Introduction to AWS Lambda

Overview

The lab provides a basic explanation of AWS Lambda. It will demonstrate the steps required to get started to create a Lambda function in an event-driven environment.

AWS Lambda is a compute service that runs your code in response to events and automatically manages the compute resources for you, making it easy to build applications that respond quickly to new information. AWS Lambda starts running your code within milliseconds of an event such as an image upload, in-app activity, website click, or output from a connected device. You can also use AWS Lambda to create new back-end services where compute resources are automatically triggered based on custom requests.

Topics covered

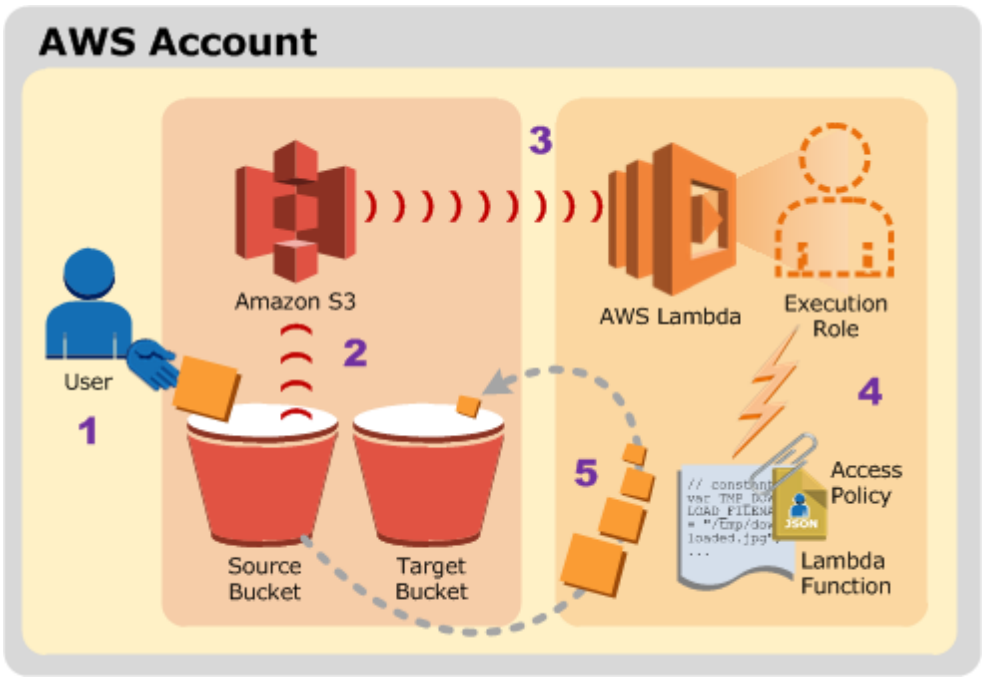
By the end of this lab you will be able to:

- Create an AWS Lambda function
- Configure an Amazon S3 bucket as a Lambda Event Source
- Trigger a Lambda function by uploading an object to Amazon S3
- Monitor AWS Lambda S3 functions through Amazon CloudWatch Log

Scenario

This lab demonstrates AWS Lambda by creating a serverless **image thumbnail application**.

The following diagram illustrates the application flow:



1 A user uploads an object to the source bucket in **Amazon S3** (object-created event).

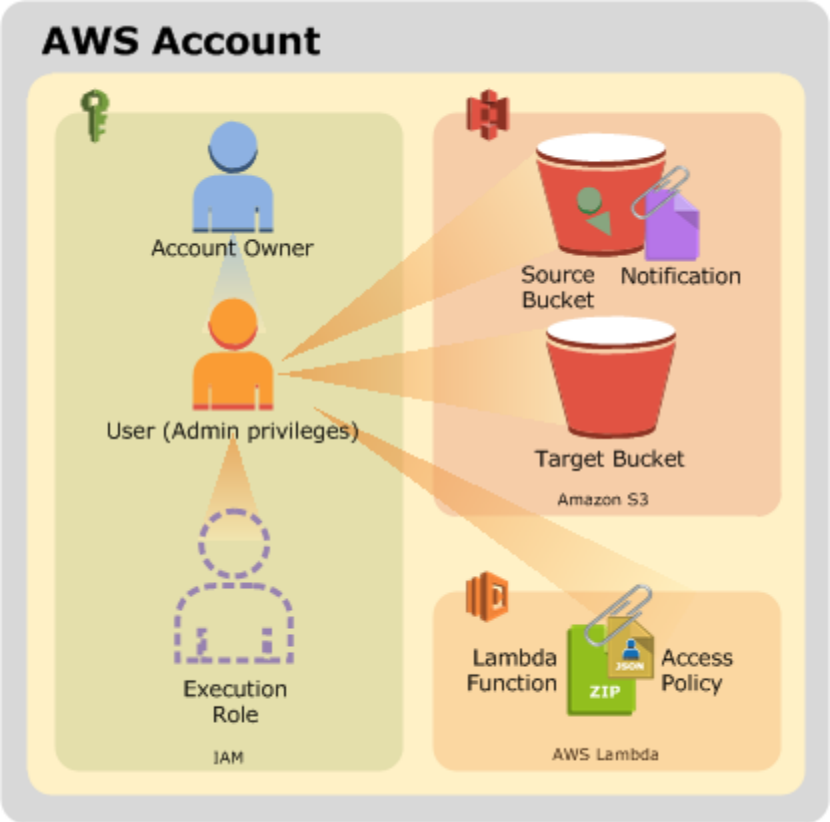
2 Amazon S3 detects the object-created event.

3 Amazon S3 publishes the object-created event to AWS Lambda by invoking the Lambda function and passing event data as a function parameter.

4 AWS Lambda executes the Lambda function.

5 From the event data it receives, the Lambda function knows the source bucket name and object key name. The Lambda function reads the object and creates a thumbnail using graphics libraries, then saves the thumbnail to the target bucket.

Upon completing this tutorial, you will have the following resources in your account:



The steps in this lab will show you how to create the Amazon S3 buckets and the Lambda function. You will then test the service by uploading images for resizing.

Task 1: Create the Amazon S3 Buckets

In this task, you will create two Amazon S3 buckets -- one for input and one for output.

3. In the **AWS Management Console**, on the **Services** menu, click **S3**.
4. If you see a message at the top of the screen that says *We've temporarily re-enabled the previous version of the S3 console while we continue to improve the new S3 console experience. Switch to the new console.*, click **Switch to the new console**.
5. In the **AWS Management Console**, on the **Services** menu, click **S3**.
6. Click **Create bucket** and then configure:

*Amazon S3 buckets require unique names, so you will add a random number to the bucket name such as *images-123456789*.

- **Bucket name:** images-123456789

*Replace **123456789** with a random number

- Copy the name of your bucket to a text editor
- Click **Create bucket**

If you receive an error stating **The requested bucket name is not available**, then click the first **Edit** link, change the bucket name with a different **random number** and try again until it works.

- Scroll to the bottom of the screen to click **Create bucket** leaving the rest of the options as default.

7. You will now create another bucket for **output**. Click **Create bucket** with similar steps as the previous bucket, now configure:

   - **Bucket name:** Paste the name of your *images* bucket
   - At the end of the bucket name, append -resized
   - Click **Create bucket**

   *Do not change the Region.

8. You will now upload a picture for testing purposes.

   - Right-click this link and download the picture to your computer: HappyFace.jpg

   - Name the file **HappyFace.jpg**.

   *Firefox users: Make sure the saved filename is *HappyFace.jpg* (not *.jpeg)*.

9. Open the image on your computer.

   - It is a large picture, with dimensions of 1280 x 853.

10. In the **S3 Management Console**, click the **images-** bucket. (Not the *-resized* bucket)

11. Click Upload

12. In the **Upload** window, click Add files

13. Browse to and select the **HappyFace.jpg** picture you downloaded.

14. Click Upload

Later in this lab you will invoke the Lambda function manually by passing sample event data to the function. The sample data will refer to this *HappyFace.jpg* image.

Task 2: Create an AWS Lambda Function

In this task, you will create an AWS Lambda function that reads an image from Amazon S3, resizes the image and then stores the new image in Amazon S3.

15. On the **Services** menu, click **Lambda**.

Do not change the Region. You must use **us-east-1** for this lab.

16. Click **Create function**

Blueprints are code templates for writing Lambda functions. Blueprints are provided for standard Lambda triggers such as creating Alexa skills and processing Amazon Kinesis Firehose streams. This lab provides you with a pre-written Lambda function, so you will **Author from scratch**.

17. Choose **Author from scratch**

18. In the **Create function** window, configure:

- **Function name:** Create-Thumbnail
- **Runtime:** *Python 3.7*
- Expand **Change default execution role**
- **Execution role:** Select **Use an existing role**
- **Existing role:** Choose **TeamRole**

*Make sure to select Python 3.7 under **Other Supported** runtime. If you select Python 3.8 from **Latest supported** list, the code will fail.

This **role** grants permission to the Lambda function to access Amazon S3 to read and write the images.

19. Click **Create function**

A page will be displayed with your function configuration.

AWS Lambda functions can be **triggered** automatically by activities such as data being received by Amazon Kinesis or data being updated in an Amazon DynamoDB database. For this lab, you will trigger the Lambda function whenever a new object is created in your Amazon S3 bucket.

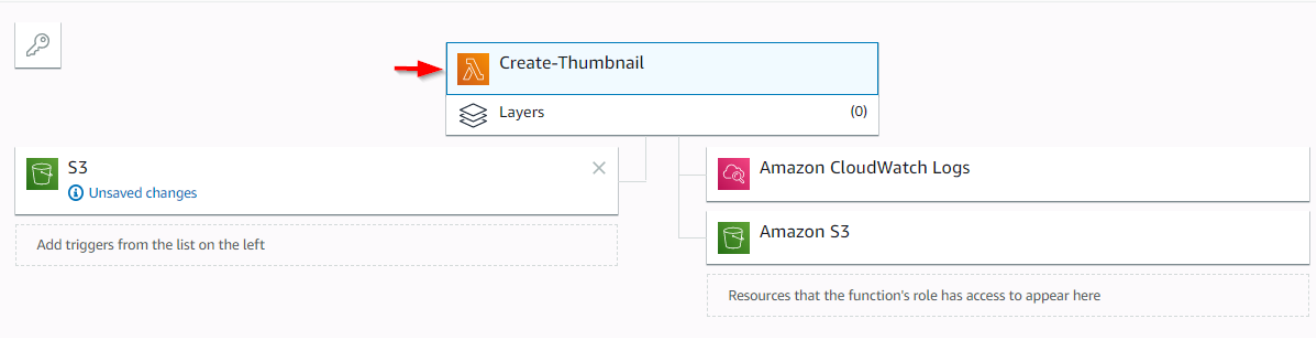
20. Click **Add trigger** then configure:

- **Select a trigger:** *S3*

- **Bucket:** Select your **images-** bucket (e.g. *images-123456789*)
- **Event type:** *All object create events*
- For **Recursive invocation**, Select **I acknowledge that ...**

21. Scroll to the bottom of the screen, then click **Add**

22. Click **Create-Thumbnail** at the top of the diagram:



You will now configure the Lambda function.

23. Scroll down to the **Function code** section and configure the following settings (and ignore any settings that aren't listed):

- Click **Actions** menu and select **Upload a file from Amazon S3**
- **Amazon S3 link URL:** Copy and paste this URL into the field:

  https://dchristian3188-us-east-1-random.s3.amazonaws.com/CreateThumbnail.zip

- Click **Save**

The *CreateThumbnail.zip* file contains the following Lambda function:

Do not copy this code -- it is just showing you what is in the Zip file.

```
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((128, 128))
        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)
```

24. Examine the above code. It is performing the following steps:

    - Receives an Event, which contains the name of the incoming object (Bucket, Key)
    - Downloads the image to local storage
    - Resizes the image using the *Pillow* library
    - Uploads the resized image to the *-resized* bucket

25. In the **Runtime settings** section, click **Edit**

    - **Handler** enter: CreateThumbnail.handler
    - Click **Save**

*Make sure you set the **Handler** field to the above value, otherwise the Lambda function will not be found.

26. In the **Basic settings** section towards the bottom of the page, click **Edit**

    - **Description** enter: Create a thumbnail-sized image

You will leave the other settings as default, but here is a brief explanation of these settings:

- **Memory** defines the resources that will be allocated to your function. Increasing memory also increases CPU allocated to the function.
- **Timeout** sets the maximum duration for function execution.

- Click **Save**

Your Lambda function has now been configured.

Task 3: Test Your Function

In this task, you will test your Lambda function. This is done by simulating an event with the same information normally sent from Amazon S3 when a new object is uploaded.

27. At the top of the screen, click **Test** then configure:

    - **Event template:** *Amazon S3 Put*
    - **Event name:** TestS3Event

A sample template will be displayed that shows the event data sent to a Lambda function when it is triggered by an upload into Amazon S3. You will need to edit the bucket name so that it uses the bucket you created earlier.

28. Replace **example-bucket** with the name of your images bucket (e.g. *images-123456789*) that you copied to your text editor.

Be sure to replace **example-bucket** in both locations.

29. Replace **test/key** with the name of the picture that you uploaded. This should be

```
{
  "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": "images-234531245234",
          "ownerIdentity": {
            "principalId": "EXAMPLE"
          },
          "arn": "arn:aws:s3:::images-234531245234"
        },
        "object": {
          "key": "HappyFace.jpg",
```

30. Click **Create**

31. Click **Test**

AWS Lambda will now trigger your function, using *HappyFace.jpg* as the input image.

Towards the top of the page you should see the message: *Execution result: succeeded*

Result returned by your function will show as *null*.

*If your test did not succeed, the error message will explain the cause of failure.

For example, a *Forbidden* message means that the image was not found possibly due to an incorrect bucket name. Review the previous steps to confirm that you have configured the function correctly.

32. Click **Details** to expand it (towards the top of the screen).

    You will be shown information including:

- Execution duration
- Resources configured
- Maximum memory used
- Log output

You can now view the resized image that was stored in Amazon S3.

33. On the **Services** menu, click **S3**.

34. Click the name of your ***-resized*** bucket (which is the second bucket you created), then:

    - Select **HappyFace.jpg**
    - Click **Actions** menu and select **Open** (If the image does not open, disable your pop-up blocker.)

The image should now be a smaller thumbnail of the original image.

You are welcome to upload your own images to the *images-* bucket and then check for thumbnails in the *-resized* bucket.

Task 4: Monitoring and Logging

You can monitor AWS Lambda functions to identify problems and view log files to assist in debugging.

35. On the **Services** menu, click **Lambda**.

36. Click your **Create-Thumbnail** function.

37. Click the **Monitoring** tab.

The console displays graphs showing:

- **Invocations:** The number of times the function has been invoked.
- **Duration:** How long the function took to execute (in milliseconds).
- **Errors:** How many times the function failed.
- **Throttles:** When too many functions are invoked simultaneously, they will be throttled. The default is 1000 concurrent executions.
- **Iterator Age:** Measures the age of the last record processed from streaming triggers (Amazon Kinesis and Amazon DynamoDB Streams).

Log messages from Lambda functions are retained in **Amazon CloudWatch Logs**.

38. Click **View logs in CloudWatch**

39. Click the **Log Stream** that appears.

40. Expand each message to view the log message details.

The Event Data includes the Request Id, the duration (in milliseconds), the billed duration (rounded up to the nearest 100 ms, the Memory Size of the function and the Maximum Memory that the function used. In addition, any logging messages or print statements from the functions are displayed in the logs. This assists in debugging Lambda functions.

Conclusion

Congratulations! You have successfully:

- Created an AWS Lambda function
- Configured an Amazon S3 bucket as a Lambda Event Source
- Triggered a Lambda function by uploading an object to Amazon S3
- Monitored AWS Lambda S3 functions through Amazon CloudWatch Log