PROGRAMMING ASSIGNMENT -02

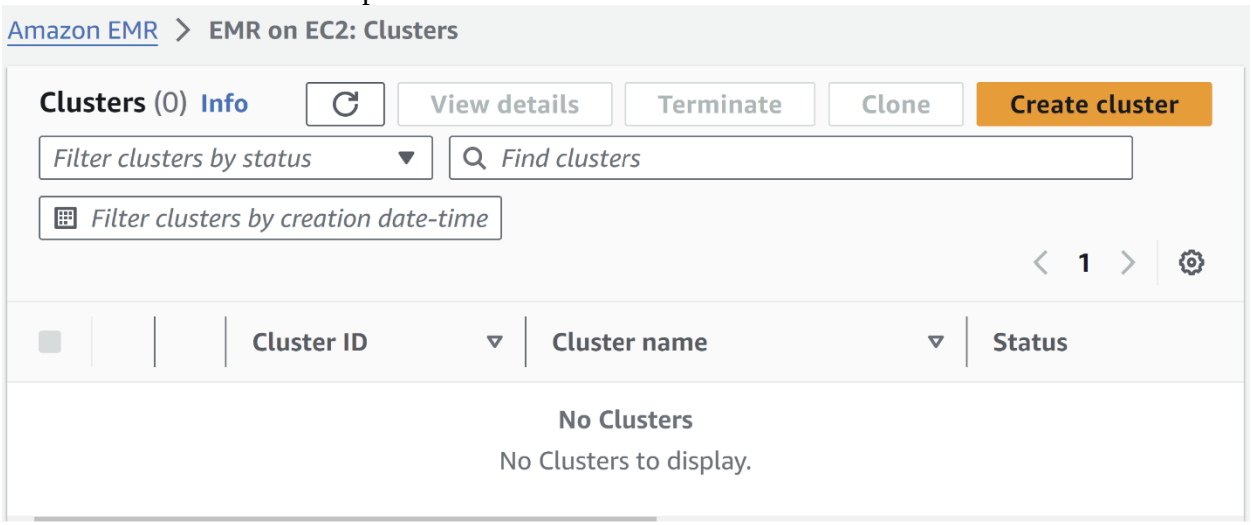
Wine Quality Prediction

Email Id: sk3568@njit.edu

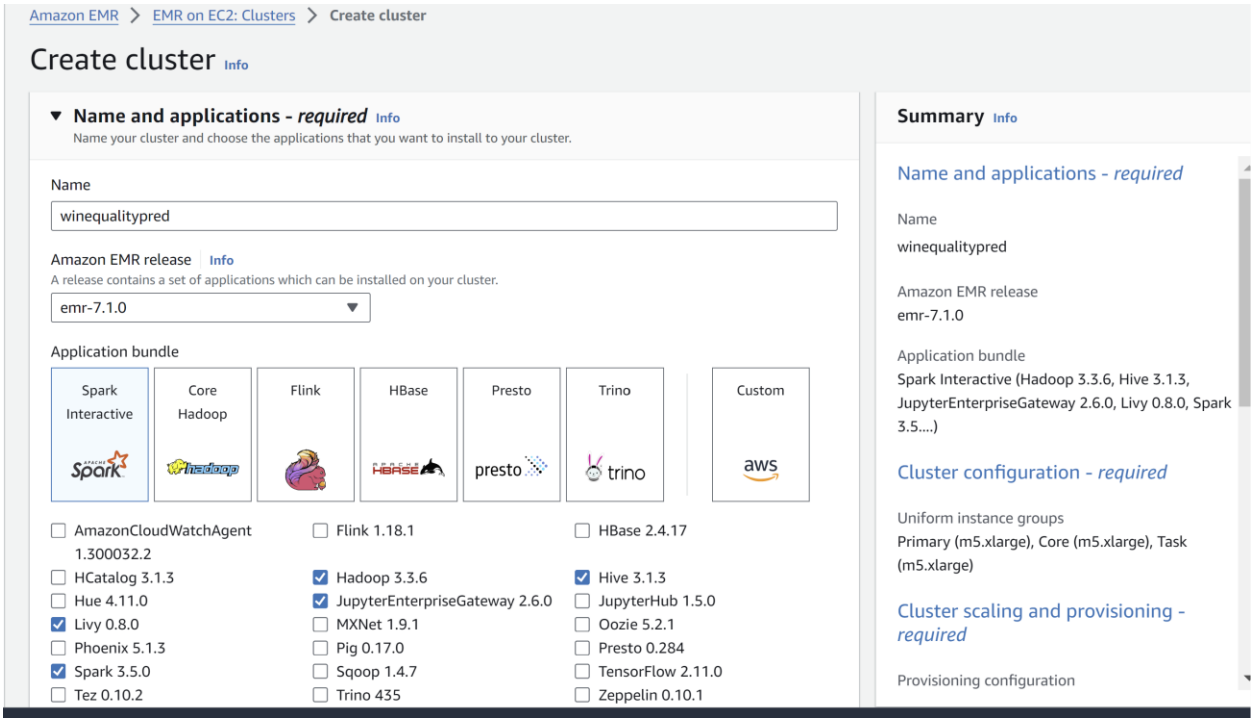
GitHub: https://github.com/sravyakganti/Wine-quality-prediction-

Docker Hub: https://hub.docker.com/repository/docker/sravyakganti/cs643-programming-assignment-2/general

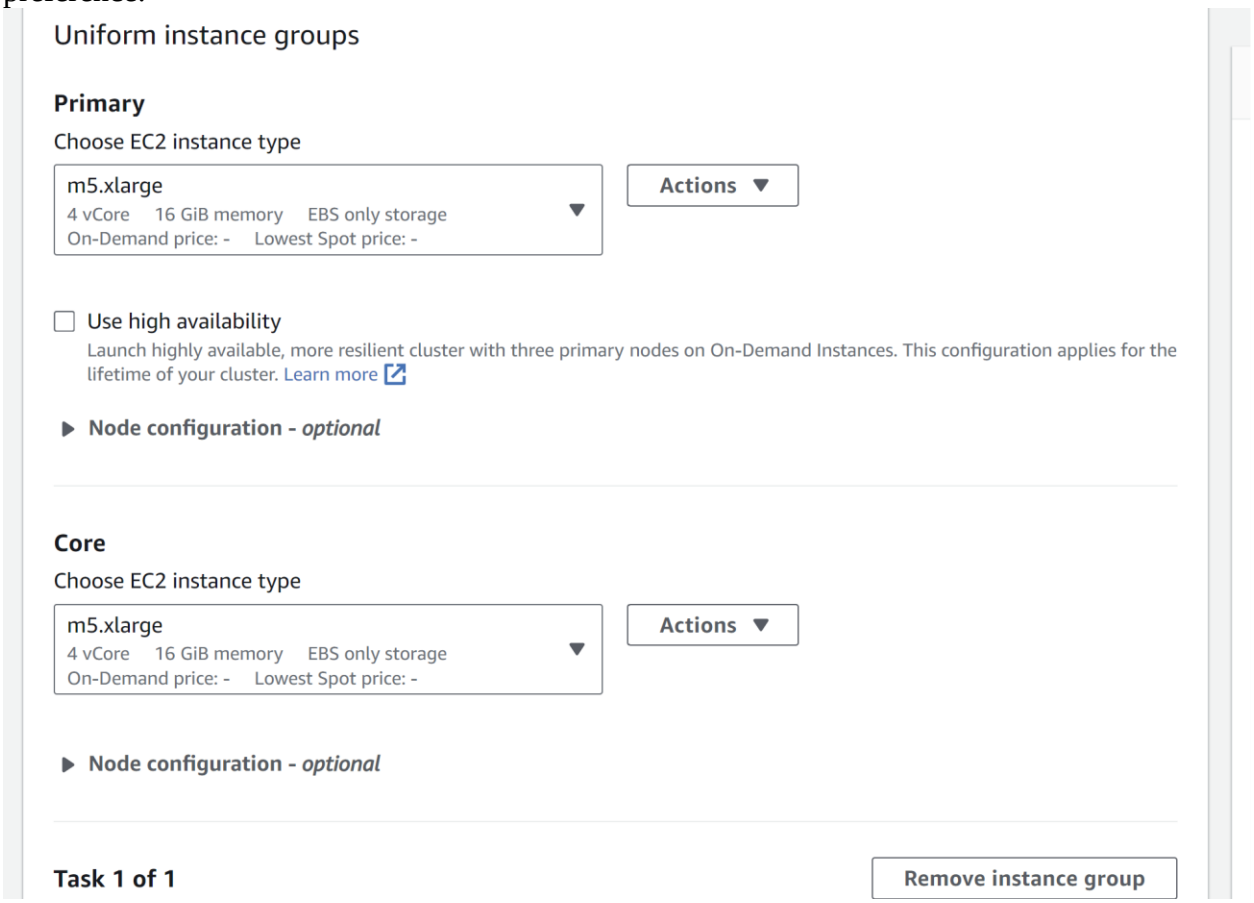
Open the AWS Console and log in.
Choose AWS EMR service from the list of AWS services, and then choose EMR on EC2 Clusters.
You'll see the Clusters page. It is evident that there aren't any clusters in use. We must establish a cluster.
Select the "Create cluster" option to start a cluster.



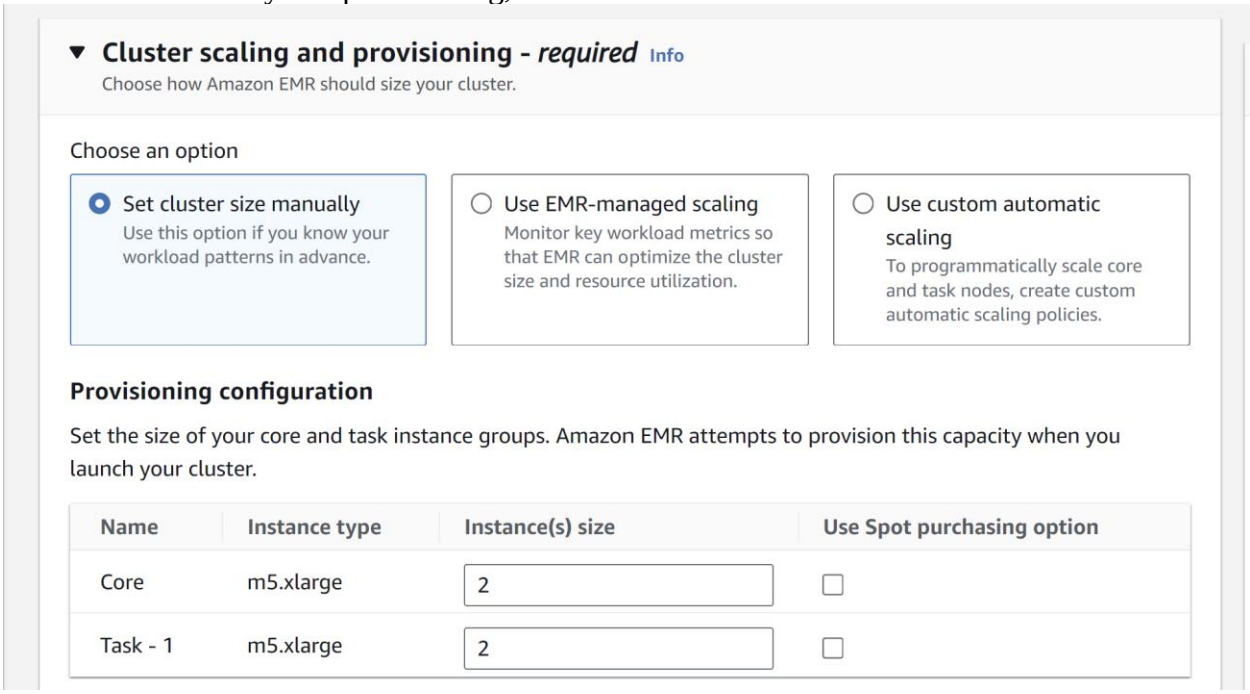
The cluster can have any name you choose. Additionally, confirm that the EMR version is the most recent one, as indicated by the figure below. Additionally, choose Spark Interactive from the Application package menu.



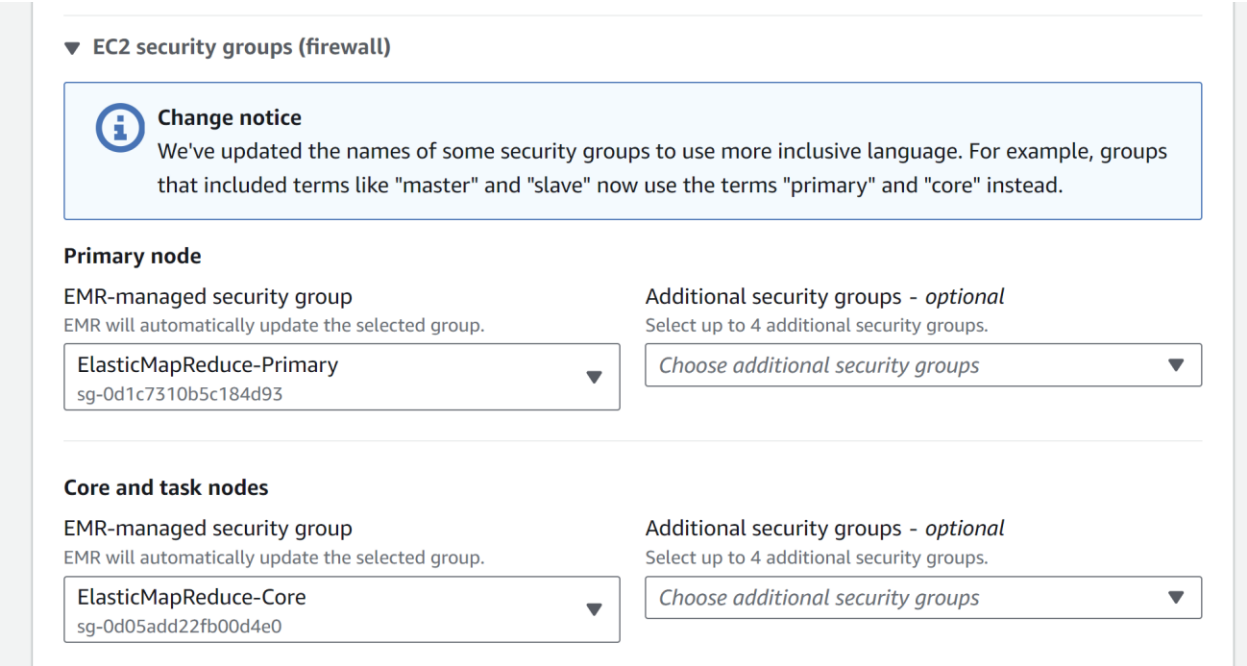
For instance type, I have selected m5.xlarge. We can choose anything according to our preference.



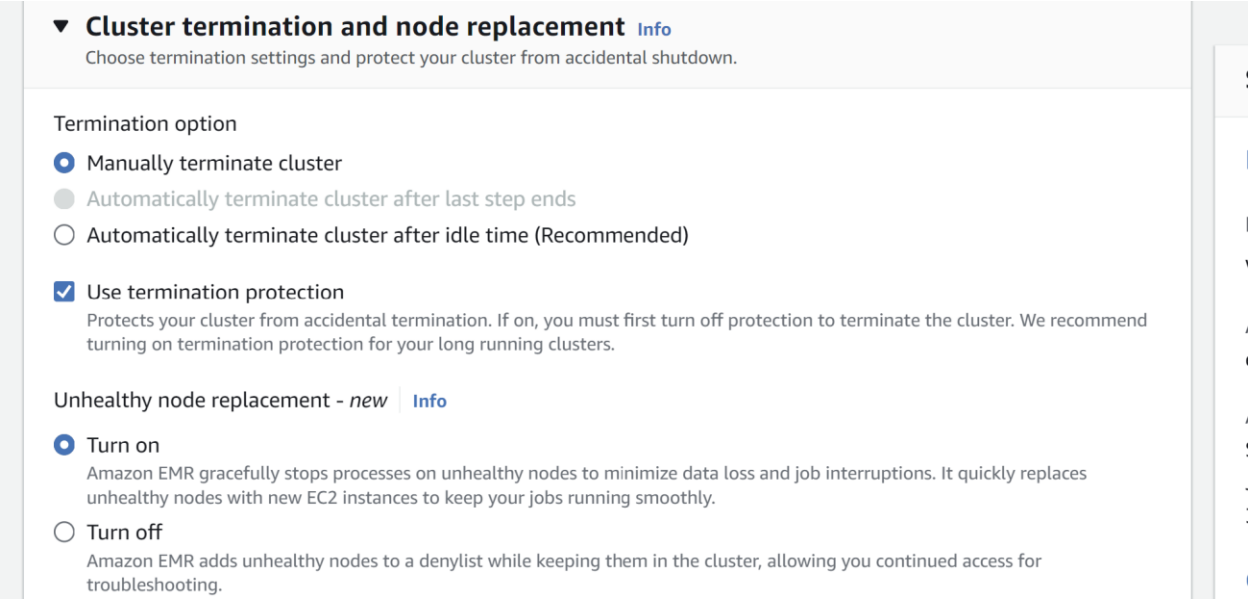
For cluster scalability and provisioning, set the Instance size for Core to 1 and Task-2 to 3.



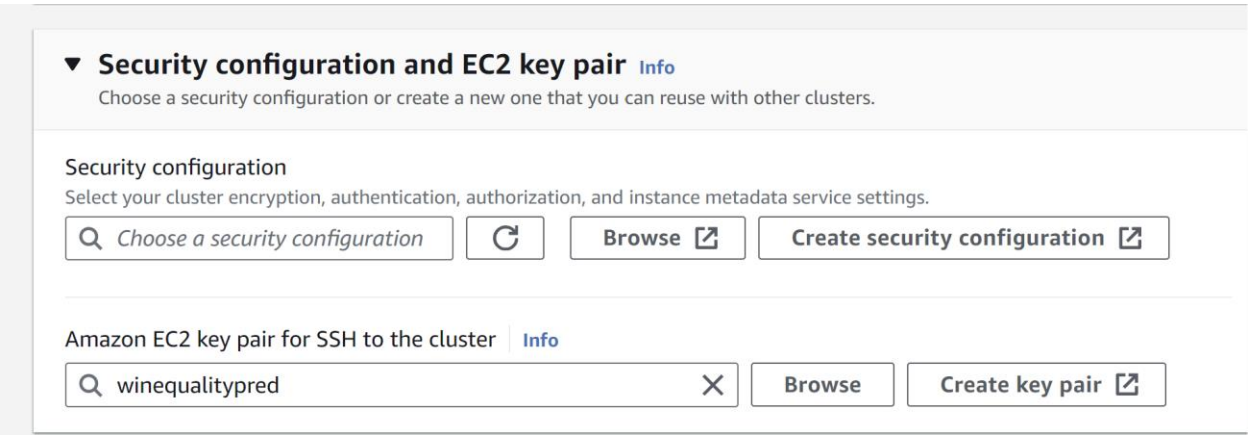
Choose the security groups for the Primary, Core, and Task nodes in the EC2 security groups as indicated below.



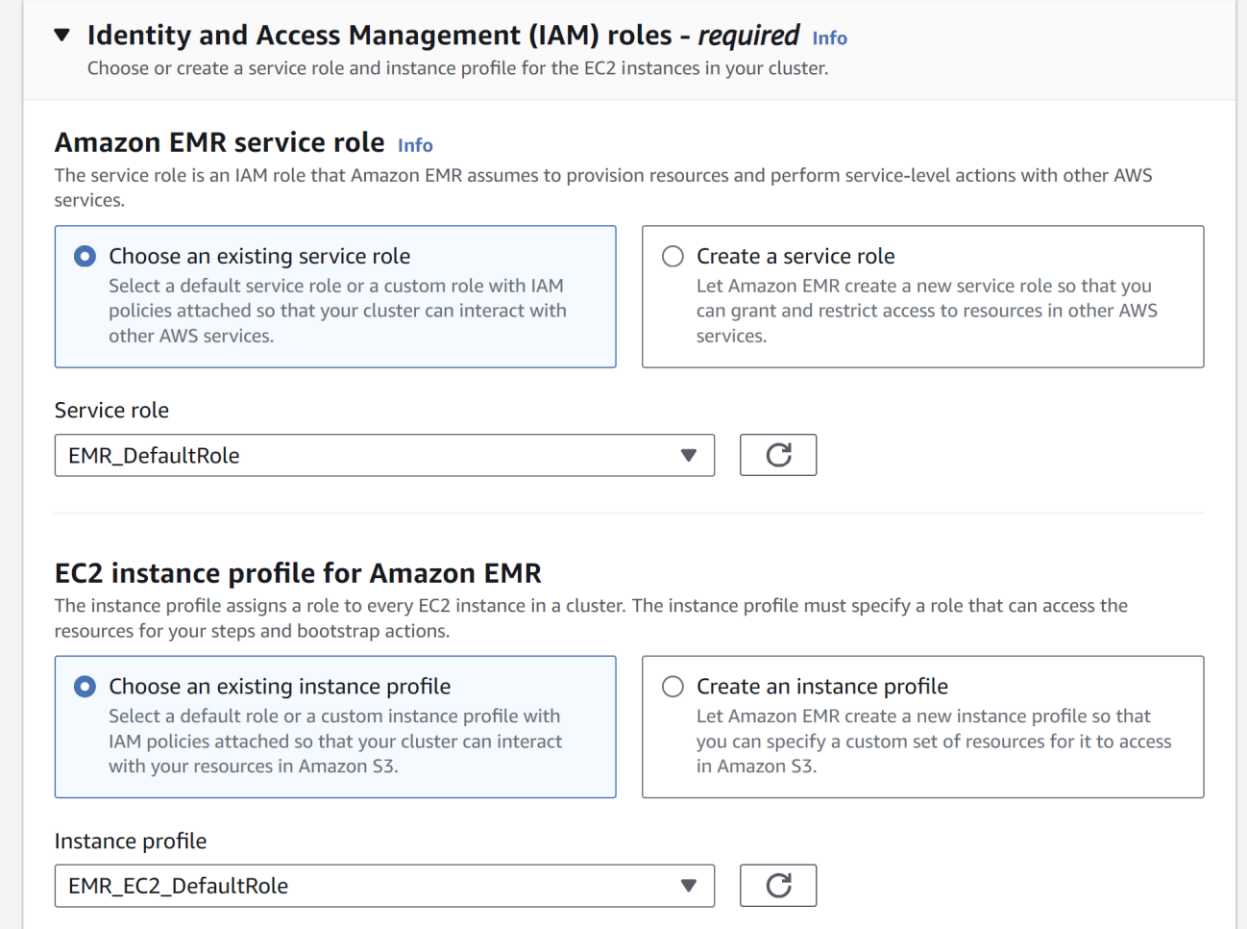
To stop the cluster from ending automatically, make sure you choose the option to manually terminate the cluster. But this is not a suggested option.



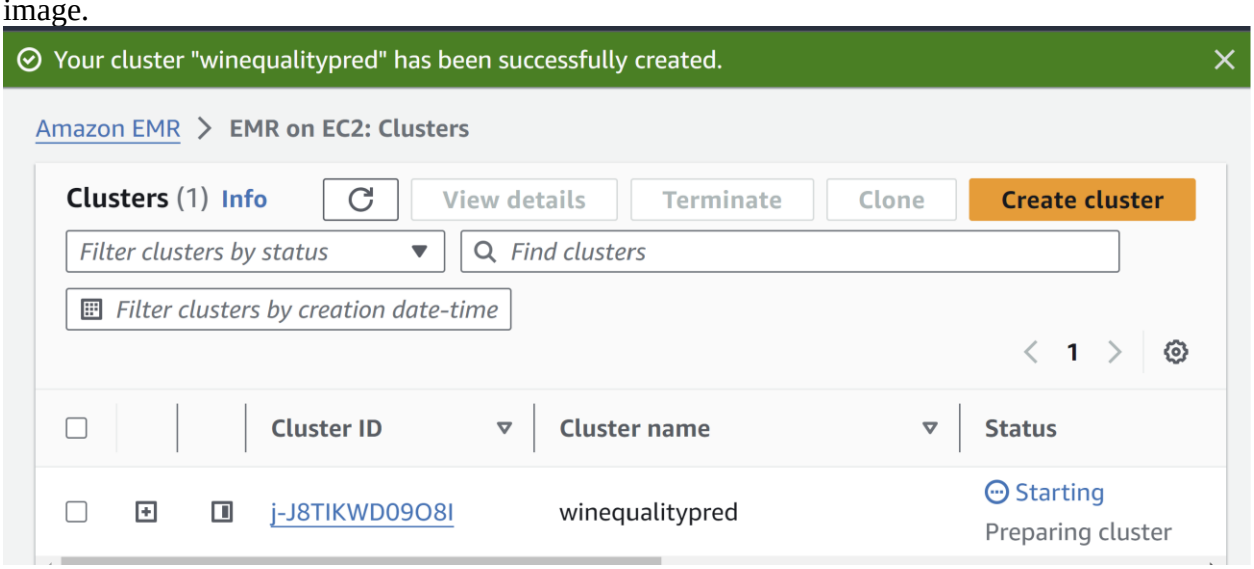
To access the cluster via SSH, create an EC2 key pair in AWS. Ensure that you securely store the .pem file associated with the key pair, as it will be required for establishing an SSH connection to the cluster.

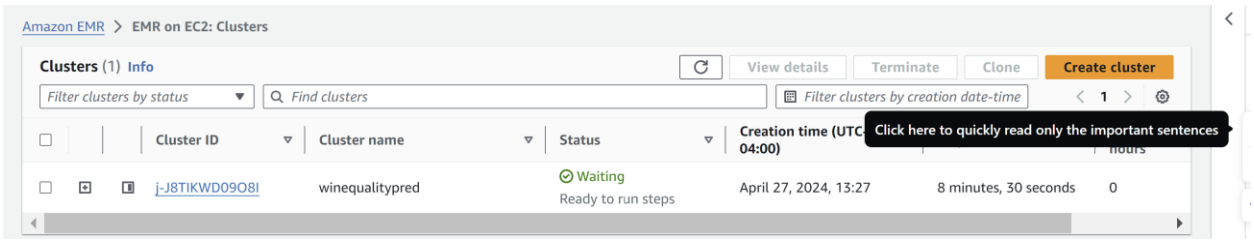


Choose the IAM Roles accordingly:

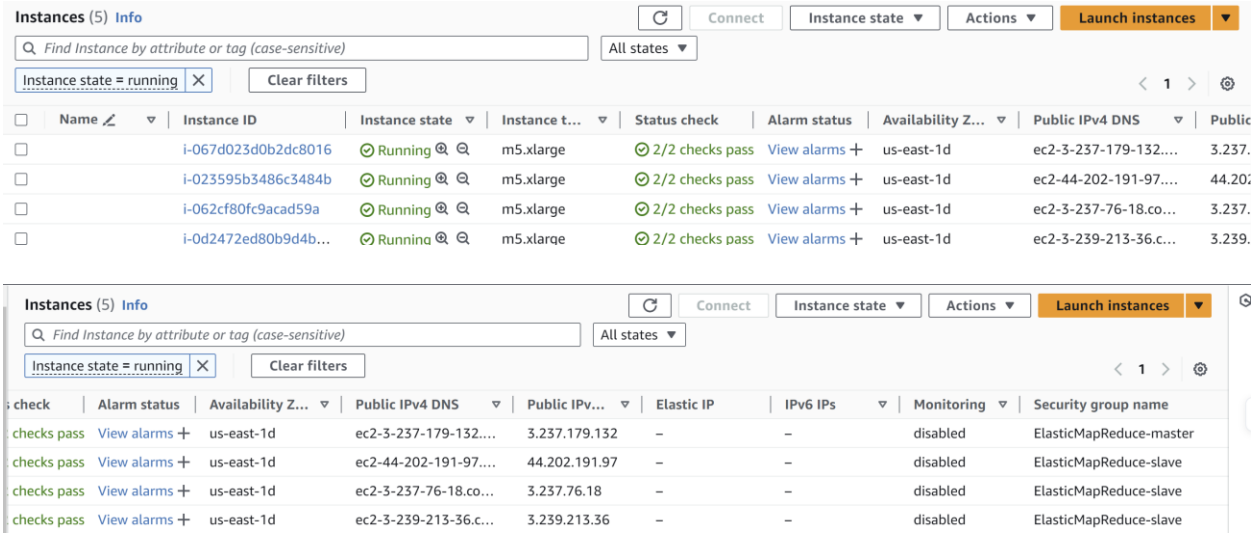


After initiating the cluster creation process, you can monitor its progress on the Clusters page of the EMR service. Initially, the cluster's status will be displayed as "Starting," indicating that it is in the process of being provisioned. Over time, the status will transition to "Waiting," signifying that the cluster has been successfully created and is ready for use, as illustrated in the provided image.

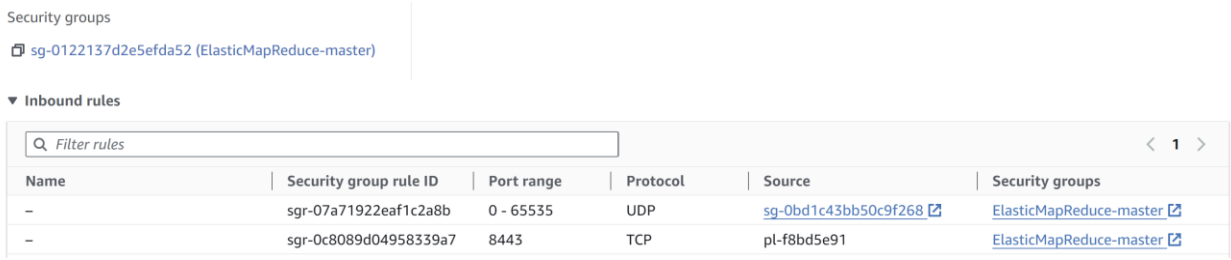




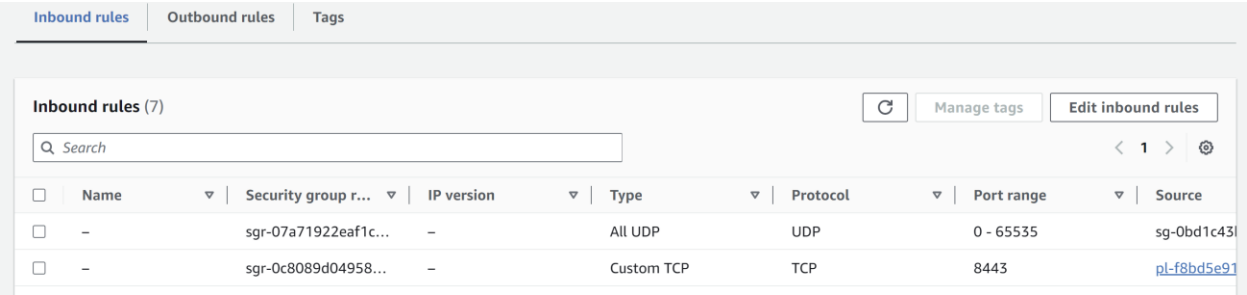
Navigate to the EC2 Instances page in the AWS Management Console. On this page, you will observe that a total of 4 EC2 instances have been launched as part of the EMR cluster creation process. Among these instances, one functions as the Master node, responsible for coordinating and managing the cluster operations. The remaining four instances serve as Slave nodes, dedicated to executing distributed tasks and computations within the cluster, as illustrated in the accompanying image.



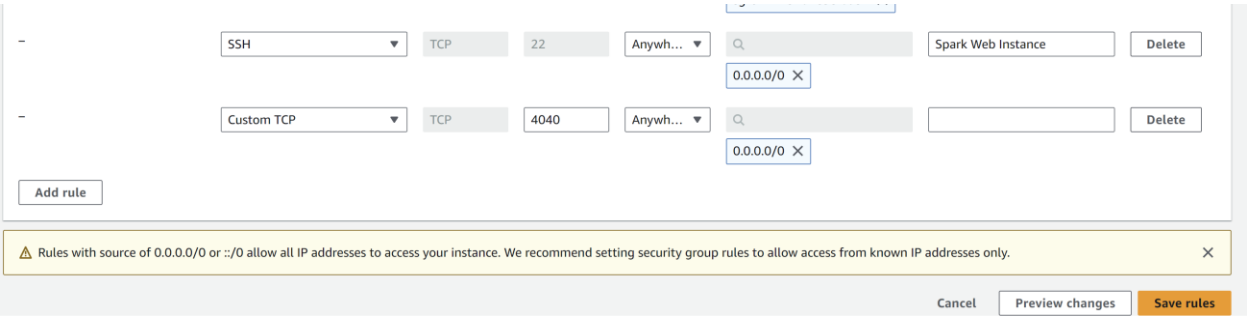
Within the EC2 service in the AWS Management Console, locate the security group named "ElasticMapReduce-Master" associated with the EMR cluster you created. Once you have identified this security group, click on its corresponding Security Group ID to access its configuration details.



Once you have accessed the configuration details of the "ElasticMapReduce-Master" security group, navigate to the section that displays the inbound network traffic rules. In this section, locate and click on the option that allows you to modify or edit the existing inbound rules.



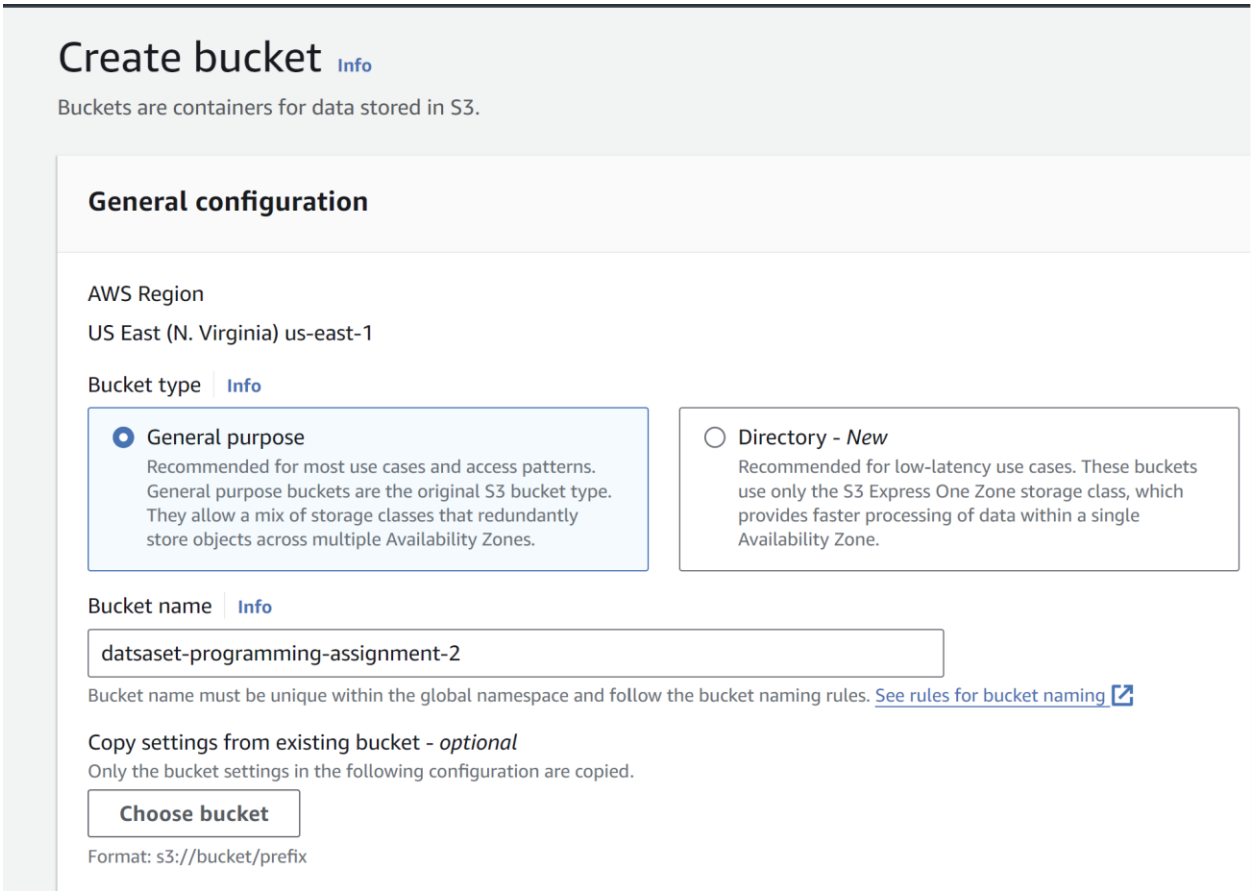
Within the inbound rules configuration section, locate the button or option that allows you to create a new rule. Click on this button, and then specify port numbers 22 and 4040 as the ports to be opened for inbound traffic. Configure any additional settings for these rules according to the provided instructions or screenshot. Once you have added these new rules, click on the "Save rules" button to apply the changes to the security group.



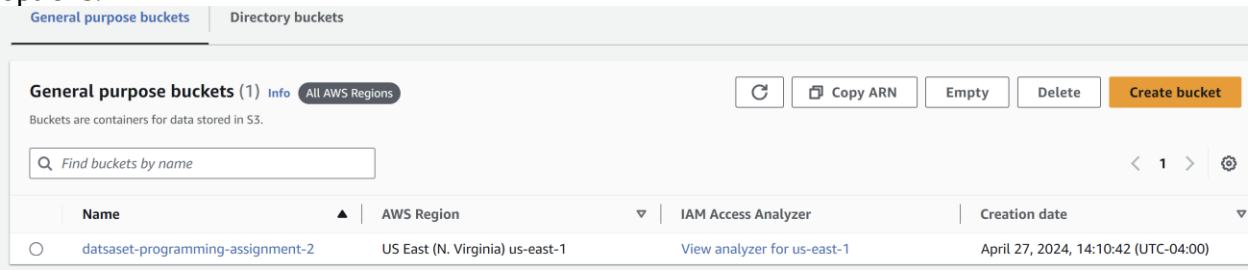
Next, navigate to the Amazon Simple Storage Service (S3) within the AWS Management Console. The purpose of this step is to create a new S3 bucket, which will serve as a storage location for the dataset files required.

Once you have accessed the Amazon S3 service, locate and click on the option or button labeled "Create Bucket" to initiate the process of creating a new S3 bucket.

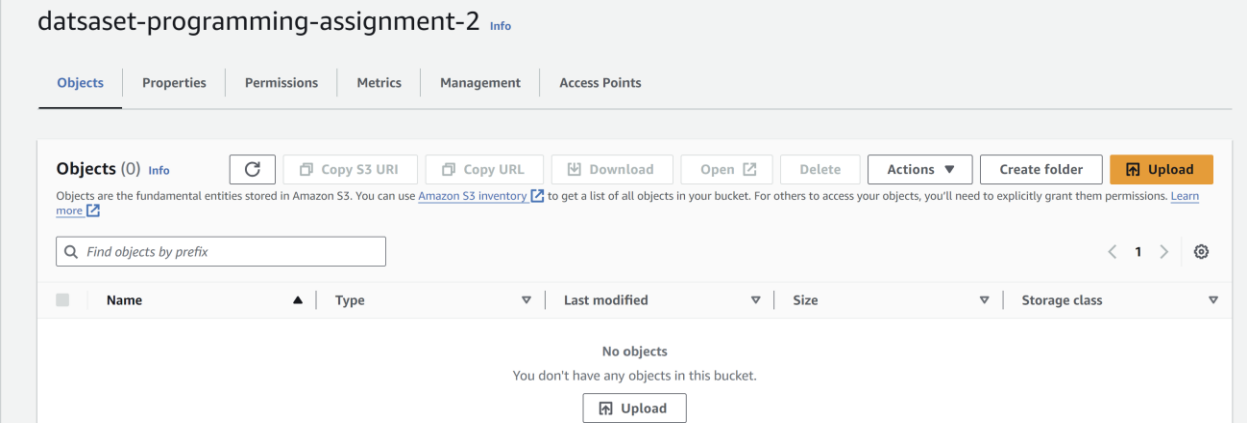
When prompted to provide a name for the new S3 bucket, enter "dataset-programming-assignment-2" as the bucket name. After specifying the bucket name, scroll down to the bottom of the page, and click on the "Create bucket" button to finalize the creation of the S3 bucket with the given name.



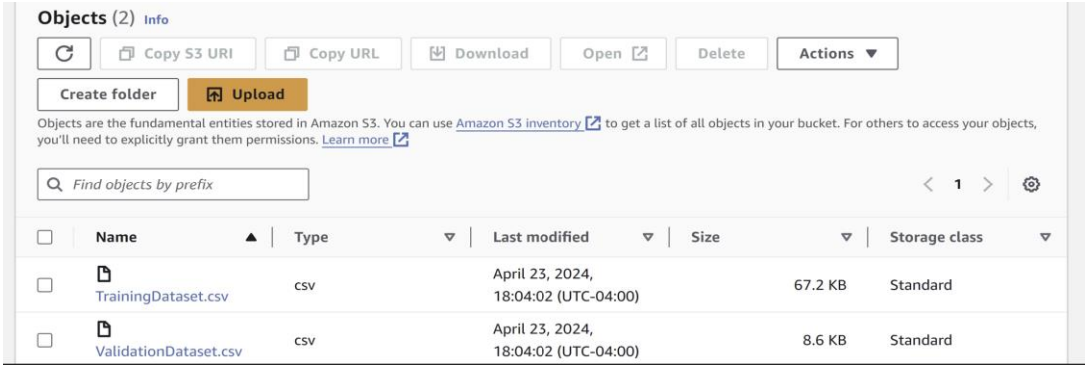
Locate the newly created S3 bucket named "dataset-programming-assignment-2" in the list of buckets displayed and click on its name to access the bucket's contents and configuration options.



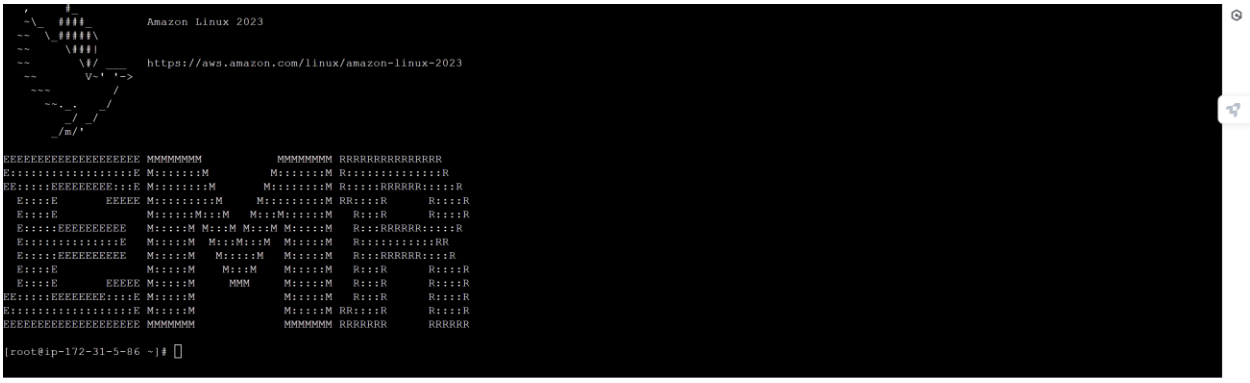
Once you have accessed the contents of the "dataset-programming-assignment-2" S3 bucket, look for the "Upload" button or option, and click on it to initiate the process of uploading files to the bucket.



From the upload interface, locate the option to add or select files for upload. Use this option to browse and choose the .csv files containing the dataset from the assignment. After selecting the desired files, proceed to click on the "Upload" button to initiate the transfer of the dataset files to the "dataset-programming-assignment-2" S3 bucket. Upon successful upload, you should have two .csv files stored in the S3 bucket, named "ValidationDataset.csv" and "TrainingDataset.csv," respectively.

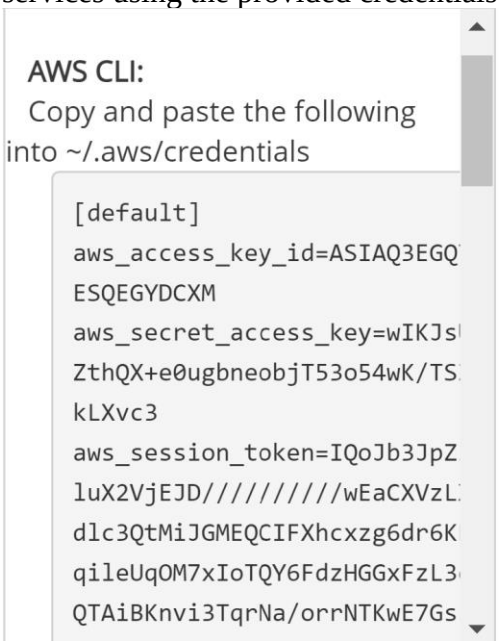


Goto EC2 instances (running), select the Instance ID which is corresponding to the master node. Click on that and select Connect. It will establish the connection as per below fig.

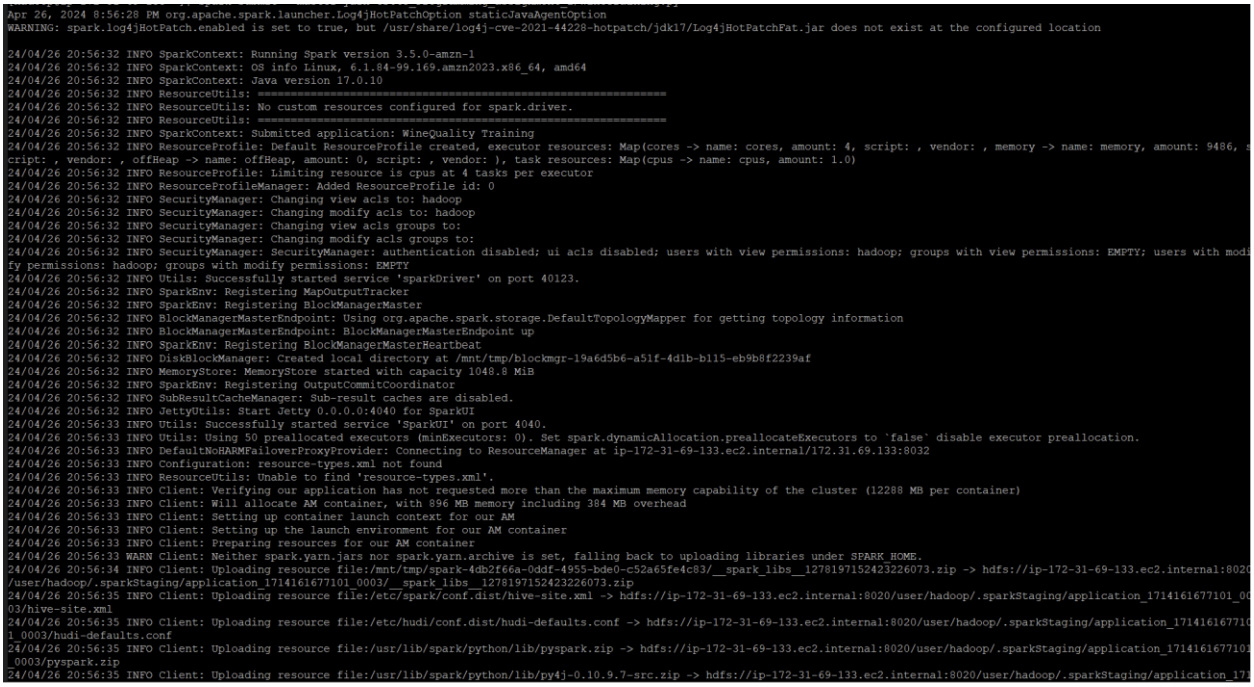


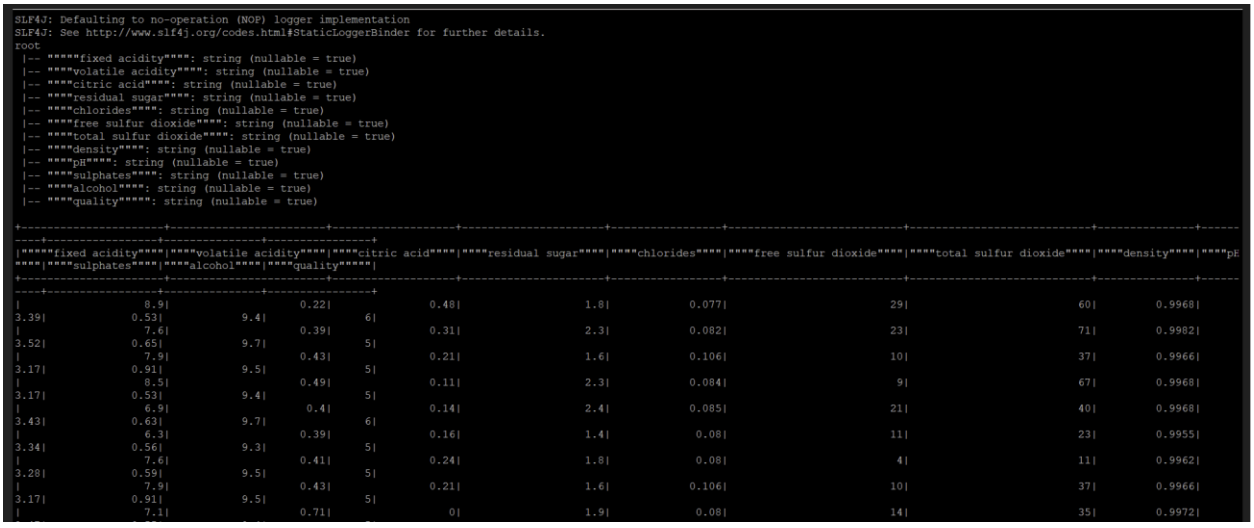
To set up the necessary credentials for accessing AWS services from the Master node EC2 instance, we need to configure the AWS credentials on that instance. Follow these steps:
Open a terminal window and connect to the Master node EC2 instance using SSH.
In the terminal of the Master node, execute the following command to create a new directory named. **aws": `mkdir .aws**
Next, create an empty file named "credentials" inside the ".aws" directory by running the command: **touch .aws/credentials**
Open the "credentials" file in a text editor by executing: **vi .aws/credentials**

You can now paste your AWS access credentials (access key ID and secret access key) into this file and save the changes. This will allow the Master node instance to authenticate with AWS services using the provided credentials.



sudo yum update: This command updates all the installed packages on your system to their latest versions using the YUM package manager.
sudo yum install git: This command installs Git, a distributed version control system, on your system using the YUM package manager.
pip install pyspark findspark boto3 numpy pandas scikit-learn datetime
git clone https://github.com/sravyakganti/Wine-quality-prediction-.git
By executing this command, Git will download a complete copy of the remote repository, including all its files and revision history, and create a new directory named "CS643_Programming_assignment_2" in your current working directory. This local copy allows you to work on the project files, make changes, and potentially contribute those changes back to the remote repository.
Now, execute the below commands to launch the Spark Application.
spark-submit --master yarn CS643_Programming_assignment_2/WineTraining.py
spark-submit --master yarn CS643_Programming_assignment_2/WineTesting.py > output.txt





Now, execute the following command to see the results:

grep F1 cat output.txt

The outcomes of the applied machine learning methods, including accuracy and F1 scores, are displayed below.

```
Validation Training Set Metrics
+-----------------------------------------------------------+-----+----------+
|features                                                   |label|prediction|
+-----------------------------------------------------------+-----+----------+
|[9.4,0.56,3.51,0.9978,11.0,34.0,0.076,1.9,0.0,0.7,7.4]    |5.0  |5.0       |
|[9.8,0.68,3.2,0.9968,25.0,67.0,0.098,2.6,0.0,0.88,7.8]    |5.0  |5.0       |
|[9.8,0.65,3.26,0.997,15.0,54.0,0.092,2.3,0.04,0.76,7.8]   |5.0  |5.0       |
|[9.8,0.58,3.16,0.998,17.0,60.0,0.075,1.9,0.56,0.28,11.2]  |6.0  |5.0       |
|[9.4,0.56,3.51,0.9978,11.0,34.0,0.076,1.9,0.0,0.7,7.4]    |5.0  |5.0       |
+-----------------------------------------------------------+-----+----------+
only showing top 5 rows

The accuracy of the model is 0.575
F1: 0.5619407071339173
```

```
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
TestingDataSet Metrics

+-----------------------------------------------------------+-----+----------+
|features                                                   |label|prediction|
+-----------------------------------------------------------+-----+----------+
|[9.4,0.56,3.51,0.9978,11.0,34.0,0.076,1.9,0.0,0.7,7.4]    |5    |5.0       |
|[9.8,0.68,3.2,0.9968,25.0,67.0,0.098,2.6,0.0,0.88,7.8]    |5    |5.0       |
|[9.8,0.65,3.26,0.997,15.0,54.0,0.092,2.3,0.04,0.76,7.8]   |5    |5.0       |
|[9.8,0.58,3.16,0.998,17.0,60.0,0.075,1.9,0.56,0.28,11.2]  |6    |5.0       |
|[9.4,0.56,3.51,0.9978,11.0,34.0,0.076,1.9,0.0,0.7,7.4]    |5    |5.0       |
+-----------------------------------------------------------+-----+----------+
only showing top 5 rows


The accuracy of the model is 0.6271186440677966
F1: 0.593151718932272
```

DOCKER IMPLEMENTATION –

Steps for Installing Docker:

sudo yum update -y
sudo service docker start
start docker.service

```
[root@ip-172-31-5-86 ~]# sudo yum update -y
Last metadata expiration check: 2:32:04 ago on Sat Apr 27 17:28:59 2024.
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-172-31-5-86 ~]# sudo yum install -y docker
Last metadata expiration check: 2:32:18 ago on Sat Apr 27 17:28:59 2024.
Package docker-25.0.3-1.amzn2023.0.1.x86_64 is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-172-31-5-86 ~]# sudo service docker start
Redirecting to /bin/systemctl start docker.service
[root@ip-172-31-5-86 ~]# sudo service docker status
Redirecting to /bin/systemctl status docker.service
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; preset: disabled)
     Active: active (running) since Sat 2024-04-27 20:01:36 UTC; 51s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
    Process: 104192 ExecStartPre=/bin/mkdir -p /run/docker (code=exited, status=0/SUCCESS)
    Process: 104193 ExecStartPre=/usr/libexec/docker/docker-setup-runtimes.sh (code=exited, status=0/SUCCESS)
   Main PID: 104194 (dockerd)
      Tasks: 10
     Memory: 108.8M
        CPU: 367ms
     CGroup: /system.slice/docker.service
             └─104194 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32768:65536

Apr 27 20:01:35 ip-172-31-5-86.ec2.internal systemd[1]: Starting docker.service - Docker Application Container Engine...
Apr 27 20:01:36 ip-172-31-5-86.ec2.internal dockerd[104194]: time="2024-04-27T20:01:36.240827520Z" level=info msg="Starting up"
```

```
[root@ip-172-31-5-86 ~]# sudo service docker start
Redirecting to /bin/systemctl start docker.service
[root@ip-172-31-5-86 ~]# sudo service docker status
Redirecting to /bin/systemctl status docker.service
● docker.service - Docker Application Container Engine
     Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; preset: disabled)
     Active: active (running) since Sat 2024-04-27 20:01:36 UTC; 51s ago
TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
    Process: 104192 ExecStartPre=/bin/mkdir -p /run/docker (code=exited, status=0/SUCCESS)
    Process: 104193 ExecStartPre=/usr/libexec/docker/docker-setup-runtimes.sh (code=exited, status=0/SUCCESS)
   Main PID: 104194 (dockerd)
      Tasks: 10
     Memory: 108.8M
        CPU: 367ms
     CGroup: /system.slice/docker.service
             └─104194 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32768:65536

Apr 27 20:01:35 ip-172-31-5-86.ec2.internal systemd[1]: Starting docker.service - Docker Application Container Engine...
Apr 27 20:01:36 ip-172-31-5-86.ec2.internal dockerd[104194]: time="2024-04-27T20:01:36.240827520Z" level=info msg="Starting up"
Apr 27 20:01:36 ip-172-31-5-86.ec2.internal dockerd[104194]: time="2024-04-27T20:01:36.458407840Z" level=info msg="Loading containers: start."
Apr 27 20:01:36 ip-172-31-5-86.ec2.internal dockerd[104194]: time="2024-04-27T20:01:36.847155017Z" level=info msg="Loading containers: done."
Apr 27 20:01:36 ip-172-31-5-86.ec2.internal dockerd[104194]: time="2024-04-27T20:01:36.910102916Z" level=info msg="Docker daemon" commit=f417435 containerd-snapshotter=false s
Apr 27 20:01:36 ip-172-31-5-86.ec2.internal dockerd[104194]: time="2024-04-27T20:01:36.910535531Z" level=info msg="Daemon has completed initialization"
Apr 27 20:01:36 ip-172-31-5-86.ec2.internal dockerd[104194]: time="2024-04-27T20:01:36.963477443Z" level=info msg="API listen on /run/docker.sock"
Apr 27 20:01:36 ip-172-31-5-86.ec2.internal systemd[1]: Started docker.service - Docker Application Container Engine.
```

To create a Docker container image from the provided Dockerfile, execute the following command in your terminal while in the same directory as the Dockerfile.

sudo docker build -t sravyakganti/cs643-programming-assignment-2 .

Executing the provided command will initiate the process of constructing a Docker image based on the instructions specified in the Dockerfile, and the resulting image will be locally stored and available within your EC2 instance.

After attempting to build the Docker image, you can verify its successful creation by running the following command: **sudo docker image ls**. This command will display a list of all Docker images present on your EC2 instance, allowing you to confirm the existence of the newly built image among the listed images.

```
[root@ip-172-31-5-86 CS643_Programming_assignment_2]# docker image ls
REPOSITORY                                TAG       IMAGE ID       CREATED         SIZE
sk3568/cs643-programming-assignment-2   latest    2d7b4954c93e   3 minutes ago   2.42GB
```

To upload and store the Docker image you've created on the Docker Hub repository, execute the following instruction in your terminal or command prompt.

Run the following command to launch this Docker image:

sudo docker run -it sravyakganti/cs643-programming-assignment-2

Here, you have the option to utilize your image ID in place of the image name.

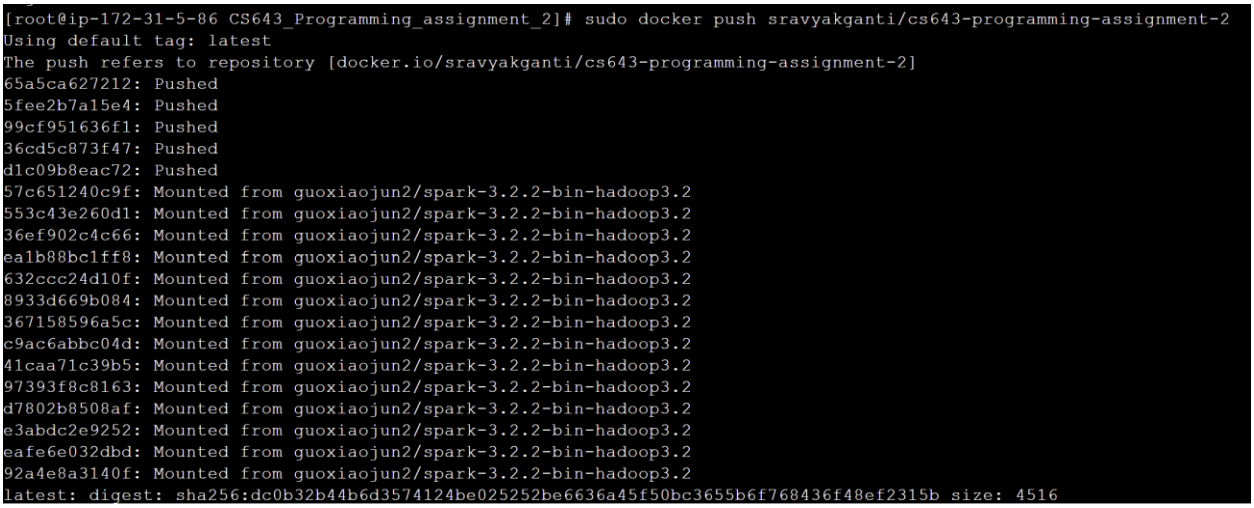
sudo docker run -it <IMAGE_ID>

```
TestingDataSet Metrics

+--------------------------------------------------------------+-----+----------+
|features                                                      |label|prediction|
+--------------------------------------------------------------+-----+----------+
|[9.4,0.56,3.51,0.9978,11.0,34.0,0.076,1.9,0.0,0.7,7.4]       |5.0  |5.0       |
|[9.8,0.68,3.2,0.9968,25.0,67.0,0.098,2.6,0.0,0.88,7.8]       |5.0  |5.0       |
|[9.8,0.65,3.26,0.997,15.0,54.0,0.092,2.3,0.04,0.76,7.8]      |5.0  |5.0       |
|[9.8,0.58,3.16,0.998,17.0,60.0,0.075,1.9,0.56,0.28,11.2]     |6.0  |5.0       |
|[9.4,0.56,3.51,0.9978,11.0,34.0,0.076,1.9,0.0,0.7,7.4]       |5.0  |5.0       |
+--------------------------------------------------------------+-----+----------+
only showing top 5 rows


The accuracy of the model is 0.6271186440677966
F1: 0.593151718932272
```

Login into Docker Hub credentials in terminal

docker login

sudo docker push sravyakganti/cs643-programming-assignment-2



Now that you have the docker image downloaded from the DockerHub repository, you may start it by following the directions on the DockerHub website.