This lab demonstrates the steps from *Demo: Lambda in the AWS Console.*
My full AWS Architect Associate course can be found here:
https://www.udemy.com/course/ultimateaws/?referralCode=7ED214B795C444141361

Lab Guide: Creating and Testing Your First AWS Lambda Function

This lab guide will walk you through the process of creating, configuring, and testing your first AWS Lambda function. It explains the key concepts behind AWS Lambda and provides a hands-on demonstration to help you understand how Lambda operates and integrates with other AWS services.

Lab Objectives

1. Understand the basics of the AWS Lambda service.
2. Create a Lambda function from scratch using the AWS Management Console.
3. Modify and deploy code for the Lambda function.
4. Test the function using a custom test event.
5. Optimize function performance by configuring memory and analyzing execution metrics.

Step 1: Accessing the AWS Lambda Console

1. **Log into the AWS Console** and ensure you're in a specific region (e.g., **Ohio (us-east-2)**).
    - **Note**: AWS Lambda is regional, meaning Lambda functions are tied to the region in which they are created.
2. Use the search bar at the top of the AWS Console and search for **Lambda**.
3. Click on **Lambda** to open the Lambda Dashboard.

Step 2: Creating Your First Lambda Function

2.1 Start the Function Creation Process

1. On the Lambda Dashboard, click **Create Function**.
2. Choose **Author from scratch**.

2.2 Configure the Function

1. **Function Name**: Enter a name for your function (e.g., `MyFirstLambda`).
2. **Runtime**: Select the programming language for your function. For this lab, choose **Python 3.9**.
3. Leave the **Permissions** as default to create a new execution role.
4. Click **Create Function**.

Step 3: Writing and Deploying Code

3.1 Edit the Default Code

1. In the Lambda function editor, you'll see some default code preloaded by AWS.
2. Replace the default code with the following simple script:

```
def lambda_handler(event, context):
    print("Hello from Lambda!")
    return {
        'statusCode': 200,
        'body': 'Lambda executed successfully!'
    }
```

3.2 Deploy the Code

1. After editing the code, click the **Deploy** button.
    - Deploying saves the updated code to the Lambda function, making it ready for execution.

Step 4: Testing the Lambda Function

4.1 Configure a Test Event

1. Click the **Test** button at the top of the function editor.
2. Create a new test event:
    - **Event Name**: Enter `MyTestEvent`.
    - **Template**: Select **CloudWatch Logs** or leave the default JSON template.
3. Click **Save**.

4.2 Run the Test

1. Click the **Test** button again to execute the function.

2. Review the output:
    - Expand the **Execution Results** section to see the function's response and logs.
    - The print statement (`"Hello from Lambda!"`) should appear in the logs.

Step 5: Analyzing Function Performance

5.1 View Execution Metrics

1. After testing, review the **Details** section in the execution results:
    - **Duration**: The time taken to execute the function.
    - **Billed Duration**: The time rounded up to the nearest millisecond for billing.
    - **Max Memory Used**: The amount of memory consumed during execution.

5.2 Adjust Memory Allocation

1. Scroll to the **Configuration** tab and select **General Configuration**.
2. Increase the memory from the default **128 MB** to **512 MB**.
3. Save the changes.

5.3 Retest the Function

1. Click **Test** again to execute the function with the updated memory allocation.
2. Compare the new execution metrics to the previous test:
    - Check whether the increased memory improved execution time or had no significant effect.

Key Concepts

AWS Lambda

- **Serverless Computing**: Lambda allows you to run code without provisioning or managing servers.
- **Event-Driven**: Lambda functions are triggered by events from AWS services like S3, DynamoDB, and CloudWatch.
- **Pay-per-Use**: Billing is based on the number of requests and execution time (rounded to the nearest millisecond).

Test Events

- **Purpose**: Simulate triggers for Lambda functions during development.
- **Templates**: AWS provides sample JSON templates for common triggers (e.g., S3, CloudWatch).

Performance Tuning

- **Memory Allocation**: Adjusting memory can impact performance and cost.
- **Metrics**:
    - **Duration**: How long the function takes to run.
    - **Max Memory Used**: Indicates whether the allocated memory is sufficient for the workload.

Step 6: Clean Up

1. To avoid incurring charges:
    - Delete the Lambda function:
        - Go to the **Functions** page, select your function, and click **Delete**.
    - Delete any associated resources (e.g., IAM roles).

Summary

In this lab, you created and tested your first AWS Lambda function, gaining insights into its configuration, execution, and performance. You also learned how to analyze metrics and optimize memory settings to improve efficiency. This hands-on experience provides a foundation for understanding serverless computing in AWS.