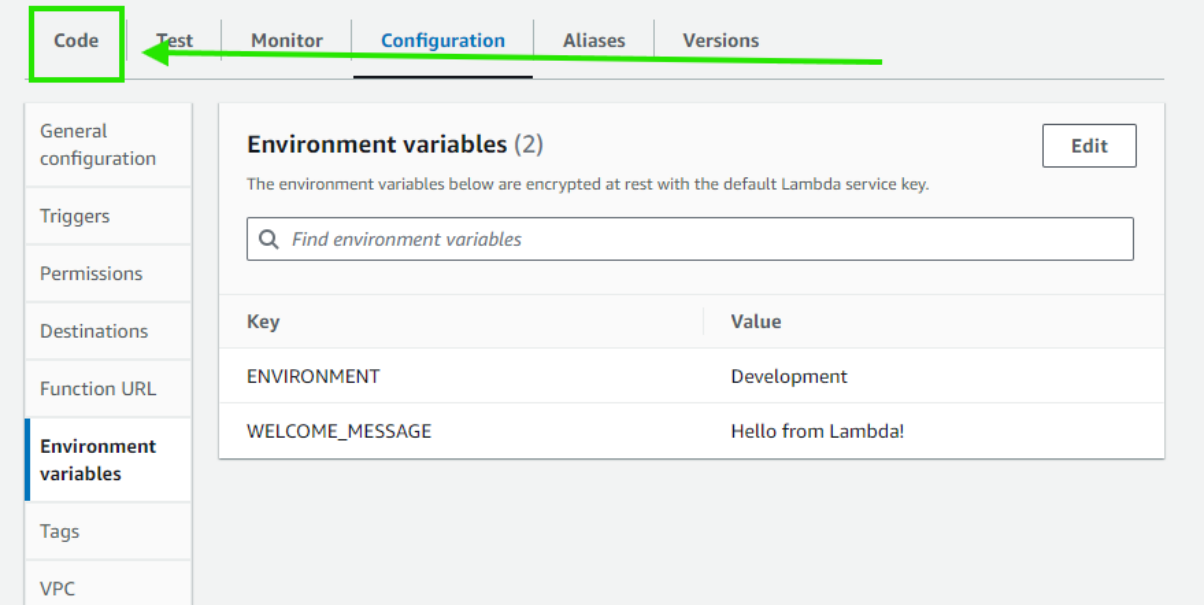
- Add the following new environment variable:
    - WELCOME_MESSAGE: **Hello from Lambda!**
    - ENVIRONMENT: **Development**



- Save the changes.

4. Modify the Lambda Code to Use Environment Variables:

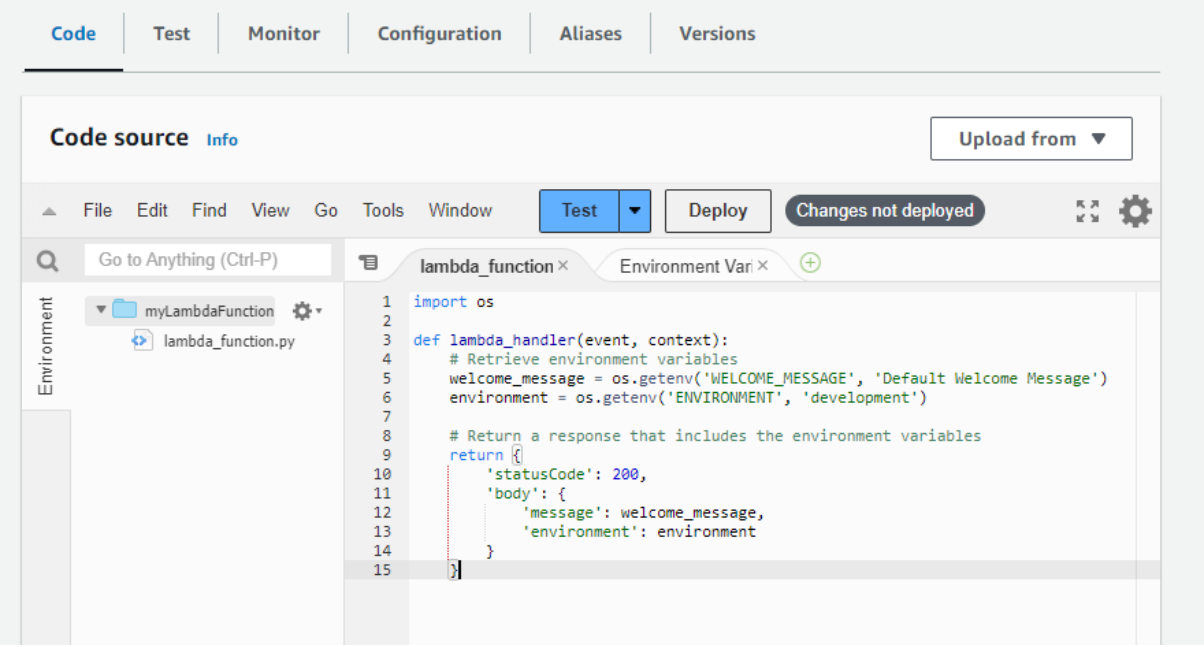
- Navigate back to the **“Code” Tab**



- Copy and Paste the following code to the code editor

***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.*

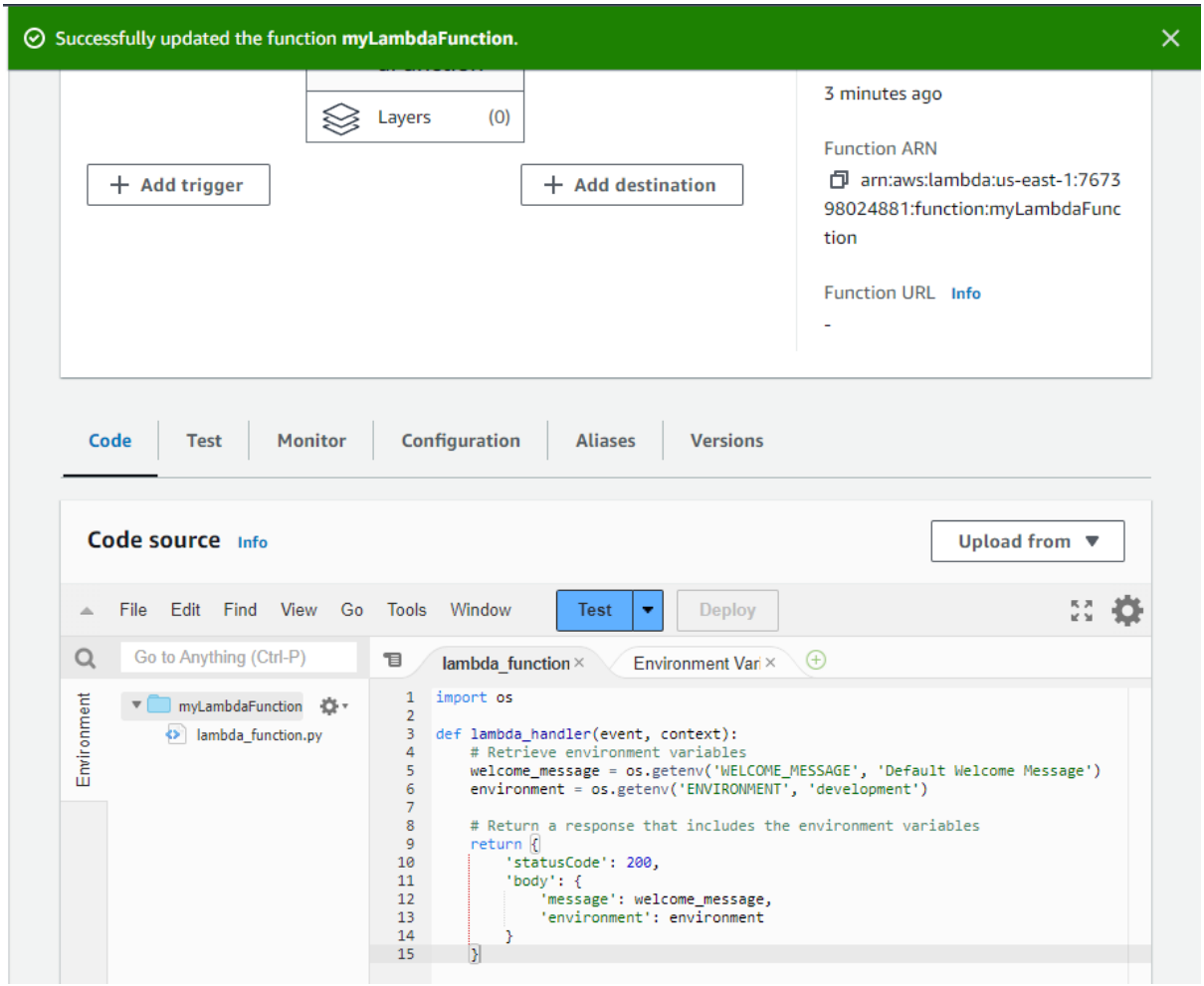
```
import os


def lambda_handler(event, context):
    # Retrieve environment variables
    welcome_message = os.getenv('WELCOME_MESSAGE', 'Default Welcome Message')
    environment = os.getenv('ENVIRONMENT', 'development')

    # Return a response that includes the environment variables
    return {
        'statusCode': 200,
        'body': {
            'message': welcome_message,
            'environment': environment
        }
    }
```



This function fetches the values of WELCOME_MESSAGE and ENVIRONMENT from the environment variables using the os module. The os.getenv() function allows the Lambda function to retrieve these environment variables securely, enhancing maintainability and security, especially for sensitive data like API tokens or passwords.

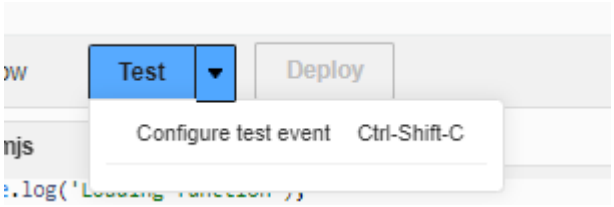
---

5. Click on **Deploy** to save changes.



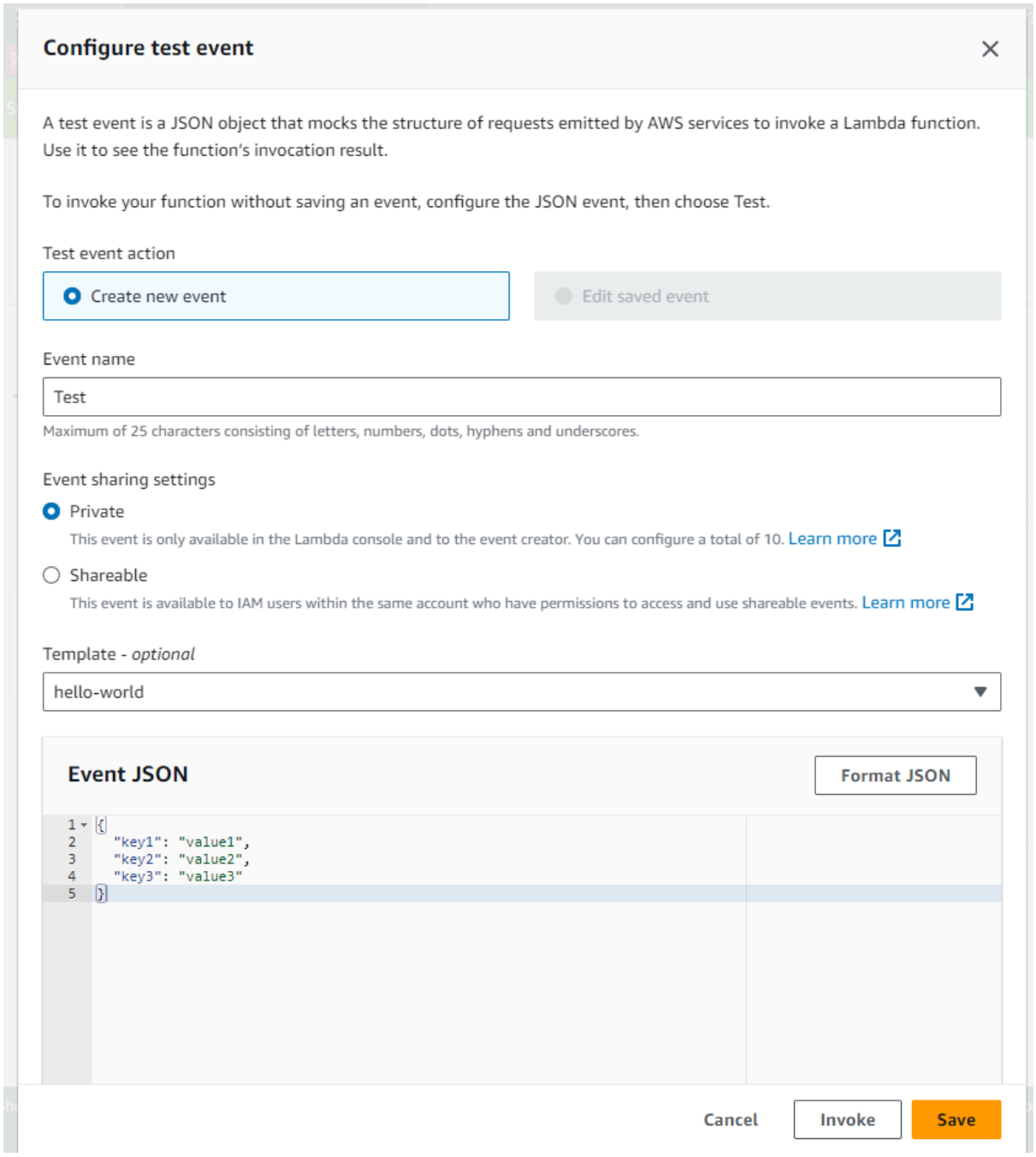
Test the Lambda Function

1. Once your function is deployed. Click the arrow dropdown of the **Test button**



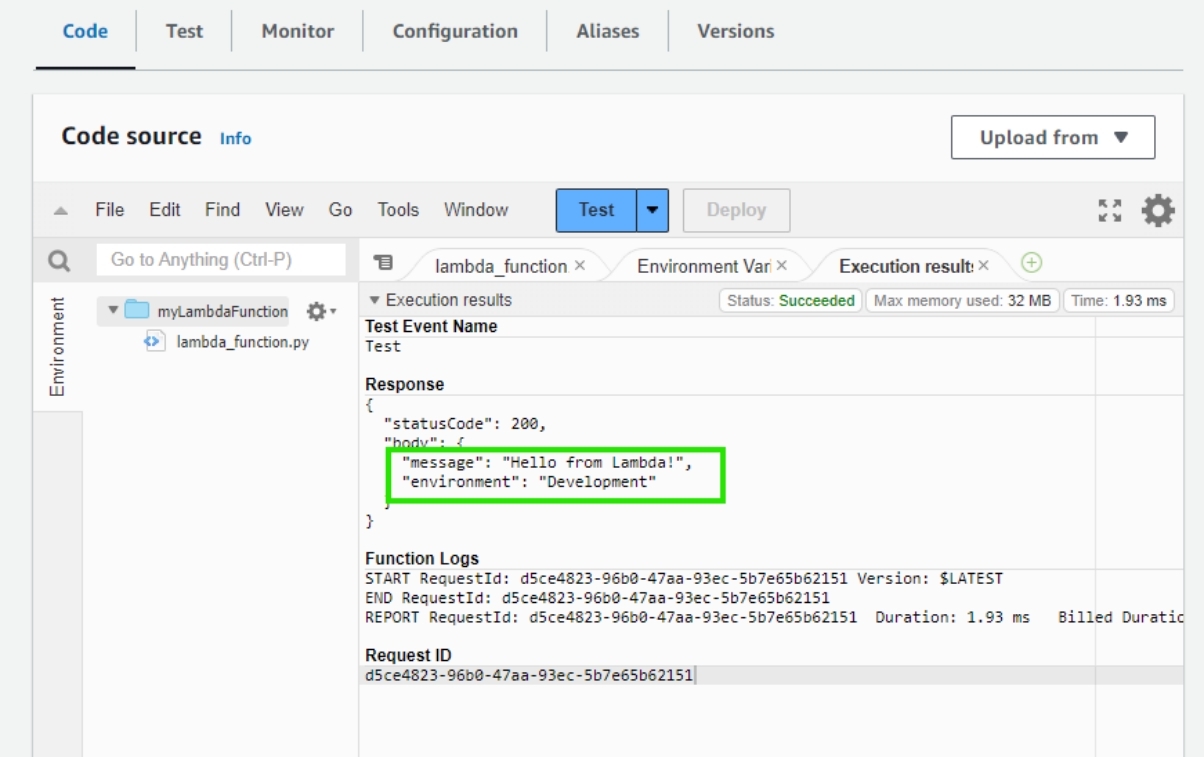
2. Click on **Configure test event,** and follow the configuration below:

- Event name: Test
- Template- optional: hello-world
    - Leave the rest as default



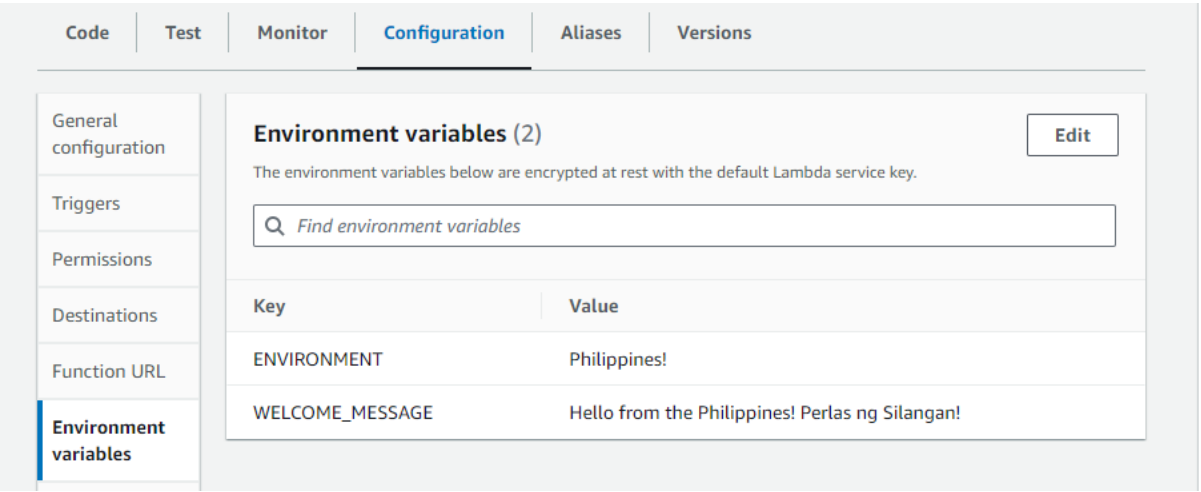
- Click on Save

3. Now, click on **Test.** Check the output to see if it returns the message and environment value defined by the environment variables.

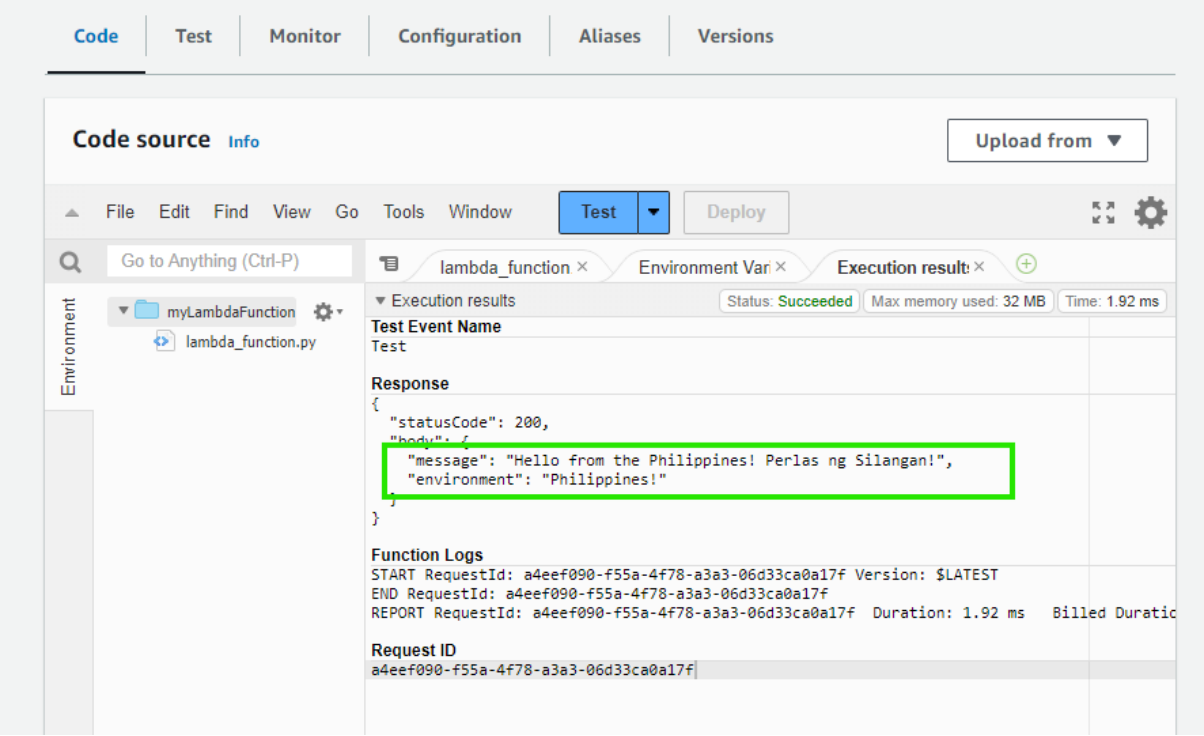


4. Update Environment Variables:

- Go back to the **"Configuration"** tab.
- Update:
    - WELCOME_MESSAGE: **Hello from the Philippines! Perlas ng Silangan!**
    - ENVIRONMENT: **Philippines!**
- Save the changes.



- Retest the function to confirm it reflects the updated environment variable values.



That's it! Congratulations! You have learned how to use environment variables in Python-based AWS Lambda functions. This approach allows for dynamic configuration, making your code more maintainable and flexible without requiring changes to the function's code whenever a configuration setting needs to be updated.

Environment variables are instrumental in real-world scenarios, such as securely managing sensitive information like API tokens, database connection strings, or passwords. By storing these values as environment variables, you avoid hardcoding them directly into your code, which enhances security.

Incorporating environment variables in your Lambda functions is a best practice for managing configuration and sensitive data, ensuring your functions are secure and easy to maintain. Happy Learning!