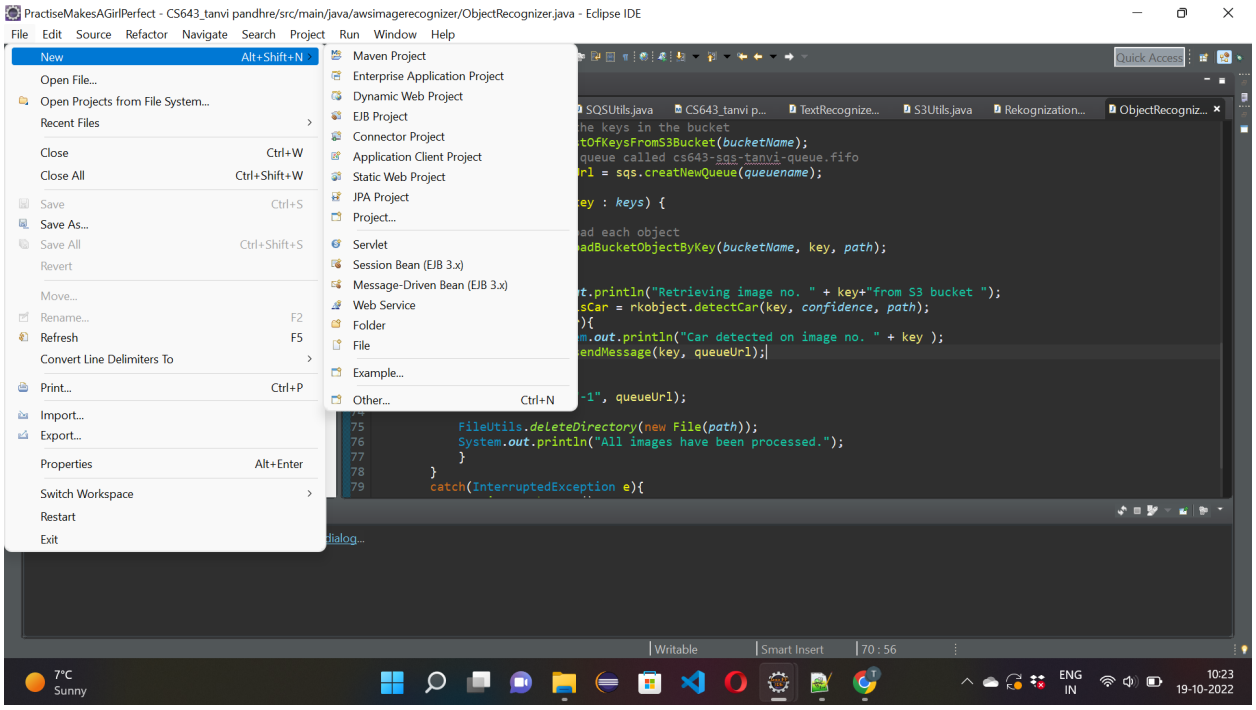
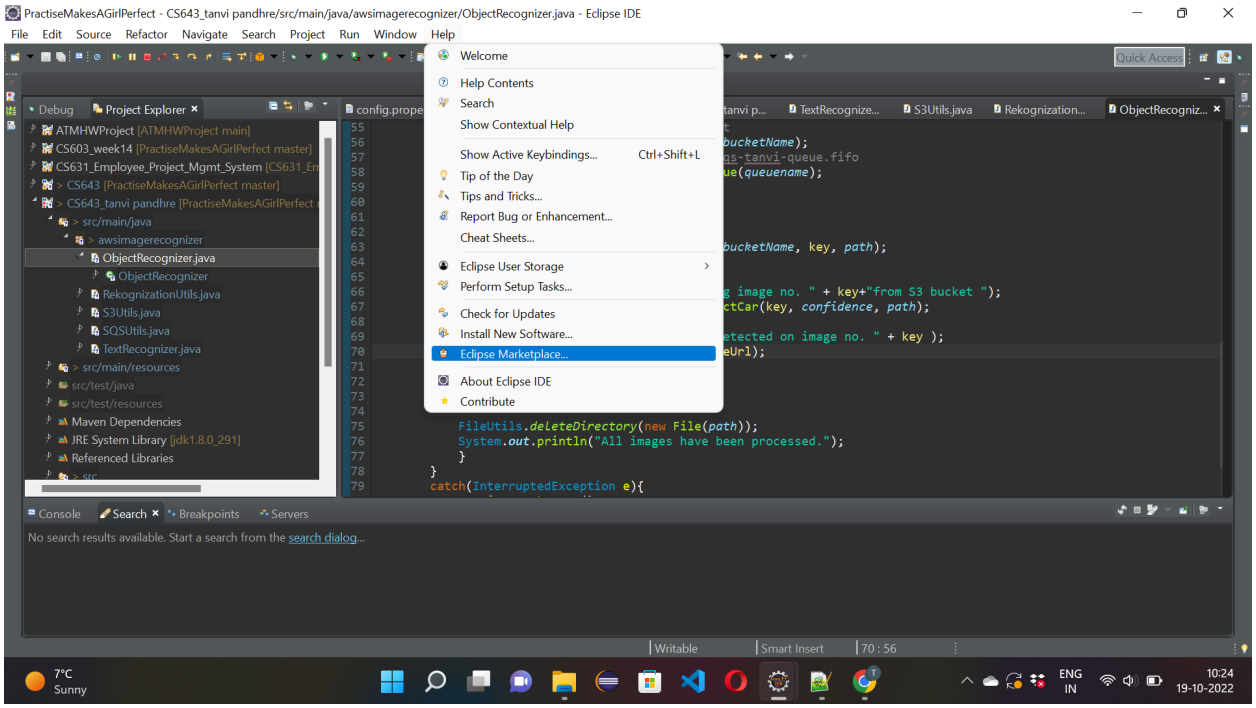
#Installation details.

1. Use eclipse IDE.

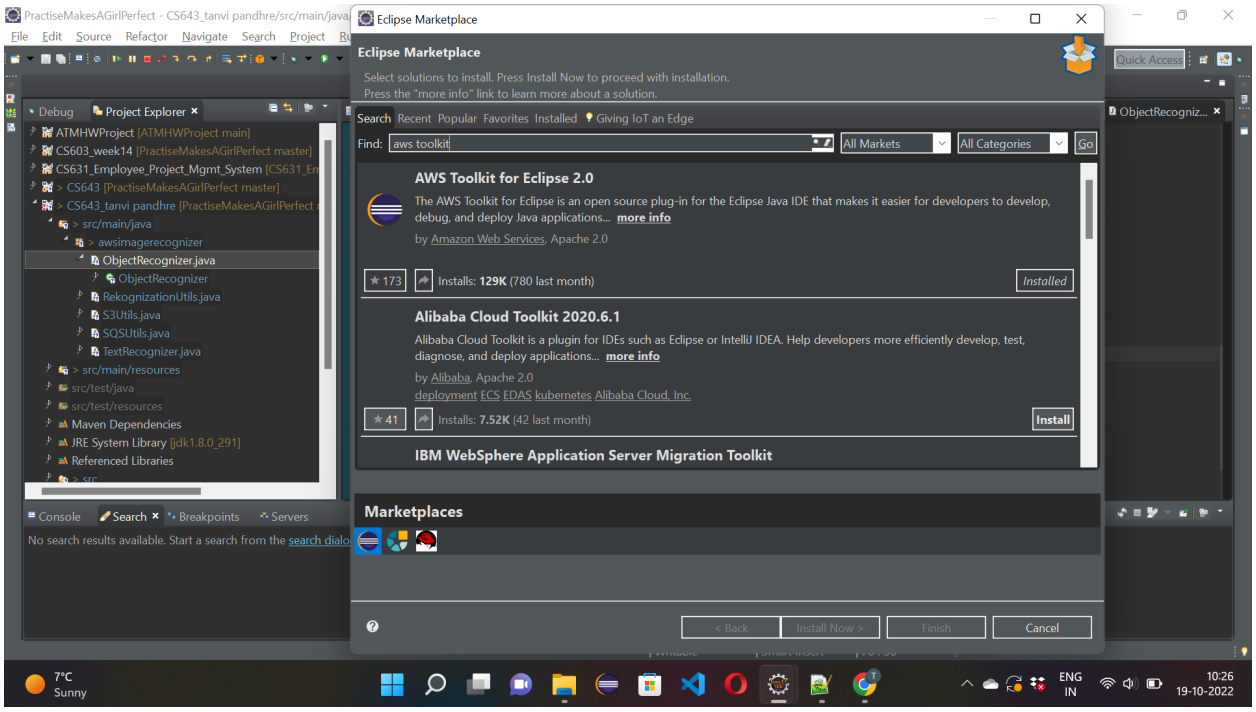
2. Create a maven project called CS643_tanvi pandhre. By choosing File -> New -> Maven Project.



3. Install aws toolkit in eclipse. By choosing Help -> Eclipse Marketplace. Search aws toolkit to install the software.



When it asks for aws access key. Provide from aws details of the learner's lab aws module.



File information - Packages, Java files, properties file, xml.

1. config.properties file -

Mention bucketname, confidenceLevel, pathOfImages, resultantPath, nameOfSqsQueue.

2. Create package awsimagerecognizerUtils for all the utility methods.

3. RekognizationUtils.java - Methods related to aws recognition are written in this class.

- detectObject() - This method will accept image, confidence level and path as parameters. In this method AmazonRekognition agent is created that will detect the object label 'Car' with the required confidence level.
- detectText() - This method will accept image and path as parameters. In this method AmazonRekognition agent is created that will detect text from the image that is passed.

4. S3Utils.java - Methods related to aws s3 bucket are written in this class. It provides a simple interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web.

- S3Utils() - In this constructor s3Client is created with the credentials and region is created.
- isBucket() - checks if bucket exists with the passed bucketName using the s3Client's function doesBucketExistV2()
- listOfKeysFromS3Bucket() - it returns a list of keys from the s3 bucket. using the s3Client's function listObjectsV2().
- downloadBucketObjectByKey() - This method will accept bucket name, key name and path as parameters and we can download the particular image from the given bucket.

5. SQSUtils.java - Methods related to aws Simple Queue Service are written in this class.
- getQueueUrl() - It generates sqs queue url from the passed queue name
- creatNewQueue() - This method will create a new fifo queue with a certain retention period. Will also check if the queue already exists or not. Create one if not
- sendMessage() - It will generate a random unique identifier uuid. It will generate a message to be sent to the sqs queue.
- receiveMessage() - It will receive all the messages passing through the sqs queue.
- deleteMessage() - it will delete message passing through the sqs queue
- deleteQueue() - it will delete the entire queue whose url is passed as parameter.

6. Create package awsimagerecognizer for the image and text recognizer files.

7. ObjectRecognizer.java - This file is for Instance a to detect object Car and send the indexes to sqs queue.
- Instantiate all the above utility file objects to use tits methods and load the config property file.
- First it checks if the bucket exists. If it does exist then all the keys from the s3 bucket have to be retrieved.

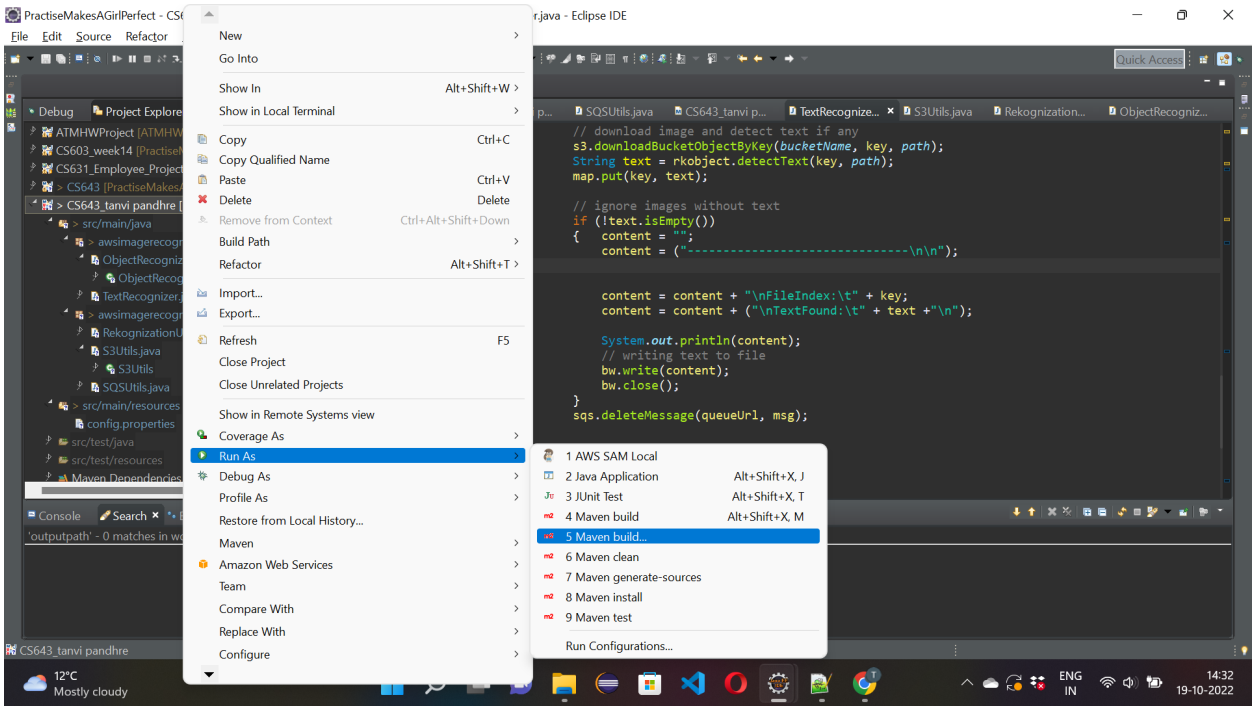
- A new queue is created.
- Each object is loaded from the s3 bucket.
- Once we get the object it is checked whether the object is a car and send its index key to the sqs queue.
- Once instance A terminates its image processing, it adds index -1 to the queue to signal to instance B that no more indexes will come.

8. TextRecognizer.java -

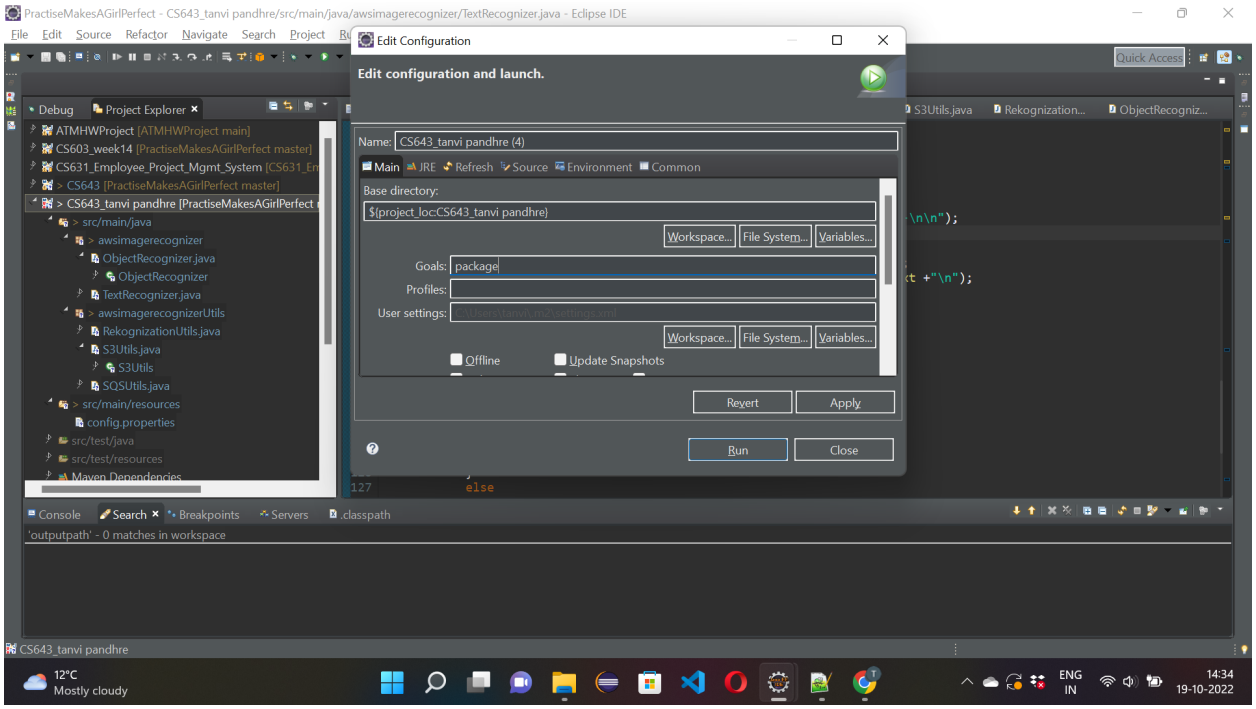
- Instantiate all the above utility file objects to use tits methods and load the config property file.
- check if S3 bucket exists
- Each message received from sqs is iterated.
- New file output.txt is created.
- All the images from the s3 bucket are downloaded whose index is present in the sqs messages.
- Text of that particular image is detected from car image
- Hashmap is created that will store text containing index and text that is generated. This is stored in output.txt file.

9. Pom.xml
In the tag mainClass - package name and class Name is mentioned
In the tag finalname name of the jar is mentioned.

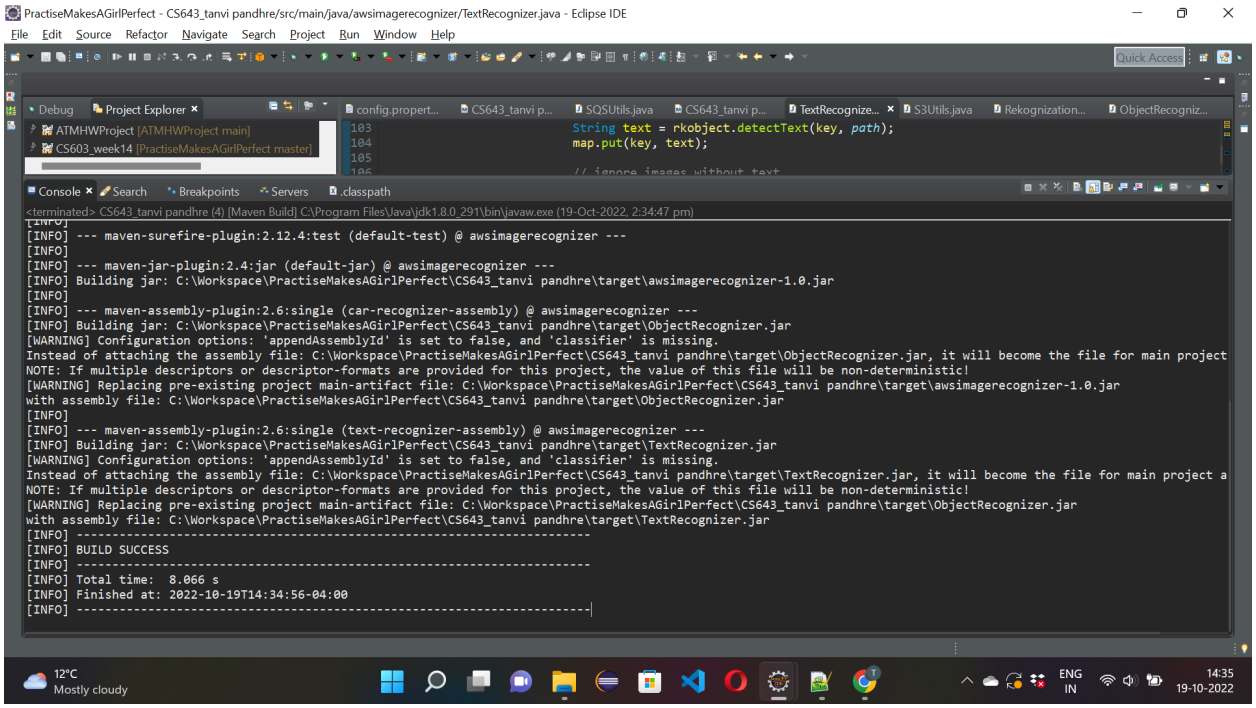
Instruction to use:
1. Create jars by doing Maven build.

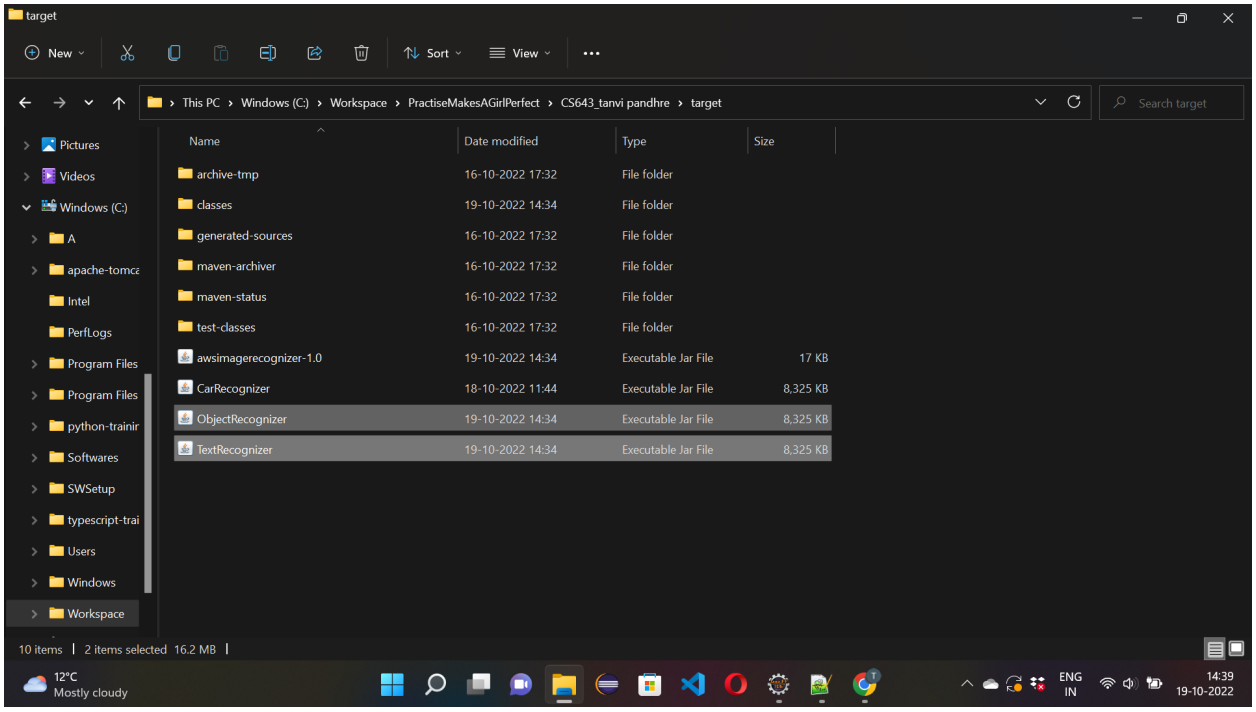


2. In goals type package.



3. Jars should get successfully created in the target location. You Can check console.

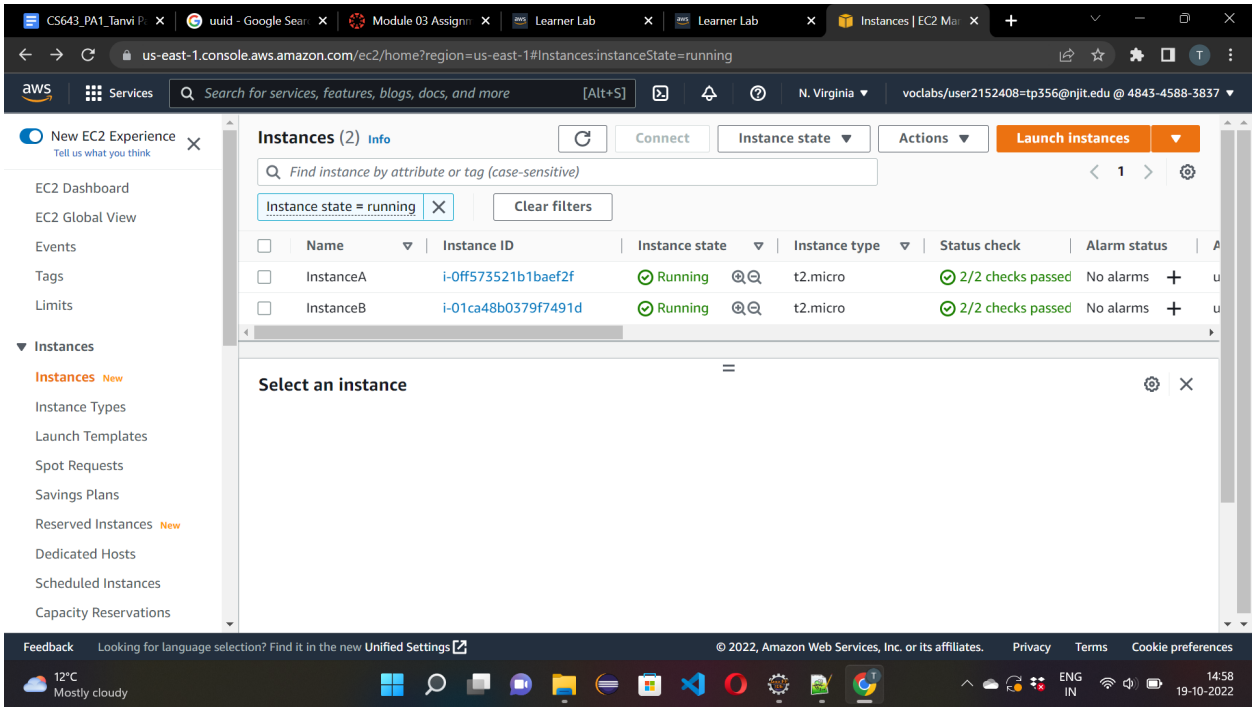




Setup cloud Environment:

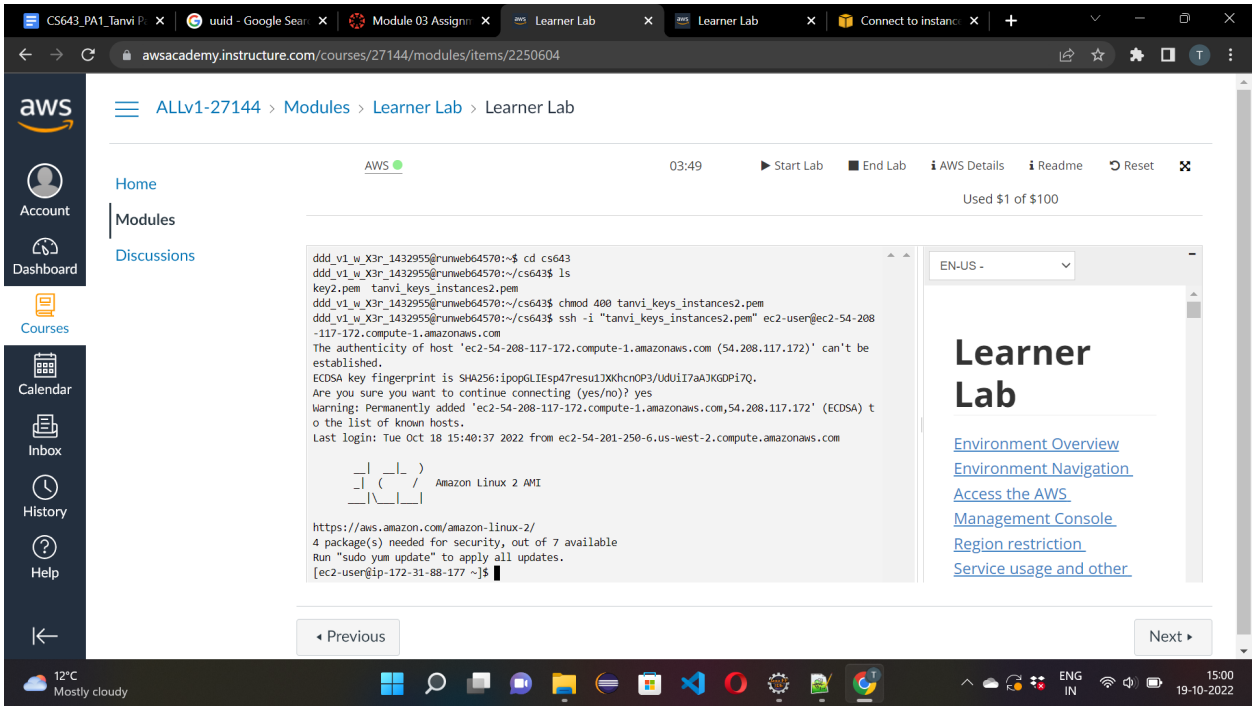
1. To create new EC2 instances go on aws console. Click on launch instances.
2. Select Amazon Linux Amazon Linux 2 Kernel 5.10 AMI free tier.

3. Keep instance type default - t2 tier.
4. Create new key-pair in the .pem format as tanvi_keys_instances2.pem
5. Allow SSH, HTTP, HTTPS traffic.
6. In security group chose MYIP - 128.235.85.0/32
7. Launch instances and you can see them running.

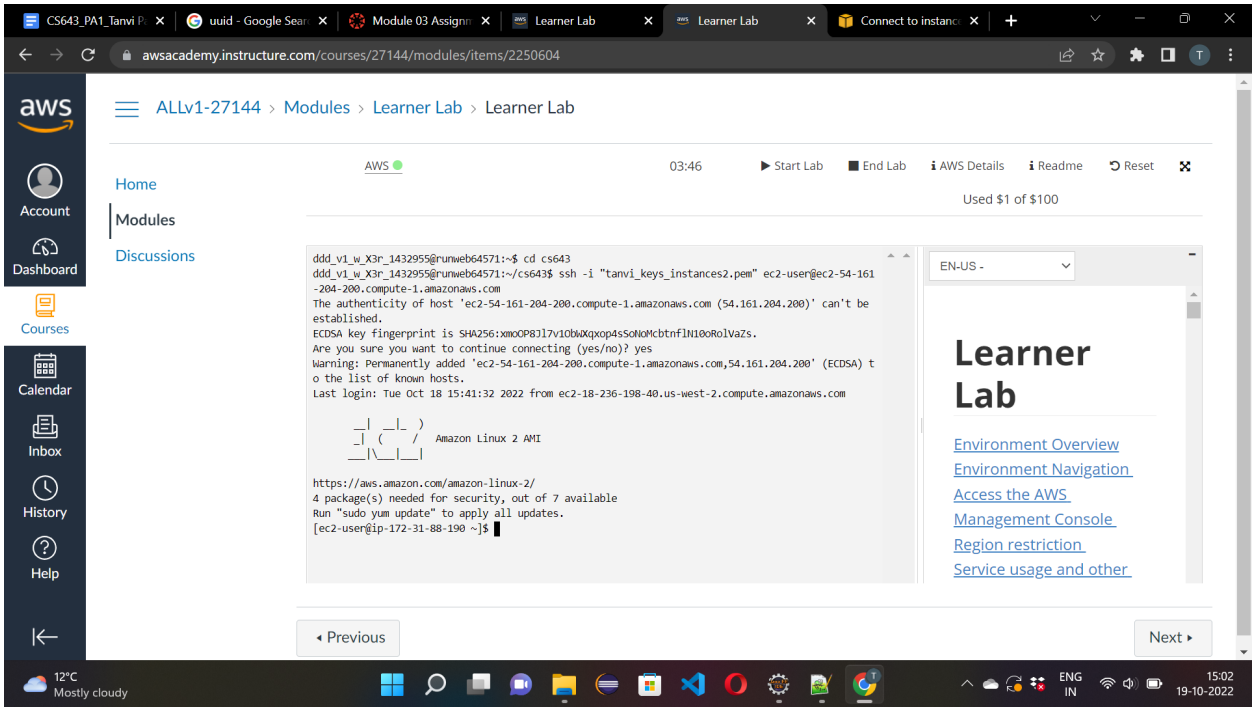


8. Create a folder cs643 in AWS learners lab. Keep tanvi_keys_instances2.pem file in that folder.
9. Connect the EC2 command using the following commands.
chmod 400 tanvi_keys_instances2.pem
ssh -i "tanvi_keys_instances2.pem" ec2-user@ec2-100-26-106-230.compute-1.amazonaws.com

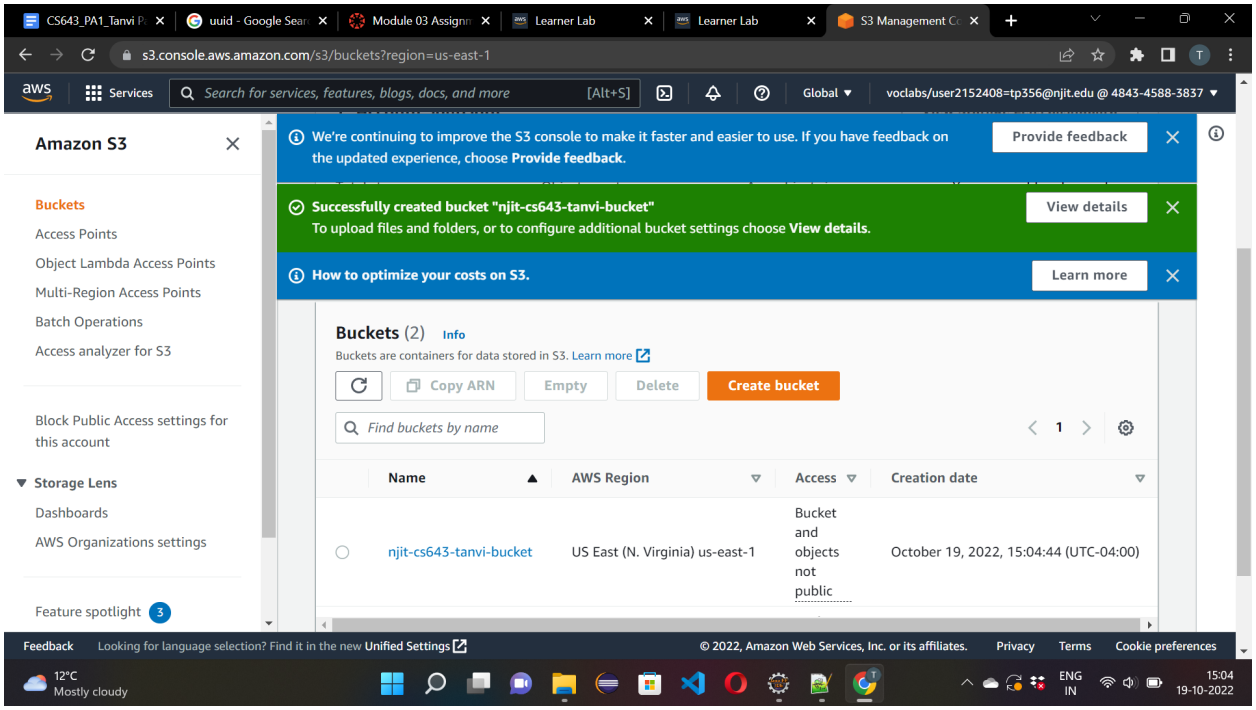
Instance A is running



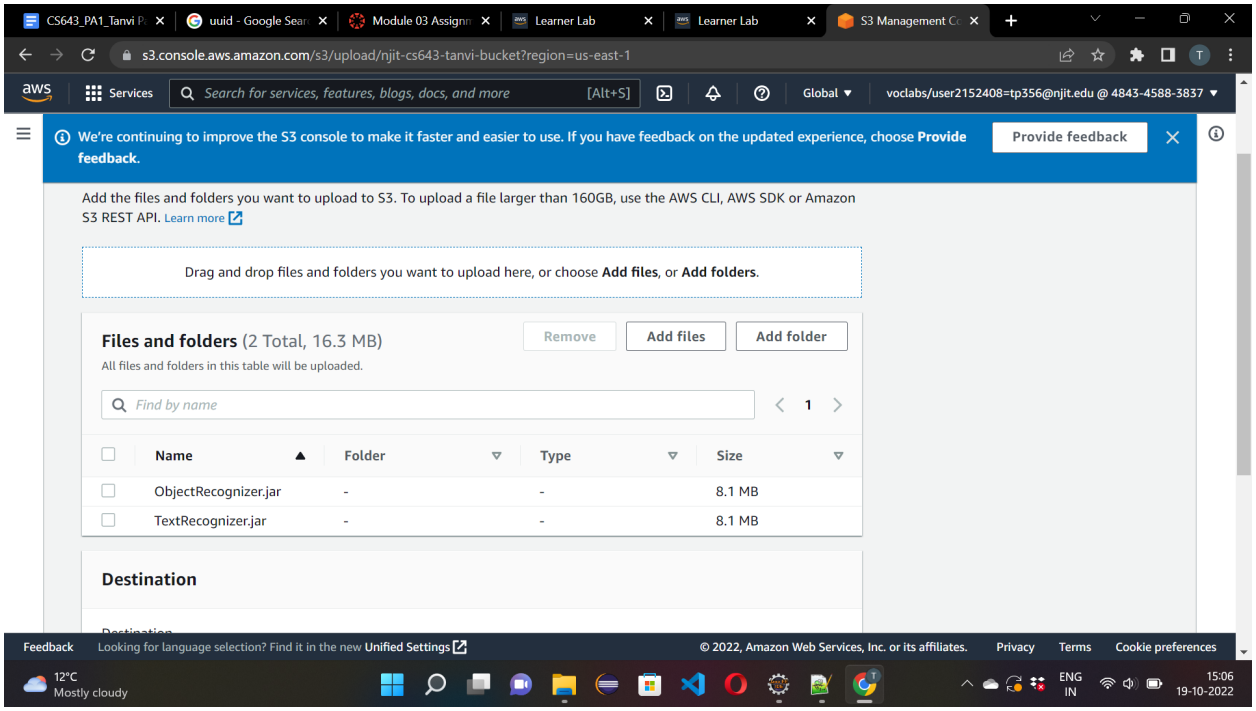
Instance B is running



10. Create s3 bucket -



11. Drag and drop the jars. Upload it then in the s3 bucket.



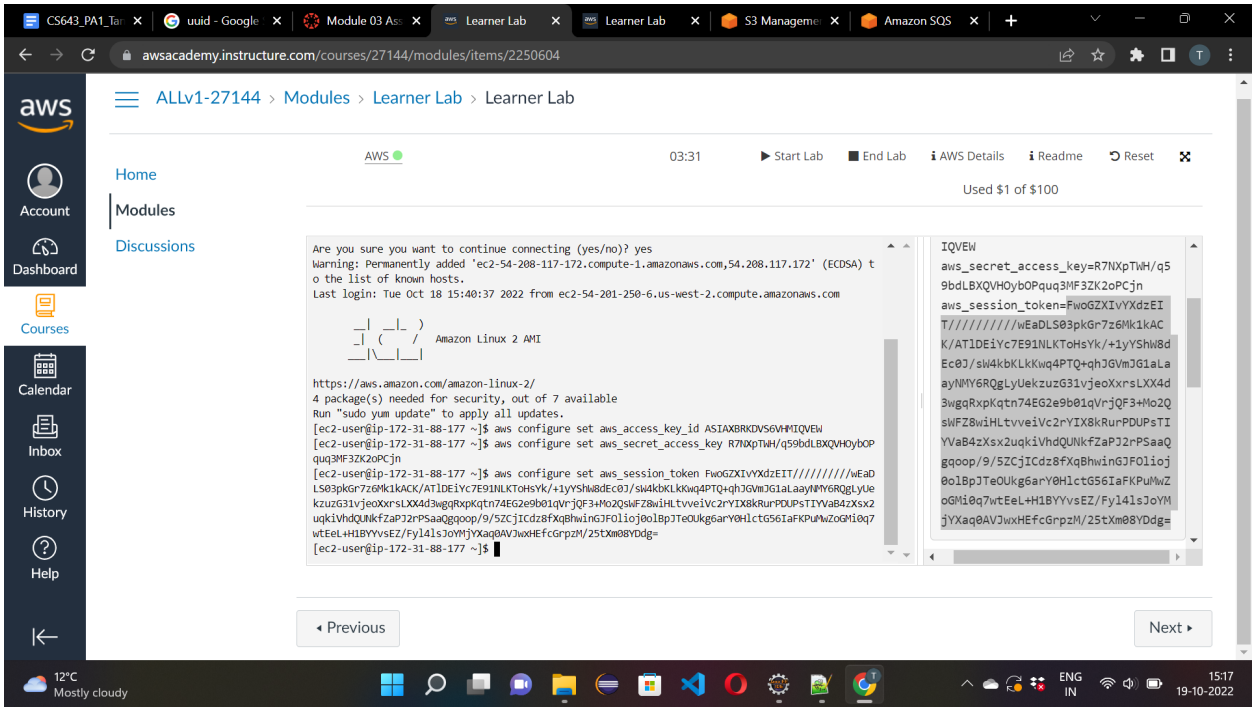
12. Install jdk for both instances using this cmd-

sudo amazon-linux-extras install java-openjdk11

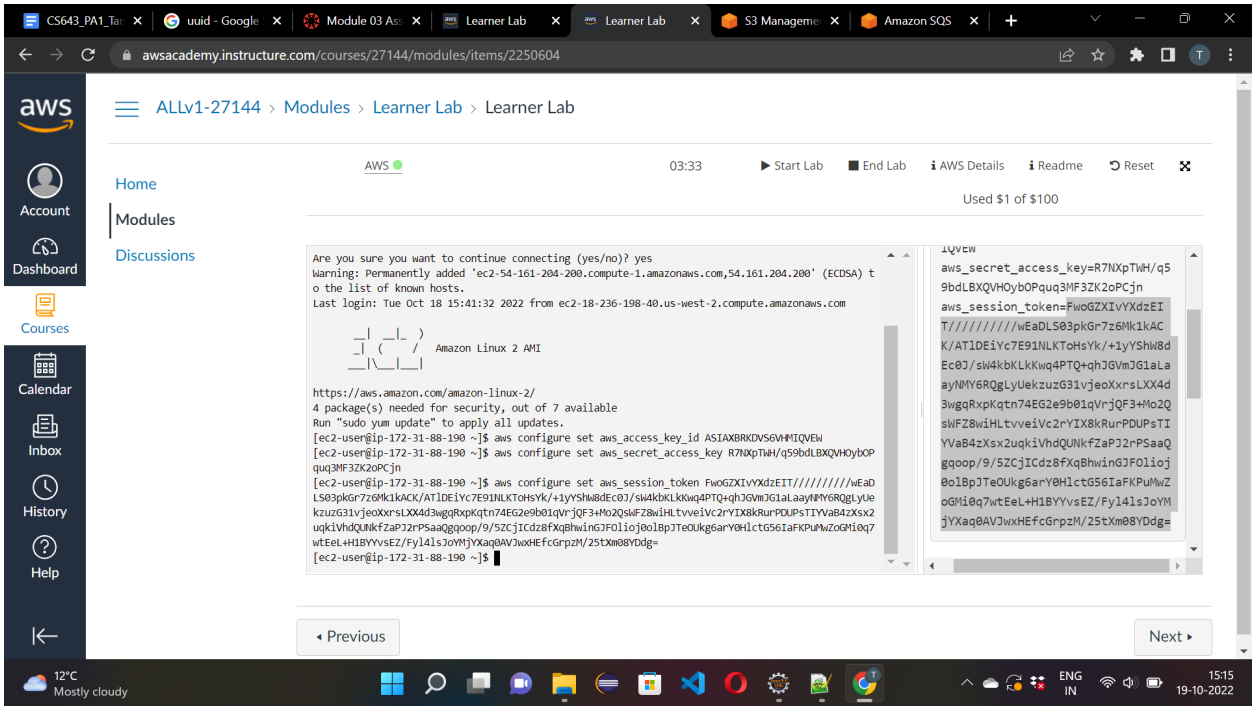
13. Set up aws configuration -

$ aws configure set aws_access_key_id <<access key >>
$ aws configure set aws_secret_access_key <<secret access key>>
$ aws configure set aws_session_token <<aws session token>>

For instance A



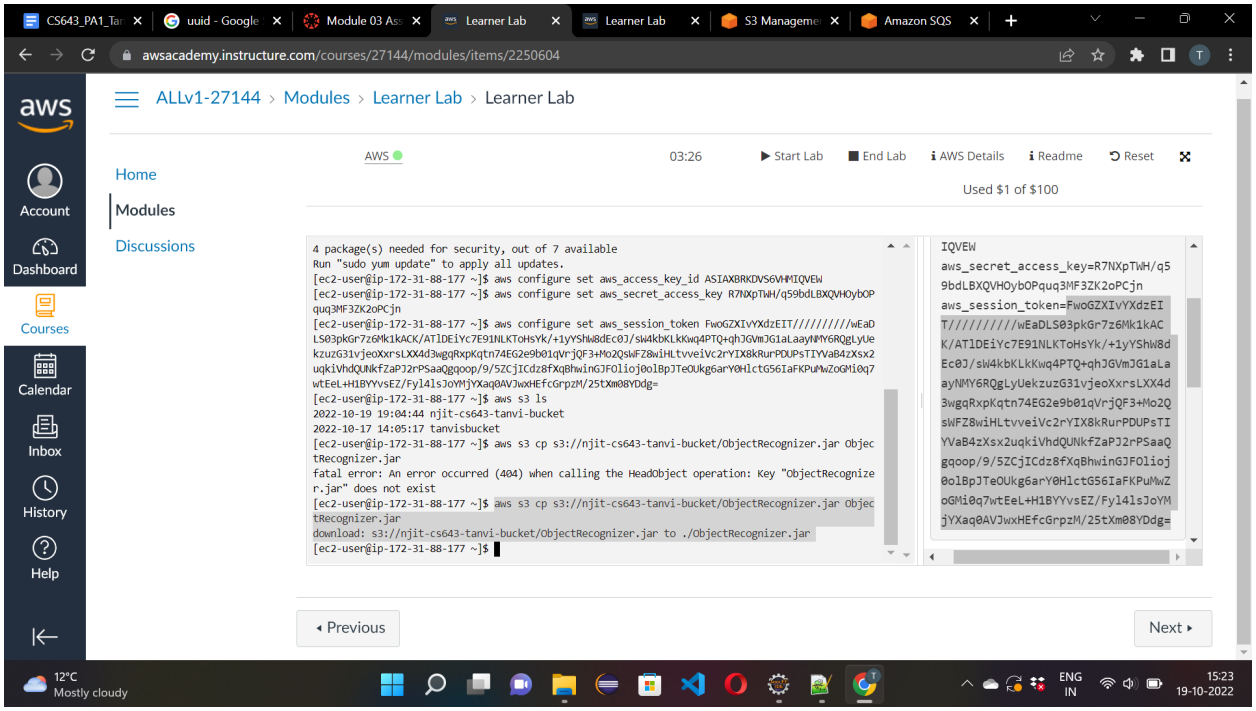
For instance B



14. Copy the required jars to EC2 instance using below command:

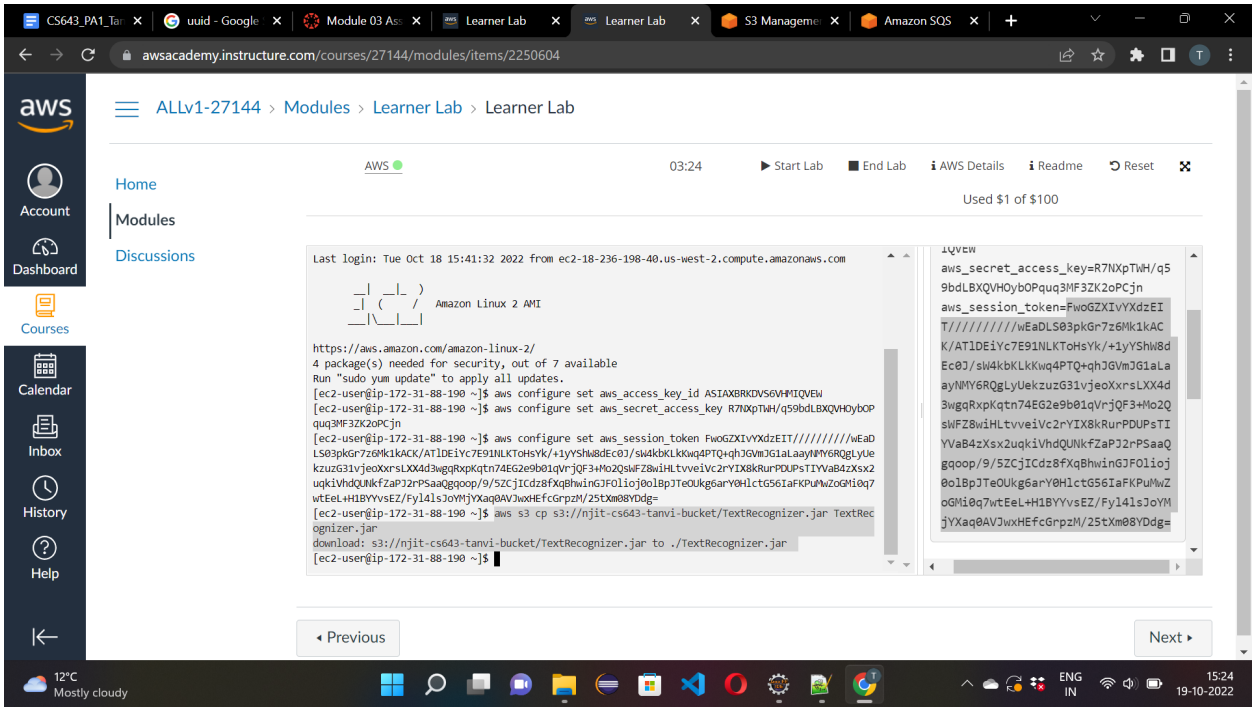
On instance A

aws s3 cp s3://njit-cs643-tanvi-bucket/ObjectRecognizer.jar ObjectRecognizer.jar



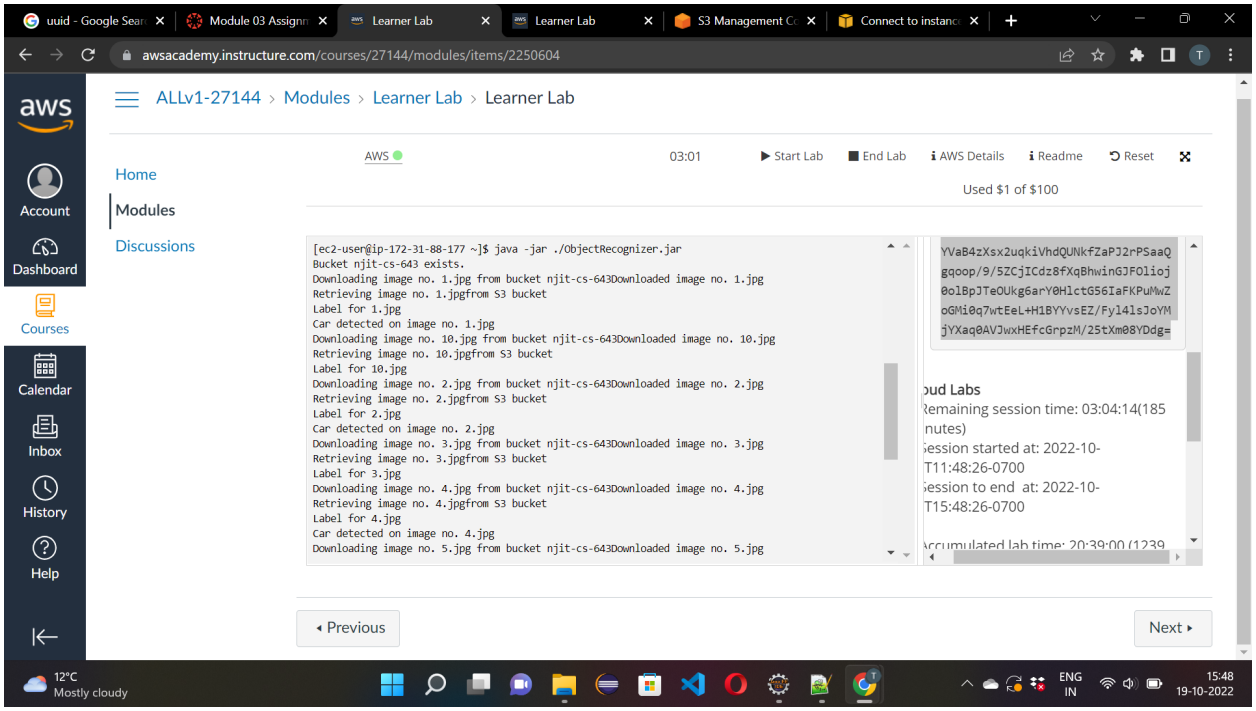
On instance B

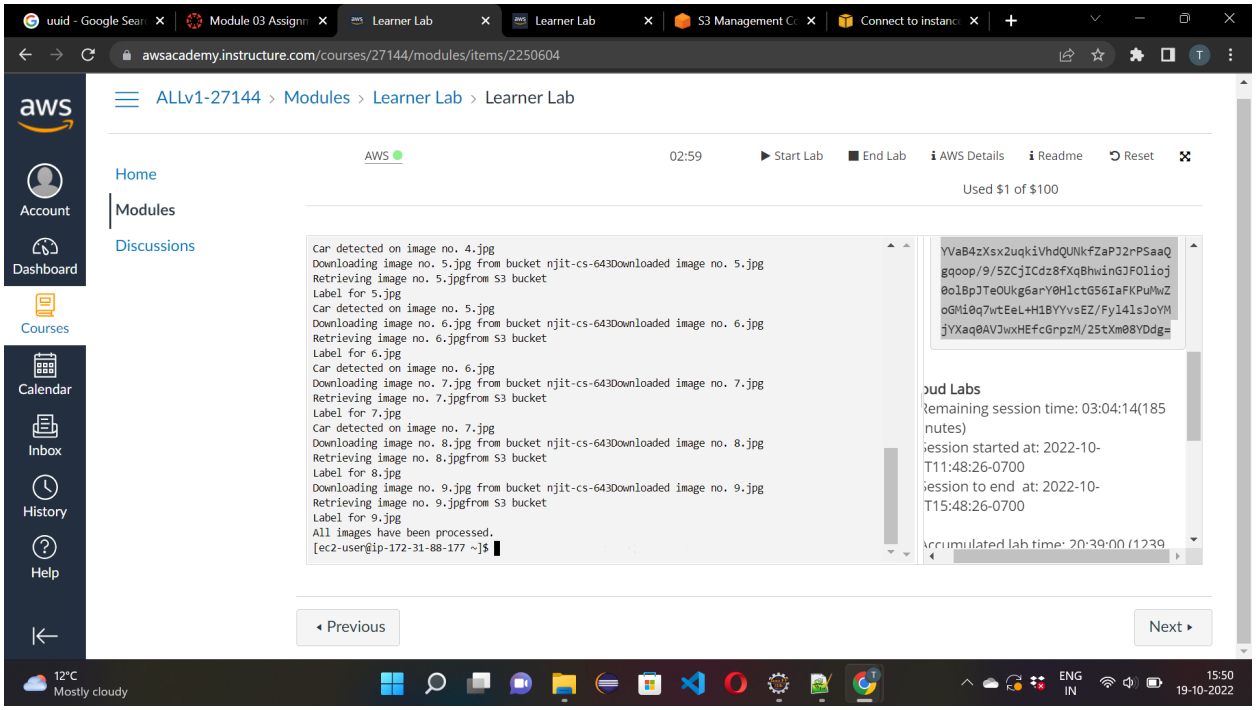
aws s3 cp s3://njit-cs643-tanvi-bucket/TextRecognizer.jar TextRecognizer.jar



15.Run ObjectRecognizer jar using below command on Instance A:

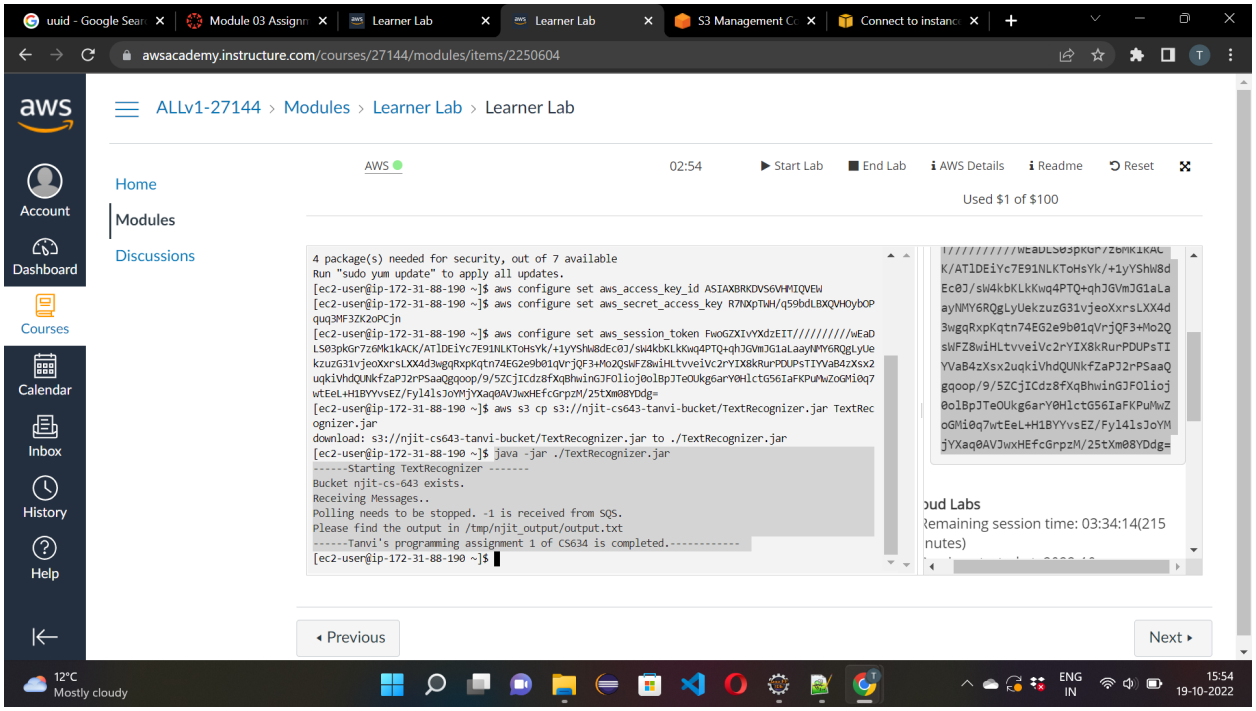
java -jar ./ObjectRecognizer.jar



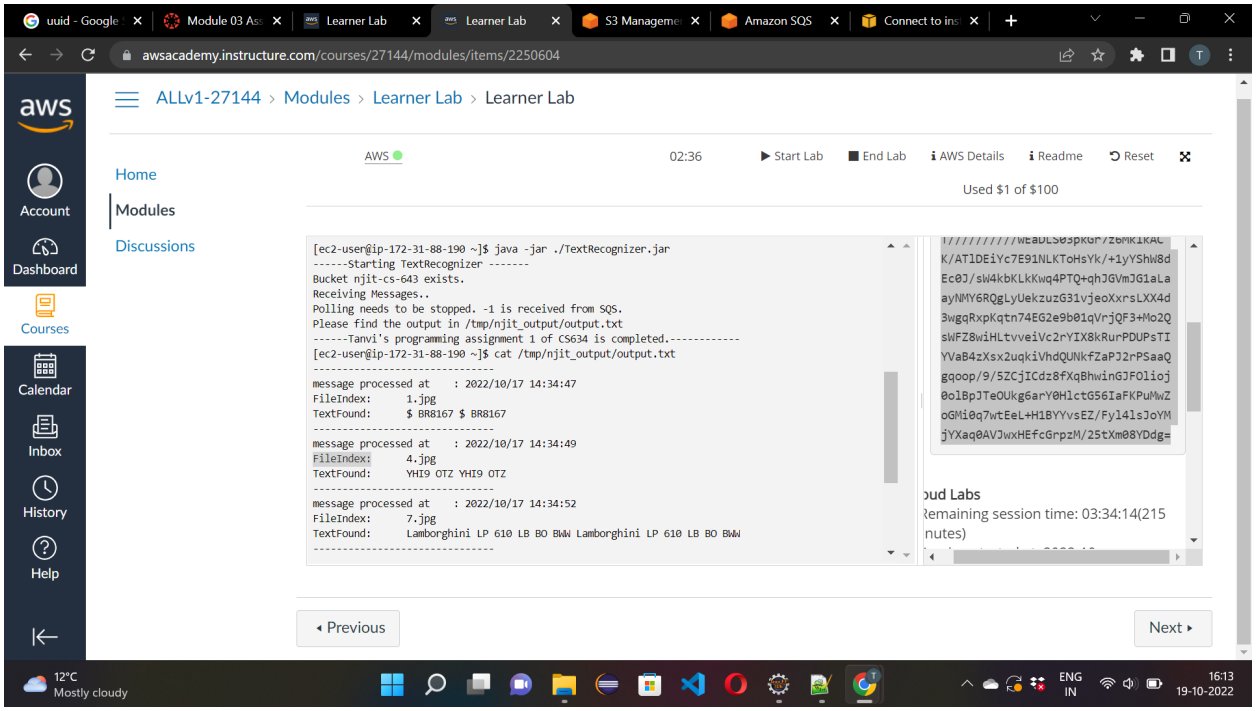


16. Run Text Recognizer jar using below command on Instance B:

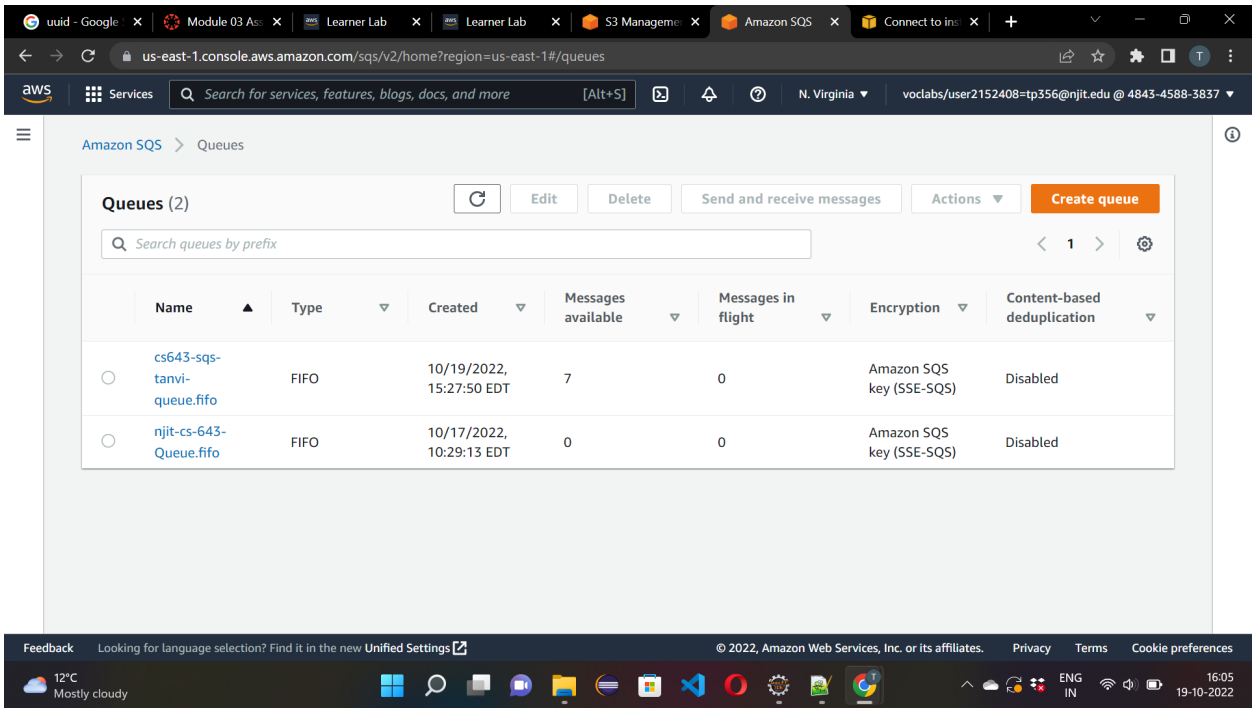
java -jar ./TextRecognizer.jar



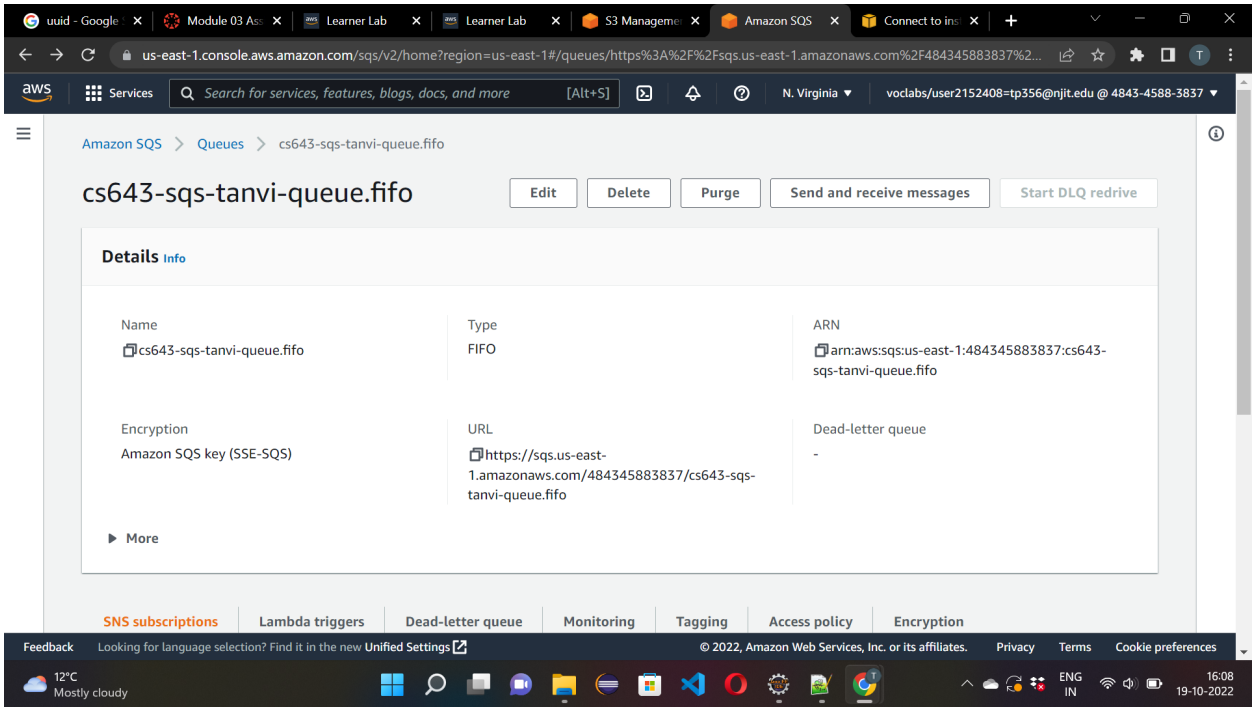
17. Output of TextRecognizer.jar is a text file named output.txt. It will saved at /tmp/njit-output/.
cat /tmp/njit_output/output.txt



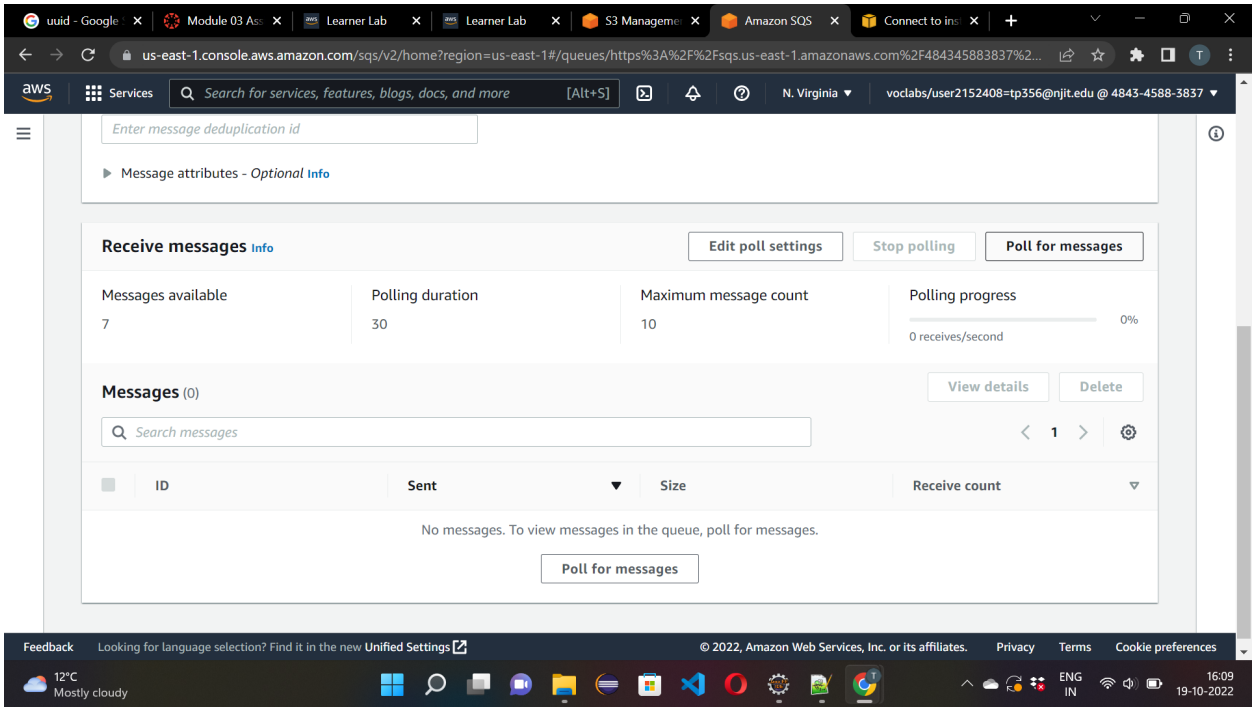
18. We can see the messages that are passed through the sqs queue. Choose cs643-sqs-tanvi-queue.fifo. Click on Send and Receive messages.



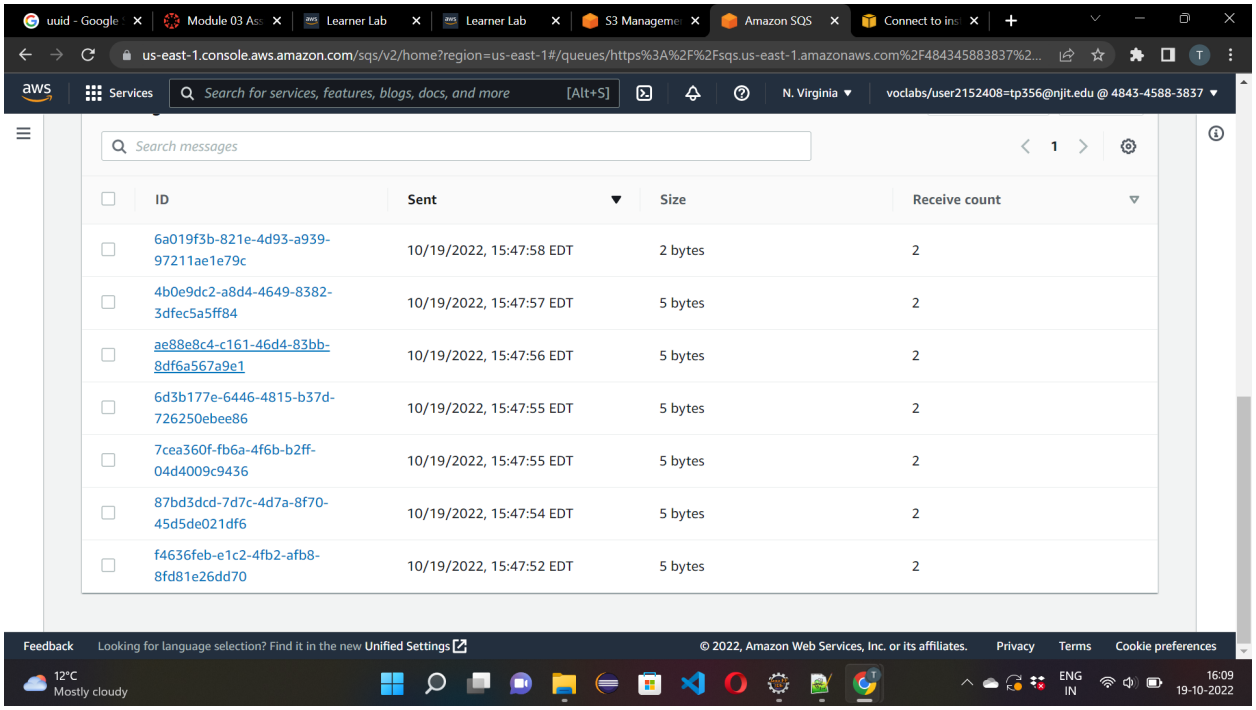
Click on send and receive messages



Click on poll for messages. Here we see 7 messages are passed.



If we click on anyone UUID



We can see the image index that was passed in the queue.