

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

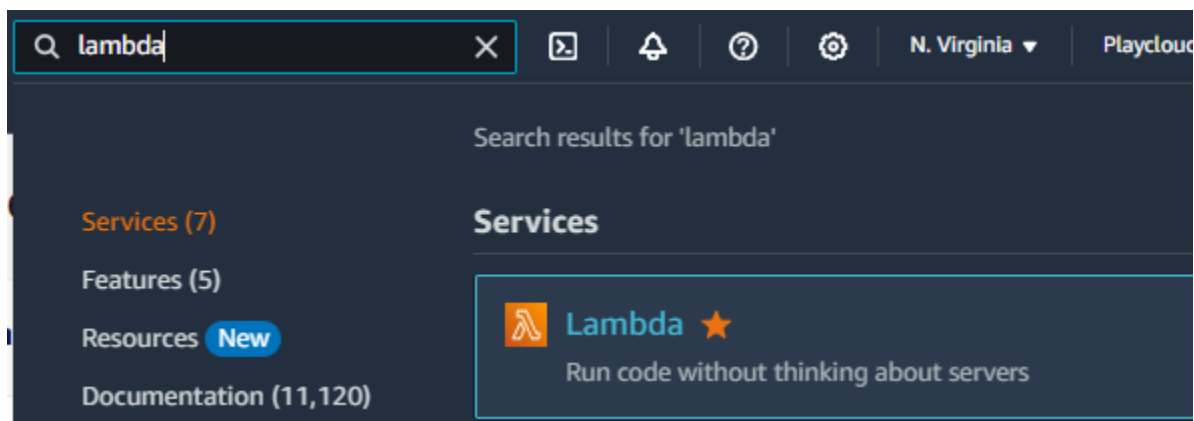
In this lab, you will:

- 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-Sandbox

Basic information

Function name
Enter a name that describes the purpose of your function.

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.

▼

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

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.

▼

[View the PlayCloud-Sandbox role](#) on the IAM console.

- Click **Create function**

Adding Environment Variables

1. Navigate to the **“Configuration”** tab of the Lambda function.

Lambda > Functions > myLambdaFunction

myLambdaFunction

Throttle Copy ARN Actions

Function overview Info

Export to Application Composer Download

Diagram Template

myLambdaFunction

Layers (0)

+ Add trigger + Add destination

Description

Last modified 1 minute ago

Function ARN
arn:aws:lambda:us-east-1:767398024881:function:myLambdaFunction

Function URL Info

Code Test Monitor Configuration Aliases Versions

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

```
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     return {}
```

2. Under “Environment Variables,” click **Edit**.

Code Test Monitor Configuration Aliases Versions

General configuration

Triggers

Permissions

Destinations

Function URL

Environment variables

Tags

Environment variables

Find environment variables


Key	Value
No environment variables	
No environment variables associated with this function.	
Edit	

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

[Lambda](#) > [Functions](#) > [myLambdaFunction](#) > Edit environment variables

Edit environment variables

Environment variables

You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. [Learn more](#) 

There are no environment variables on this function.

Add environment variable

► Encryption configuration

Cancel


Save

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

[Lambda](#) > [Functions](#) > [myLambdaFunction](#) > Edit environment variables

Edit environment variables

Environment variables

You can define environment variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. [Learn more](#) 

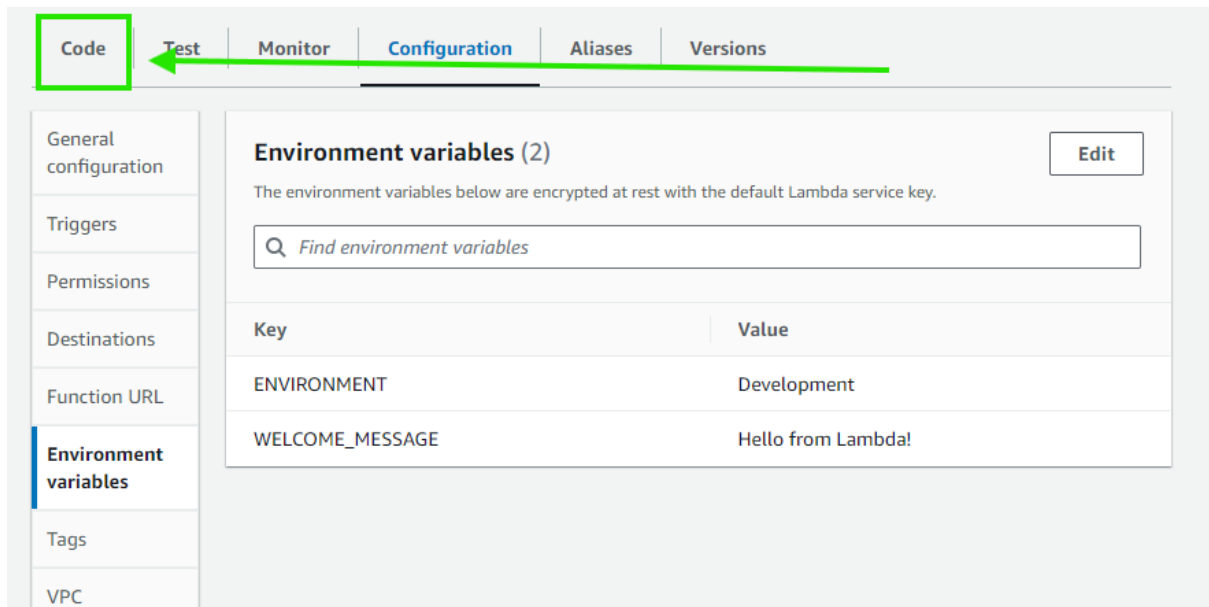
Key	Value	
WELCOME_MESSAGE	Hello from Lambda!	Remove
ENVIRONMENT	Development	Remove
Add environment variable		

► Encryption configuration

Cancel

Save

- 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,
```

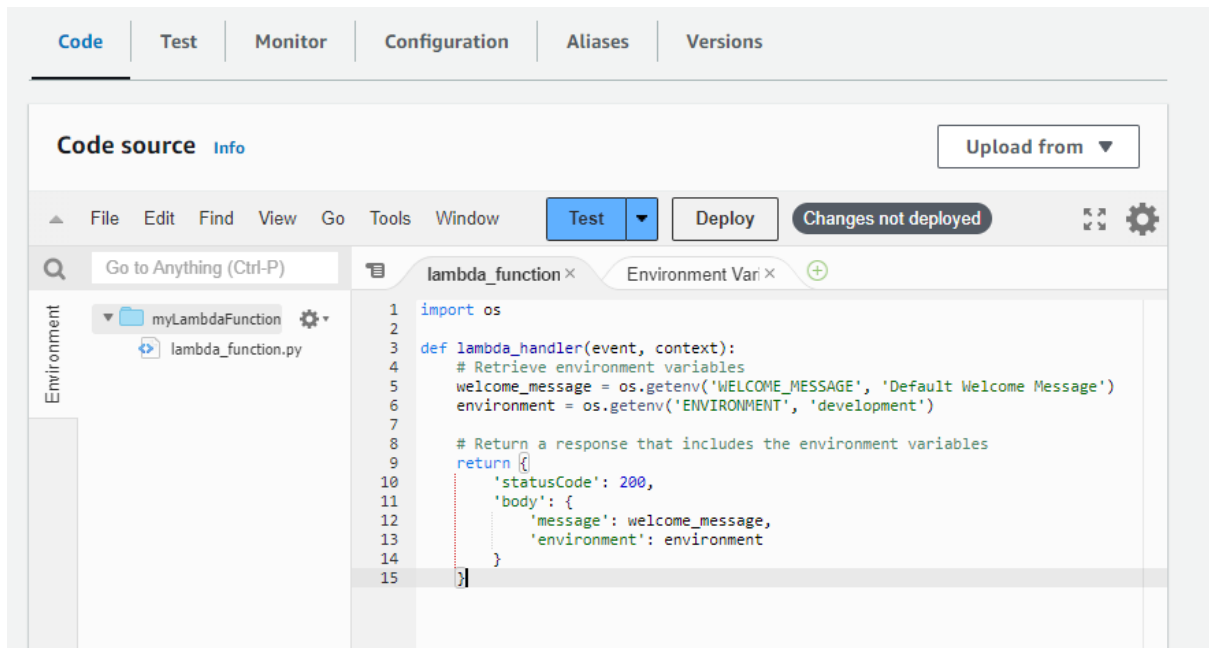
```
        'body': {
```

```
            'message': welcome_message,
```

```
            'environment': environment
```

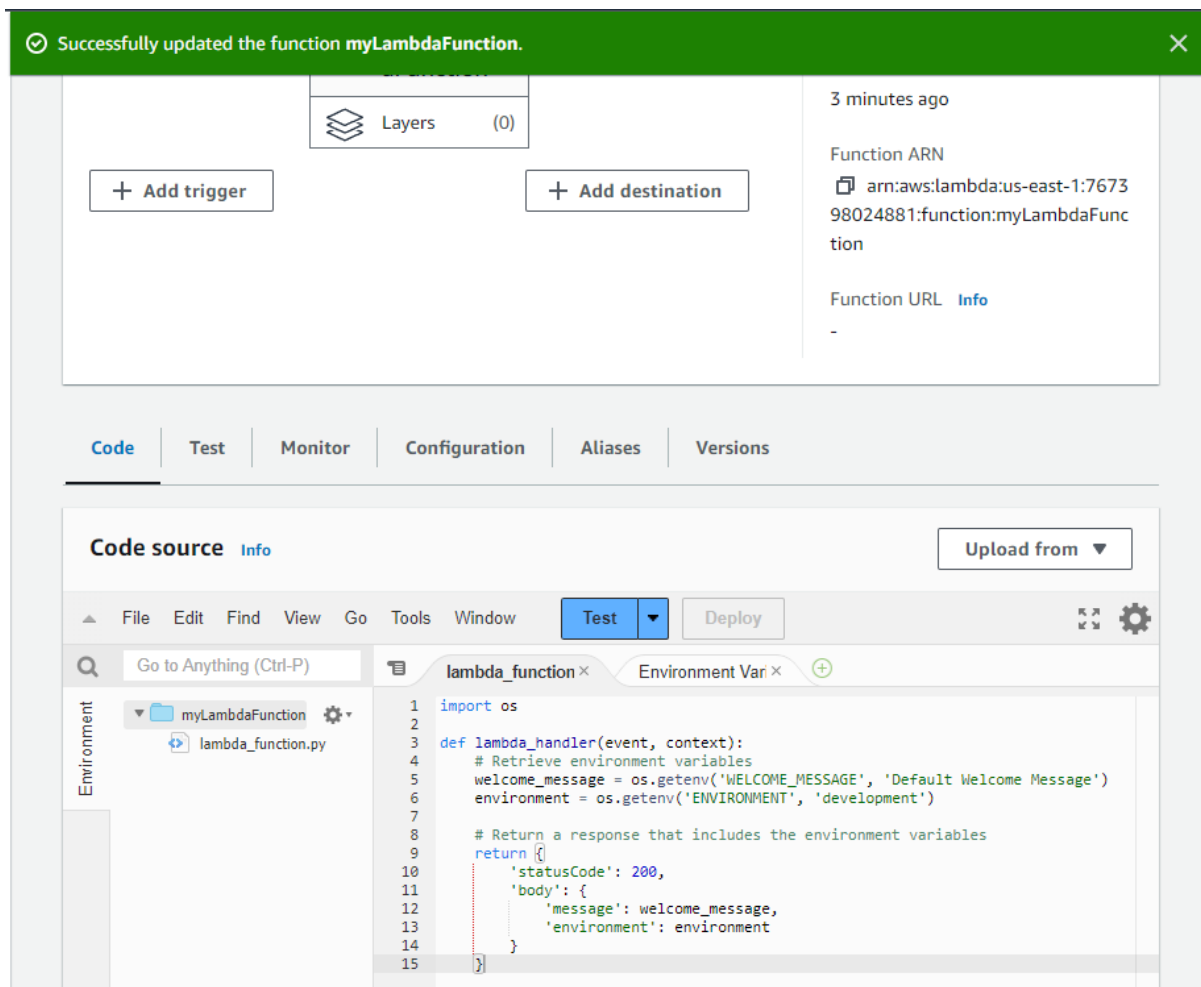
```
        }
```

```
    }
```



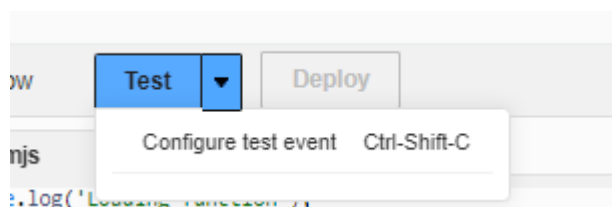
This function fetches the values of `WELCOME_MESSAGE` and `ENVIRONMENT` from the environment variables using the `os` module. The `os.getenv()` function allows the Lambda function to retrieve these environment variables securely, enhancing maintainability and security, especially for sensitive data like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

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

Test

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

Format JSON

1 {

2 "key1": "value1",

3 "key2": "value2",

4 "key3": "value3"

5 }

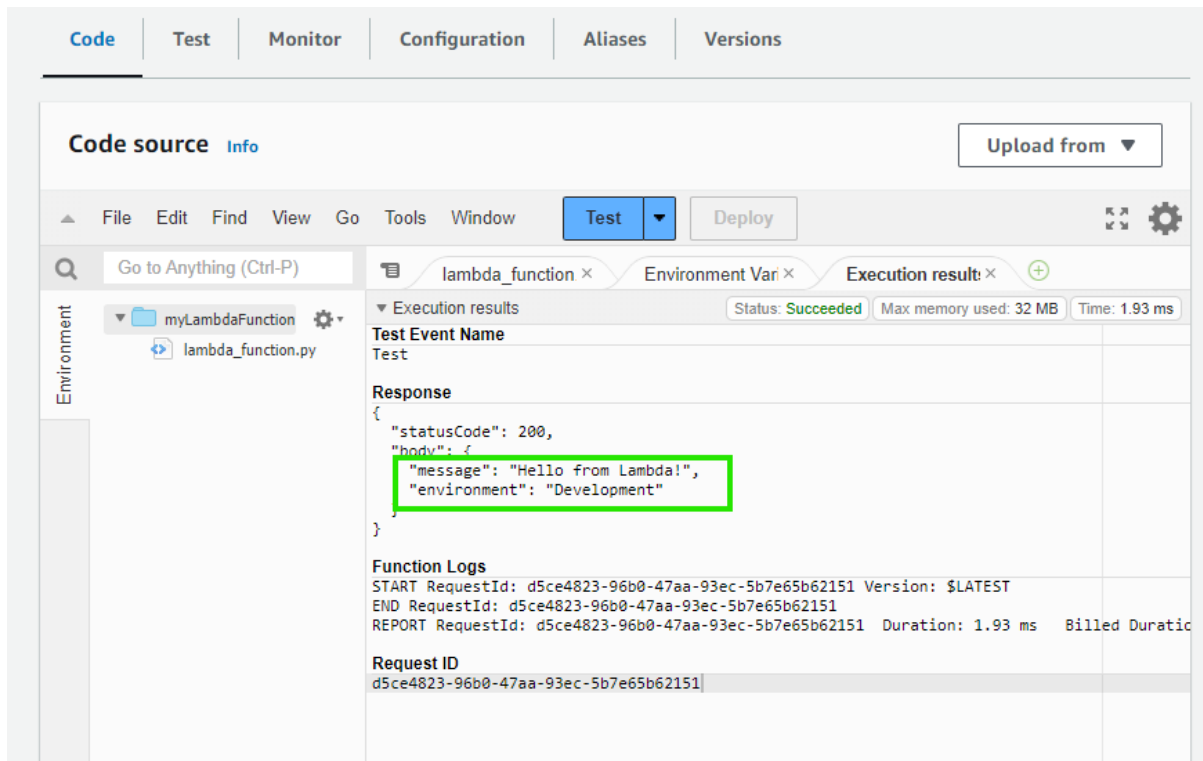
Cancel

Invoke

Save

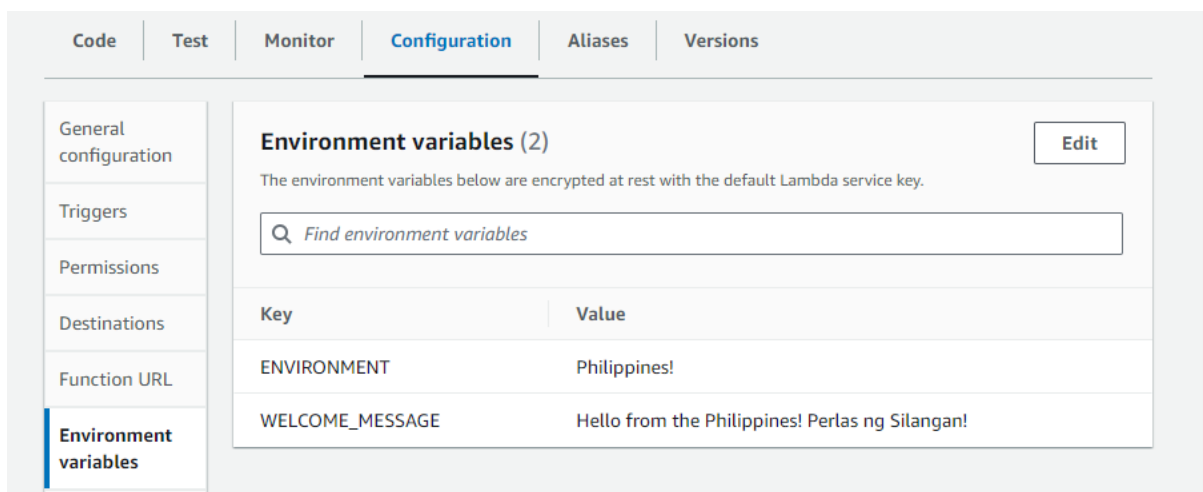
- Click on Save

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

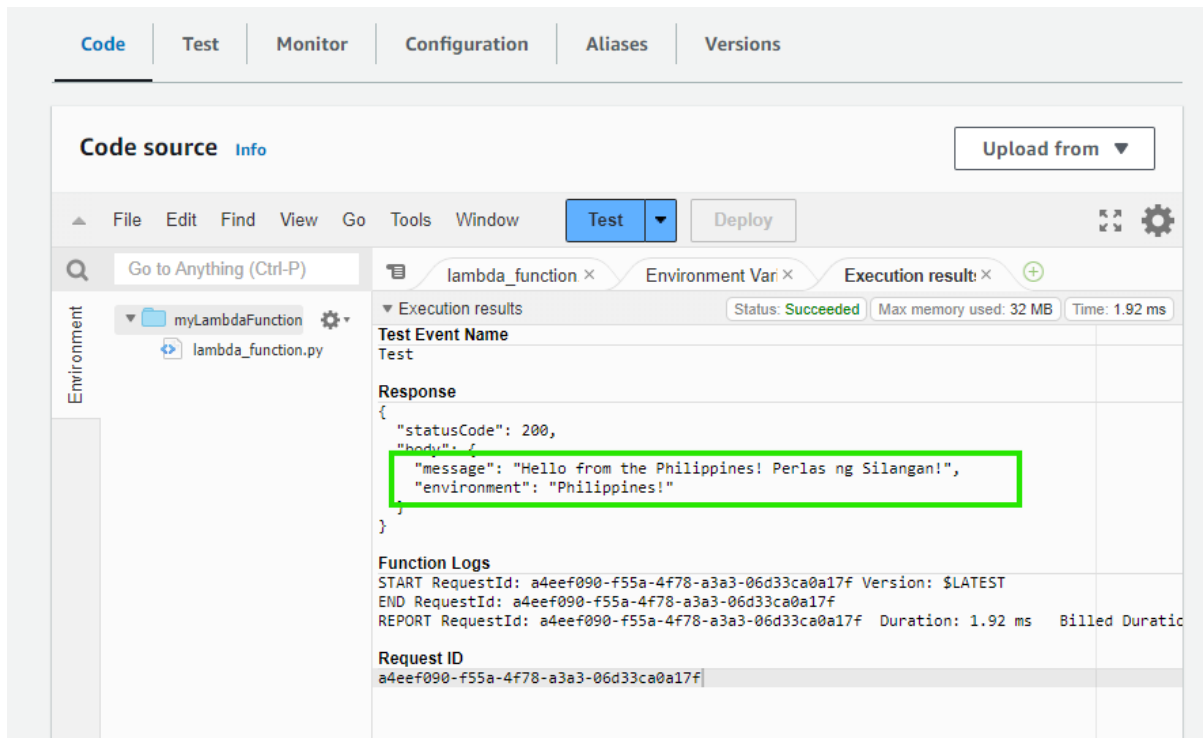


4. Update Environment Variables:

- 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!