



AWS Image Recognition Pipeline Project Walkthrough

21/02/2025

Kurudunje Deekshith Shetty
Programming Assignment 1
CS643

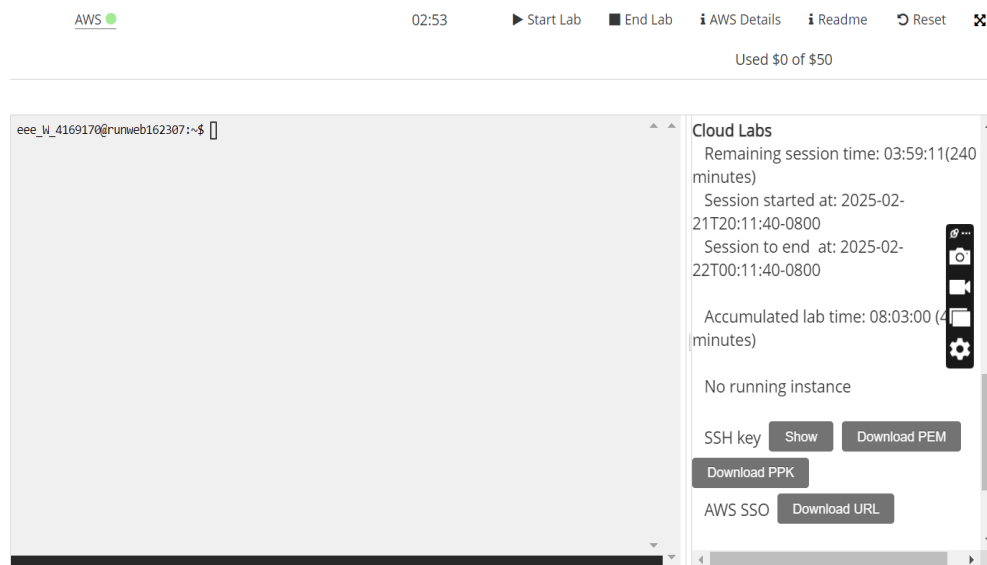
Overview

This document provides a step-by-step guide to setting up and running the AWS image recognition pipeline project using EC2 instances, S3, SQS, Rekognition, and Textract.

AWS Learner Lab Setup

Access AWS Learner Lab:

- Log in to your AWS Academy Learner Lab course using the provided link: <https://awsacademy.instructure.com/courses/109197>
- Navigate to Modules -> Learner Lab -> Learner Lab.
- Click Start Lab (in the top menu).



- Connect to the AWS Management Console by clicking the AWS link by downloading the URL within the AWS details menu(AWS SSO).

EC2 Instances Setup

Get EC2 Keys:

- Click on AWS Details (top menu, after starting the lab).
- Download the .pem key (for Linux/Mac) or .ppk key (for Windows/PuTTY). This key will be used for SSH access to both instances.

Launch EC2 Instances:

- Navigate to the EC2 service in the AWS Management Console.
- Launch two EC2 instances ("ec2A" and "ec2B").
- Select Amazon Linux AMI as the operating system. Choose the latest version that you see in free tier use.
- Choose an appropriate instance type (t2.micro).

▼ **Instance type**
[Info](#) | [Get advice](#)

Instance type

t2.micro

Free tier eligible

Family: t2 1 vCPU 1 GiB Memory Current generation: true
On-Demand Windows base pricing: 0.0162 USD per Hour
On-Demand Ubuntu Pro base pricing: 0.0134 USD per Hour
On-Demand SUSE base pricing: 0.0116 USD per Hour
On-Demand RHEL base pricing: 0.026 USD per Hour
On-Demand Linux base pricing: 0.0116 USD per Hour

☒ All generations
[Compare instance types](#)

[Additional costs apply for AMIs with pre-installed software](#)

▼ **Key pair (login)**
[Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - required

vockey

▼

[Create new key pair](#)

- When launching instances, select the key that already exists in the EC2 dashboard->Key pairs ("vockey").

Configure Security Groups:

- Configure the Security Group to prevent unauthorized access.
- In the Security Group tab, set the "Source" to "MY IP" for SSH to restrict access to your IP address only.

Firewall (security groups) | [Info](#)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

☒ Create security group

☐ Select existing security group

We'll create a new security group called **'launch-wizard-3'** with the following rules:

☒ Allow SSH traffic from
Helps you connect to your instance

My IP
108.53.25.28/32

☒ Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server

☒ Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server

- Open the following ports:
 - SSH (22)
 - HTTP (80)
 - HTTPS (443)

Programmer's Keys Setup

Access AWS Credentials:

- Go to AWS Details in the Learner Lab.
- Find the section related to AWS CLI credentials.

AWS

02:47

▶ Start Lab

■ End Lab

i AWS Details

i Readme

↺ Reset

⋮

Used \$0 of \$50

4169170@runweb162307:~\$

Copy and paste the following into ~/.aws/credentials

```
[default]
aws_access_key_id=ASIAQKARQHAWKUXKKPT
aws_secret_access_key=xndCYEW1uhAXm1Lp2pLwxehIFQm19AXKJDg5wx1D
aws_session_token=IQoJb3JpZ2luX2VjELZ////////wEaCXVzLXd1c3QtMiJIMEYCIQCI13JGYQBGT7bXxorKBhbt5A7+CM1R+LAGOQuu2VoC5gIhANKgDUXnEfAdiR0WULIISYcGM9puYPEHNsN1YKZ/SSDnKr4CCOX////////wEQARoMMDIXNTExNTUxNDI1IgwHj99Moeqfs9w6mGMqgL U3Akn+SDFDfVZshodhPqxqPi6fEiAYEqPYuAvqu0hJJt2pjMapEmoLJ9wK8kOQEjPjV68sA82tQnDPTIQE3VaQ6UR6H4jJLZaUuY4VuhP11kmKRwy8KuFcYnV6eWJdGdukjL6t1fupdE+i913TYYF1a9MQKP4vHTv9rTAfw11RIWvOdrCjoPLPqMTTMea3TMEAnP7iNUBE0JJAMmr8KGMRDhigpDjaoyLSyMrtYAFhcQAscSLAS17A0B3W+P2p52bkpTkVq1iDvx2rL/C/ErICPMup5teDvD/Gk7/SaLchqNCI85n0g08WU8pe4L5swiBfQKJ3BH5CsHB8GOPZDVhd4yBZRE5wC3ivGg361VtPMP2c5b0GOpwB0qm+nmP80NHsz26mMgeImg
```

- ## Configure AWS CLI

-

- ```

ec2-user@ip-172-31-89-254:~
Using username "ec2-user".
Authenticating with public key "imported-openssh-key"

A newer release of "Amazon Linux" is available.
Version 2023.6.20250218:
Run "/usr/bin/dnf check-release-update" for full release and version update info

#
~\##### Amazon Linux 2023
~~~\#####\
~~~\####|
~~~\#/ https://aws.amazon.com/linux/amazon-linux-2023
~~~~V~'-'>
~~~~
~~~~.-.
~~~~/m/'-/-/-
Last login: Sat Feb 22 04:56:44 2025 from 108.53.25.28
[ec2-user@ip-172-31-89-254 ~]$ aws configure
AWS Access Key ID [*****KKPT]: 

```

- Enter the access key ID, secret access key, default region (in my case : us-east-1), and output format (in my case : json).

## Configure SQS

- Navigate to the SQS service in the AWS Management Console.
- Click on the create queue option. Select the “fifo” option and give a custom name to it(e.g. MySQLS).

**Details**

**Type**  
Choose the queue type for your application or cloud infrastructure.

☐ **Standard** [Info](#)  
At-least-once delivery, message ordering isn't preserved
 

- At-least once delivery
- Best-effort ordering

☒ **FIFO** [Info](#)  
First-in-first-out delivery, message ordering is preserved
 

- First-in-first-out delivery
- Exactly-once processing

*You can't change the queue type after you create a queue.*

**Name**

MySQLS.fifo

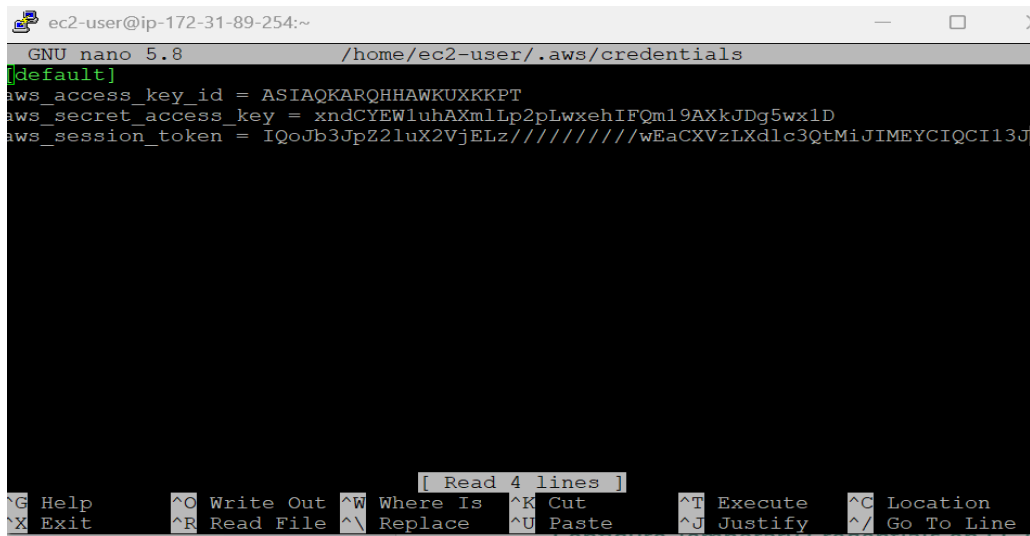
A queue name is case-sensitive and can have up to 80 characters. You can use alphanumeric characters, hyphens (-), and underscores (\_).

- Turn on the “Content-based deduplication” option. Not required to edit the other options. Save the queue configuration.

## Temporary Credentials Configuration

### Configure Temporary Credentials on EC2 Instances:

- Connect to your EC2 instances via SSH.
- Edit the ~/.aws/credentials file. If the file doesn't exist, create it.



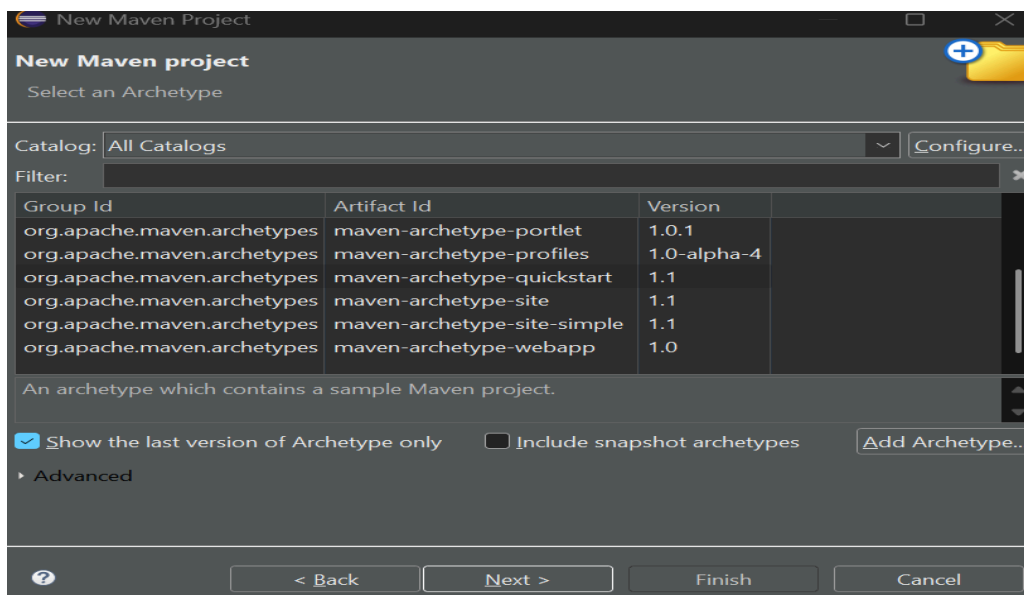
ec2-user@ip-172-31-89-254:~  
 GNU nano 5.8 /home/ec2-user/.aws/credentials  
 [default]  
 aws\_access\_key\_id = ASIAQKARQHHAWKUXKKPT  
 aws\_secret\_access\_key = xndCYEW1uhAXmlIp2pLwxehIFQm19AXkJDg5wx1D  
 aws\_session\_token = IQoJb3JpZ2luX2VjELz////////wEaCXVzLXdlc3QtMiJIMEYCIQCI13Jf  
 [ Read 4 lines ]  
 Help Write Out Where Is Cut Execute Location  
 Exit Read File Replace Paste Justify Go To Line

- Add a new profile with the copied temporary credentials : `aws_access_key_id`, `aws_secret_access_key`, and `aws_session_token`.

## Java Application Implementation

### Development Environment:

- Use a Java IDE (Eclipse, etc.) on your local machine.
- Create a new Project. (in my case : Maven project)



- Include the AWS SDK for Java in your project. If you're using Maven too, There will be an existing "pom.xml" file. You'll be required to add the required dependencies of S3, Rekognition and SQS (refer to submitted "pom.xml" for entire code).

```
<dependencies>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>s3</artifactId>
  </dependency>

  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>rekognition</artifactId>
  </dependency>

  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>sqs</artifactId>
  </dependency>
</dependencies>
```

### Instance ec2A (Face Detection):

- Use the AWS SDK for Java to read 10 images from the public S3 bucket: "cs643-sp25-project1".
- We are going to use the queue "MySQL.fifo".
- For each image, use the Rekognition service to detect faces.
- Set the confidence threshold to 75%.
- If a face is detected with confidence > 75%, store the image index in the SQS queue.
- After processing all images, send the index "-1" to the SQS queue to signal the end of processing.
- For code, refer to submitted "FaceRecognition.java" file

### Instance ec2B (Text Recognition):

- Read Image Indexes from SQS: Continuously read image indexes from the SQS queue.
- For each image index received, retrieve the corresponding image from the S3 bucket.
- Use the Textract service to extract text from the image.
- Store the image index and the extracted text.
- Stop Processing: Stop processing when the index "-1" is received from the SQS queue.
- Write the indexes of images containing both a face and text, along with the extracted text, to the output.txt file on ec2B.

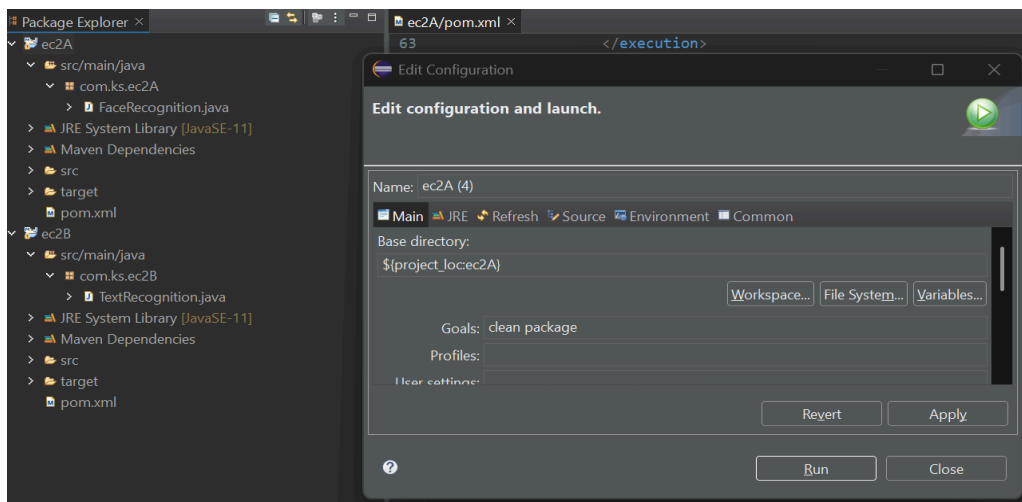


- For code, refer to submitted “TextRecognition.java” file

## Code Deployment and Execution

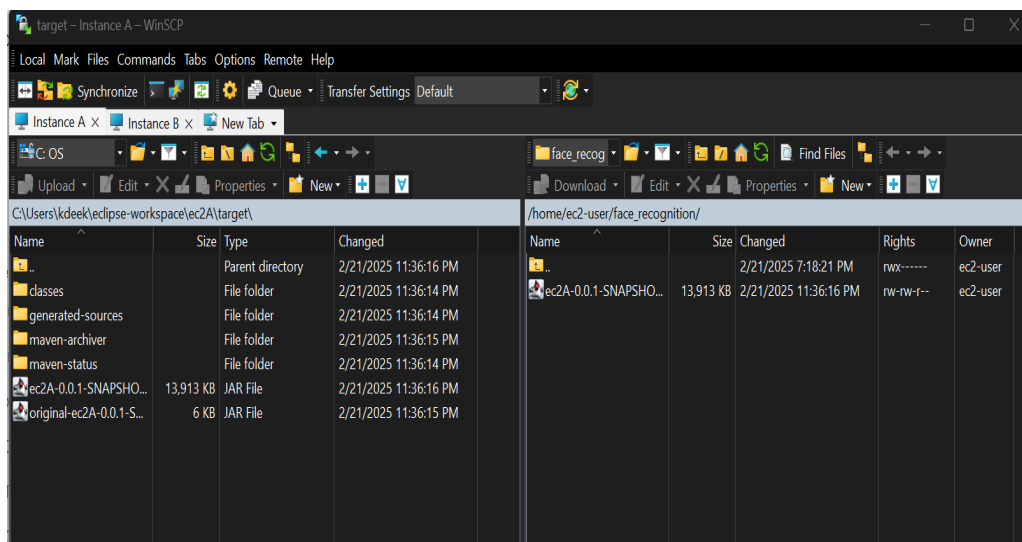
### Compile Java Code:

- Compile your Java code into executable JAR files.
- Click on run as, Use the Maven build, then select the “clean package” option for the group. Click on run to build and save it.



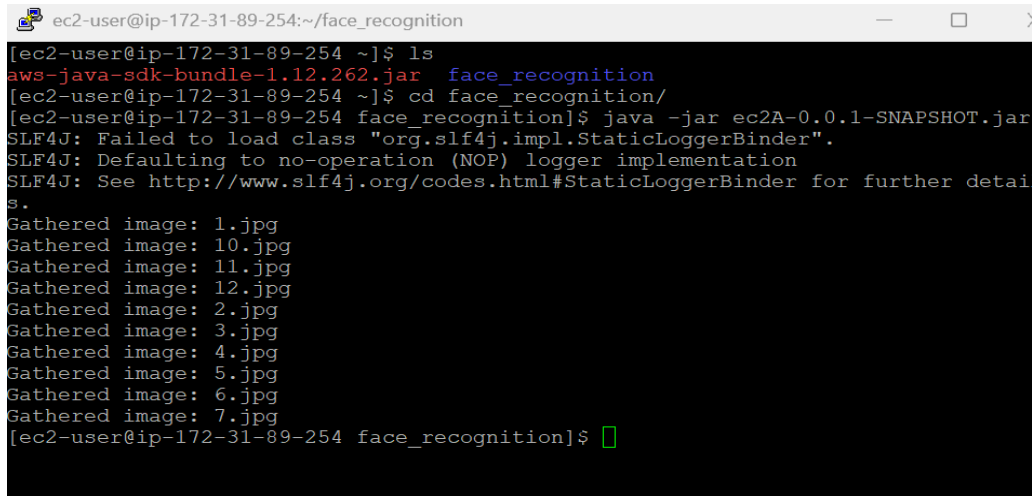
### Transfer JAR Files to EC2 Instances:

- Use the “scp” command to transfer the JAR files to your EC2 instances. For convenience, you can use the winSCP application on windows.



## Run Java Applications:

- Connect to your EC2 instances via SSH.
- Run the Java applications using the following command: "java -jar example\_app.jar" (ec2A-0.0.1-SNAPSHOT.jar on ec2A and c2A-0.0.1-SNAPSHOT.jar on ec2B)
- I've renamed it to "FaceRecognition.jar" and "TextRecognition.jar" in my submission.



```

ec2-user@ip-172-31-89-254:~/face_recognition
[ec2-user@ip-172-31-89-254 ~]$ ls
aws-java-sdk-bundle-1.12.262.jar  face_recognition
[ec2-user@ip-172-31-89-254 ~]$ cd face_recognition/
[ec2-user@ip-172-31-89-254 face_recognition]$ java -jar ec2A-0.0.1-SNAPSHOT.jar
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further details.
Gathered image: 1.jpg
Gathered image: 10.jpg
Gathered image: 11.jpg
Gathered image: 12.jpg
Gathered image: 2.jpg
Gathered image: 3.jpg
Gathered image: 4.jpg
Gathered image: 5.jpg
Gathered image: 6.jpg
Gathered image: 7.jpg
[ec2-user@ip-172-31-89-254 face_recognition]$
  
```

## Verification and Output

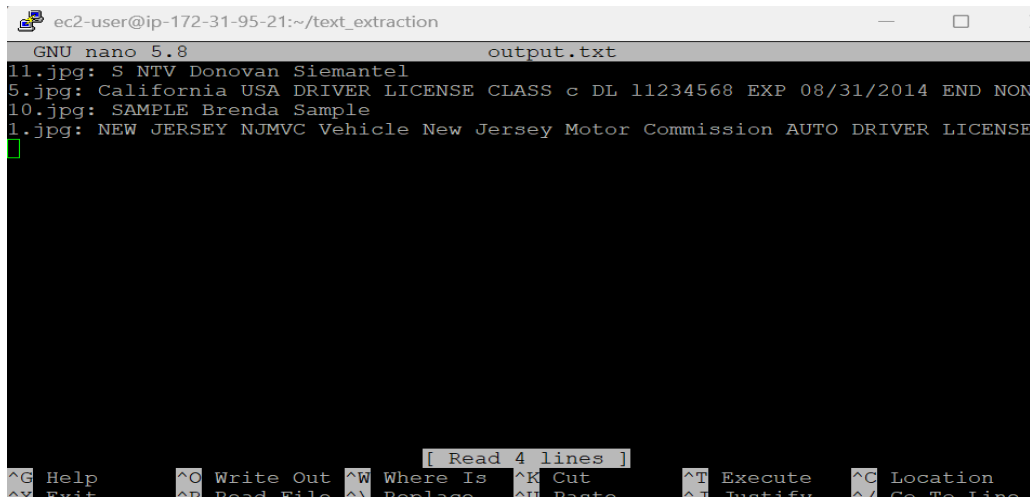
### Monitor Execution:

- Monitor the output of your Java applications in the terminal to ensure they are running correctly.
- Check the SQS queue to see messages being added and consumed.

| Queues (1) <span>Edit</span> <span>Delete</span> <span>Send and receive</span> |                            |      |                        |                    |                    |  |
|--------------------------------------------------------------------------------|----------------------------|------|------------------------|--------------------|--------------------|--|
| <input type="text" value="Search queues by prefix"/>                           |                            |      |                        |                    |                    |  |
|                                                                                | Name                       | Type | Created                | Messages available | Messages in flight |  |
|                                                                                | <a href="#">MySQL.fifo</a> | FIFO | 2025-02-21T18:30-05:00 | 6                  | 0                  |  |

## Verify Output:

- After TextRecog.jar finishes processing on ec2B, check the output.txt file on it.



The screenshot shows a terminal window titled "ec2-user@ip-172-31-95-21:~/text\_extraction". Inside, the GNU nano 5.8 editor is open, displaying the file "output.txt". The file contains the following text:

```
11.jpg: S NTV Donovan Siemantel
5.jpg: California USA DRIVER LICENSE CLASS c DL 11234568 EXP 08/31/2014 END NON
10.jpg: SAMPLE Brenda Sample
1.jpg: NEW JERSEY NJMVC Vehicle New Jersey Motor Commission AUTO DRIVER LICENSE
```

The bottom of the screen shows the nano editor's command palette with options like Help, Write Out, Where Is, Cut, Execute, and Location.

- Verify that the file contains the correct image indexes and extracted text.

## Demonstration Link

[AWS Image Recognition Pipeline Project Demo](#)

## Convenience case:

- For ease, I've attached the ".jar" file within the zip file for both the codes built with all the required dependencies.
- Copy the "FaceRecognition.jar" to instance 1, "TextRecognition.jar" to instance 2 and run the following command : `java -jar <application_name>.jar` on both the instances. You should be able to see the output.txt file on instance 2.