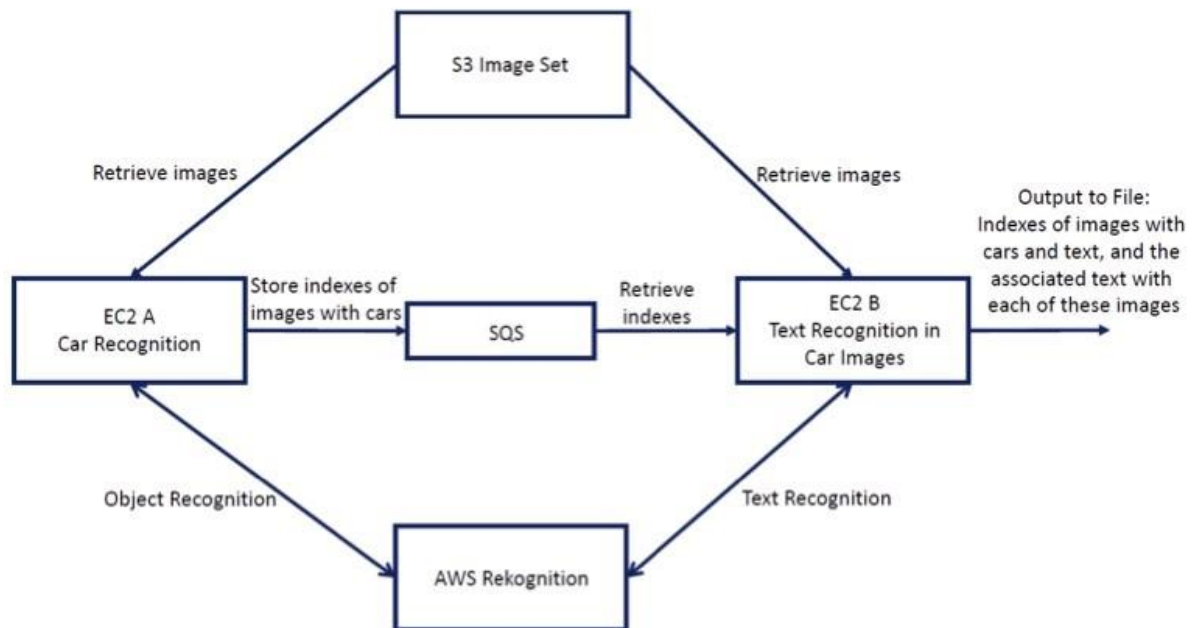


CS643 CLOUD PROGRAMMING ASSIGNMENT 1

Goal: The purpose of this individual assignment is to learn how to use the Amazon AWS cloud platform and how to develop an AWS application that uses existing cloud services. Specifically, you will learn: (1) how to create VMs (EC2 instances) in the cloud; (2) how to use cloud storage (S3) in your applications; (3) how to communicate between VMs using a queue service (SQS); (4) how to program distributed applications in Java on Linux VMs in the cloud; and (5) how to use a machine learning service (AWS Rekognition) in the cloud.

Description: You have to build an image recognition pipeline in AWS, using two EC2 instances, S3, SQS, and Rekognition. The assignment can be done in your preferred language on Amazon Linux VMs. For the rest of the description, you should refer to the figure below:

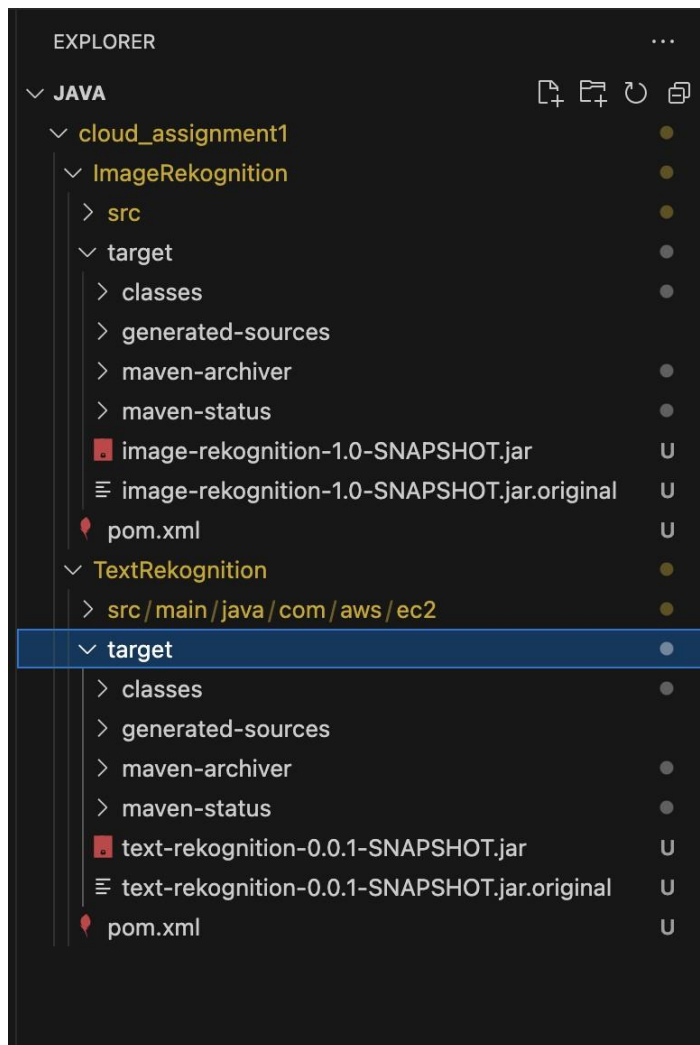


You have to create 2 EC2 instances (EC2 A and B in the figure), with Amazon Linux AMI, that will work in parallel. Each instance will run a Java application. Instance A will read 10 images from an S3 bucket that we created (<https://njit-cs-643.s3.us-east-1.amazonaws.com>) and perform object detection in the images. When a car is detected using Rekognition, with confidence higher than 90%, the index of that image (e.g., 2.jpg) is stored in SQS. Instance B reads indexes of images from SQS as soon as these indexes become available in the queue, and performs text recognition on these images (i.e., downloads them from S3 one by one and uses Rekognition for text recognition). Note that the two instances work in parallel: for example, instance A is processing image 3, while instance B is processing image 1 that was recognized as a car by instance A. When instance A terminates its image processing, it adds index -1 to the queue to signal to instance B that no more indexes will come. When instance B finishes, it prints to a file, in its associated EBS, the indexes of the images that have both cars and text, and also prints the actual text in each image next to its index.

Solution:

The below Image shows the VSCODE SpringBoot Application **project folder structure** of Image and Text recognition that has been written using Amazon SDK.

ImageRecognition and **TextRecognition** has been built separately into individual projects and generated two separate snapshots of JAR files to upload to the cloud. Pom.xml consists of all the dependencies required for the project.



These projects contain code to recognize images and text using AWS tools. They also include code for communicating via SQS, a messaging service in AWS. This communication logic is intertwined with the image and text recognition code because they need to work together on the EC2 instances for everything to function simultaneously.

The below Image shows the maven build of the **ImageRecognition** project.

```
2.jar (821 kB at 693 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/com/amazonaws/aws-java-sdk-iam/1.11.792/aws-java-sdk-iam-1.11.792.jar (1.4 MB at 1.2 M B/s)
[INFO]
[INFO] --- clean:3.1.0:clean (default-clean) @ image-rekognition ---
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/2.0.4/plexus-utils-2.0.4.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/2.0.4/plexus-utils-2.0.4.jar (222 kB at 11 MB/s)
[INFO]
[INFO] --- resources:3.1.0:resources (default-resources) @ image-rekognition ---
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.1.0/plexus-utils-3.1.0.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.1.0/plexus-utils-3.1.0.pom (4.7 kB at 311 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-filtering/3.1.1/maven-filtering-3.1.1.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-filtering/3.1.1/maven-filtering-3.1.1.pom (5.7 kB at 631 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.0.24/plexus-utils-3.0.24.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/3.0.24/plexus-utils-3.0.24.pom (4.1 kB at 500 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-toolchain/2.2.1/maven-toolchain-2.2.1.jar (38 kB at 373 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/com/thoughtworks/qdox/qdox/2.0-M8/qdox-2.0-M8.jar (316 kB at 2.3 MB/s)
[INFO] No tests to run.
[INFO]
[INFO] --- jar:3.2.0:jar (default-jar) @ image-rekognition ---
[INFO] Building jar: /Users/gurudata/Java/cloud_assignment1/ImageRekognition/target/image-rekognition-1.0-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot:2.3.4.RELEASE:repackage (repackage) @ image-rekognition ---
Downloading from central: https://repo.maven.apache.org/maven2/org/springframework/boot/spring-framework-boot/spring-boot-buildpack-platform/2.3.4.RELEASE/spring-boot-buildpack-platform-2.3.4.RELEASE.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/springframework/boot/spring-framework-boot/spring-boot-buildpack-platform/2.3.4.RELEASE/spring-boot-buildpack-platform-2.3.4.RELEASE.pom (3.1 kB at 256 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/com/fasterxml/jackson/module/jackson-module-parameter-names/2.11.2/jackson-module-parameter-names-2.11.2.pom
Downloaded from central: https://repo.maven.apache.org/maven2/com/fasterxml/jackson/module/jackson-module-parameter-names/2.11.2/jackson-module-parameter-names-2.11.2.pom
Downloaded from central: https://repo.maven.apache.org/maven2/com/fasterxml/jackson/module/jackson-module-parameter-names/2.11.2/jackson-module-parameter-names-2.11.2.jar (183 kB at 6.8 MB/s)
● Downloading from central: https://repo.maven.apache.org/maven2/com/fasterxml/jackson/module/jackson-module-parameter-names/2.11.2/jackson-module-parameter-names-2.11.2.jar
● ameter-names-2.11.2.jar
● Downloading from central: https://repo.maven.apache.org/maven2/net/java/dev/jna/jna-platform/5.5.0/jna-platform-5.5.0.jar
Downloaded from central: https://repo.maven.apache.org/maven2/net/java/dev/jna/jna-platform/5.5.0/jna-platform-5.5.0.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/springframework/boot/spring-framework-boot/spring-boot-loader-tools/2.3.4.RELEASE/spring-boot-loader-tools-2.3.4.RELEASE.jar (242 kB at 7.1 MB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/eclipse/sisu/org.eclipse.sisu.inject/0.3.4/org.eclipse.sisu.inject-0.3.4.jar (9.3 kB at 927 kB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-plugin-api/3.6.3/maven-plugin-api-3.6.3.jar (9.6 MB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-utils/3.1.0/maven-shared-utils-3.1.0.jar (262 kB at 4.7 MB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-classworlds/2.6.0/plexus-classworlds-2.6.0.jar (1.5 MB at 9.3 MB/s)
Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-classworlds/2.6.0/plexus-classworlds-2.6.0.jar (1.5 MB at 9.3 MB/s)
[INFO] Replacing main artifact with repackaged archive
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 7.077 s
[INFO] Finished at: 2024-03-02T18:38:54-05:00
[INFO]
```

The below Image shows the maven build of the **TextRekognition** project.

```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS CODE REFERENCE LOG zsh - ImageRekognition + v
gurudatta@Gurudattas-MBP cloud_assignment1 % cd TextRekognition
gurudatta@Gurudattas-MBP TextRekognition % mvn clean package
[INFO] Scanning for projects...
[INFO]
[INFO] -----< com.aws.ec2:text-rekognition >-----
[INFO] Building TextRekognition 0.0.1-SNAPSHOT
[INFO] from pom.xml
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- clean:3.1.0:clean (default-clean) @ text-rekognition ---
[INFO]
[INFO] --- resources:3.1.0:resources (default-resources) @ text-rekognition ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/gurudatta/Java/cloud_assignment1/TextRekognition/src/main/resources
[INFO] skip non existing resourceDirectory /Users/gurudatta/Java/cloud_assignment1/TextRekognition/src/main/resources
[INFO]
[INFO] --- compiler:3.8.1:compile (default-compile) @ text-rekognition ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to /Users/gurudatta/Java/cloud_assignment1/TextRekognition/target/classes
[INFO]
[INFO] --- resources:3.1.0:testResources (default-testResources) @ text-rekognition ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/gurudatta/Java/cloud_assignment1/TextRekognition/src/test/resources
[INFO]
[INFO] --- compiler:3.8.1:testCompile (default-testCompile) @ text-rekognition ---
[INFO] No sources to compile
[INFO]
[INFO] --- surefire:2.22.2:test (default-test) @ text-rekognition ---
[INFO] No tests to run.
[INFO]
[INFO] --- jar:3.2.0:jar (default-jar) @ text-rekognition ---
[INFO] Building jar: /Users/gurudatta/Java/cloud_assignment1/TextRekognition/target/text-rekognition-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot:2.3.4.RELEASE:repackage (repackage) @ text-rekognition ---
[INFO] Replacing main artifact with repackaged archive
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 1.143 s
[INFO] Finished at: 2024-03-02T18:39:04-05:00
[INFO]

```

STEPS:

Log into AWS Student Account.

Start Lab in the vocareum and click on the green signal of AWS button to start the console.

This will automatically head over to the AWS Management console

Process of Creating EC2 Instances:

Navigate to EC2 Dashboard: From the services menu, select EC2 to navigate to the EC2 Dashboard.

Launch Instance: Click on the "Launch Instance" button to start the process of creating a new EC2 instance.

Choose Linux AMI: In the "Choose an Amazon Machine Image (AMI)" step, select "Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type". This is a stable and widely used AMI for general-purpose computing.

Choose default Instance Type: Select the instance type to be "t2.micro". This is a low-cost, general-purpose instance type suitable for small applications and testing purposes. **Select**

Key Pair: Create a new key-pair. This key pair will be used to securely connect to your instance via SSH from a local machine.

Configure Security Group: Create a new security group with the following settings:

Allow SSH traffic (port 22) from your IP address (select "My IP").

Allow HTTP traffic (port 80) from the internet.

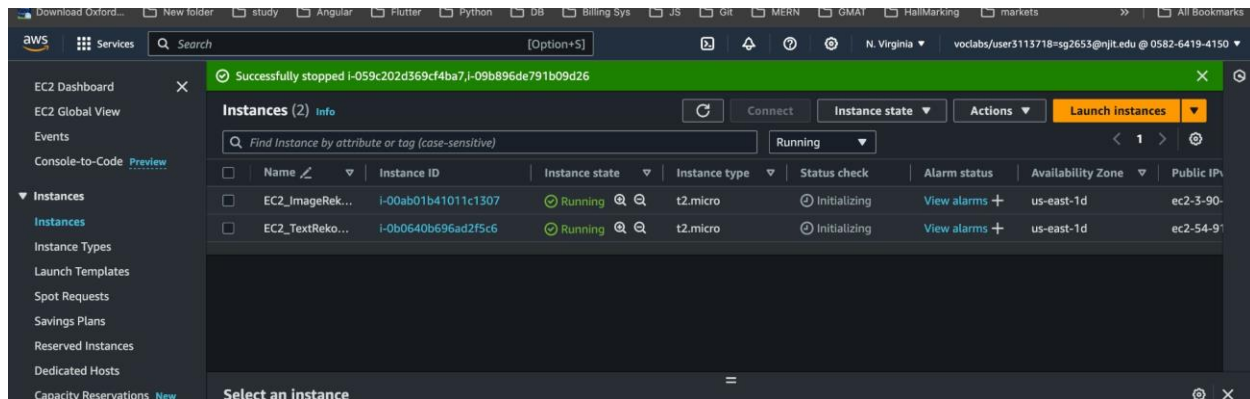
Allow HTTPS traffic (port 443) from the internet.

Advanced Options -> IAM Role: Select the LabInstanceRole in the dropdown of AWS Student account.

Launch Instance: Click on the "Launch" button to launch the instance after ensuring all the correct configurations are made.

The image below shows that I successfully created two instances with the same above setup configuration.

Instances -> **EC2_ImageRekognition** and **EC2_TextRekognition**



Accessing Instance: Once the instance is launched, you can access it using SSH. Use the public IP address or public DNS provided by AWS to connect to your instance.

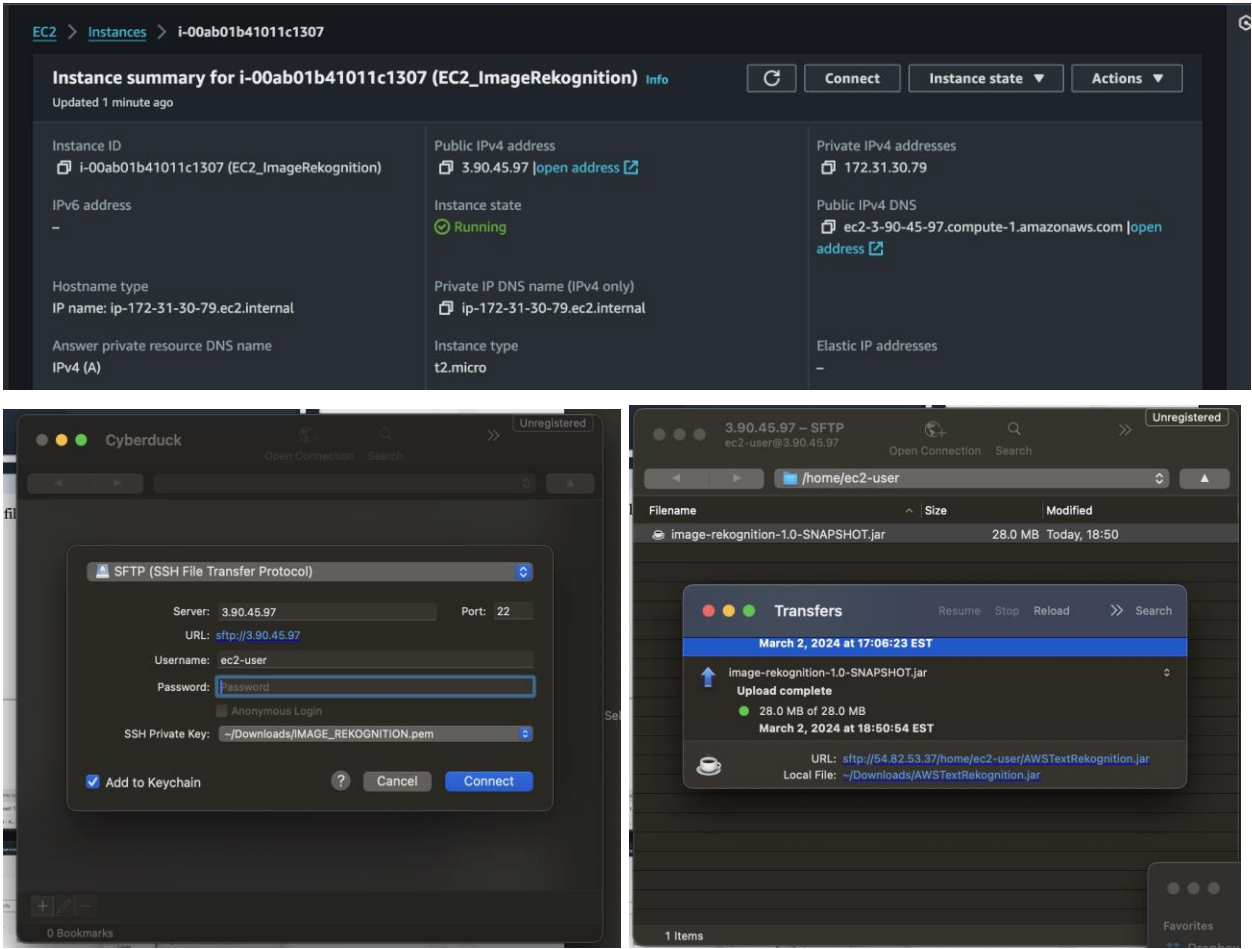
I used CyberDuck to connect to SSH because I am using mac.

Select SFTP which is a file transfer protocol to upload to EC2 via cyberduck.

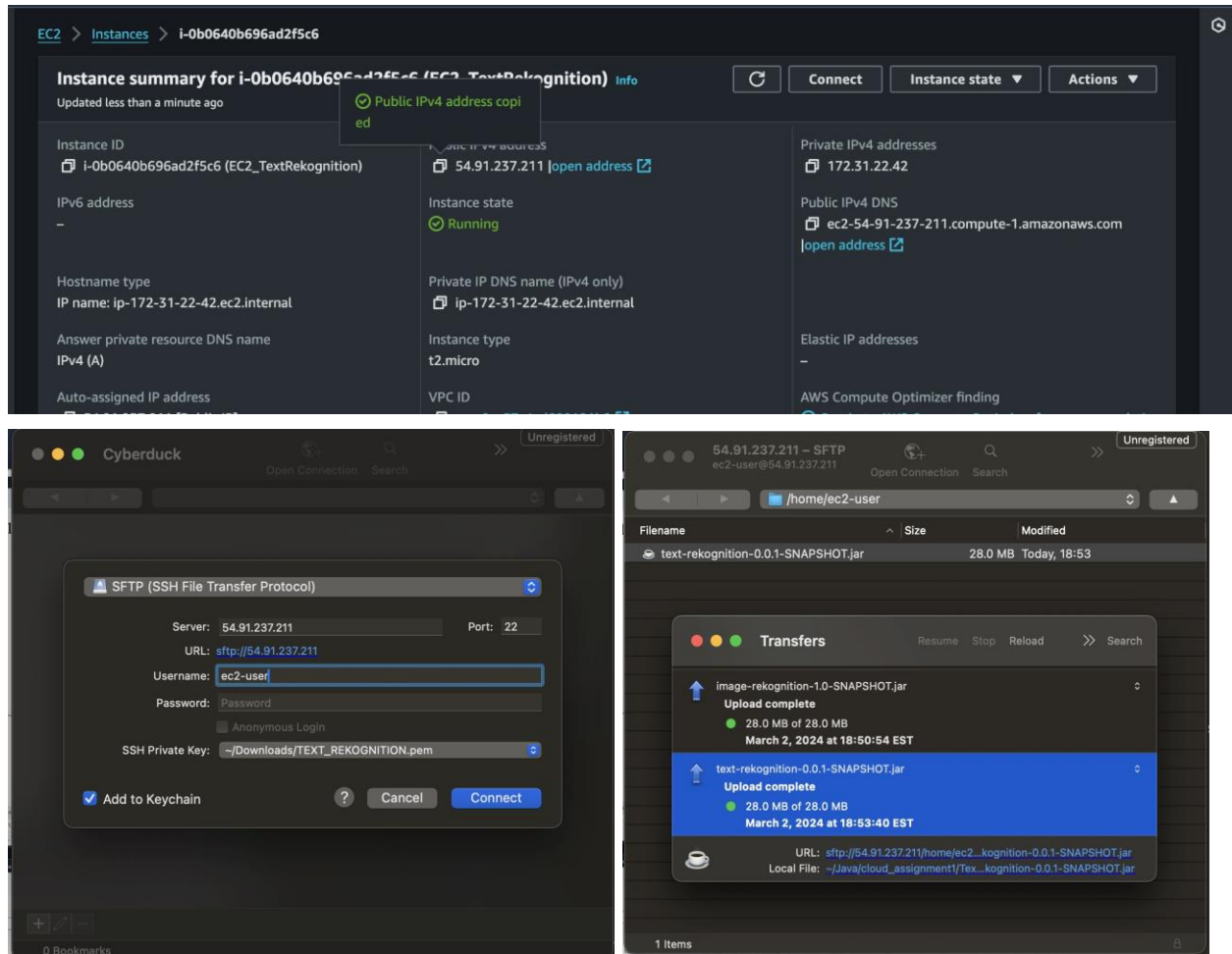
Enter Public DNS and then attach the .pem file of the respective EC2 wanted to connect.

Once connected , drag and drop the JAR files of the projects separately to respective EC2s.

The below Image shows the upload of ImageRekognition JAR to EC2_ImageRekognition



The below Image shows the upload of TextRekognition JAR to EC2_TextRekognition



Connecting SSH: The below steps to be done to both EC2s created

Headover to the EC2 and click on connect, and select SSH Client Tab.

Open the Terminal or Command line in the local system and traverse to the path where .pem files are located.

Then change the permissions of .pem file using the below command

```
chmod 400 "IMAGE_REKOGNITION".pem
```

Next to connect to SSH Client, enter the below command and type yes to connect ssh -i

```
"IMAGE_REKOGNITION.pem" ec2-user@ec2-3-90-45-97.compute-1.amazonaws.com
```

The addresses change for different EC2s and the same action needs to be done to connect two EC2s that were created.

After connecting to each EC2,

Type in the command -> **ls** to see whether the uploaded jar file is present or not

To Install Java in linux, execute the below 2 commands

```
sudo amazon-linux-extras install java-openjdk11 -y  
sudo yum install java-1.8.0-openjdk -y
```

To check if java is successfully installed -> java -version

In the Vocareum, our AWS credentials are present in the AWS Details Tab.

It is required to create aws credentials file and can be done using the below commands

To create directory -> mkdir home/ec2-user/.aws

To create a file -> touch home/ec2-user/.aws/credentials

To write in the credentials -> vi ~/.aws/credentials -> Paste the credentials and press ESC key and enter :wq to save the file

EC2_ImageRekognition SSH Connection Images:


EC2 Instance Connect

Session Manager


SSH client


EC2 serial console

Instance ID


 i-00ab01b41011c1307 (EC2_ImageRekognition)


1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is IMAGE_REKOGNITION.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.

 `chmod 400 "IMAGE_REKOGNITION.pem"`
4. Connect to your instance using its Public DNS:

 `ec2-3-90-45-97.compute-1.amazonaws.com`

Example:

 `ssh -i "IMAGE_REKOGNITION.pem" ec2-user@ec2-3-90-45-97.compute-1.amazonaws.com`

 **Note:** In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

```

Downloads — ec2-user@ip-172-31-30-79:~ — ssh -i IMAGE_REKOGNITION.pem ec2-user@ec2-3-90-45-97.compute-1.amazonaws.com
Last login: Sat Mar  2 17:07:02 on ttys005
gurudatta@Gurudattas-MBP ~ % cd Downloads
gurudatta@Gurudattas-MBP Downloads % chmod 400 "IMAGE_REKOGNITION.pem"
gurudatta@Gurudattas-MBP Downloads % ssh -i "IMAGE_REKOGNITION.pem" ec2-user@ec2-3-90-45-97.compute-1.amazonaws.com
The authenticity of host 'ec2-3-90-45-97.compute-1.amazonaws.com (3.90.45.97)' can't be established.
ED25519 key fingerprint is SHA256:FLyPeOWGCF6s2GzIBsPp7FC7wq9j5LzzNEsFz88hp4Q.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-3-90-45-97.compute-1.amazonaws.com' (ED25519) to the list of known hosts.

      #_
     ~\  #####      Amazon Linux 2
    ~ ~ \#####\
    ~ ~  \###|
    ~ ~   \#/
    ~ ~    V~'  -->
    ~ ~
    ~ ~ .-.-
    ~ ~ _/_/_/_/_/
    ~ ~ _/m/'

A newer version of Amazon Linux is available!

Amazon Linux 2023, GA and supported until 2028-03-15.
https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@ip-172-31-30-79 ~]$ ls
image-rekognition-1.0-SNAPSHOT.jar
[ec2-user@ip-172-31-30-79 ~]$
  
```

```

1-0 p1xman:K00_04 010.0410 1.0m2n2.012
P [
Complete!
[ec2-user@ip-172-31-30-79 ~]$ java -version
openjdk version "1.8.0_402"
OpenJDK Runtime Environment (build 1.8.0_402-b06)
OpenJDK 64-Bit Server VM (build 25.402-b06, mixed mode)
[ec2-user@ip-172-31-30-79 ~]$

[ec2-user@ip-172-31-30-79 .aws]$ ls
credentials
[ec2-user@ip-172-31-30-79 .aws]$ cd ..
[ec2-user@ip-172-31-30-79 ~]$ ls
image-rekognition-1.0-SNAPSHOT.jar
[ec2-user@ip-172-31-30-79 ~]$ ls ~/.aws
credentials
[ec2-user@ip-172-31-30-79 ~]$ cat ~/.aws/credentials
[ec2-user@ip-172-31-30-79 ~]$ vi ~/.aws/credentials
[ec2-user@ip-172-31-30-79 ~]$ cat ~/.aws/credentials
[default]
aws_access_key_id=ASIAQ3EGRBRTKQ6JHUJJ
aws_secret_access_key=y6JzUKuaR3E71qRzoljaZgy6tqcM/1onkyg6V6Bm
aws_session_token=FwoGZXIvYXdzEGUaDF0sC2kZ7ZtOSQueHyLAA0IRWwraAuBNEuKdbMumhwfIFHBvwIKi4j4jAMN4u/PMdFTfP
CIPnXn8sVpR0kLfI9fBn9fX0JznKgQQafiuRoUzgW3Q39D3K1xHwsHxBEPo9kUTOHwC13CM6jHwRT00PVJ1y0JJBFHbi0jYkyDvKPrpVs
r7Di6DJvRPDIETwh8Rt/UA3+Wf5WjZMy8i61l0Vty83uJN87RQpXuf/5WkKVaGDLH7/h4LEfe0MX9/7e9bCyD0qG4QnPE5YA9oBNCyi09
I2vBjIt6klgAUPgK1vLYIWISL1qSYki0G2k6Fob2baYI5WWGG+6EZe6xPUjk8dF40gl
[ec2-user@ip-172-31-30-79 ~]$

```

EC2_TextRecognition SSH Connection Images:


EC2 Instance Connect

Session Manager


SSH client


EC2 serial console

Instance ID


 i-0b0640b696ad2f5c6 (EC2_TextRecognition)


1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is TEXT_REKOGNITION.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.


```
 chmod 400 "TEXT_REKOGNITION.pem"
```
4. Connect to your instance using its Public DNS:


```
 ec2-54-91-237-211.compute-1.amazonaws.com
```

Example:

```
 ssh -i "TEXT_REKOGNITION.pem" ec2-user@ec2-54-91-237-211.compute-1.amazonaws.com
```

 **Note:** In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

```

Downloads — ec2-user@ip-172-31-22-42:~ — ssh -i TEXT_REKOGNITION.pem ec2-user@ec2-54...
Last login: Sat Mar  2 18:56:36 on ttys006
gurudatta@Gurudattas-MBP ~ % cd Downloads
gurudatta@Gurudattas-MBP Downloads % chmod 400 "TEXT_REKOGNITION.pem"
gurudatta@Gurudattas-MBP Downloads % ssh -i "TEXT_REKOGNITION.pem" ec2-user@ec2-54-91-237-211.compute-1.amazonaws.com
The authenticity of host 'ec2-54-91-237-211.compute-1.amazonaws.com (54.91.237.211)' can't be established.
ED25519 key fingerprint is SHA256:CqLYY0nVby+KQw4iaPU/fb1FQ9uxANSyUZr5c607xiE.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-91-237-211.compute-1.amazonaws.com' (ED25519) to the list of known hosts.

#_
~\_#####_ Amazon Linux 2
~~\_#####\
~~\###| AL2 End of Life is 2025-06-30.
~~\#/
~~\#/_-->
~~~V~' '---
~~~_/_/_/_/_/
~~~_/_/_/_/_/
_/_/_/_/_/_/

A newer version of Amazon Linux is available!

Amazon Linux 2023, GA and supported until 2028-03-15.
https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@ip-172-31-22-42 ~]$ ls
text-rekognition-0.0.1-SNAPSHOT.jar
[ec2-user@ip-172-31-22-42 ~]$
  
```

```
Complete!
[ec2-user@ip-172-31-22-42 .aws]$ ls
credentials
[ec2-user@ip-172-31-22-42 .aws]$ cd ..
[ec2-user@ip-172-31-22-42 ~]$ java -version
openjdk version "1.8.0_402"
OpenJDK Runtime Environment (build 1.8.0_402-b06)
OpenJDK 64-Bit Server VM (build 25.402-b06, mixed mode)
[ec2-user@ip-172-31-22-42 ~]$
```

```
Downloads — ec2-user@ip-172-31-22-42:~/aws — ssh -i TEXT_REKOGNITION.pem ec2-user@ec...
hed.
ED25519 key fingerprint is SHA256:CqLYY0nVby+KQw4iaPU/fb1FQ9uxANSyUZr5c607xiE.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-54-91-237-211.compute-1.amazonaws.com' (ED25519) to the list of known hosts.

#_
~\_#####_ Amazon Linux 2
~~~\_#####\_
~~~\_###| AL2 End of Life is 2025-06-30.
~~~\_#/
~~~\_V~'--->
~~~~\_./\_/_/
~~~~\_./\_/_/ A newer version of Amazon Linux is available!
~~~~\_./\_/_/ Amazon Linux 2023, GA and supported until 2028-03-15.
~~~~\_./\_/_/ https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@ip-172-31-22-42 ~]$ ls
text-rekognition-0.0.1-SNAPSHOT.jar
[ec2-user@ip-172-31-22-42 ~]$ mkdir ~/.aws
[ec2-user@ip-172-31-22-42 ~]$ cd .aws
[ec2-user@ip-172-31-22-42 .aws]$ touch credentials
[ec2-user@ip-172-31-22-42 .aws]$ ls
credentials
[ec2-user@ip-172-31-22-42 .aws]$ vi ~/.aws/credentials
[ec2-user@ip-172-31-22-42 .aws]$ cat ~/.aws/credentials
[default]
aws_access_key_id=ASIAQ3EGRBRTKQ6JHUJJ
aws_secret_access_key=y6JzUKuaR3E71qRzojjaZgy6tqcM/1onkyg6V6Bm
aws_session_token=FwoGZXIvYXdzEGUaDF0sC2kZ7ZtOSQueHyLAA0IRWwrrwaAuBNEuKdbMumhWfIFHBvwIKi4j4jAMN4u/PMdF
TfPCIPnXn8sVpR0klfI9fBn9fX0JznKgQqafiuRoUzgW3Q39D3K1xHwShxBEPo9kUTOHwC13CM6jHwRT00PVJ1yOJJ8fHbi0jYkyDv
KPrpVsr7Di6DJvRPDIETwh8Rt/UA3+Wf5WjZMy8i61l0Vty83uJN87RQpXuf/5WKKVaGDLH7/h4LEfe0MX9/7e9bCyd0qG4QnPE5YA
9oBNCyi09I2vBjIt6klgAUPgKlvLYIWISL1qSYki0G2k6Fob2baYI5WWGG+6EZe6xPUjk8dF40gl
[ec2-user@ip-172-31-22-42 .aws]$
```

Execution of JAR Files:

EC2_ImageRekognition:

- This component is responsible for processing images using Amazon Rekognition.
- It starts by fetching car images from a public S3 bucket (<https://njit-cs-643.s3.us-east-1.amazonaws.com>).
- Once an image is fetched, it performs object recognition on the image using Amazon Rekognition.
- After processing, it prints the list of objects detected in the image.
- Finally, it pushes the processed image and its results to an SQS Queue.

EC2_TextRekognition:

- This component is responsible for processing text from car images.

- It listens to the SQS Queue where the processed images are pushed by EC2_ImageRekognition.
- When an image is received from the queue, it fetches the respective image.
- Then, it performs text recognition on the image using Amazon Rekognition.
- After processing, it recognizes the text present in the image and prints the output.

SQS Queue: The SQS Queue is used as a mechanism for communication between EC2_ImageRekognition and EC2_TextRekognition, allowing the latter to know when new images are ready for text recognition.

Workflow in Below Images:

Started by executing EC2_ImageRekognition to process the car images and push the results to the SQS Queue.

Next, executed EC2_TextRekognition to listen to the SQS Queue and process the images for text recognition.

EC2_TextRekognition fetches the processed images and recognizes the text present in the images.

Finally, it prints the output of the text recognition process.

Commands to Execute JAR files:

First : Run ImageRekognition JAR in EC2_ImageRekognition instance

-> java -jar image-rekognition-1.0-SNAPSHOT.jar

Second : Run TextRekognition JAR in EC2_TextRekognition instance

-> java -jar text-rekognition-1.0-SNAPSHOT.jar

EXECUTION OUTPUT in SSH Client:

The below Images show the output after execution of JAR files using above commands.

[illegible]

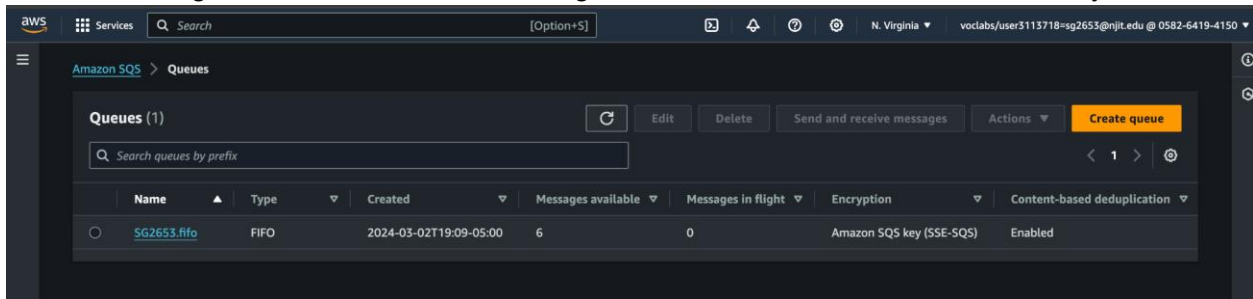
Once the EC2 ImageRekognition is executed, it creates a Queue from code.

I have created a **SG2653.fifo** queue in code and this created a queue automatically in AWS and uses it.

```
public class ImageRecognition {
    Run | Debug
    public static void main(String[] args) throws IOException, JMSEException, InterruptedException {
        SpringApplication.run(ImageRekognition.class, args);
        Regions clientRegion = Regions.US_EAST_1;
        String bucketName = "njit-cs-643";
        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withRegion(clientRegion)
                .build();
            SQSConnectionFactory connectionFactory = new SQSConnectionFactory(new ProviderConfiguration(),
                AmazonSQSClientBuilder.defaultClient());
            SQSConnection connection = connectionFactory.createConnection();
            AmazonSQSMessagingClientWrapper client = connection.getWrappedAmazonSQSClient();
            if (!client.queueExists("SG2653.fifo")) {
                Map<String, String> attributes = new HashMap<String, String>();
                attributes.put(key:"FifoQueue", value:"true");
                attributes.put(key:"ContentBasedDeduplication", value:"true");
                client.createQueue(new CreateQueueRequest().withQueueName("SG2653.fifo").withAttributes(attributes));
            }
            Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
            Queue queue = session.createQueue("SG2653.fifo");
            MessageProducer producer = session.createProducer(queue);
            System.out.println("Validation objects created.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Navigate to services in AWS console and search for SQS, and this shows up that QUEUE is created.

The below image shows **SG2653.fifo** running as it is created from code automatically.



The images below depict the outputs generated by executing the JAR files concurrently, illustrating the communication between EC2_ImageRekognition and EC2_TextRekognition. This communication is facilitated by utilizing an SQS Queue as a pipeline for pushing and fetching processed image data.

[illegible]

The below images shows the output of EC2_ImageRekognition. This process involves recognition during the push to the queue, listing the objects, and detecting labels with the condition that the label is 'car' and the confidence level is above 90%.


```

Downloads — ec2-user@ip-172-31-30-79:~ — ssh -i IMAGE_REKOGNITION.pem ec2-user@ec2-3-90...

:: Spring Boot :: (v2.3.4.RELEASE)

2024-03-03 00:11:56.800 INFO 9433 --- [main] com.aws.ec2.ImageRekognition : Start
ing ImageRekognition v1.0-SNAPSHOT on ip-172-31-30-79.ec2.internal with PID 9433 (/home/ec2-user/image-re
kognition-1.0-SNAPSHOT.jar started by ec2-user in /home/ec2-user)
2024-03-03 00:11:56.809 INFO 9433 --- [main] com.aws.ec2.ImageRekognition : No ac
tive profile set, falling back to default profiles: default
2024-03-03 00:11:57.105 INFO 9433 --- [main] com.aws.ec2.ImageRekognition : Start
ed ImageRekognition in 1.274 seconds (JVM running for 2.276)
Listing objects...
Detected labels for: 1.jpg => Label: Car ,Confidence: 99.94879
Pushed to SQS.
2024-03-03 00:12:00.686 INFO 9433 --- [main] c.a.s.javamessaging.SQSMessageProducer : Messa
ge sent to SQS with SQS-assigned messageId: e6cf4152-b640-4cc8-a86b-cdd807071ced
JMS Message ID:e6cf4152-b640-4cc8-a86b-cdd807071ced
JMS Message Sequence Number 18884356760477423616
Detected labels for: 2.jpg => Label: Car ,Confidence: 99.703156
Pushed to SQS.
2024-03-03 00:12:01.851 INFO 9433 --- [main] c.a.s.javamessaging.SQSMessageProducer : Messa
ge sent to SQS with SQS-assigned messageId: 05cf5224-f97d-4955-93fc-f30931eab964
JMS Message ID:05cf5224-f97d-4955-93fc-f30931eab964
JMS Message Sequence Number 18884356760686319616
Detected labels for: 4.jpg => Label: Car ,Confidence: 99.47948
Pushed to SQS.
2024-03-03 00:12:02.843 INFO 9433 --- [main] c.a.s.javamessaging.SQSMessageProducer : Messa
ge sent to SQS with SQS-assigned messageId: c0e175e8-f891-493a-b5fa-47d472d88091
JMS Message ID:c0e175e8-f891-493a-b5fa-47d472d88091
JMS Message Sequence Number 18884356760944879616
Detected labels for: 5.jpg => Label: Car ,Confidence: 99.51721
Pushed to SQS.
2024-03-03 00:12:03.271 INFO 9433 --- [main] c.a.s.javamessaging.SQSMessageProducer : Messa
ge sent to SQS with SQS-assigned messageId: 534fb3e0-05e2-4ba2-b9f7-c652918f2f5e
JMS Message ID:534fb3e0-05e2-4ba2-b9f7-c652918f2f5e
JMS Message Sequence Number 18884356761076463616
Detected labels for: 6.jpg => Label: Car ,Confidence: 98.79461
Pushed to SQS.
2024-03-03 00:12:04.025 INFO 9433 --- [main] c.a.s.javamessaging.SQSMessageProducer : Messa
ge sent to SQS with SQS-assigned messageId: dfbb2178-96c1-4d3e-9128-838de890793a
JMS Message ID:dfbb2178-96c1-4d3e-9128-838de890793a
JMS Message Sequence Number 18884356761232879872
Detected labels for: 7.jpg => Label: Car ,Confidence: 99.999916
Pushed to SQS.
2024-03-03 00:12:04.606 INFO 9433 --- [main] c.a.s.javamessaging.SQSMessageProducer : Messa
ge sent to SQS with SQS-assigned messageId: 75bdf672-ea8b-4480-b2ed-f9717e541b9c
JMS Message ID:75bdf672-ea8b-4480-b2ed-f9717e541b9c
JMS Message Sequence Number 18884356761378287616

```

```

Downloads — ec2-user@ip-172-31-30-79:~ — ssh -i IMAGE_REKOGNITION.pem ec2-user@ec2-3-90...
JMS Message Sequence Number 18884356761378287616
*([ec2-user@ip-172-31-30-79 ~]$ java -jar image-rekognition-1.0-SNAPSHOT.jar

:: Spring Boot :: (v2.3.4.RELEASE)

2024-03-03 00:11:56.800 INFO 9433 --- [main] com.aws.ec2.ImageRekognition : Start
ing ImageRekognition v1.0-SNAPSHOT on ip-172-31-30-79.ec2.internal with PID 9433 (/home/ec2-user/image-re
kognition-1.0-SNAPSHOT.jar started by ec2-user in /home/ec2-user)
2024-03-03 00:11:56.809 INFO 9433 --- [main] com.aws.ec2.ImageRekognition : No ac
tive profile set, falling back to default profiles: default
2024-03-03 00:11:57.105 INFO 9433 --- [main] com.aws.ec2.ImageRekognition : Start
ed ImageRekognition in 1.274 seconds (JVM running for 2.276)
Listing objects...
Detected labels for: 1.jpg => Label: Car ,Confidence: 99.94879
Pushed to SQS.
2024-03-03 00:12:00.686 INFO 9433 --- [main] c.a.s.javamessaging.SQSMessageProducer : Messa
ge sent to SQS with SQS-assigned messageId: e6cf4152-b640-4cc8-a86b-cdd807071ced
JMS Message ID:e6cf4152-b640-4cc8-a86b-cdd807071ced
JMS Message Sequence Number 18884356760477423616
Detected labels for: 2.jpg => Label: Car ,Confidence: 99.703156
Pushed to SQS.
2024-03-03 00:12:01.851 INFO 9433 --- [main] c.a.s.javamessaging.SQSMessageProducer : Messa
ge sent to SQS with SQS-assigned messageId: 05cf5224-f97d-4955-93fc-f30931eab964
JMS Message ID:05cf5224-f97d-4955-93fc-f30931eab964
JMS Message Sequence Number 18884356760686319616
Detected labels for: 4.jpg => Label: Car ,Confidence: 99.47948
Pushed to SQS.
2024-03-03 00:12:02.843 INFO 9433 --- [main] c.a.s.javamessaging.SQSMessageProducer : Messa
ge sent to SQS with SQS-assigned messageId: c0e175e8-f891-493a-b5fa-47d472d88091
JMS Message ID:c0e175e8-f891-493a-b5fa-47d472d88091
JMS Message Sequence Number 18884356760944879616
Detected labels for: 5.jpg => Label: Car ,Confidence: 99.51721
Pushed to SQS.
2024-03-03 00:12:03.271 INFO 9433 --- [main] c.a.s.javamessaging.SQSMessageProducer : Messa
ge sent to SQS with SQS-assigned messageId: 534fb3e0-05e2-4ba2-b9f7-c652918f2f5e
JMS Message ID:534fb3e0-05e2-4ba2-b9f7-c652918f2f5e
JMS Message Sequence Number 18884356761076463616
Detected labels for: 6.jpg => Label: Car ,Confidence: 98.79461
Pushed to SQS.
2024-03-03 00:12:04.025 INFO 9433 --- [main] c.a.s.javamessaging.SQSMessageProducer : Messa
ge sent to SQS with SQS-assigned messageId: dfbb2178-96c1-4d3e-9128-838de890793a
JMS Message ID:dfbb2178-96c1-4d3e-9128-838de890793a
JMS Message Sequence Number 18884356761232879872
Detected labels for: 7.jpg => Label: Car ,Confidence: 99.999916
Pushed to SQS.
2024-03-03 00:12:04.606 INFO 9433 --- [main] c.a.s.javamessaging.SQSMessageProducer : Messa
ge sent to SQS with SQS-assigned messageId: 75bdf672-ea8b-4480-b2ed-f9717e541b9c
JMS Message ID:75bdf672-ea8b-4480-b2ed-f9717e541b9c
JMS Message Sequence Number 18884356761378287616

```

The Below Image shows the output of EC2_TextRekognition, the indexes of the images that have both cars and text, and also prints the actual text in each image next to its index.

```
Text Detected lines and words for: 7.jpg ==> Text Detected: Lamborghini , Confidence: 97.139915
Text Detected: LP 610 LB , Confidence: 93.50435
Text Detected: BO , Confidence: 78.93724
Text Detected: BWW , Confidence: 11.18303
Text Detected: Lamborghini , Confidence: 97.139915
Text Detected: LP , Confidence: 99.59495
Text Detected: 610 LB , Confidence: 87.41375
Text Detected: BO , Confidence: 78.93724
Text Detected: BWW , Confidence: 11.18303

[+] Spring Boot !! (v2.1.3.RELEASE)

2024-03-03 00:30:29.524 INFO 16101 --- [ main] com.aws.ec2.ImageRekognition
ted ImageRekognition.v1.0-SNAPSHOT.jar started by ec2-user in /home/ec2-user/Star
reognition-1.0-SNAPSHOT.jar started by ec2-user in /home/ec2-user
2024-03-03 00:30:29.533 INFO 16101 --- [ main] com.aws.ec2.ImageRekognition
active profile set, falling back to default profiles: default
2024-03-03 00:30:29.607 INFO 16101 --- [ main] com.aws.ec2.ImageRekognition
ted ImageRekognition in 1.428 seconds (JVM running for 2.43s)
Listing objects...
Detected labels for: 1.jpg ==> Label: Car , Confidence: 99.94679
Pushed to SQS.
2024-03-03 00:30:33.544 INFO 16101 --- [ main] c.a.s.javamessaging.SQSMessageProducer
age sent to SQS with SQS-assigned messageId: ad7448a4-7467-408b-B692-26b20897bc7
JMS Message ID: ad7448a4-7467-408b-B692-26b20897bc7
JMS Message Sequence Number 1888435708793999516
Detected labels for: 2.jpg ==> Label: Car , Confidence: 79.83156
Pushed to SQS.
2024-03-03 00:30:34.447 INFO 16101 --- [ main] c.a.s.javamessaging.SQSMessageProducer
age sent to SQS with SQS-assigned messageId: 69b52a20-7f49-aae2-bb4c-ab99bee8b824
JMS Message ID: d9b52a20-7f49-aae2-bb4c-ab99bee8b824
JMS Message Sequence Number 1888435708793999516
Detected labels for: 4.jpg ==> Label: Car , Confidence: 99.67948
Pushed to SQS.
2024-03-03 00:30:35.389 INFO 16101 --- [ main] c.a.s.javamessaging.SQSMessageProducer
age sent to SQS with SQS-assigned messageId: 72b464c8-479a-4f1f-b5c2-5f635964b3f5
JMS Message ID: 72b464c8-479a-4f1f-b5c2-5f635964b3f5
JMS Message Sequence Number 1888435708793999516
Detected labels for: 5.jpg ==> Label: Car , Confidence: 99.51721
Pushed to SQS.
2024-03-03 00:30:35.782 INFO 16101 --- [ main] c.a.s.javamessaging.SQSMessageProducer
age sent to SQS with SQS-assigned messageId: 8e9a3b61-2374-494b-b19a-88b8214dc093
JMS Message ID: 8e9a3b61-2374-494b-b19a-88b8214dc093
JMS Message Sequence Number 1888435708793999516
Detected labels for: 6.jpg ==> Label: Car , Confidence: 98.79461
Pushed to SQS.
2024-03-03 00:30:36.437 INFO 16101 --- [ main] c.a.s.javamessaging.SQSMessageProducer
age sent to SQS with SQS-assigned messageId: 6f5c2758-b2c6-471b-be47-7d565c700388
JMS Message ID: 6f5c2758-b2c6-471b-be47-7d565c700388
JMS Message Sequence Number 1888435708793999516
Detected labels for: 7.jpg ==> Label: Car , Confidence: 99.99916
Pushed to SQS.
2024-03-03 00:30:37.007 INFO 16101 --- [ main] c.a.s.javamessaging.SQSMessageProducer
age sent to SQS with SQS-assigned messageId: ea23d678-af06-4278-811b-b9c6cd7819c8
JMS Message ID: ea23d678-af06-4278-811b-b9c6cd7819c8
JMS Message Sequence Number 1888435708793999516

[+] Spring Boot !! (v2.1.3.RELEASE)

2024-03-03 00:30:29.524 INFO 16101 --- [ main] com.aws.ec2.ImageRekognition
ted ImageRekognition.v1.0-SNAPSHOT.jar started by ec2-user in /home/ec2-user/Star
reognition-1.0-SNAPSHOT.jar started by ec2-user in /home/ec2-user
2024-03-03 00:30:29.533 INFO 16101 --- [ main] com.aws.ec2.ImageRekognition
active profile set, falling back to default profiles: default
2024-03-03 00:30:29.607 INFO 16101 --- [ main] com.aws.ec2.ImageRekognition
ted ImageRekognition in 1.428 seconds (JVM running for 2.43s)
Listing objects...
Detected labels for: 1.jpg ==> Label: Car , Confidence: 99.94679
Pushed to SQS.
2024-03-03 00:30:33.544 INFO 16101 --- [ main] c.a.s.javamessaging.SQSMessageProducer
age sent to SQS with SQS-assigned messageId: ad7448a4-7467-408b-B692-26b20897bc7
JMS Message ID: ad7448a4-7467-408b-B692-26b20897bc7
JMS Message Sequence Number 1888435708793999516
Detected labels for: 2.jpg ==> Label: Car , Confidence: 79.83156
Pushed to SQS.
2024-03-03 00:30:34.447 INFO 16101 --- [ main] c.a.s.javamessaging.SQSMessageProducer
age sent to SQS with SQS-assigned messageId: 69b52a20-7f49-aae2-bb4c-ab99bee8b824
JMS Message ID: d9b52a20-7f49-aae2-bb4c-ab99bee8b824
JMS Message Sequence Number 1888435708793999516
Detected labels for: 4.jpg ==> Label: Car , Confidence: 99.67948
Pushed to SQS.
2024-03-03 00:30:35.389 INFO 16101 --- [ main] c.a.s.javamessaging.SQSMessageProducer
age sent to SQS with SQS-assigned messageId: 72b464c8-479a-4f1f-b5c2-5f635964b3f5
JMS Message ID: 72b464c8-479a-4f1f-b5c2-5f635964b3f5
JMS Message Sequence Number 1888435708793999516
Detected labels for: 5.jpg ==> Label: Car , Confidence: 99.51721
Pushed to SQS.
2024-03-03 00:30:35.782 INFO 16101 --- [ main] c.a.s.javamessaging.SQSMessageProducer
age sent to SQS with SQS-assigned messageId: 8e9a3b61-2374-494b-b19a-88b8214dc093
JMS Message ID: 8e9a3b61-2374-494b-b19a-88b8214dc093
JMS Message Sequence Number 1888435708793999516
Detected labels for: 6.jpg ==> Label: Car , Confidence: 98.79461
Pushed to SQS.
2024-03-03 00:30:36.437 INFO 16101 --- [ main] c.a.s.javamessaging.SQSMessageProducer
age sent to SQS with SQS-assigned messageId: 6f5c2758-b2c6-471b-be47-7d565c700388
JMS Message ID: 6f5c2758-b2c6-471b-be47-7d565c700388
JMS Message Sequence Number 1888435708793999516
Detected labels for: 7.jpg ==> Label: Car , Confidence: 99.99916
Pushed to SQS.
2024-03-03 00:30:37.007 INFO 16101 --- [ main] c.a.s.javamessaging.SQSMessageProducer
age sent to SQS with SQS-assigned messageId: ea23d678-af06-4278-811b-b9c6cd7819c8
JMS Message ID: ea23d678-af06-4278-811b-b9c6cd7819c8
JMS Message Sequence Number 1888435708793999516
```

The below Image shows the SQS Queue communication statistics in graph metrics in SQS services in AWS console.

