Getting Started with Serverless Compute on AWS

Hands-on with AWS Lambda

November 2020

Table of Contents

Overview

With services like AWS Lambda and Amazon API Gateway, developers can very easily develop, debug, and deploy serverless applications in the cloud.

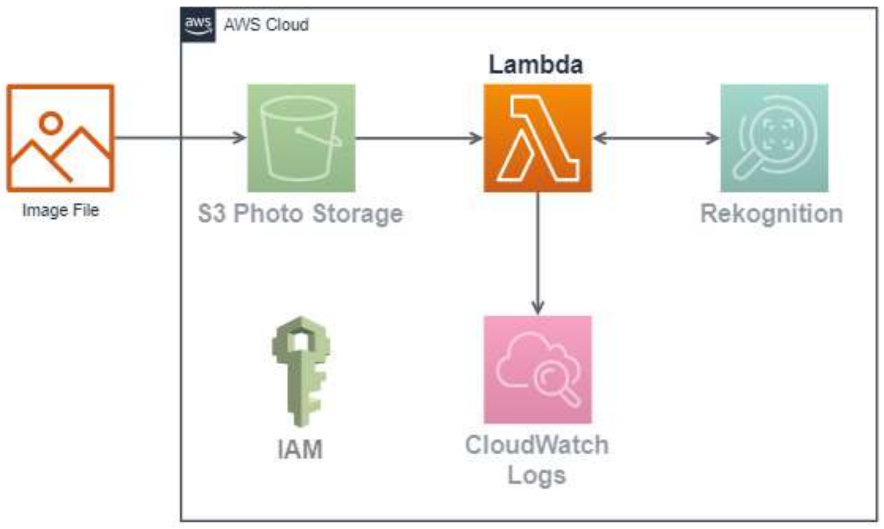
AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume - there is no charge when your code is not running. With Lambda, you can run code for virtually any type of application or backend service - all with zero administration. Just upload your code and Lambda takes care of everything required to run and scale your code with high availability.

In this Lab, you will experience:

- Developing a simple AWS Lambda function
- Configuring S3 bucket as Lambda trigger event source
- Testing your AWS Lambda function
- Triggering Lambda function by uploading image to S3

Introduction

In this hands-on lab, we are going to create a Lambda function using the Python programming language. This function will be linked to an S3 buck so that any time a new object arrives in S3, the lambda function will be called to evaluate the object using Rekognition image analysis. The high level architecture is as follows:



Login to AWS

1. Sign into the AWS Management Console using Event Engine

Event Hash: **a091-15aec7b154-53**

Note: This account has been created specifically for this immersion day event, and all resources created will be deleted after the immersion day event is over

Create S3 bucket for images

2. Use the AWS console to navigate to S3 service

3. Click the

button

4. Choose any name you would like such as **image-repo-813449418529**

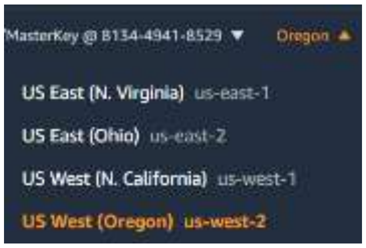
   a. S3 bucket names must be unique so you can appenx account number as suffix

5. Accept defaults including **US West (Oregon) us-west-2** region and press

Create your Lambda function

6. Use the AWS console to navigate to Lambda service

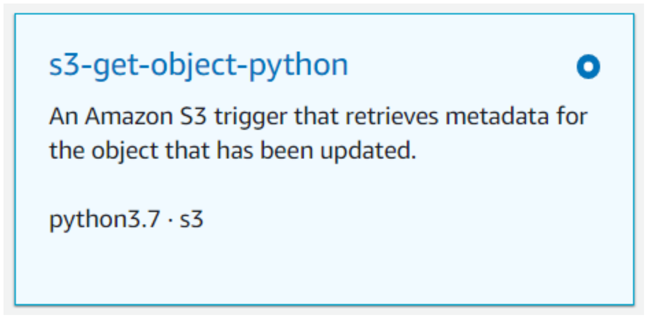
7. Double-check that you are in **Oregon** region as your bucket using region selector at top-right of the screen


MasterKey @ 8134-4941-8529
Oregon
US East (N. Virginia) us-east-1
US East (Ohio) us-east-2
US West (N. California) us-west-1
US West (Oregon) us-west-2

8. Click the

button

9. Use a blueprint to give a head start on code creation


s3-get-object-python

An Amazon S3 trigger that retrieves metadata for the object that has been updated.

python3.7 · s3

10. Move to next step by clicking

11. Name your function **computer-vision-example**
    a. Lambda function names only need to be unique within your account
12. Create a new role called **LambdaExecutionRole** from AWS policy templates
13. Select your S3 bucket as the trigger
14. Accept remaining defaults and click

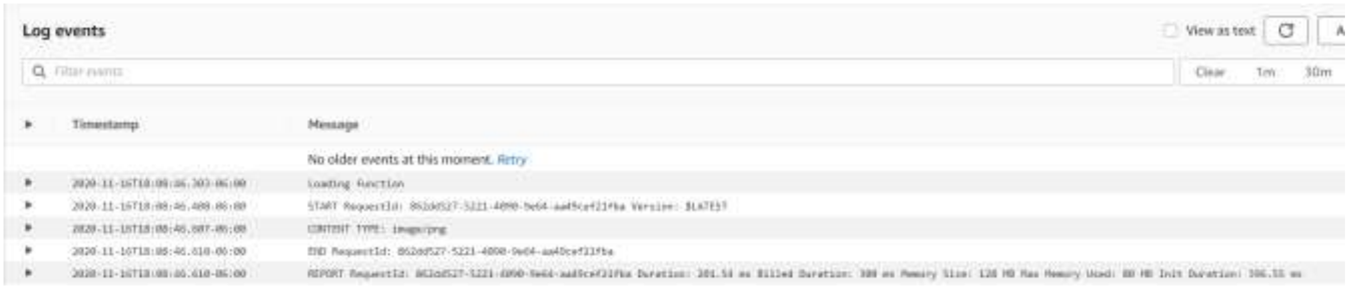
15. After function is created, click on the *Latest* link in the Aliases table to work on latest version of the function code
16. Click on the **Code** tab to view the sample code provided
17. Click on the **Latest Configuration** tab and enable the S3 trigger

Triggering your function from S3

18. After your function is created, you can upload an image to S3 to trigger execution. Navigate back to S3 and upload any image file to the bucket.
19. Go back to the Lambda function, navigate to the Monitor tab and click the

button to see the logs for execution of the function.

20. Review the most recent log entries to validate the function did not have any errors

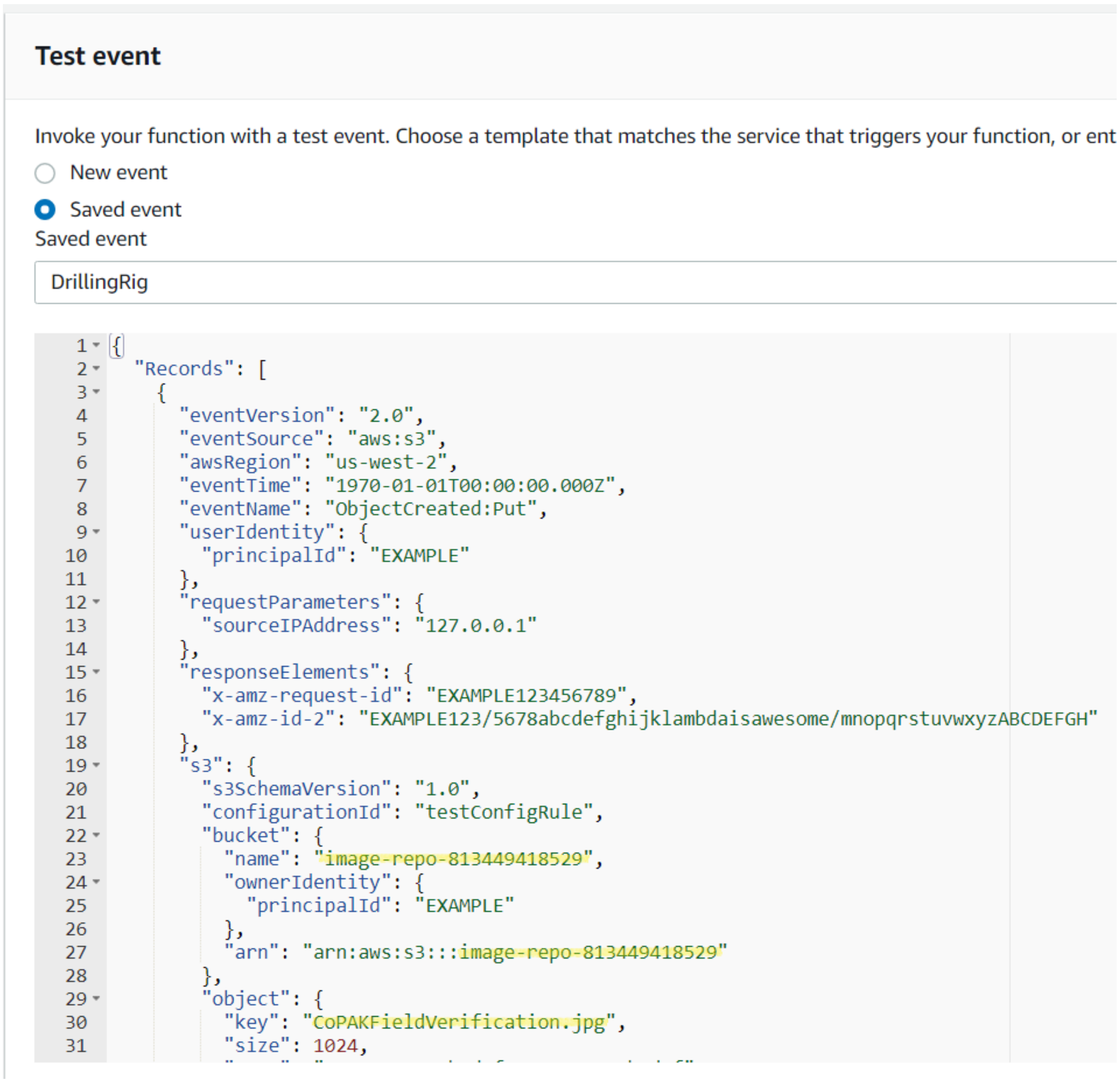


```
Timestamp                        Message
                                 No older events at this moment. Retry
2020-11-16T18:08:46.303-06:00    Loading function
2020-11-16T18:08:46.408-06:00    START RequestId: 862dd527-5221-4090-9e64-aa49cef21fba Version: $LATEST
2020-11-16T18:08:46.607-06:00    CONTENT TYPE: image/png
2020-11-16T18:08:46.610-06:00    END RequestId: 862dd527-5221-4090-9e64-aa49cef21fba
2020-11-16T18:08:46.610-06:00    REPORT RequestId: 862dd527-5221-4090-9e64-aa49cef21fba Duration: 201.54 ms Billed Duration: 300 ms Memory Size: 128 MB Max Memory Used: 80 MB Init Duration: 306.55 ms
```

Testing and Monitoring your function

21. Go to the **Test** tab of your Lambda function. We will setup a trigger that can be simulated instead of having to place a file in S3 every time we want the function to execute.

22. Change template to **Saved Event** and start with the S3-Put example. replace the bucket name and file to match the bucket and uploaded file in your environment. Your test data should look something like this:



Test event

Invoke your function with a test event. Choose a template that matches the service that triggers your function, or ent

New event

[✓ Saved event]

Saved event

DrillingRig

```
{
  "Records": [
    {
      "eventVersion": "2.0",
      "eventSource": "aws:s3",
      "awsRegion": "us-west-2",
      "eventTime": "1970-01-01T00:00:00.000Z",
      "eventName": "ObjectCreated:Put",
      "userIdentity": {
        "principalId": "EXAMPLE"
      },
      "requestParameters": {
        "sourceIPAddress": "127.0.0.1"
      },
      "responseElements": {
        "x-amz-request-id": "EXAMPLE123456789",
        "x-amz-id-2": "EXAMPLE123/5678abcdefghijklambdaisawesome/mnopqrstuvwxyzABCDEFGH"
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "testConfigRule",
        "bucket": {
          "name": "image-repo-813449418529",
          "ownerIdentity": {
            "principalId": "EXAMPLE"
          },
          "arn": "arn:aws:s3:::image-repo-813449418529"
        },
        "object": {
          "key": "CoPAKFieldVerification.jpg",
          "size": 1024,
```

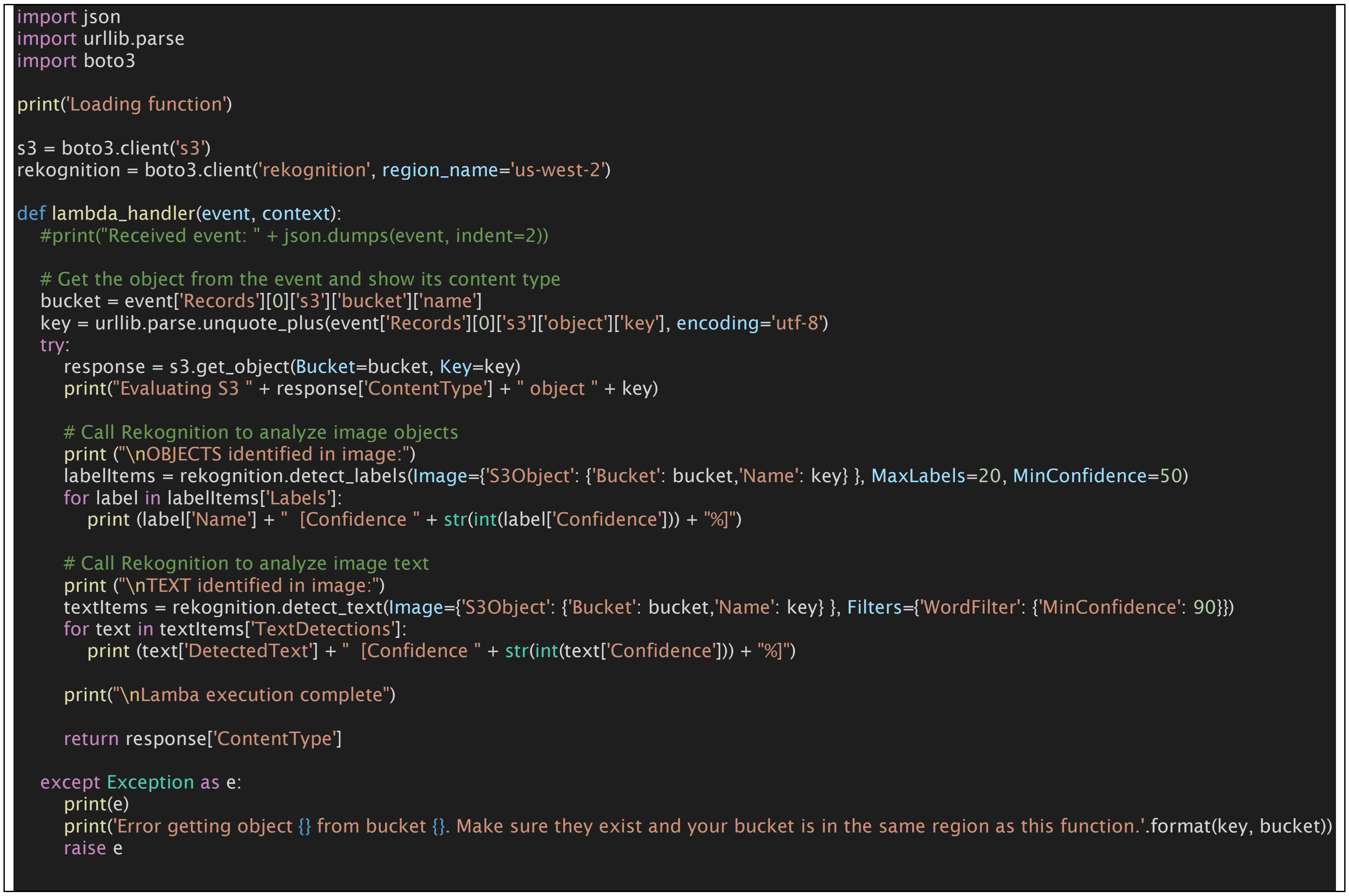
Once your test event is configured, you can save, deploy & test from the Code tab using this test event data.

Adding Functionality

Now that we have a Lambda function and a triggering mechanism, let's make the compute activites a bit more interesting. We are going to add a few lines of code (and necessary permissions) for Lambda to call the Amazon Rekognition service.

23. Go to the Lambda function configuration and find the IAM role used by the function
24. Modify this IAM role to include the AmazonRekognitionFullAccess policy
25. Modify the Lambda code to call Rekognition to identify objects and text in the image. This step is just to get familiar with making changes to Lambda functions and deploying/testing them. Don't worry too much about what the code actually does (or doesn't), but focus on getting a feel for using Lambda

You completed code should look something like this:



```python
import json
import urllib.parse
import boto3

print('Loading function')

s3 = boto3.client('s3')
rekognition = boto3.client('rekognition', region_name='us-west-2')

def lambda_handler(event, context):
    #print("Received event: " + json.dumps(event, indent=2))

    # Get the object from the event and show its content type
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'], encoding='utf-8')
    try:
        response = s3.get_object(Bucket=bucket, Key=key)
        print("Evaluating S3 " + response['ContentType'] + " object " + key)

        # Call Rekognition to analyze image objects
        print ("\nOBJECTS identified in image:")
        labelItems = rekognition.detect_labels(Image={'S3Object': {'Bucket': bucket,'Name': key} }, MaxLabels=20, MinConfidence=50)
        for label in labelItems['Labels']:
            print (label['Name'] + "  [Confidence " + str(int(label['Confidence'])) + "%]")

        # Call Rekognition to analyze image text
        print ("\nTEXT identified in image:")
        textItems = rekognition.detect_text(Image={'S3Object': {'Bucket': bucket,'Name': key} }, Filters={'WordFilter': {'MinConfidence': 90}})
        for text in textItems['TextDetections']:
            print (text['DetectedText'] + "  [Confidence " + str(int(text['Confidence'])) + "%]")

        print("\nLamba execution complete")

        return response['ContentType']

    except Exception as e:
        print(e)
        print('Error getting object {} from bucket {}. Make sure they exist and your bucket is in the same region as this function.'.format(key, bucket))
        raise e
```

Clean up

After you are done with your lab, feel free to practice deleting all the resources we created during this lab. This is not a requirement as the account will automatically deleted after 3 days.

Conclusion

In this lab you have learned about serverless compute using AWS Lambda. For additional information or questions, reach out to your account Solutions Architect. Happy building!