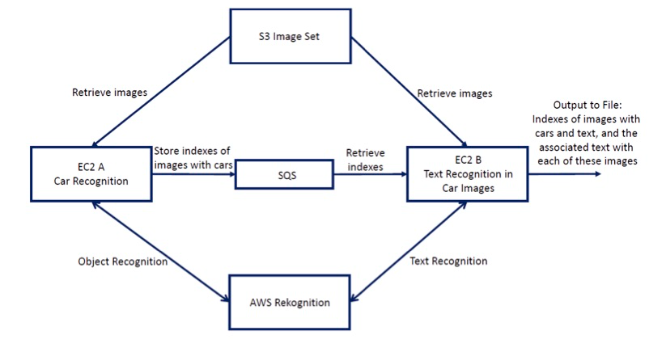
**CS - 643 Introduction to Cloud computing**

**Programming Assignment-1**

The objective of this individual assignment is to gain proficiency in utilizing the Amazon AWS cloud platform and develop an AWS application that leverages various cloud services such as EC2 instances, S3, SQS, and Rekognition.

The assignment entails building an image recognition pipeline utilizing two EC2 instances on Amazon Linux VMs. The process involves using S3 for storage, SQS for communication between instances, and Rekognition for machine learning. The entire development process should be done in Java. Please refer to the diagram below for further details.



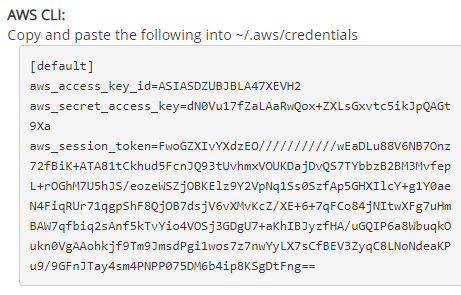
The assignment requires the creation of two EC2 instances (EC2 A and B) using Amazon Linux AMI to work in parallel.

* The first instance (A) will read ten images from an S3 bucket and perform object detection using Rekognition. If a car is detected in an image with a confidence level greater than 90%, the index of that image will be added to the SQS queue.
* The second instance (B) will read the indexes of images from the SQS queue and perform text recognition using Rekognition. It will download the images from S3 one by one and recognize the text. The two instances will work in parallel, and when each one is finished, it will signal the other instance by adding -1 to the queue.

**Implementation**

**Step 1: Accessing the AWS account details through AWS Academy**

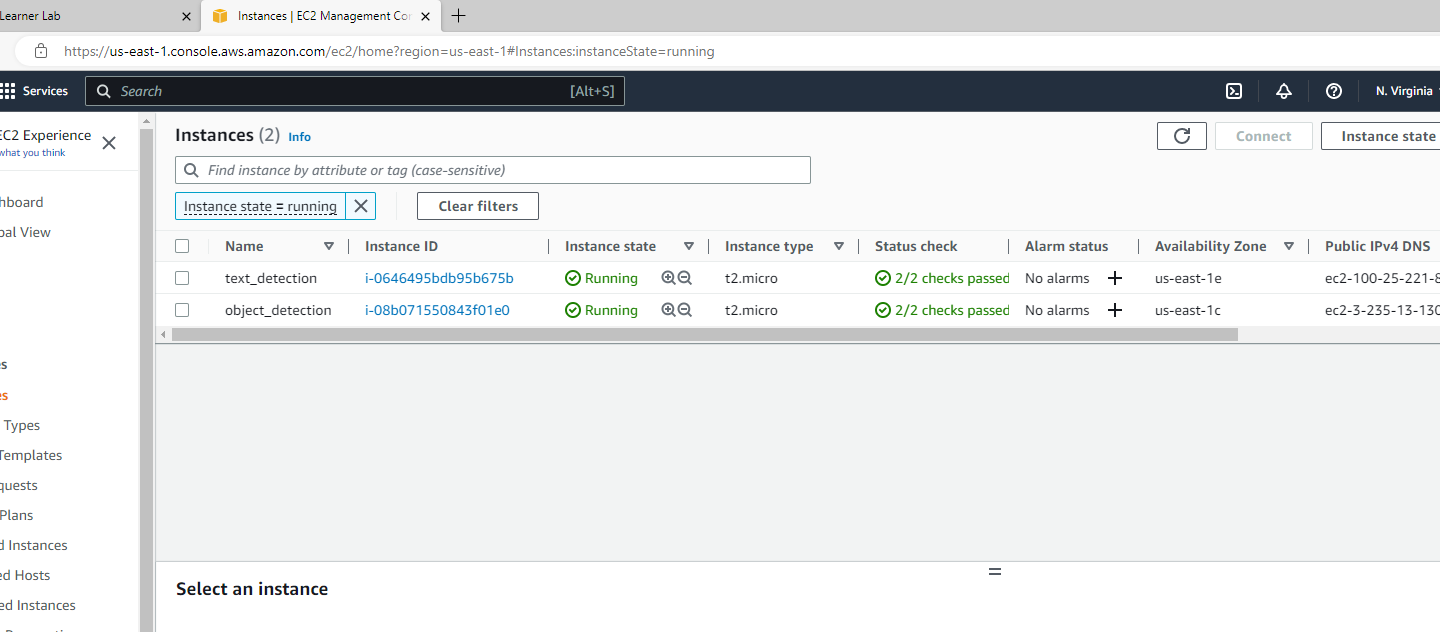
* First, we sign up for the AWS Academy course using the NJIT email by using the invite link.
* Then we proceed to the Learner Lab after logging in to your AWS Lab and begin the AWS session there. Under "AWS details," we now copy the **AWS access\_key, secret\_key, and session\_token.** We must also obtain the **ppk and pem files** from this location. These will be our means of gaining access to the EC2 instances and running the services.

****

**Step 2: Launching and Configuring EC2 instances**

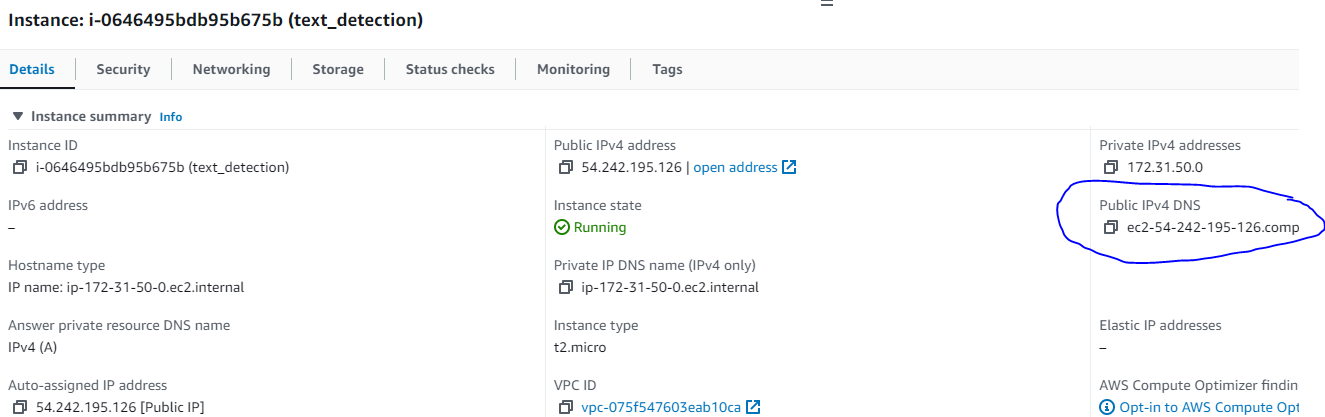
1. We first click on the launch instance and then give this instance a name. **I have used object\_recognition and text\_recognition as the names.**
2. Then we have to select the Amazon Machine image. Here for both instances, **I have used Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type.**
3. Then we need to select the EC2 instance type. For this assignment, I have used **t2.micro** as the instance type as it is a cost-effective and versatile option that offers a standard level of CPU performance.
4. Next, we select key-value pairs which refer to metadata that can be added to instances during their creation. These key-value pairs are used to specify additional information about the instances, such as their owner, purpose, or any other characteristics that could be helpful in identifying or managing them. Here I selected **vockey** as the key-value pair.
5. Then we configure the network settings for EC2 instances. Here, we create a new security group which is basically a set of firewall rules that control the traffic for your instance. Here I have configured both instances to allow incoming **SSH, HTTPS, and HTTP traffic from the internet.**
6. The IP address from which this instance can be accessed is listed in a column called "Source" under this heading: Here we should choose "MY IP" from the drop-down menu.
7. Then once we are done configuring the instances we can **launch them.**

**Then we can head over to EC2 instances and check if the instance state is running. If we encounter any problems here we can stop and start the instance again to ensure it is running smoothly.**

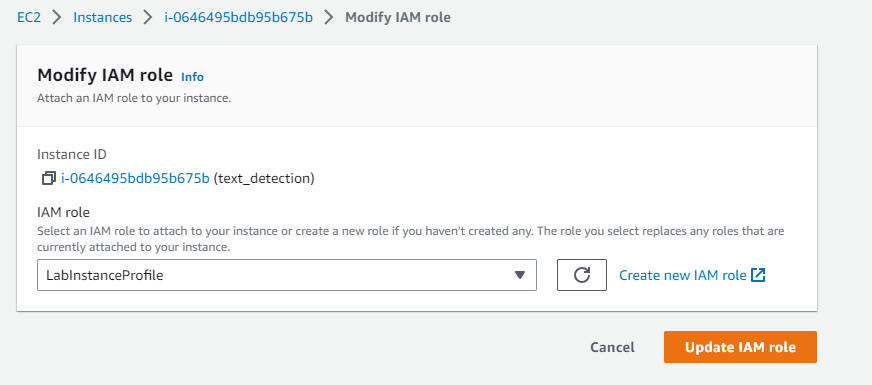


**Fig1. EC2 Instances**

8. Then here we need to copy the Public IPv4 DNS as shown in the picture below, as we will be using this to SSH into our instance.



**Fig2. Public IPv4 DNS for text\_recognition EC2 instance**



**Fig3. Choosing IAM role for EC2 instances**

9. After the EC2 instances are launched, you can select an instance from the list and then navigate to the "Actions" button on the console. From there, you can select the "Security" option and then choose "Modify IAM role" for that instance. Here we can add or change the instance's IAM (Identity and Access Management) role. The instance's permissions and the tasks it can carry out with AWS resources are defined by the IAM role we choose. As shown in the picture above we choose the **“LabInstanceProfile”** Role here.

**STEP 3: Creating JAVA programs and Maven projects**

Now for both text recognition and image recognition, we will need to create two different JAVA programs. The object detection is going to be run on the first EC2 instance while the text detection program will be executed on the second EC2 instance that we have created.

For this, we will need to create an executable. JAR files for both programs. Here. JAR files will be having the compressed version of all the .class files. So the first step here would be to install JAVA and Maven on PC. The main reason to do this is to set up the required AWS SDK. For implementing these programs I have used **JAVA 19.0.2 and Apache Maven 3.9.0 on Windows OS.**

i) Once we are done installing JAVA and Maven we first create a new directory to hold the Maven project

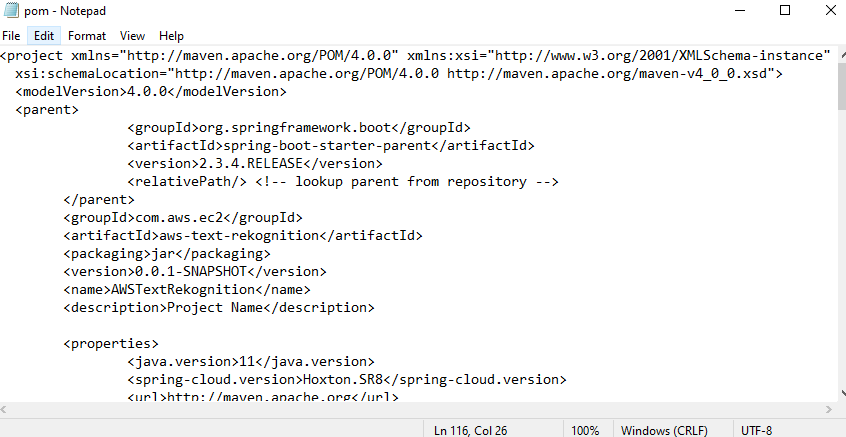
ii) Then from the terminal, we move to this directory to create a new Maven project (Which can be customized based on how we want the project to be configured). On creating the project we can see that Maven creates the project file with the src directory and pom.xml file. We will be using this pom.xml file as our configuration file in which we basically describe the project and its dependencies and src will be holding our . JAVA file.

iii) Then we update the pom.xml and. JAVA file with the required code

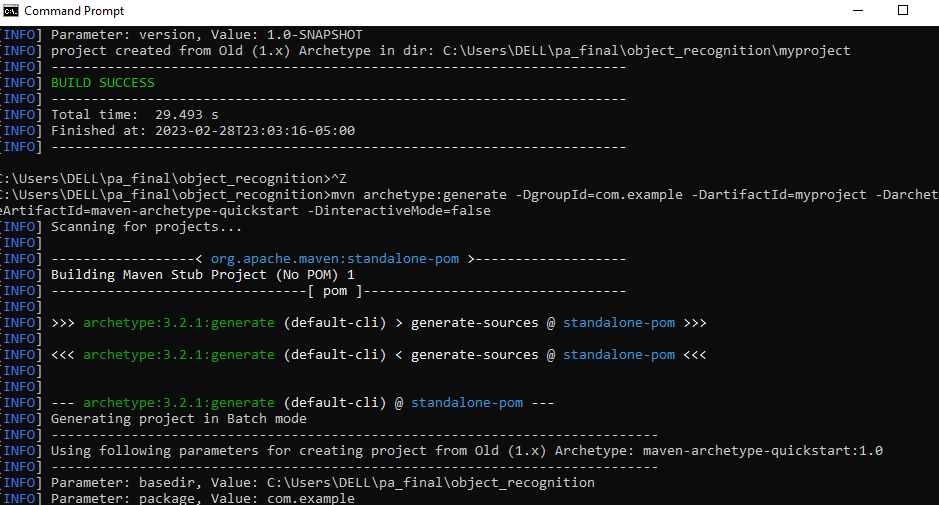
iv) In the final step we will build a maven project from terminal using the command **"mvn clean package"**

Once the project has been built successfully a . JAR file will be created in a new target directory implying that we have successfully built our JAVA application

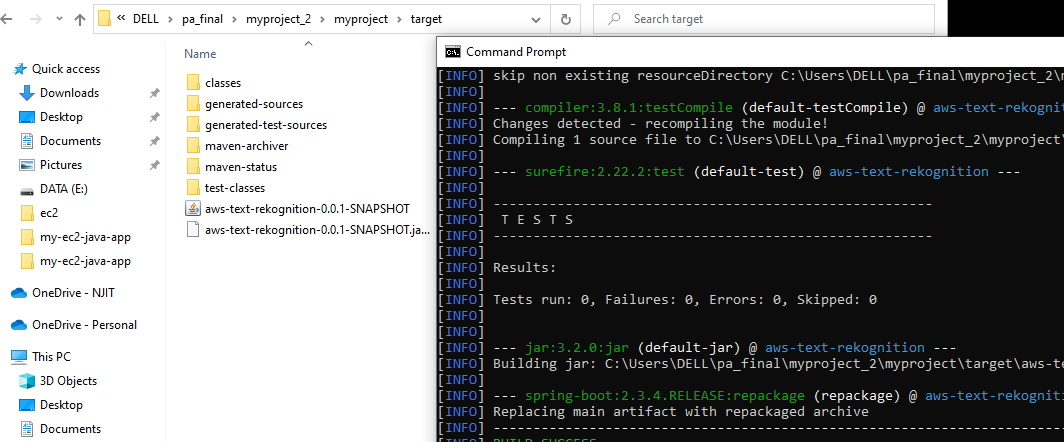
v) We repeat all the above steps for both Object recognition and Text recognition applications so that we get. JAR files to upload to our EC2 instances.



**Fig4. Configuring the pom.xml file**



**Fig5. Creating Maven project from terminal**



**Fig6. JAR file in the Maven project target directory**

**STEP 4: Uploading. JAR files to EC2 instances**

Then we will need to upload these. JAR files into their respective ec2 instances. The way we do it will depend on the operating system that we use.

**As I am working on a windows operating system I have used WinSCP.** It is basically a software tool that allows users to transfer files between their local computer and a remote computer, such as a web or cloud server. It uses the secure SSH protocol for safe file management and supports various file transfer protocols such as SFTP, FTP, SCP, and WebDAV.

Following are the step for transferring .jar files into an EC2 instance,

i) Download and install WinSCP on your local computer.

ii) Open WinSCP and click the "New Site" button to create a new connection.

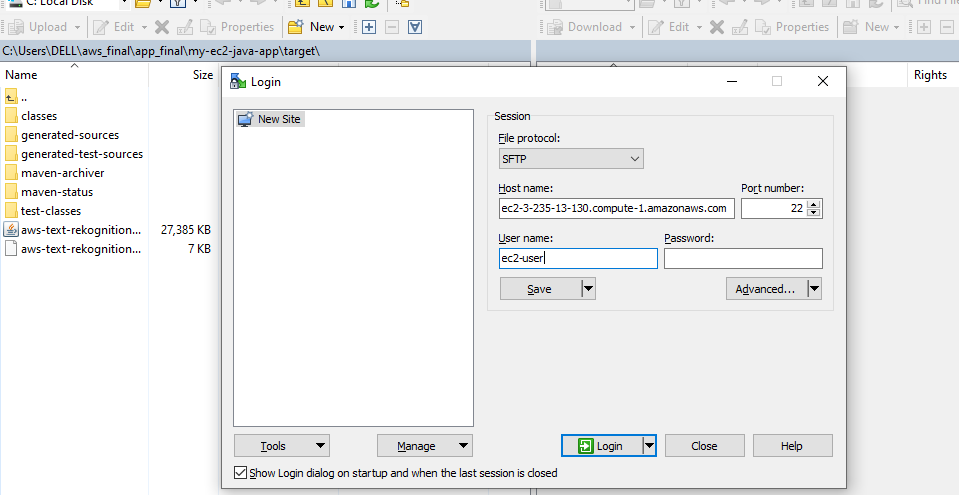
iii) Enter the hostname, username, and password (or key) for the EC2 instance.

iv) Connect to the EC2 instance using WinSCP.

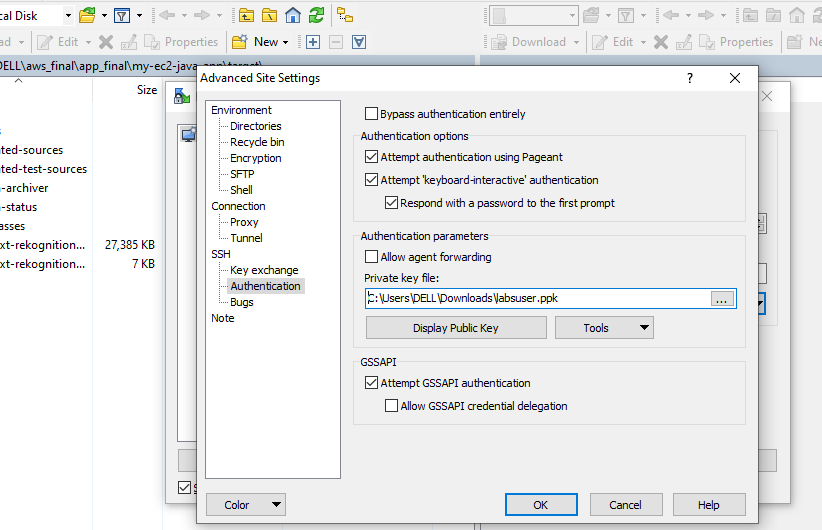
v) Navigate to the destination directory on the EC2 instance where you want to transfer the .jar files.

vi) Drag and drop the .jar files from your local computer to the remote directory on the EC2 instance.

vii) Monitor the file transfer progress in the WinSCP transfer window and wait for it to complete.



**Fig7. Entering hostname and username in WinSCP**



**Fig8. Using .ppk file for authentication in WinSCP**

**STEP 5: SSH into EC2 instances**

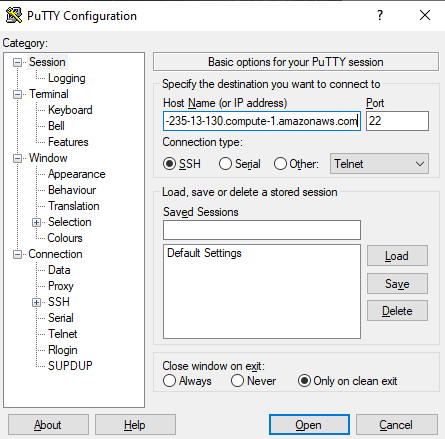
**Here I have used PuTTY to SSH into both EC2 instances.** PuTTY is an open-source software used to remotely access and manage Unix or Linux-based systems, such as servers and cloud instances like Amazon EC2, in the Windows operating system.

It also provides support for a variety of encryption algorithms, as well as session management and public-key authentication.

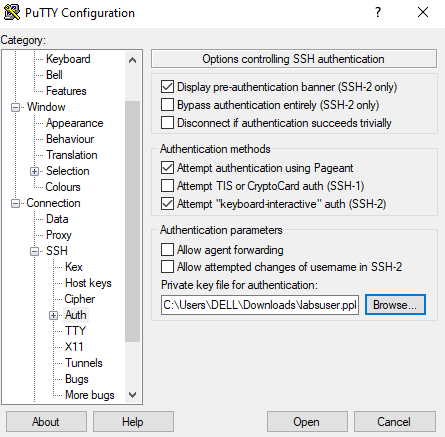
To SSH into an EC2 instance using Putty, follow the steps below:

* First, we need to download and install Putty from the official website.
* Then we have to launch PuTTY and enter the public DNS of your EC2 instance ( We have copied this earlier from EC2 instances ) in the "Host Name" field.
* Next, we head to Connection>SSH>Auth and choose the .ppk file that we had earlier downloaded.
* Then we click on “Open” to start the SSH session
* Now since we are using Amazon Linux AMI for both EC2 instances we need to enter “ec2-user” in the username. Then on clicking Login we will be logged in to our EC2 instance.

**Note:** If we are facing problems while connecting to your EC2 instance using SSH, we need to make sure that the security group linked to the instance has been configured to permit incoming traffic on port 22 (SSH) specifically from our IP address.



**Fig9. Entering hostname, connection type, and port number in PuTTY**



**Fig10. Using .ppk file for SSH authentication in PuTTY**

**STEP 6: Running Object and Text recognition programs**

First, before running the programs on both instances we need to make sure that we have updated the access\_key, secret\_key, and access\_token. For this, we use the details that we initially copied and enter them into the .aws/configure file under the home directory. For this, we can use these commands in the terminal.

**cd ~**

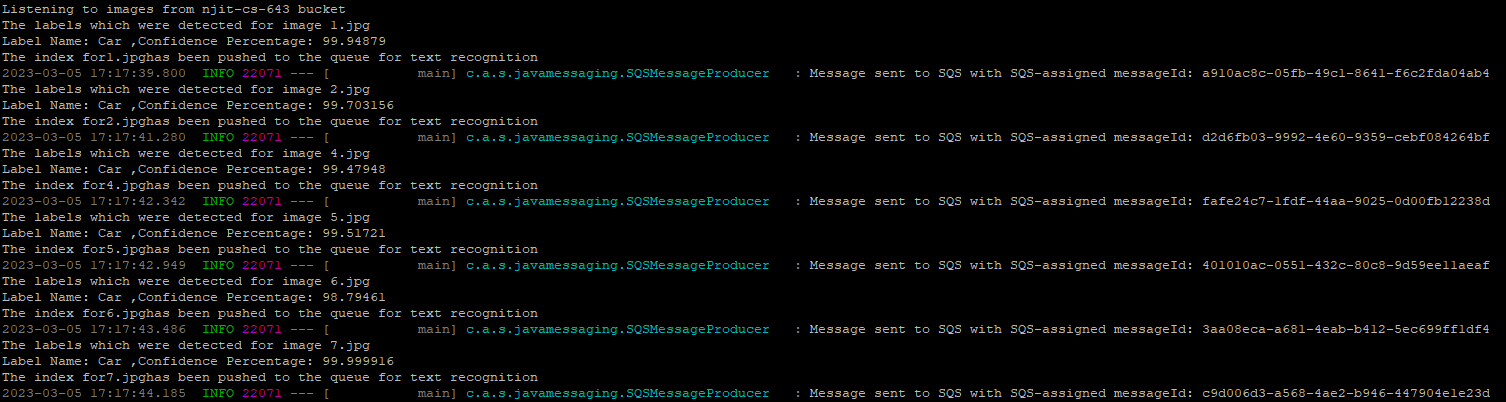
**nano .aws/configure**

Then we also need to make sure that both our machines have JAVA installed in them.

**sudo apt-get install default-jdk**

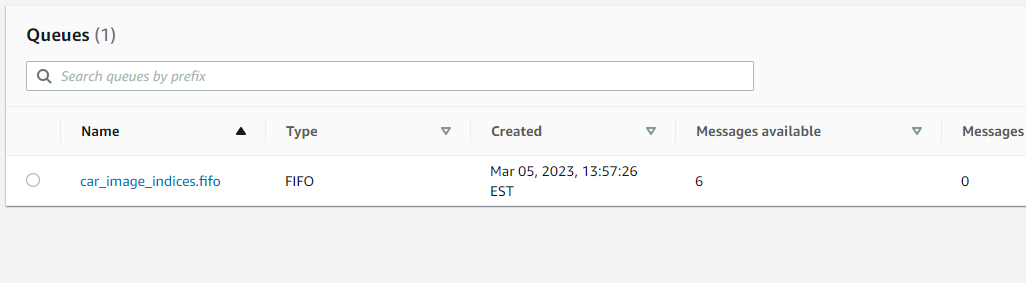
Once we are done with the required setup, we can run the Object recognition file using the command,

**java -jar aws-object-rekognition-0.0.1-SNAPSHOT.jar**



**Fig11. Object Recognition output ( using Amazon Rekognition )**

As we can see from the picture above image, On running the JAR file a Queue named car\_image\_indices.fifo is created and images that have objects with label= Car and Confidence >90% are Pushed into the SQS. From the above output, we can also see the Message ID of each message being pushed into the SQS.



**Fig12. Populated FIFO Queue after running the first EC2 instance**

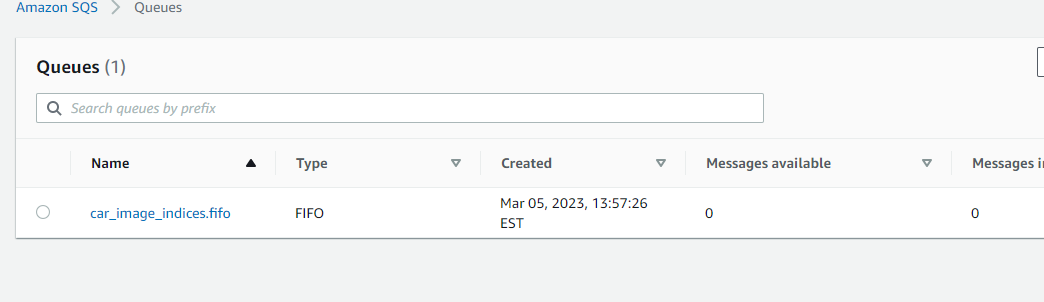
As we can see Amazon Rekognition has detected 6 images that have Cars and confidence greater than 90% ( To be precise img1.jpg, img2.jpg, img4.jpg, img5.jpg, img6.jpg, img7.jpg are the images that were pushed into the queue).

Then we SSH into our second instance and run the Text recognition file,

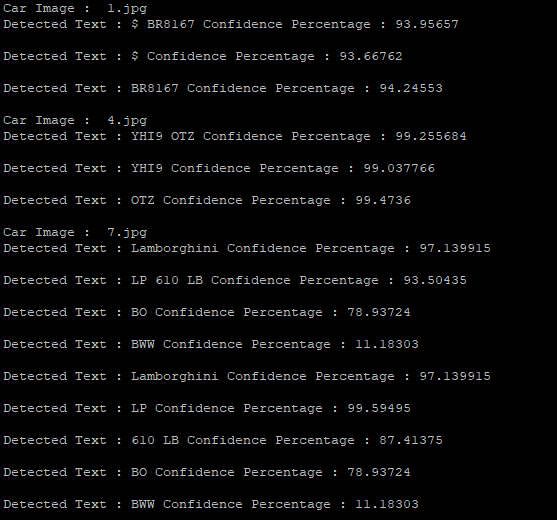
**java -jar aws-text-rekognition-0.0.1-SNAPSHOT.jar**

When the second instance is run, it uses the images from the queue and then prints the indexes of the images that contain both cars and text as well as prints the actual text in each image.

As we can now see from the below image, after running the text recognition file the available messages in the queue become zero as they are retrieved asynchronously by the second instance.



**Fig13. FIFO queue after messages were retrieved by the second EC2 instance**



**Fig14. Text Recognition output**

Now in order to verify if the object and text recognition pipelines are working and are performing as desired I have retrieved the images from the “njit-cs-643” bucket. We can do this in a couple of ways,

**i) Using URI**

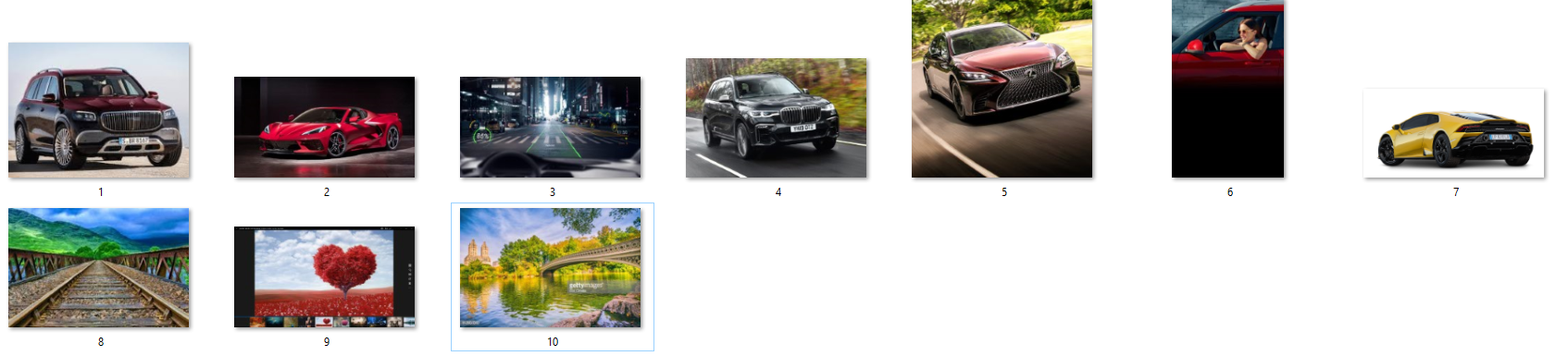
For example, to retrieve img2.jpg we can use the URI,

<https://s3.us-east-1.amazonaws.com/njit-cs-643/10.jpg>

**ii) JAVA application**

Here we can use another .JAR file to download images from S3 bucket. For this we need to repeat all the steps above and run the .JAR file on EC2 instance where we’d like to download our images.

**Now once the images are downloaded we can validate if our program is functioning in the desired way.**



**Fig 15. Images that were used for text and car recognition**

As we can see from Fig15, The given images contain car images and some of them have registration numbers as text. The object recognition algorithm has correctly identified the images that have cars in them, and the text recognition algorithm has accurately detected the registration numbers from the images. Hence, it can be concluded that the program's output is consistent with the expected output, indicating that the program is functioning correctly.