**AWS Lambda Unit Testing with Python and Node.js**

**Objective**

The objective of this document is to demonstrate how to perform unit testing of AWS Lambda functions using **Python**and **Node.js**. We will use:

* **Python**: unittest with moto to mock AWS services.
* **Node.js**: jest with aws-sdk-mock for mocking AWS services.

This approach allows testing Lambda functions locally without making actual calls to AWS, making the testing process faster, cost-effective, and isolated.

**Overview**

**AWS Lambda**

AWS Lambda is a serverless computing service that allows you to run code in response to events.

**Python Tools**

* **moto**: A Python library that mocks AWS services to allow local testing of AWS-related code without actual AWS calls.
* **unittest**: The built-in testing framework in Python used for writing test cases in an object-oriented style.

**Node.js Tools**

* **jest**: A JavaScript testing framework for unit and integration testing.
* **aws-sdk-mock**: A library that helps mock AWS SDK calls for testing.

**Python Lambda Function and Unit Test**

**Installation**

Ensure the required libraries are installed by running:

pip install boto3 moto unittest

**1. Create the AWS Lambda Function (Python)**

The Lambda function will upload a file to an S3 bucket.

import boto3

def lambda\_handler(event, context):

bucket\_name = event['bucket\_name']

file\_name = event['file\_name']

file\_content = event['file\_content']

s3 = boto3.client('s3')

s3.put\_object(Bucket=bucket\_name, Key=file\_name, Body=file\_content)

return {

'statusCode': 200,

'body': f'File {file\_name} uploaded successfully to {bucket\_name}'

}

**2. Create the Unit Test (Python)**

Using unittest and moto to mock AWS S3.

import unittest

import boto3

from moto import mock\_s3

from lambda\_function import lambda\_handler

class TestLambdaFunction(unittest.TestCase):

@mock\_s3

def setUp(self):

self.s3 = boto3.client('s3', region\_name='us-east-1')

self.s3.create\_bucket(Bucket='test-bucket')

@mock\_s3

def test\_lambda\_handler(self):

event = {

'bucket\_name': 'test-bucket',

'file\_name': 'testfile.txt',

'file\_content': 'This is a test file.'

}

result = lambda\_handler(event, None)

self.assertEqual(result['statusCode'], 200)

self.assertIn('File testfile.txt uploaded successfully', result['body'])

response = self.s3.get\_object(Bucket='test-bucket', Key='testfile.txt')

self.assertEqual(response['Body'].read().decode('utf-8'), 'This is a test file.')

@mock\_s3

def tearDown(self):

bucket\_name = 'test-bucket'

for obj in self.s3.list\_objects\_v2(Bucket=bucket\_name).get('Contents', []):

self.s3.delete\_object(Bucket=bucket\_name, Key=obj['Key'])

self.s3.delete\_bucket(Bucket=bucket\_name)

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()

**3. Running the Unit Test (Python)**

Run the test using:

python test\_lambda\_function.py

**Node.js Lambda Function and Unit Test**

**Installation**

Install the required dependencies:

npm install --save-dev jest aws-sdk-mock

**1. Create the AWS Lambda Function (Node.js)**

const AWS = require('aws-sdk');

exports.lambdaHandler = async (event) => {

const s3 = new AWS.S3();

const { bucket\_name, file\_name, file\_content } = event;

await s3.putObject({

Bucket: bucket\_name,

Key: file\_name,

Body: file\_content

}).promise();

return {

statusCode: 200,

body: `File ${file\_name} uploaded successfully to ${bucket\_name}`

};

};

**2. Create the Unit Test (Node.js)**

Using jest and aws-sdk-mock to mock AWS S3.

const AWSMock = require('aws-sdk-mock');

const AWS = require('aws-sdk');

const { lambdaHandler } = require('./lambda\_function');

describe('Lambda Function', () => {

beforeAll(() => {

AWSMock.mock('S3', 'putObject', (params, callback) => {

callback(null, { ETag: 'mocked-etag' });

});

});

afterAll(() => {

AWSMock.restore('S3');

});

test('should upload file to S3', async () => {

const event = {

bucket\_name: 'test-bucket',

file\_name: 'testfile.txt',

file\_content: 'This is a test file.'

};

const result = await lambdaHandler(event);

expect(result.statusCode).toBe(200);

expect(result.body).toContain('File testfile.txt uploaded successfully');

});

});

**3. Running the Unit Test (Node.js)**

Run the test using:

npx jest

**4. Teardown (Node.js)**

Ensure that resources are cleaned up after tests:

afterAll(() => {

AWSMock.restore('S3');

});

**Conclusion**

In this guide, we demonstrated how to write and execute unit tests for AWS Lambda functions in both Python and Node.js. By using moto (Python) and aws-sdk-mock (Node.js), we successfully mocked AWS services, making the testing process fast, isolated, and cost-effective. Additionally, we included teardown steps to clean up any infrastructure created during testing, ensuring a proper testing environment. This approach ensures Lambda functions are thoroughly tested before deployment to AWS.