AWS Image Recognition Pipeline Project Walkthrough

21/02/2025

Kurudunje Deekshith Shetty

Programming Assignment 1

CS643

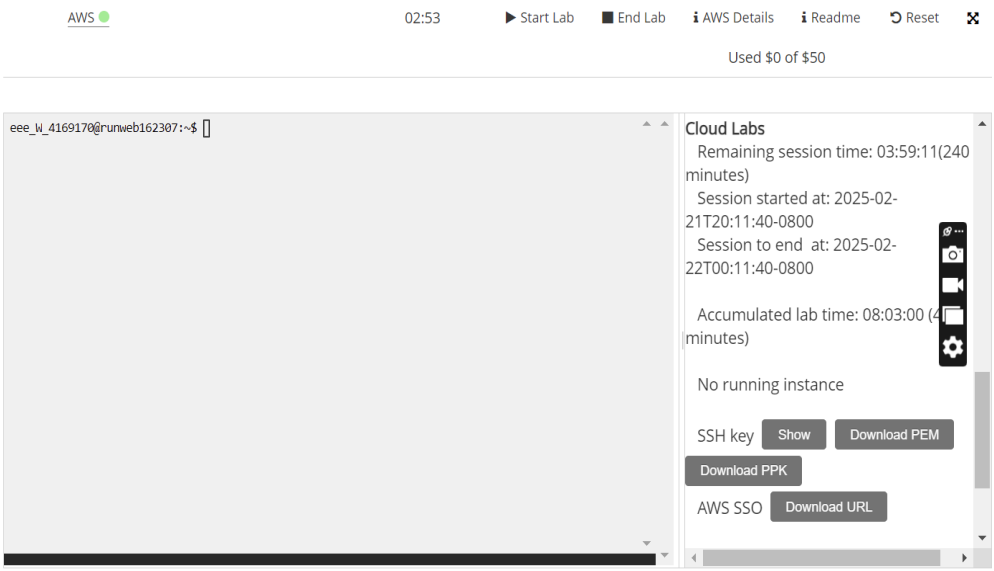
Overview

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).

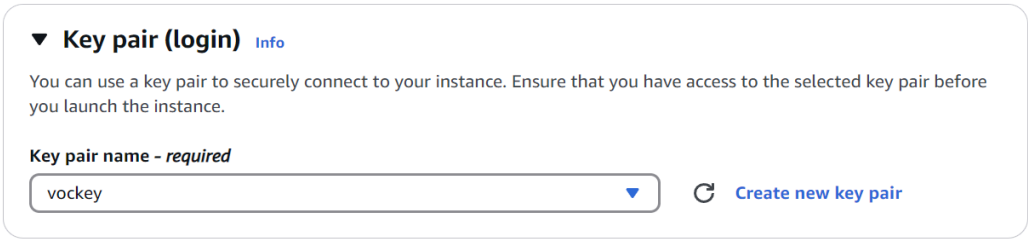
EC2 Instances Setup

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:

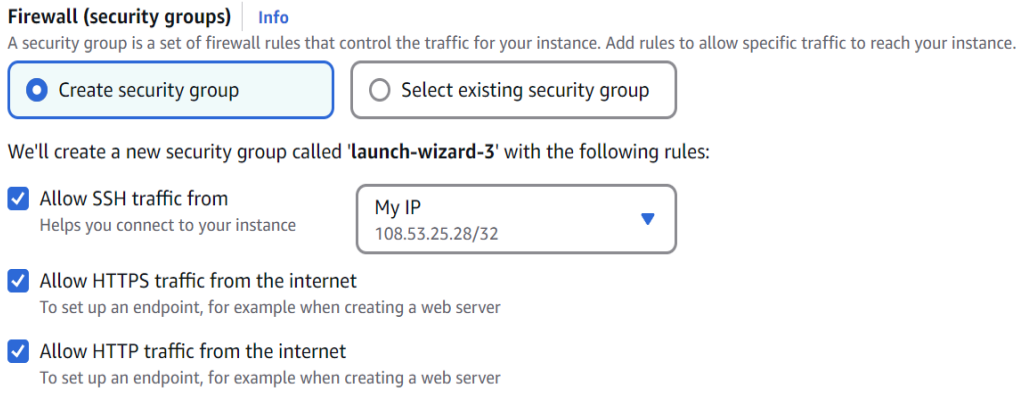
- Navigate to the EC2 service in the AWS Management Console.
- Launch two EC2 instances ("ec2A" and "ec2B").
- Select Amazon Linux AMI as the operating system. Choose the latest version that you see in free tier use.
- Choose an appropriate instance type (t2.micro).



- 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.

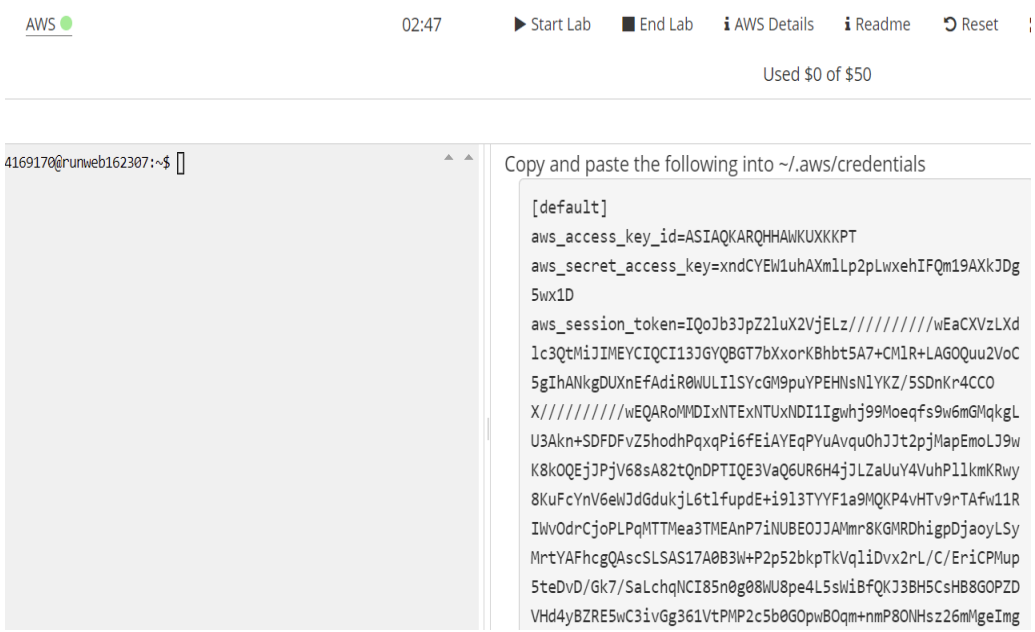


- Open the following ports:
  - SSH (22)
  - HTTP (80)
  - HTTPS (443)

Programmer's Keys Setup

Access AWS Credentials:

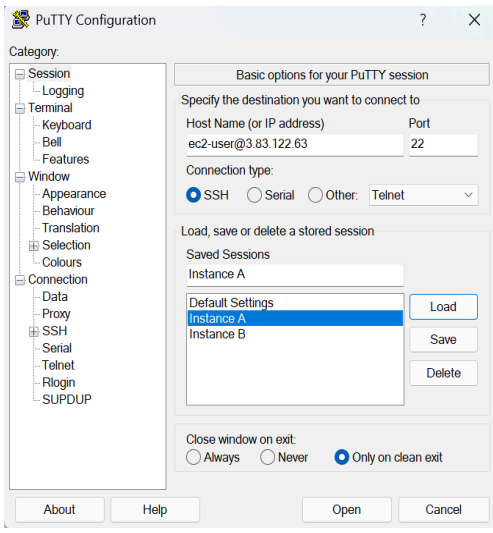
- Go to AWS Details in the Learner Lab.
- Find the section related to AWS CLI credentials.

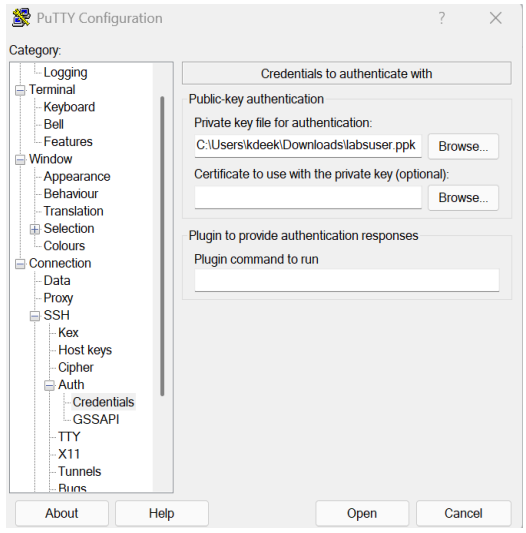


- Note the aws_access_key_id, aws_secret_access_key, and aws_session_token. This will be used later as temporary credentials and it is to be noted that these change every session.

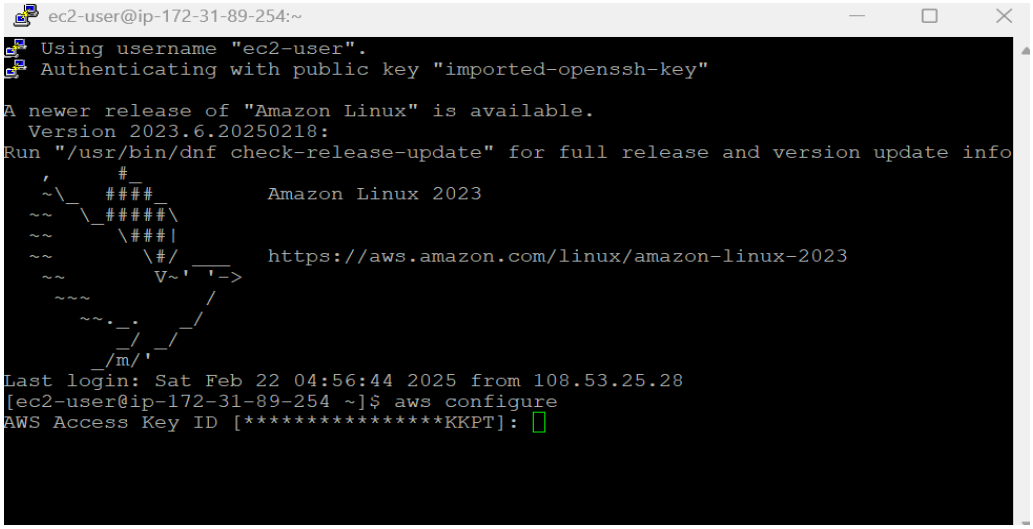
Configure AWS CLI

- Connect to your EC2 instances via SSH (using the .pem key or .ppk key). Use the public IP's of the instances for the host name in the format "ec2-user@<IP>". Use the .ppk for authentication.





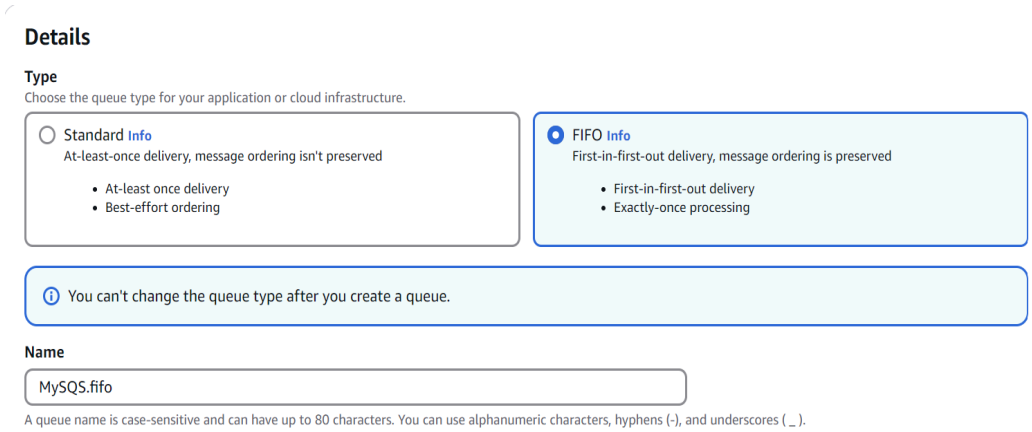
- Install the AWS CLI: sudo yum install aws-cli
- Configure the AWS CLI with the credentials we saved before using the command "aws configure"



- 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. MySQS).

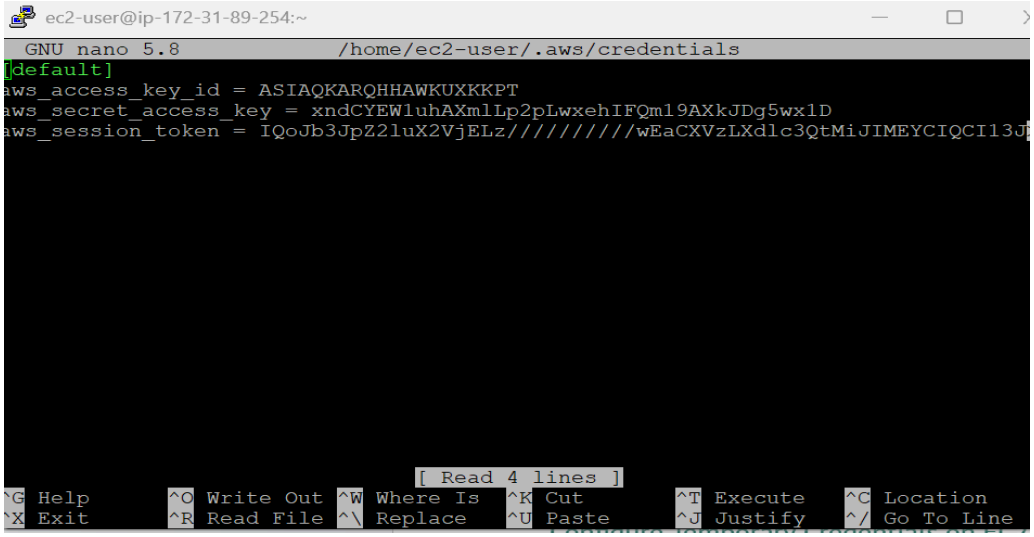


- Turn on the “Content-based deduplication” option. Not required to edit the other options. Save the queue configuration.

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.

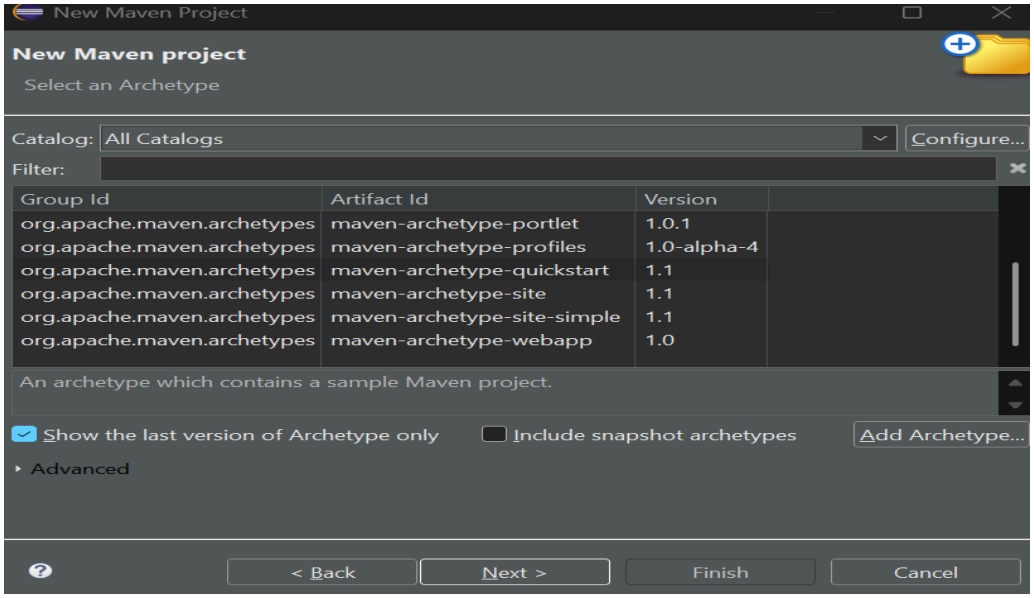


- Add a new profile with the copied temporary credentials : aws_access_key_id, aws_secret_access_key, and aws_session_token.

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 “MySQS.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):

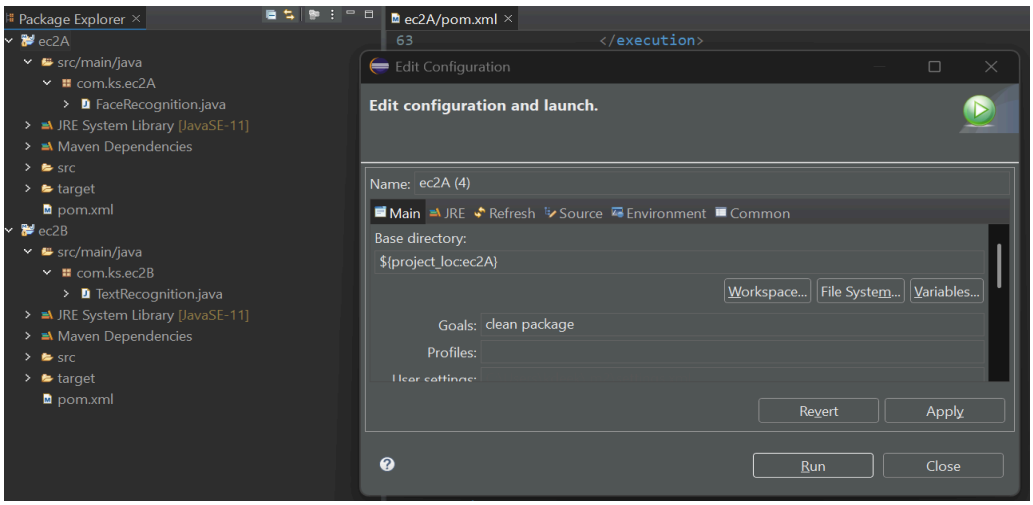
- Read Image Indexes from SQS: Continuously read image indexes from the SQS queue.
- For each image index received, retrieve the corresponding image from the S3 bucket.
- Use the Textract service to extract text from the image.
- Store the image index and the extracted text.
- Stop Processing: Stop processing 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

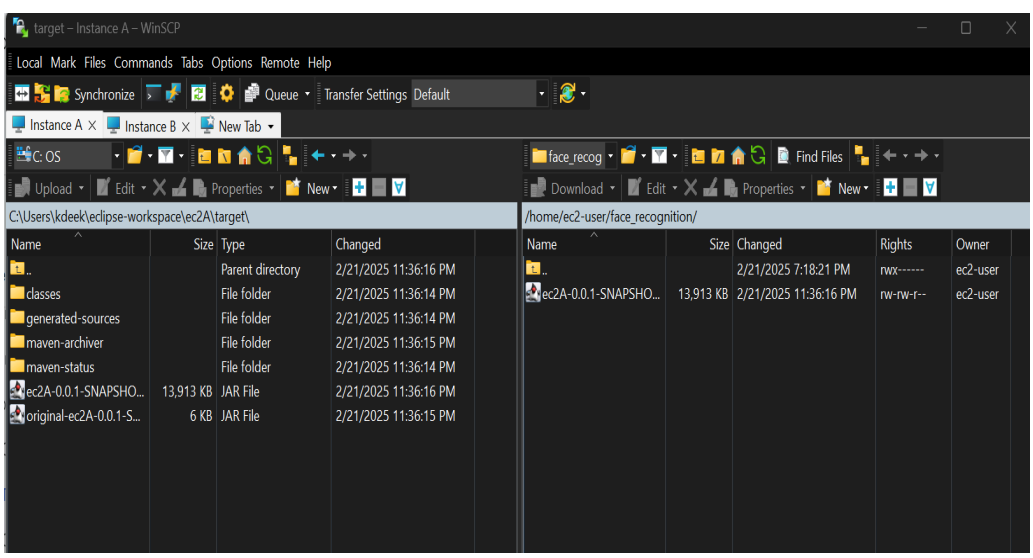
Compile Java Code:

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



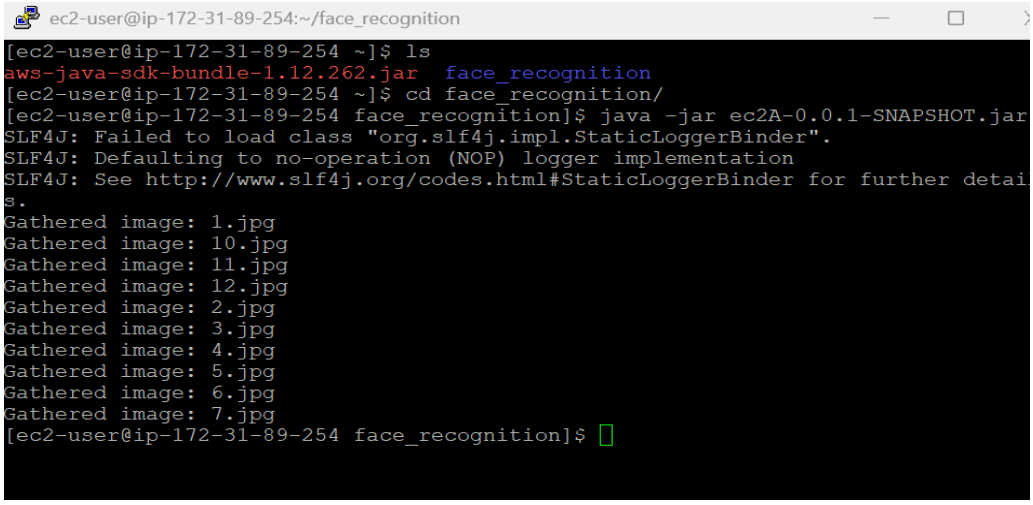
Transfer JAR Files to EC2 Instances:

- 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:

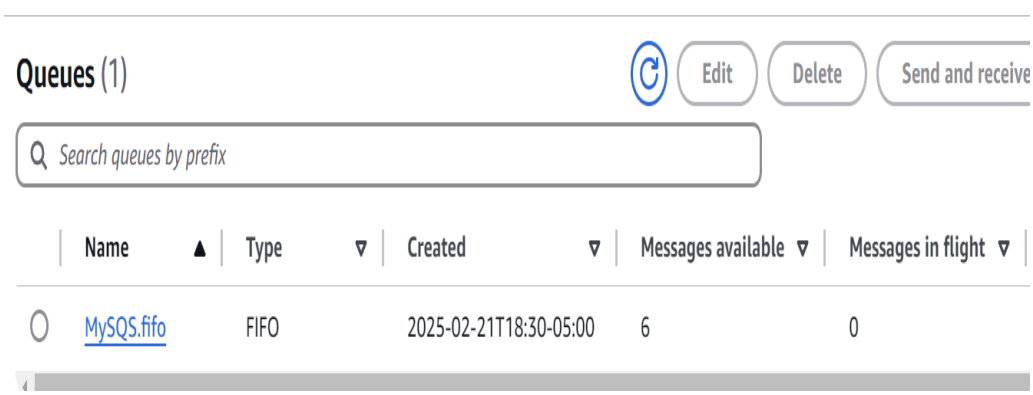
- Connect to your EC2 instances via SSH.
- Run the Java applications using the following command: “java -jar example_app.jar” (ec2A-0.0.1-SNAPSHOT.jar on ec2A and c2A-0.0.1-SNAPSHOT.jar on ec2B)
- I’ve renamed it to “FaceRecognition.jar” and “TextRecognition.jar” in my submission.



Verification and Output

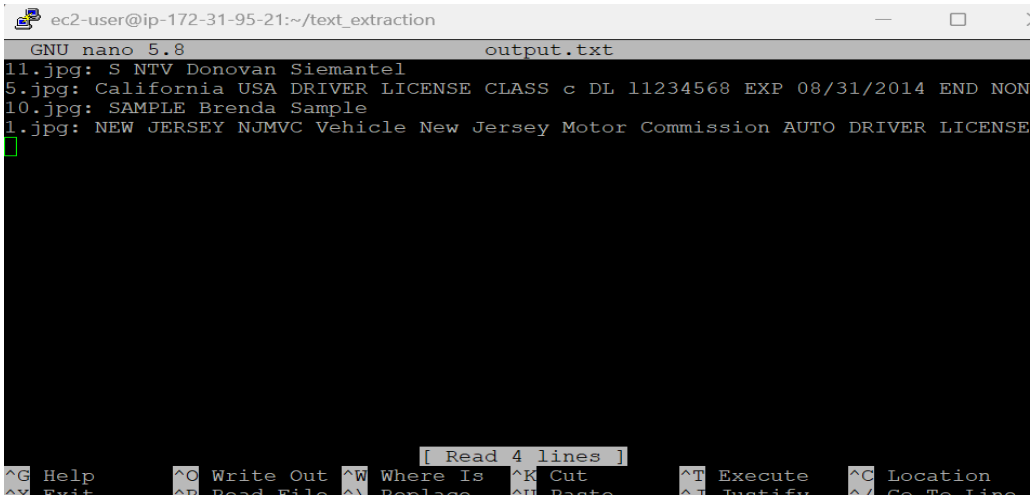
Monitor Execution:

- 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.



Verify Output:

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



- 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.