CS643 CLOUD PROGRAMMING ASSIGNMENT 2

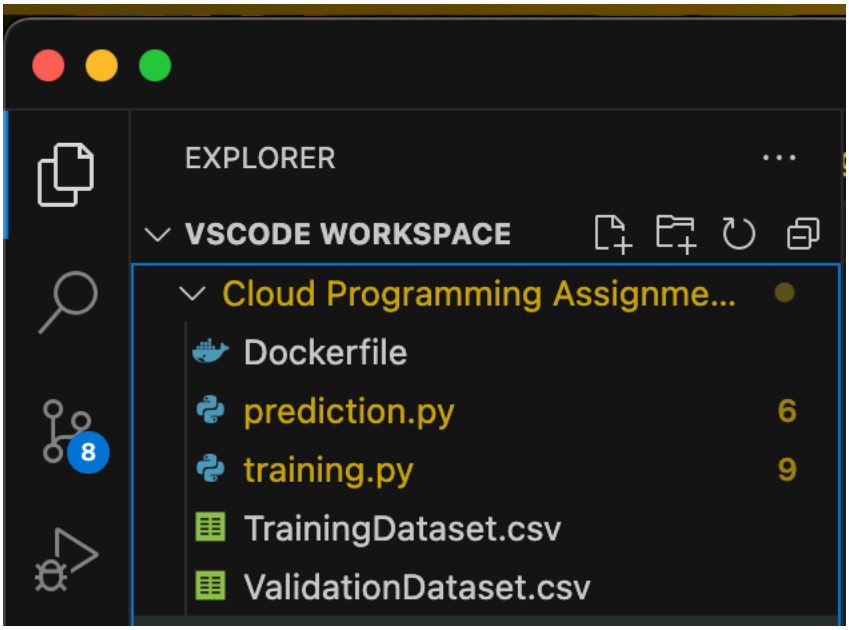
Goal: The purpose of this individual assignment is to learn how to develop parallel machine learning (ML) applications in Amazon AWS cloud platform. Specifically, you will learn: (1) how to use Apache Spark to train an ML model in parallel on multiple EC2 instances; (2) how to use Spark's MLlib to develop and use an ML model in the cloud; (3) How to use Docker to create a container for your ML model to simplify model deployment.

Description: You have to build a wine quality prediction ML model in Spark over AWS. The model must be trained in parallel using 4 EC2 instances. Then, you need to save and load the model in a Spark application that will perform wine quality prediction; this application will run on one EC2 instance. The assignment must be implemented in Java on Ubuntu Linux. The details of the assignment are presented below:

- Input for model training: we share 2 datasets with you for your ML model. Both datasets are available in Canvas, under Programming Assignment 2.
    - TrainingDataset.csv: you will use this dataset to train the model in parallel on multiple EC2 instances.
    - ValidationDataset.csv: you will use this dataset to validate the model and optimize its performance (i.e., select the best values for the model parameters).
- Input for prediction testing: TestDataset.csv. We will use this file, which has a similar structure with the two datasets above, to test the functionality and performance of your prediction application. Your prediction application should take such a file as input. This file is not shared with you, but you can use the validation dataset to make sure your application works.
- Output: The output of your application will be a measure of the prediction performance, specifically the F1 score, which is available in MLlib.
- Model Implementation: You have to develop a Spark application that uses MLlib to train for wine quality prediction using the training dataset. You will use the validation dataset check the performance of your trained model and to potentially tune your ML model parameters for best performance. You should start with a simple linear regression or logistic regression model from MLlib, but you can try multiple ML models to see which one leads to better performance. For classification models, you can use 10 classes (the wine scores are from 1 to 10). Note: there will be extra-credit for the top 5 applications/students in terms of prediction performance (see below under grading).
- Docker container: You have to build a Docker container for your prediction application. In this way, the prediction model can be quickly deployed across many different environments.
- The model training is done in parallel on 4 EC2 instances.
- The prediction with or without Docker is done on a single EC2 instance.

Solution:

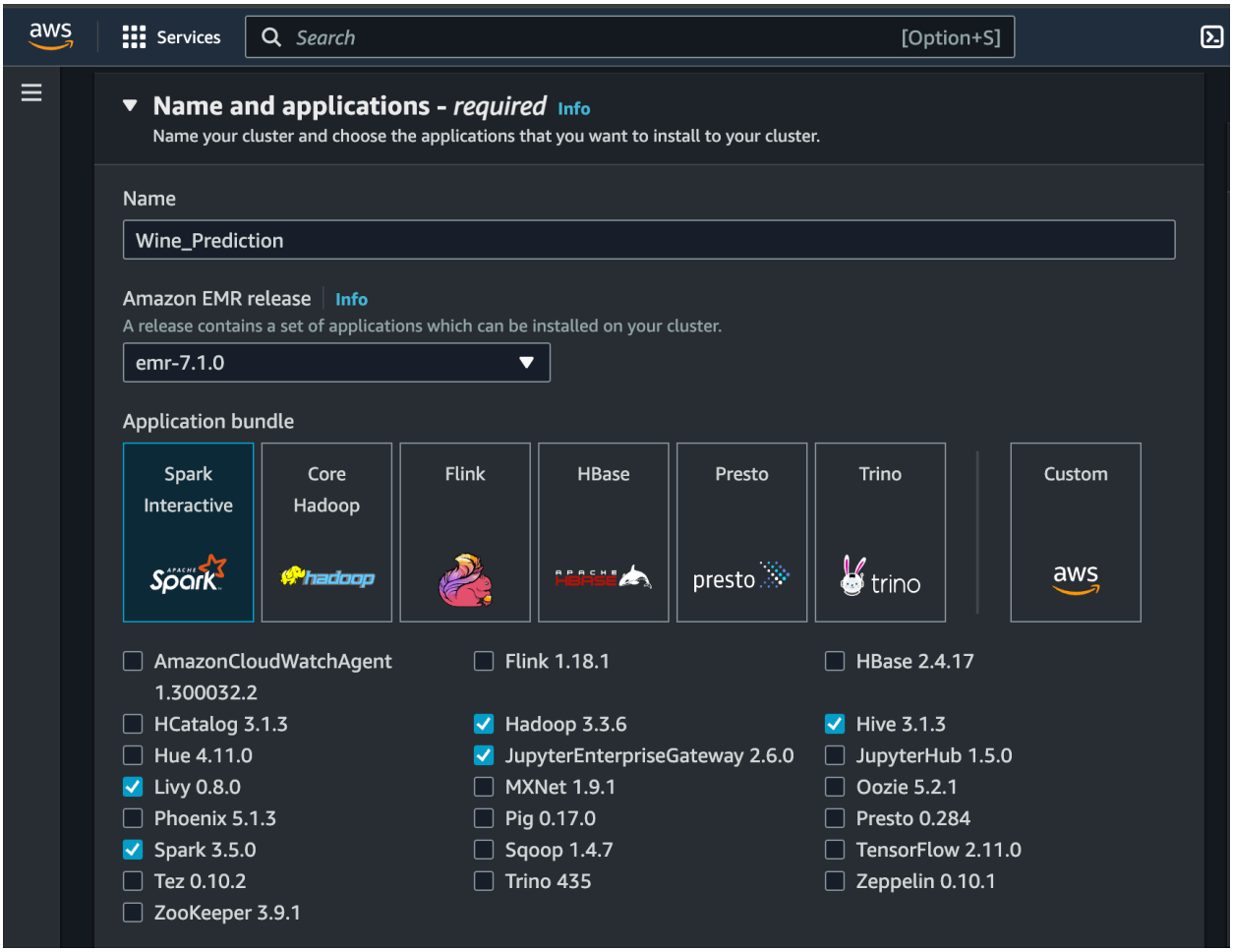
File structure of cloud programming assignment 2 in VSCODE

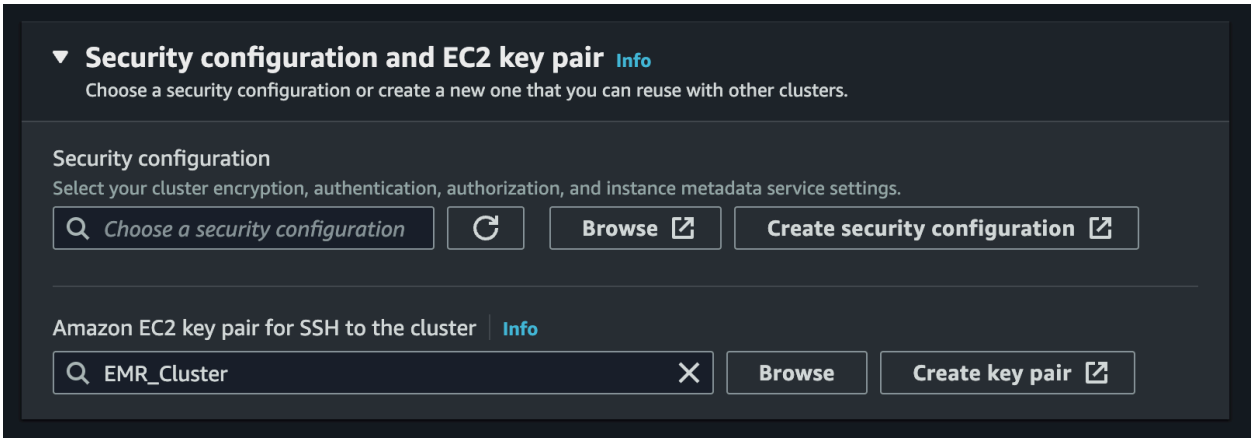


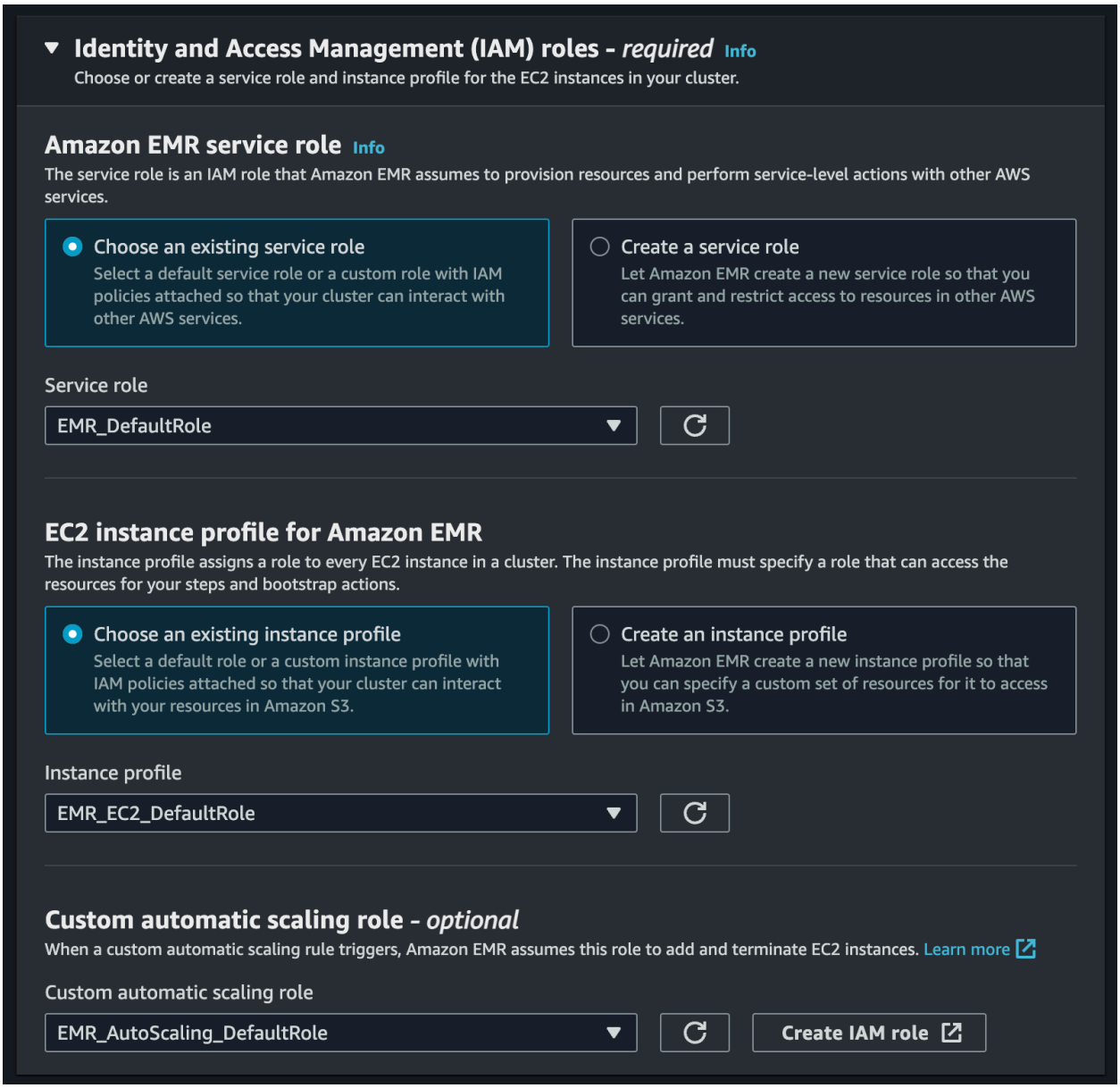
Create an EMR(Elastic Map Reduce Cluster) as shown in the steps below and include 4 task instance groups to run in parallel.

It create one core, master and 4 slave instances.
Core
Master
Task 1
Task 2
Task 3
Task 4

Let everything else be default
create a key pair .pem file
Select IAM roles as specified in the images


aws
Services
Search
[Option+S]
Name and applications - required Info
Name your cluster and choose the applications that you want to install to your cluster.
Name
Wine_Prediction
Amazon EMR release Info
A release contains a set of applications which can be installed on your cluster.
emr-7.1.0
Application bundle
Spark Interactive
Core Hadoop
Flink
HBase
Presto
Trino
Custom
AmazonCloudWatchAgent 1.300032.2
Flink 1.18.1
HBase 2.4.17
HCatalog 3.1.3
Hadoop 3.3.6
Hive 3.1.3
Hue 4.11.0
JupyterEnterpriseGateway 2.6.0
JupyterHub 1.5.0
Livy 0.8.0
MXNet 1.9.1
Oozie 5.2.1
Phoenix 5.1.3
Pig 0.17.0
Presto 0.284
Spark 3.5.0
Sqoop 1.4.7
TensorFlow 2.11.0
Tez 0.10.2
Trino 435
Zeppelin 0.10.1
ZooKeeper 3.9.1


Security configuration and EC2 key pair Info
Choose a security configuration or create a new one that you can reuse with other clusters.
Security configuration
Select your cluster encryption, authentication, authorization, and instance metadata service settings.
Choose a security configuration
Browse
Create security configuration
Amazon EC2 key pair for SSH to the cluster Info
EMR_Cluster
Browse
Create key pair

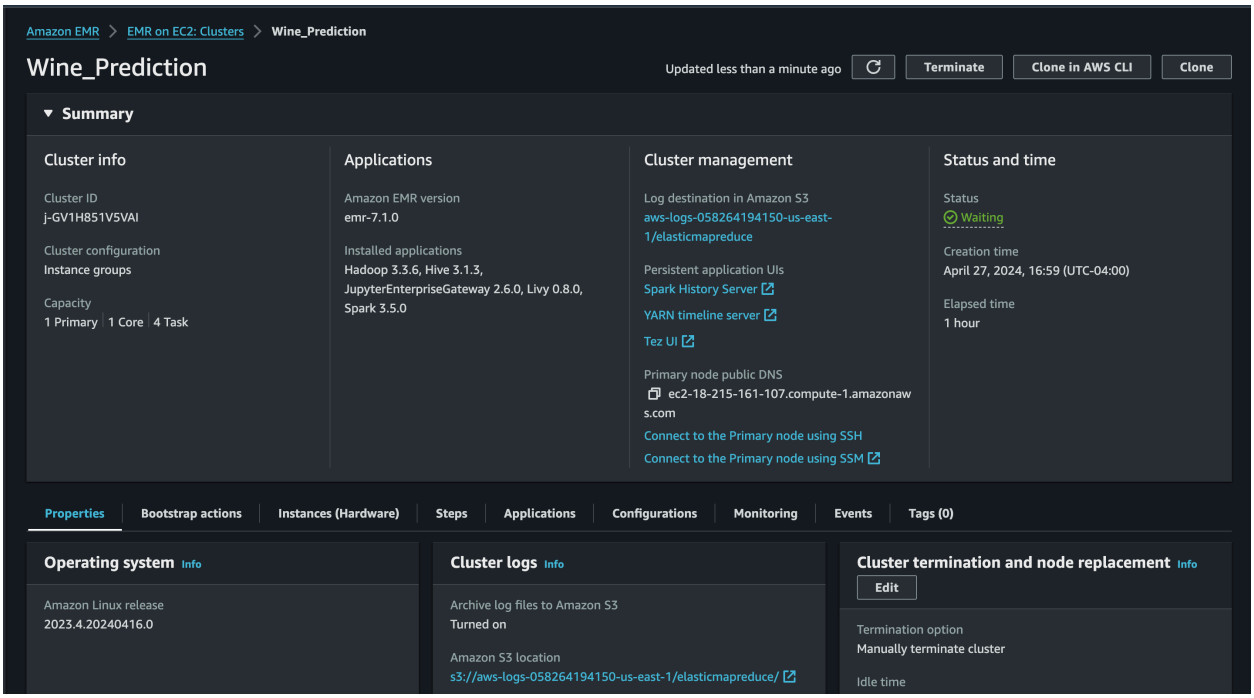


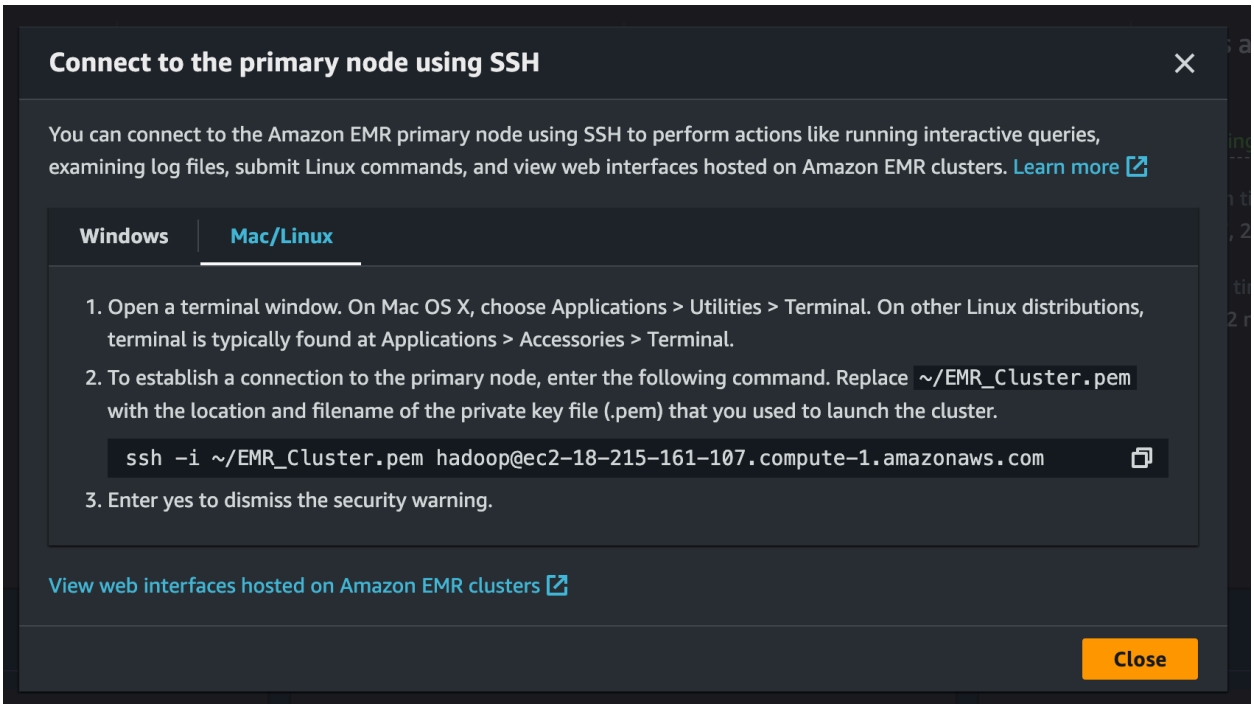
Open EMR in AWS and you can see the clusters
Go to EC2 console and check which instance is master and edit the inbound rules under security tab
Add rule -> select SSH and MyIP in two drop-downs and then select save.

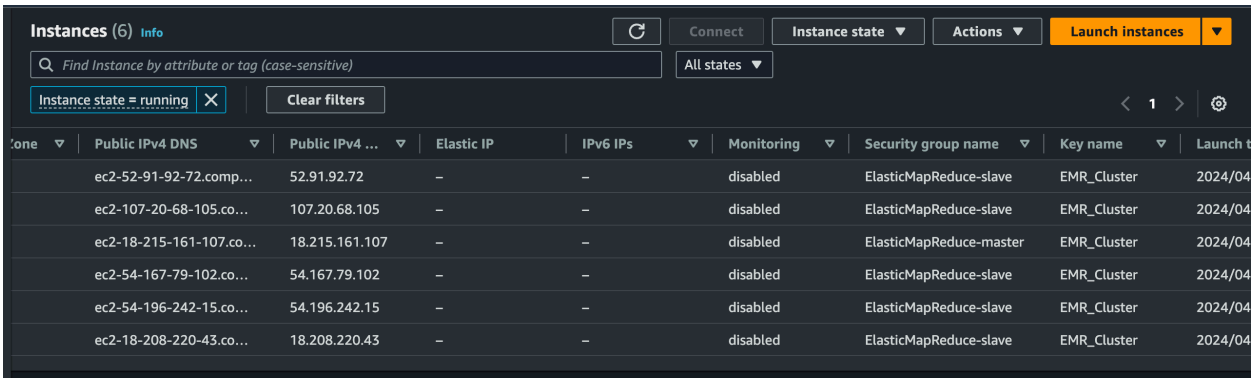
Move to Cluster Wine_Prediction now and you can see link to
“connect to the Primary node using SSH”

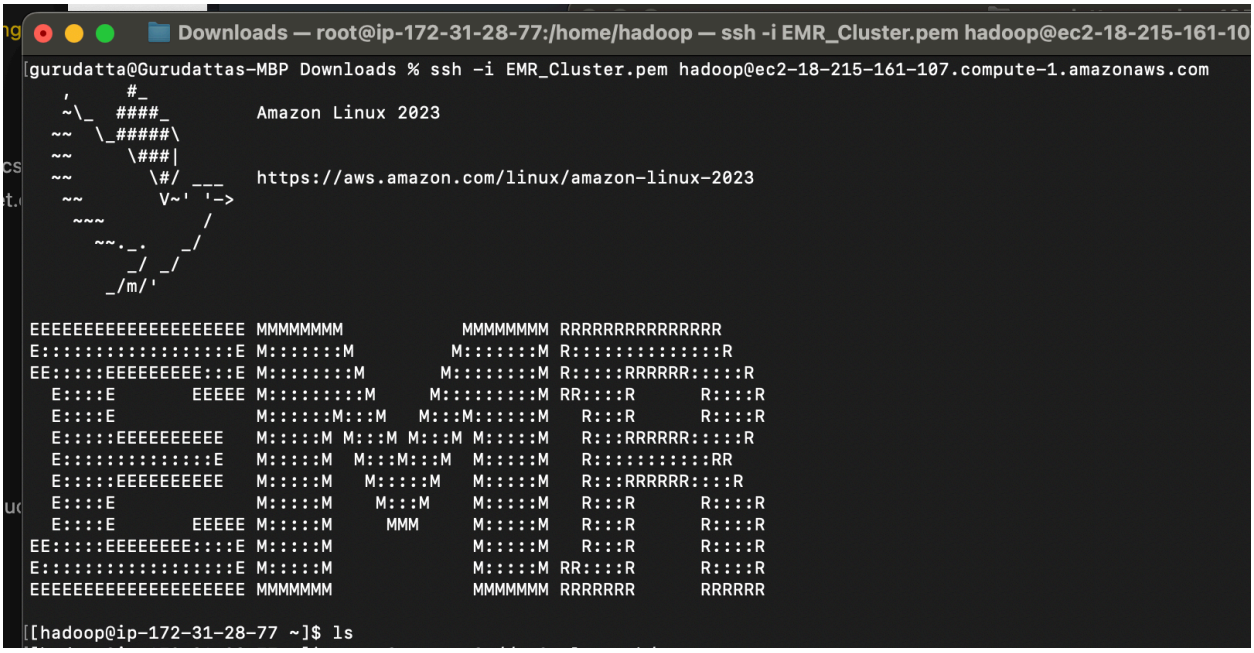
Move to folder that has key pair in terminal and execute below commands

chmod 400 EMR_Cluster.pem
ssh -i EMR_Cluster.pem
hadoop@ec2-18-215-161-107.compute-1.amazonaws.com



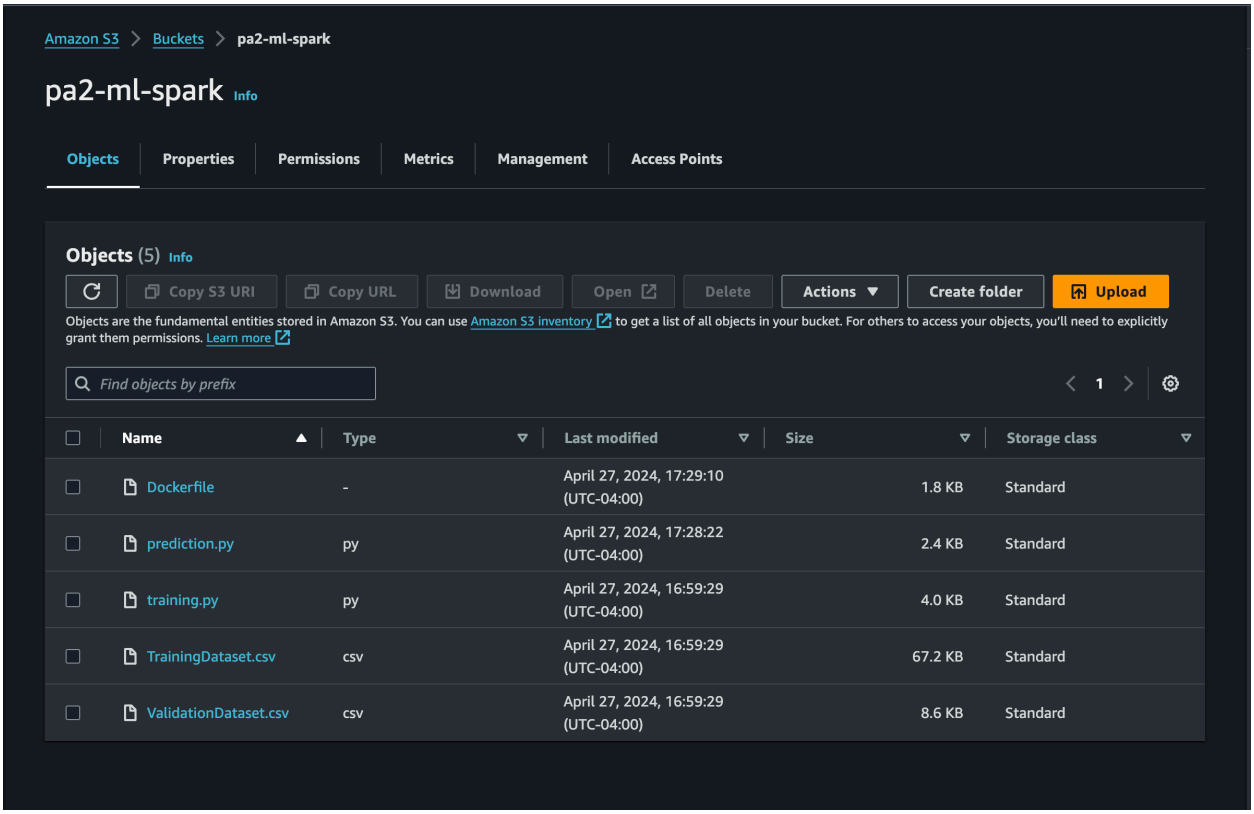






Go to S3 Console and create S3 Bucket “pa2-ml-spark” and upload all required files into the bucket

upload all the code files and datasets and docker file as we will be running everything inside EMR cluster master ec2 that we just have created.



Execution without Docker

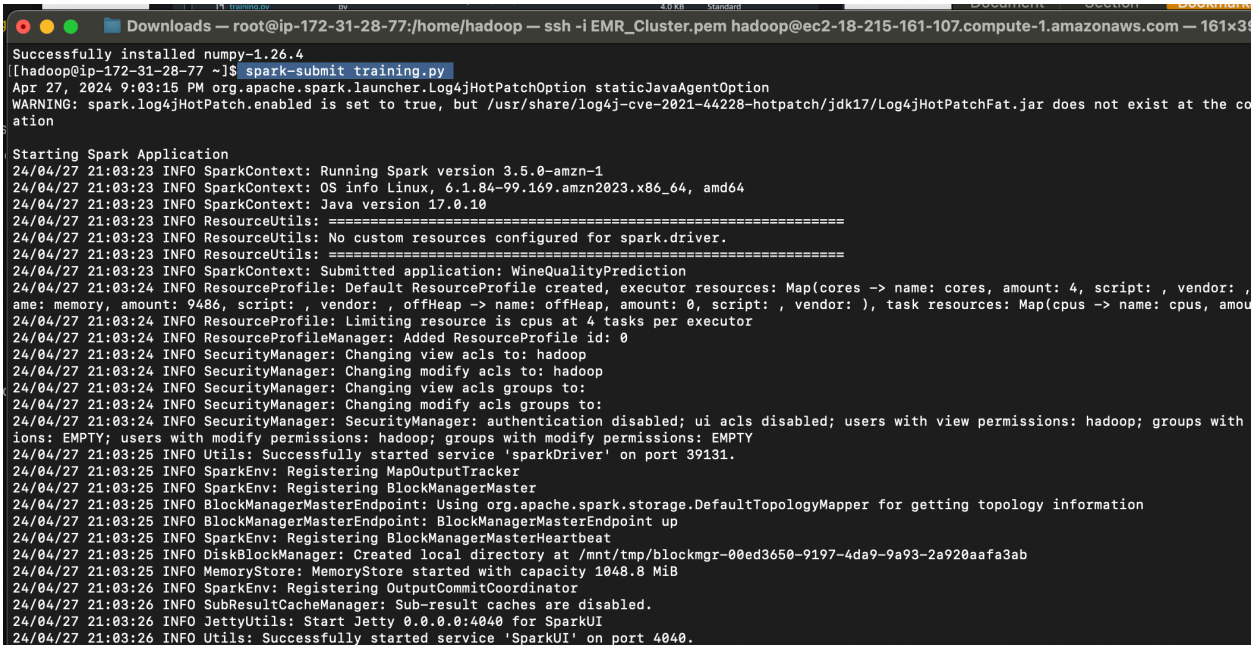
Go back to terminal now and type in the below commands

sudo su -> for changing to the root (optional)
aws s3 sync s3://pa2-ml-spark/ . -> fetching all code files into the instance
ls -> cross check if all files are dumped into instance

```
[[hadoop@ip-172-31-28-77 ~]$ ls
[[hadoop@ip-172-31-28-77 ~]$ aws s3 sync s3://pa2-ml-spark/ .
download: s3://pa2-ml-spark/TrainingDataset.csv to ./TrainingDataset.csv
download: s3://pa2-ml-spark/prediction.py to ./prediction.py
download: s3://pa2-ml-spark/Dockerfile to ./Dockerfile
download: s3://pa2-ml-spark/training.py to ./training.py
download: s3://pa2-ml-spark/ValidationDataset.csv to ./ValidationDataset.csv
[[hadoop@ip-172-31-28-77 ~]$ ls
Dockerfile  TrainingDataset.csv  ValidationDataset.csv  prediction.py  training.py
```

As this assignment requires to run the ML code of Wine_Quality prediction using SPARK ML Libraries, we use the below command

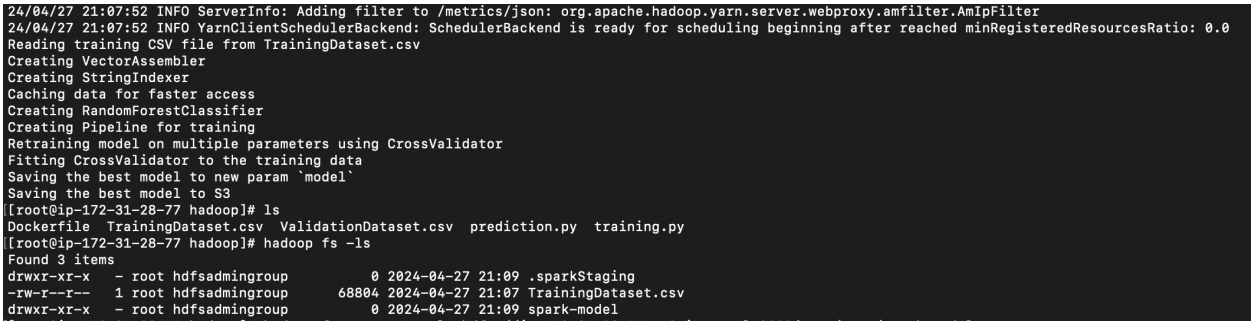
Pip install numpy —user -> for installing pyhthon numpy into instance
spark-submit training.py -> running training.py to generate a training model
In our case it creates a trained model with name spark-model



For any Hadoop related path issues

refer the below
hadoop fs -copyFromLocal TrainingDataset.csv hdfs://
ip-172-31-28-77.ec2.internal:8020/user/root/

Below images shows the list of files and generated ML model spark-model in hdfs



Now push ValidationDataset.csv to hdfs
hadoop fs -copyFromLocal ValidationDataset.csv hdfs://
ip-172-31-28-77.ec2.internal:8020/user/root/

```
[[root@ip-172-31-28-77 hadoop]# hadoop fs -copyFromLocal ValidationDataset.csv hdfs://ip-172-31-28-77.ec2.internal:8020/user/root/
[[root@ip-172-31-28-77 hadoop]# spark-submit prediction.py
Apr 27, 2024 9:12:59 PM org.apache.spark.launcher.Log4jHotPatchOption staticJavaAgentOption
WARNING: spark.log4jHotPatch.enabled is set to true, but /usr/share/log4j-cve-2021-44228-hotpatch/jdk17/Log4jHotPatchFat.jar does not exist at
ation

Starting Spark Application
24/04/27 21:13:02 INFO SparkContext: Running Spark version 3.5.0-amzn-1
24/04/27 21:13:02 INFO SparkContext: OS info Linux, 6.1.84-99.169.amzn2023.x86_64, amd64
24/04/27 21:13:02 INFO SparkContext: Java version 17.0.10
24/04/27 21:13:03 INFO ResourceUtils: ==============================================================
24/04/27 21:13:03 INFO ResourceUtils: No custom resources configured for spark.driver.
24/04/27 21:13:03 INFO ResourceUtils: ==============================================================
24/04/27 21:13:03 INFO SparkContext: Submitted application: WineQualityPrediction
24/04/27 21:13:03 INFO ResourceProfile: Default ResourceProfile created, executor resources: Map(cores -> name: cores, amount: 4, script: , ve
ame: memory, amount: 9486, script: , vendor: , offHeap -> name: offHeap, amount: 0, script: , vendor: ), task resources: Map(cpus -> name: cpu
24/04/27 21:13:03 INFO ResourceProfile: Limiting resource is cpus at 4 tasks per executor
24/04/27 21:13:03 INFO ResourceProfileManager: Added ResourceProfile id: 0
24/04/27 21:13:03 INFO SecurityManager: Changing view acls to: root
24/04/27 21:13:03 INFO SecurityManager: Changing modify acls to: root
24/04/27 21:13:03 INFO SecurityManager: Changing view acls groups to:
24/04/27 21:13:03 INFO SecurityManager: Changing modify acls groups to:
24/04/27 21:13:03 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: root; groups
ns: EMPTY; users with modify permissions: root; groups with modify permissions: EMPTY
24/04/27 21:13:03 INFO Utils: Successfully started service 'sparkDriver' on port 39923.
24/04/27 21:13:03 INFO SparkEnv: Registering MapOutputTracker
24/04/27 21:13:03 INFO SparkEnv: Registering BlockManagerMaster
24/04/27 21:13:03 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
24/04/27 21:13:03 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
24/04/27 21:13:03 INFO SparkEnv: Registering BlockManagerMasterHeartbeat
```

The below image shows the prediction.py execution with validation dataset and shows up the accuracy score as below

Test Accuracy of wine prediction model = 0.96875

```
/usr/lib/spark/python/lib/pyspark.zip/pyspark/sql/context.py:158:
FutureWarning: Deprecated in 3.0.0. Use
SparkSession.builder.getOrCreate() instead.
Weighted F1 Score of wine prediction model = 0.9541901629072682
Exiting Spark Application
```

```
24/04/27 21:13:14 INFO ServerInfo: Adding filter to /stages: org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter
24/04/27 21:13:14 INFO ServerInfo: Adding filter to /stages/json: org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter
24/04/27 21:13:14 INFO ServerInfo: Adding filter to /stages/stage: org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter
24/04/27 21:13:14 INFO ServerInfo: Adding filter to /stages/stage/json: org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter
24/04/27 21:13:14 INFO ServerInfo: Adding filter to /stages/pool: org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter
24/04/27 21:13:14 INFO ServerInfo: Adding filter to /stages/pool/json: org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter
24/04/27 21:13:14 INFO ServerInfo: Adding filter to /storage: org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter
24/04/27 21:13:14 INFO ServerInfo: Adding filter to /storage/json: org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter
24/04/27 21:13:14 INFO ServerInfo: Adding filter to /storage/rdd: org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter
24/04/27 21:13:14 INFO ServerInfo: Adding filter to /storage/rdd/json: org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter
24/04/27 21:13:14 INFO ServerInfo: Adding filter to /environment: org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter
24/04/27 21:13:14 INFO ServerInfo: Adding filter to /environment/json: org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter
24/04/27 21:13:14 INFO ServerInfo: Adding filter to /executors: org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter
24/04/27 21:13:14 INFO ServerInfo: Adding filter to /executors/json: org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter
24/04/27 21:13:14 INFO ServerInfo: Adding filter to /executors/threadDump: org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter
24/04/27 21:13:14 INFO ServerInfo: Adding filter to /executors/threadDump/json: org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter
24/04/27 21:13:14 INFO ServerInfo: Adding filter to /executors/heapHistogram: org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter
24/04/27 21:13:14 INFO ServerInfo: Adding filter to /executors/heapHistogram/json: org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter
24/04/27 21:13:14 INFO ServerInfo: Adding filter to /static: org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter
24/04/27 21:13:14 INFO ServerInfo: Adding filter to /: org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter
24/04/27 21:13:14 INFO ServerInfo: Adding filter to /api: org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter
24/04/27 21:13:14 INFO ServerInfo: Adding filter to /jobs/job/kill: org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter
24/04/27 21:13:14 INFO ServerInfo: Adding filter to /stages/stage/kill: org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter
24/04/27 21:13:14 INFO ServerInfo: Adding filter to /metrics/json: org.apache.hadoop.yarn.server.webproxy.amfilter.AmIpFilter
24/04/27 21:13:14 INFO YarnClientSchedulerBackend: SchedulerBackend is ready for scheduling beginning after reached minRegisteredResourcesRatio: 0.0
Test Accuracy of wine prediction model = 0.96875
/usr/lib/spark/python/lib/pyspark.zip/pyspark/sql/context.py:158: FutureWarning: Deprecated in 3.0.0. Use SparkSession.builder.getOrCreate() instead.
Weighted F1 Score of wine prediction model = 0.9541901629072682
Exiting Spark Application
```

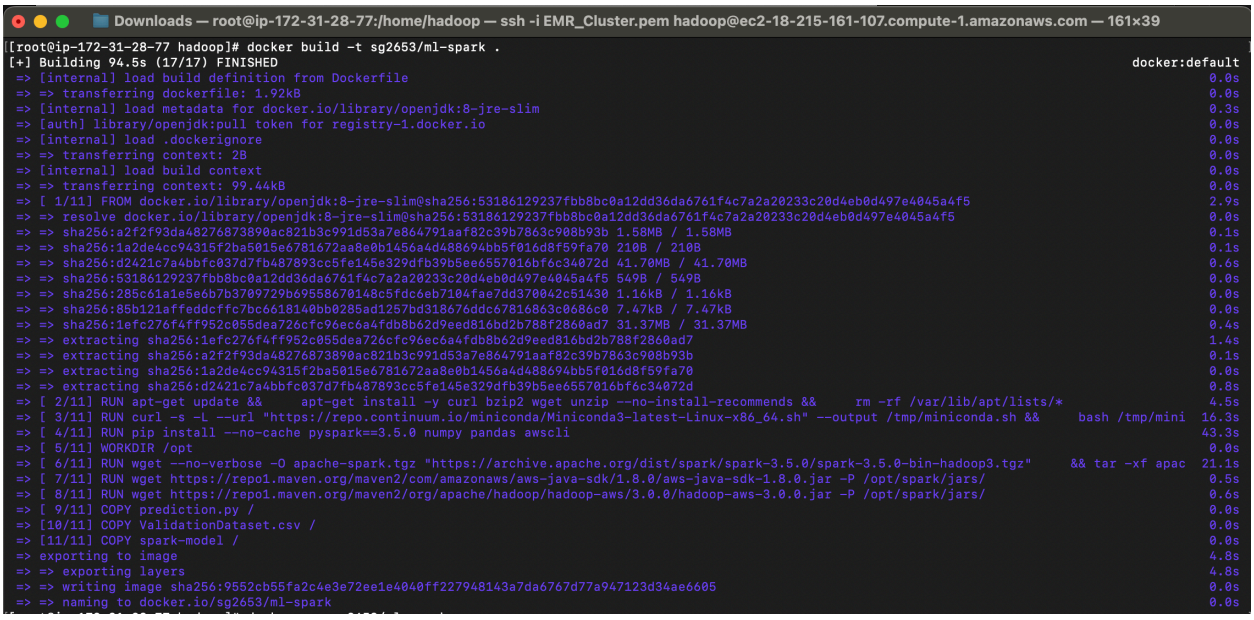
The above score shows almost 97% test accuracy of prediction in the trained spark-model and Weighted F1 score is around 95.5%

Execution with Docker

Login to docker from the instance

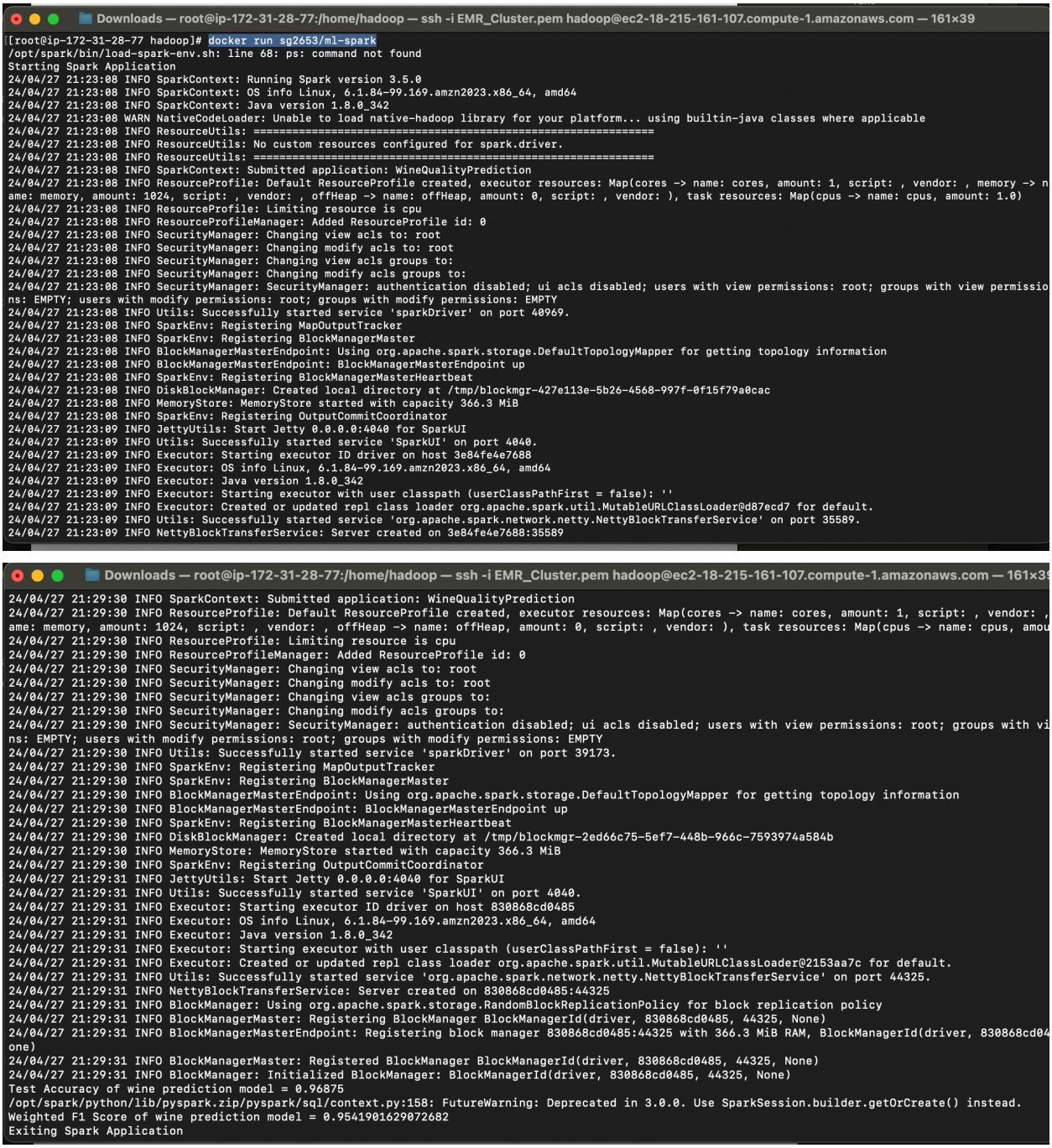
commands : run all of them below from the project directory where Dockerfile is present

docker login -> enter username and password
docker build -t sg2653/ml-spark . -> username/name_of_image

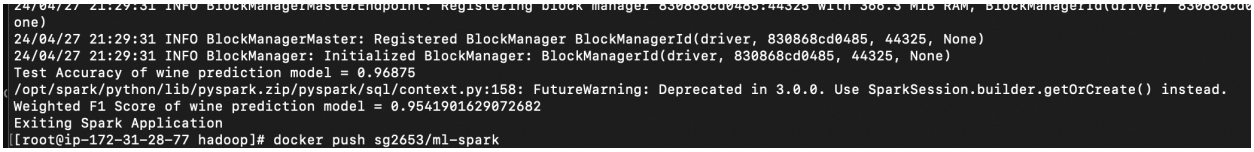


docker run sg2653/ml-spark

In the above command, we need not mention any .py file name or test dataset as arguments because everything is configured inside Dockerfile which takes care of complete execution. All we need it to be aware of file path setup.



Output from Execution with Docker



```
[[root@ip-172-31-28-77 hadoop]# docker push sg2653/ml-spark
Using default tag: latest
The push refers to repository [docker.io/sg2653/ml-spark]
94e33ba8ef7a: Pushed
47b7588ce0c1: Pushed
9780aaf4f1e4: Pushed
99952b8d544f: Pushed
a6acf7297fdc: Pushed
ce712ca78f41: Pushed
5f70bf18a086: Pushed
9e25f407b39d: Pushed
10bc2294d76f: Pushed
4d8c41736756: Pushed
b66078cf4b41: Mounted from library/openjdk
cd5a0a9f1e01: Mounted from library/openjdk
eafe6e032dbd: Mounted from library/openjdk
92a4e8a3140f: Mounted from library/openjdk
latest: digest: sha256:963e6dabb9c11d0b03fb2bb1cfcf886cc122077aa8912017c369a354384bda3f size: 3262
```

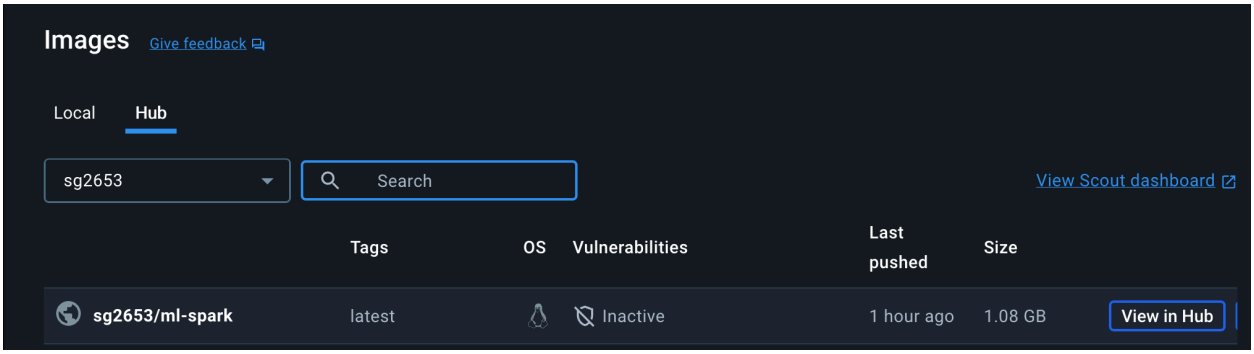
Now we can push the total docker build to docker using the below command

docker push sg2653/ml-spark

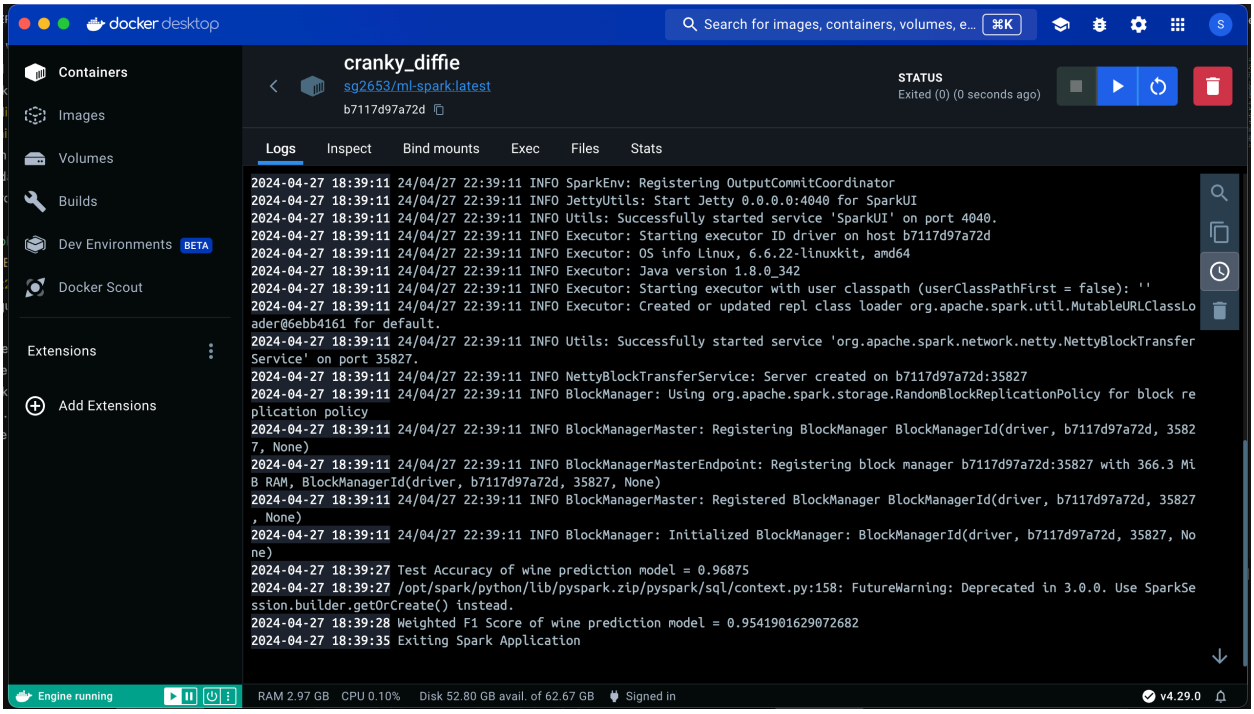
so we can even pull the same any time in any machine and run the total model with any add on files using below command

docker pull sg2653/ml-spark

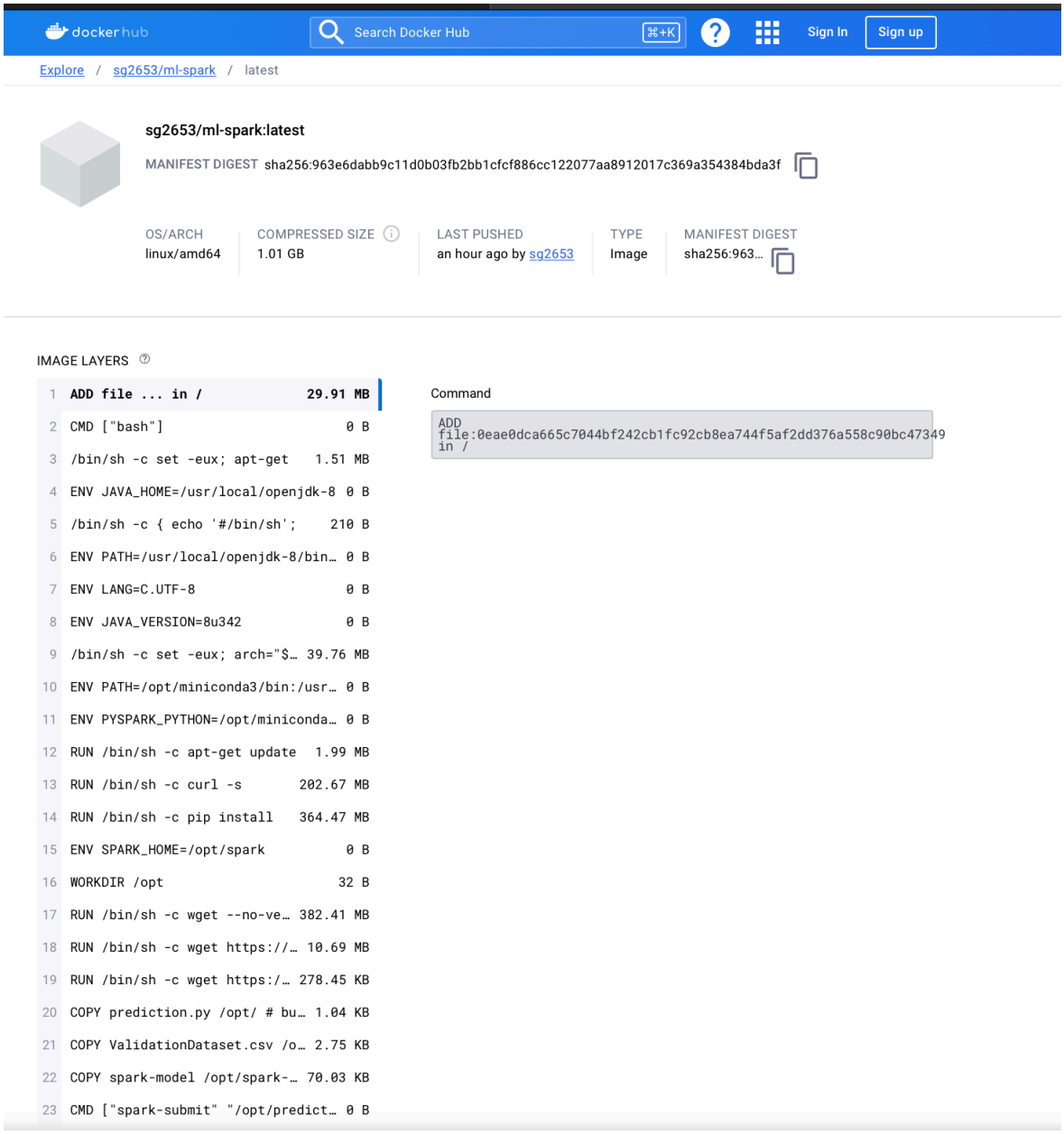
Docker Images



We can directly run in Docker Desktop too and check the ouptput as we got in the instance terminal.



Docker Hub : Latest Image Layers



To download files from instance to local machine :

scp -i EMR_Cluster.pem -r
'hadoop@ec2-18-215-161-107.compute-1.amazonaws.com:/home/hadoop/*'
/path/to/local_machine_folder

GitHub Repository link for code :

https://github.com/sainageshgowrabalaji/academic2

DockerHub Repository link :

https://hub.docker.com/r/sg2653/ml-spark/tags