

## Submitted files:-

**CarImageDetector.java** :- Logic to detect car in an image with confidence > 90.

**CarRecognitionEngine** :- The main class for controlling logic to be run on ec2 – 1.

**S3Manager** :- Performs S3 related operations.

**SqsHandler** :- Handles all sqs related operations including creation of queue if not present.

**TextDetector** :- Extracts text from an image using Rekognition.

**TextRecognitionEngine** :- Main class to control logic to be run on ec2 -2 including writing to a output file.

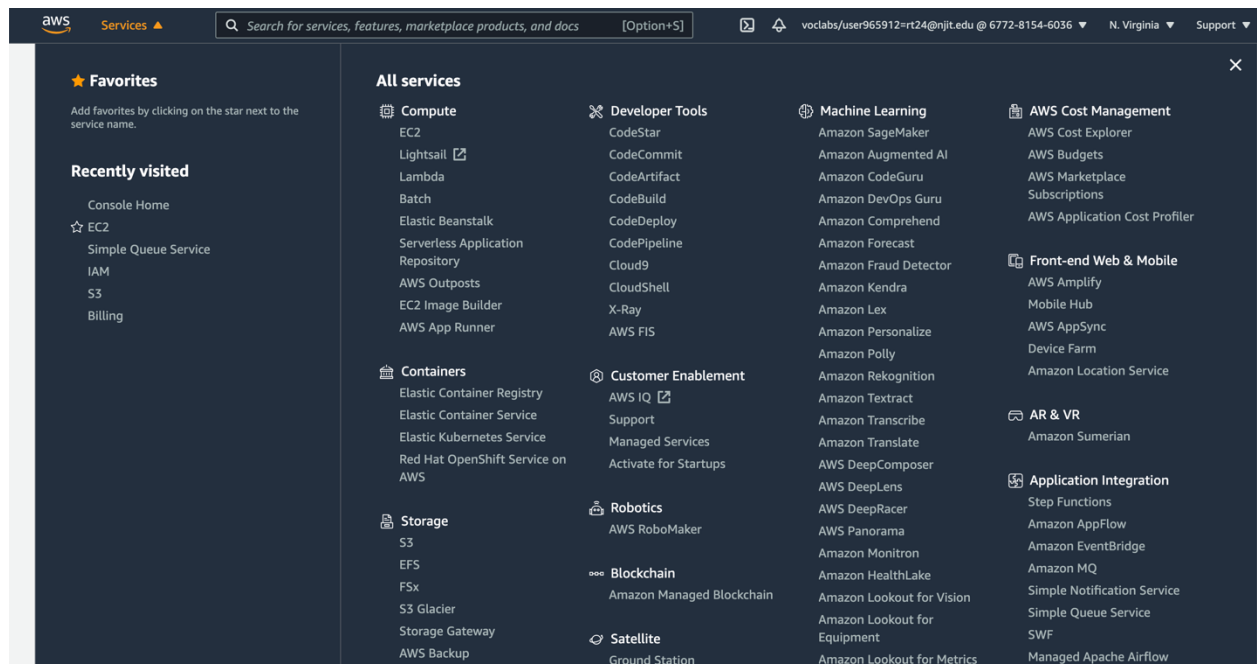
**Readme file** :- Contains instructions to setup AWS services and steps to execute the application.

## Steps to create instances:-

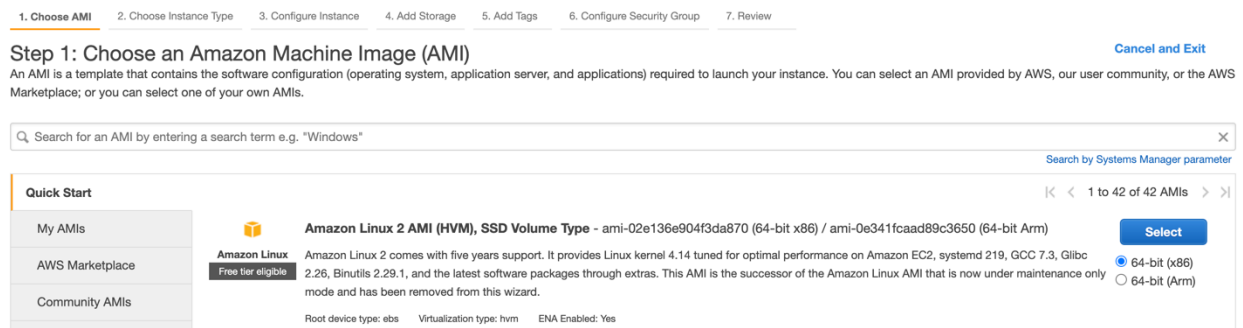
1. Start the lab session in the aws academy. Click on aws.

The screenshot shows the AWS Academy interface. On the left is a dark sidebar with the AWS logo and navigation icons for Account, Dashboard, Courses, Calendar, Inbox, History, and Help. The main content area has a header with the breadcrumb 'ALLFv1-6707 > Modules > Learner Lab Foundation Services > Learner Lab - Foundational Services'. Below the header, there's a status bar showing 'AWS' with a green dot, 'Used \$0.1 of \$100', and a timer '04:00'. A 'Launch Terminal' button is centered in the main area. On the right, a 'Cloud Access' panel is open, displaying session information: 'AWS CLI: Show', 'Cloud Labs', 'Remaining session time: 03:59:19(240 minutes)', 'Session started at: 2021-10-16T16:40:16-0700', 'Session to end at: 2021-10-16T20:40:16-0700', 'Accumulated lab time: 2 days 00:02:52 (2883 minutes)', 'No running instance', 'SSH key: Show, Download PEM', 'Download PPK', 'AWS SSO: Download URL', and a table with 'AWSAccountId: 677281546036' and 'Region: us-east-1'.

2. Go to the aws console and from the services drop down select ec2 and click launch



### 3. From the AMI machine choose the Amazon Linux 2 AMI.



### 4. Choose t2.micro(free tier) as instance type.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

## Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: All instance families Current generation Show/Hide Columns

Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, -, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GiB)	EBS-Optimized Available	Network Performance	IPv6 Support
<input type="checkbox"/>	t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	t2	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.medium	2	4	EBS only	-	Low to Moderate	Yes

## 5. Configure Instances Details by selecting number of instances as 2.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

## Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the more.

Number of instances 1 Launch into Auto Scaling Group

## 6. Configure Security Groups: Create a new security group with three ports for SSH, HTTP and HTTPS with source as My IP. We initially set the HTTP and HTTPS to Anywhere so that we can install java onto the ec2 instances.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

## Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: ☒ Create a new security group  
☐ Select an existing security group

Security group name: launch-wizard-4

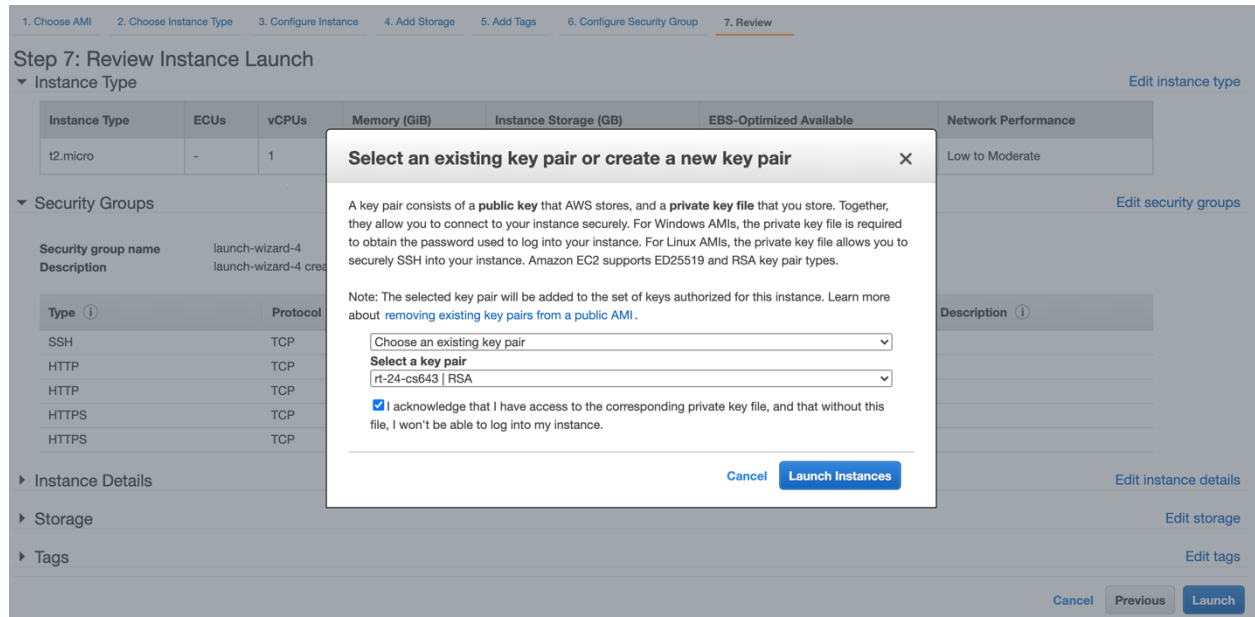
Description: launch-wizard-4 created 2021-10-16T19:55:11.755-04:00

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	My IP 173.54.74.134/32	e.g. SSH for Admin Desktop
HTTP	TCP	80	Anywhere 0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop
HTTPS	TCP	443	Anywhere 0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop

Add Rule

7. Review and Launch the instances.

8. Choose an existing key pair or create a new key pair and download the pem file. Change the permission of the pem file to 660.



9. Launch and view the instances. Rename the instances as ec2-1 and ec2-2 for easy reference.

## Connecting to the aws machine:

1. Ssh to the aws machine with the following command.

```
ssh -i <path to the pem file> ec2-user@<ec2-1 instance public IPv4 DNS name>
```

Run the above command for both the instances in two different terminals with their respective IPv4 address.

2. Create a directory ".aws" in the home directory. Create a file named "credentials" in the .aws folder. In this file add the "aws\_access\_key\_id", "aws\_secret\_access\_key", "aws\_session\_token". This can be found from the lab session we started under the AWS CLI section.

3. Check if java is installed in the ec2 instances with the command “java –version”. If not install java using the following command.

```
sudo amazon-linux-extras install java-openjdk11
```

4. Create a directory “cs643” in the home directory of both the instances.

5. Open a new tab in the terminal. Update the aws credentials file in your local machine too.

6. Copy the jar files to the ec2 instances using the following command.

- a) `scp -i “pem file location” “absolute path of cartextrekog-car-rekon.jar” ec2-user@“ec2-1 instance public IPv4 DNS name”:/home/ec2-user/cs643`
- b) `scp -i “pem file location” “absolute path of cartextrekog-text-rekon.jar” ec2-user@“ec2-2 instance public IPv4 DNS name”:/home/ec2-user/cs643`

7. Once the jar files are copied into their respective instances. Run the jar using the below command (move to the directory where jar is located).

- a) For ec2-1 run “java -jar cartextrekog-car-rekon.jar”
- b) For ec2-2 run “java -jar cartextrekog-text-rekon.jar”

## Output:

Below are snippets of the output generated.

```
/home/ec2-user/cs643
[ec2-user@ip-172-31-19-25 cs643]$ java -jar cartextrekog-car-rekon.jar
Car Detected in image: 1.jpg
Car Detected in image: 2.jpg
Car Detected in image: 4.jpg
Car Detected in image: 5.jpg
Car Detected in image: 6.jpg
Car Detected in image: 7.jpg
```

```
[ec2-user@ip-172-31-25-225 cs643]$ pwd
/home/ec2-user/cs643
[ec2-user@ip-172-31-25-225 cs643]$ java -jar cartextrekog-text-rekon.jar
[ec2-user@ip-172-31-25-225 cs643]$ ls
cartextrekog-text-rekon.jar  output1634510893670.txt
[ec2-user@ip-172-31-25-225 cs643]$ less output1634510893670.txt
[ec2-user@ip-172-31-25-225 cs643]$
[ec2-user@ip-172-31-25-225 cs643]$ cat output1634510893670.txt
Texts in file: 1.jpg are: S-BR8167
Texts in file: 4.jpg are: YHI9 OTZ
Texts in file: 7.jpg are: LP 610 LB
```