Big Data on AWS

Lab Eight: Using AWS Lambda with Amazon S3

Table of Contents

[Lab Eight: Using AWS Lambda with Amazon S3 3](#_Toc509772591)

[Task One: Create two buckets, one source bucket and another target bucket 3](#_Toc509772592)

[Task Two: Create a Deployment Package 3](#_Toc509772593)

[Task Three: Create the Execution Role (IAM Role) 5](#_Toc509772594)

[Task Four: Create the Lambda Function (Upload the Deployment Package) 6](#_Toc509772595)

[Task Five: Test the Lambda Function (Invoke Manually) 6](#_Toc509772596)

[Task Six: Add Permissions to the Lambda Function's Access Permissions Policy 8](#_Toc509772597)

[Task Seven: Configure Notification on the Bucket 8](#_Toc509772598)

[Task Eight: Test the Setup 9](#_Toc509772599)

# Lab Eight: Using AWS Lambda with Amazon S3

In this lab, you will use Amazon S3 Event Notification to invoke AWS Lambda to create a thumbnail image of the user uploaded.

Through this lab, you will complete the following tasks:

* Task 1: Create two buckets, one source bucket and another target bucket
* Task 2: Create a Deployment Package
* Task 3: Create the Execution Role (IAM Role)
* Task 4: Create the Lambda Function
* Task 5: Test the Lambda Function
* Task 6: Add Permissions to the Lambda Function's Access Permissions Policy
* Task 7: Configure Notification on the Bucket
* Task 8: Test the Setup

## Task One: Create two buckets, one source bucket and another target bucket

Both the source bucket and your Lambda function must be in the same AWS region.

1. Go to S3 console.
2. Create two buckets. The target bucket name must be source followed by resized, where source is the name of the bucket you want to use for the source.

For example, mybucket and mybucketresized.

In the source bucket, upload a .jpg object, HappyFace.jpg.

Note: When you invoke the Lambda function manually before you connect to Amazon S3, you pass sample event data to the function that specifies the source bucket and HappyFace.jpg as the newly created object so you need to create this sample object first.

## Task Two: Create a Deployment Package

In this lab, we will create an example Python function and install dependencies. The code sample is compliant with Python runtime versions 3.6 or 2.7.

1. Create an EC2 instance with Amazon Linux AMI.
2. Connect to the Amazon Linux instance via SSH.

ssh -i key.pem ec2-user@public-ip-address

1. Open a text editor, and copy the following code. The code uploads the resized image to a different bucket with the same image name, as shown following:

source-bucket/image.png -> source-bucketresized/image.png

from \_\_future\_\_ import print\_function

import boto3

import os

import sys

import uuid

from PIL import Image

import PIL.Image

s3\_client = boto3.client('s3')

def resize\_image(image\_path, resized\_path):

with Image.open(image\_path) as image:

image.thumbnail(tuple(x / 2 for x in image.size))

image.save(resized\_path)

def handler(event, context):

for record in event['Records']:

bucket = record['s3']['bucket']['name']

key = record['s3']['object']['key']

download\_path = '/tmp/{}{}'.format(uuid.uuid4(), key)

upload\_path = '/tmp/resized-{}'.format(key)

s3\_client.download\_file(bucket, key, download\_path)

resize\_image(download\_path, upload\_path)

s3\_client.upload\_file(upload\_path, '{}resized'.format(bucket), key)

1. Save the file as CreateThumbnail.py.
2. Install Python 3.6 and virtualenv using the following steps:

sudo yum install -y gcc zlib zlib-devel openssl openssl-devel

wget https://www.python.org/ftp/python/3.6.1/Python-3.6.1.tgz

tar -xzvf Python-3.6.1.tgz

cd Python-3.6.1 && ./configure && make

sudo make install

sudo /usr/local/bin/pip3 install virtualenv

Choose the virtual environment that was installed via pip3

/usr/local/bin/virtualenv ~/shrink\_venv

source ~/shrink\_venv/bin/activate

Install libraries in the virtual environment

pip install Pillow

pip install boto3

1. Add the contents of lib site-packages to your .zip file. Note that the following steps assume you used Python runtime version 3.6.

cd $VIRTUAL\_ENV/lib/python3.6/site-packages

zip -r9 ~/CreateThumbnail.zip \*

1. Add your python code to the .zip file

cd ~

zip -g CreateThumbnail.zip CreateThumbnail.py

## Task Three: Create the Execution Role (IAM Role)

1. Sign in to the AWS Management Console and open the IAM console at [https://console.aws.amazon.com/iam/](https://console.aws.amazon.com/iam/" \t "_blank).
2. Choose Create role
3. In **Select type of trusted entity**, choose **AWS service**, and then choose **Lambda**. This will allow Lambda functions to call AWS services under your account.
4. Choose Next: Permissions
5. In Filter: Policy type enter AWSLambdaExecute and choose Next: Review.
6. In **Role name\***, enter a role name that is unique within your AWS account (for example, **lambda-s3-execution-role**) and then choose **Create role**.
7. Open the service role that you just created.
8. Under the Permissions tab, choose Add inline policy.
9. In service, choose Choose a service.
10. In Select a service below, choose S3.
11. In Actions, choose Select actions.
12. Expand Write under Access level groups and then choose PutObject.
13. Choose **Resources** and then choose the **Any** checkbox.
14. Choose Review policy.
15. Enter a **Name\*** and then choose **Create policy**. Note the policy specifications:

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "VisualEditor0",

"Effect": "Allow",

"Action": "s3:PutObject",

"Resource": "arn:aws:s3:::\*/\*"

}

]

}

1. Under the **Summary** of your role, record the **Role ARN**. You will need it in the next step when you create your Lambda function.

## Task Four: Create the Lambda Function (Upload the Deployment Package)

1. Configure AWS CLI
2. You need to update the command by providing the .zip file path and the execution role ARN. For the runtime parameter, choose python3.6.

aws lambda create-function \

--region region \

--function-name CreateThumbnail \

--zip-file fileb://file-path/CreateThumbnail.zip \

--role role-arn \

--handler CreateThumbnail.handler \

--runtime runtime \

--timeout 20 \

--memory-size 1024

3. Write down the function ARN. You will need this in the next section when you add notification configuration to your Amazon S3 bucket.

## Task Five: Test the Lambda Function (Invoke Manually)

1. Save the following Amazon S3 sample event data in a file and save it as inputFile.txt.
2. You need to update the JSON by providing your *sourcebucket* name and a .jpg object key.

{

"Records":[

{

"eventVersion":"2.0",

"eventSource":"aws:s3",

"awsRegion":"us-west-2",

"eventTime":"1970-01-01T00:00:00.000Z",

"eventName":"ObjectCreated:Put",

"userIdentity":{

"principalId":"AIDAJDPLRKLG7UEXAMPLE"

},

"requestParameters":{

"sourceIPAddress":"127.0.0.1"

},

"responseElements":{

"x-amz-request-id":"C3D13FE58DE4C810",

"x-amz-id-2":"FMyUVURIY8/IgAtTv8xRjskZQpcIZ9KG4V5Wp6S7S/JRWeUWerMUE5JgHvANOjpD"

},

"s3":{

"s3SchemaVersion":"1.0",

"configurationId":"testConfigRule",

"bucket":{

"name":"sourcebucket",

"ownerIdentity":{

"principalId":"A3NL1KOZZKExample"

},

"arn":"arn:aws:s3:::sourcebucket"

},

"object":{

"key":"HappyFace.jpg",

"size":1024,

"eTag":"d41d8cd98f00b204e9800998ecf8427e",

"versionId":"096fKKXTRTtl3on89fVO.nfljtsv6qko"

}

}

}

]

}

1. Run the following Lambda CLI invoke command to invoke the function.
2. Note that the command requests asynchronous execution.
3. You can optionally invoke it synchronously by specifying RequestResponse as the invocation-type parameter value.

aws lambda invoke \

--invocation-type Event \

--function-name CreateThumbnail \

--region region \

--payload file://file-path/inputfile.txt \

outputfile.txt

Note

You are able to invoke this function because you are using your own credentials to invoke your own function. In the next section, you configure Amazon S3 to invoke this function on your behalf, which requires you to add permissions to the access policy associated with your Lambda function to grant Amazon S3 permissions to invoke your function.

1. Verify that the thumbnail was created in the target bucket and monitor the activity of your Lambda function in the AWS Lambda console as follows:
   1. The AWS Lambda console shows a graphical representation of some of the CloudWatch metrics in the **Cloudwatch Metrics at a glance** section for your function.
   2. For each graph, you can also click the **logs** link to view the CloudWatch Logs directly.

## Task 6: Add Permissions to the Lambda Function's Access Permissions Policy

1. Run the following Lambda CLI add-permission command to grant Amazon S3 service principal (s3.amazonaws.com) permissions to perform the lambda:InvokeFunction action.

aws lambda add-permission \

--function-name CreateThumbnail \

--region region \

--statement-id some-unique-id \

--action "lambda:InvokeFunction" \

--principal s3.amazonaws.com \

--source-arn arn:aws:s3:::sourcebucket \

--source-account bucket-owner-account-id \

## Task 7: Configure Notification on the Bucket

1. Go to S3 console. In the **Bucket Name** list, choose the **sourcebucket** you want to enable events for.
2. Choose **properties**
3. Under advanced settings, choose **Events**
4. Choose **Add Notification**
5. Enter a Name for the event.
6. Under Events, select **Object Created (All)**.
7. Select the type of destination as **Lambda**.
8. In **Lambda Function**, type or choose the name of the Lambda function (CreateThumbnail) that you want to receive notifications from Amazon S3.
9. Choose **Save**

## Task 8: Test the Setup

You're all done! Now you can test the setup as follows:

1. Upload .jpg objects to the source bucket using the Amazon S3 console.
2. Verify that the thumbnail was created in the target bucket using the CreateThumbnail function.
3. You can also verify the CloudWatch Logs.
4. You can monitor the activity of your Lambda function in the AWS Lambda console. For example, choose the logs link in the console to view logs, including logs your function wrote to CloudWatch Logs.