CS643852-Cloud Computing
Programming Assignment 2

Sai Praneeth Kommu
sk3592@njit.edu

Wine Quality Prediction Model with Spark on AWS

Description: In this individual assignment, the objective is to develop a wine quality prediction machine learning model on the Amazon AWS cloud platform. Utilizing Apache Spark, the model will be trained in parallel across four EC2 instances to enhance computational efficiency. Leveraging Spark's MLlib, the model will be constructed and utilized within the cloud environment. Furthermore, Docker will be employed to streamline model deployment by encapsulating it into a container. The entire implementation will be carried out in Python on Ubuntu Linux, ensuring compatibility and ease of development.

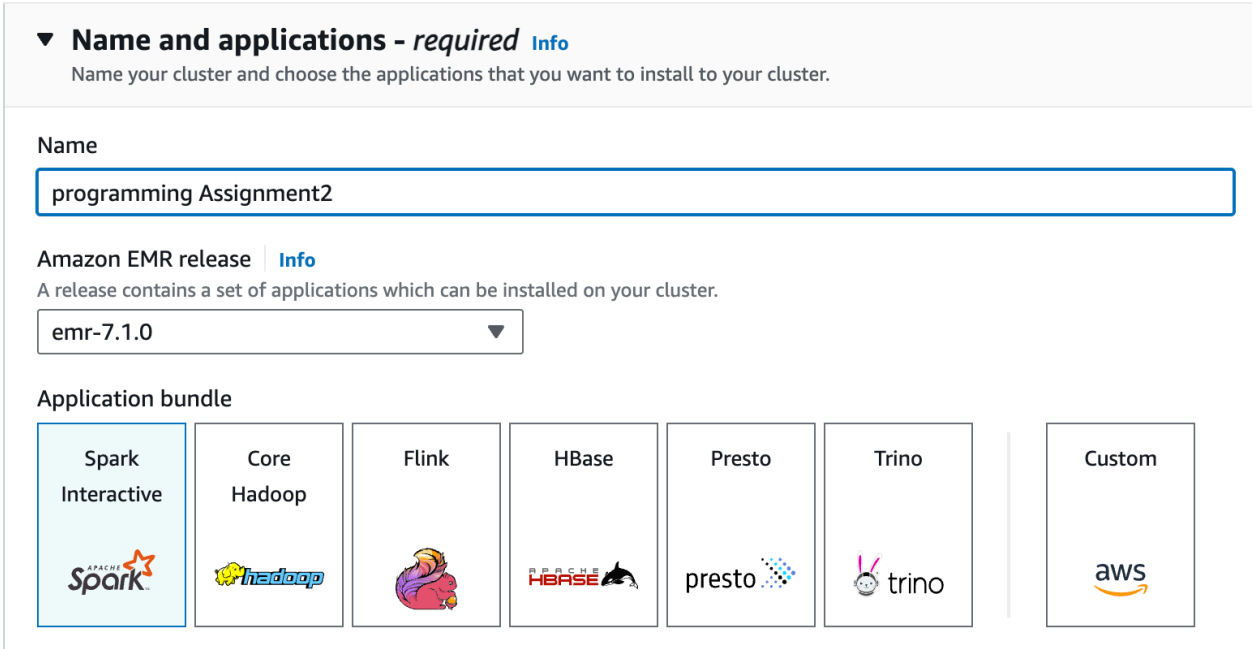
GitHub Repository: https://github.com/saipraneethkommu/Wine_prediction

Docker Hub: https://hub.docker.com/repository/docker/praneethdocker1/assignment2/general

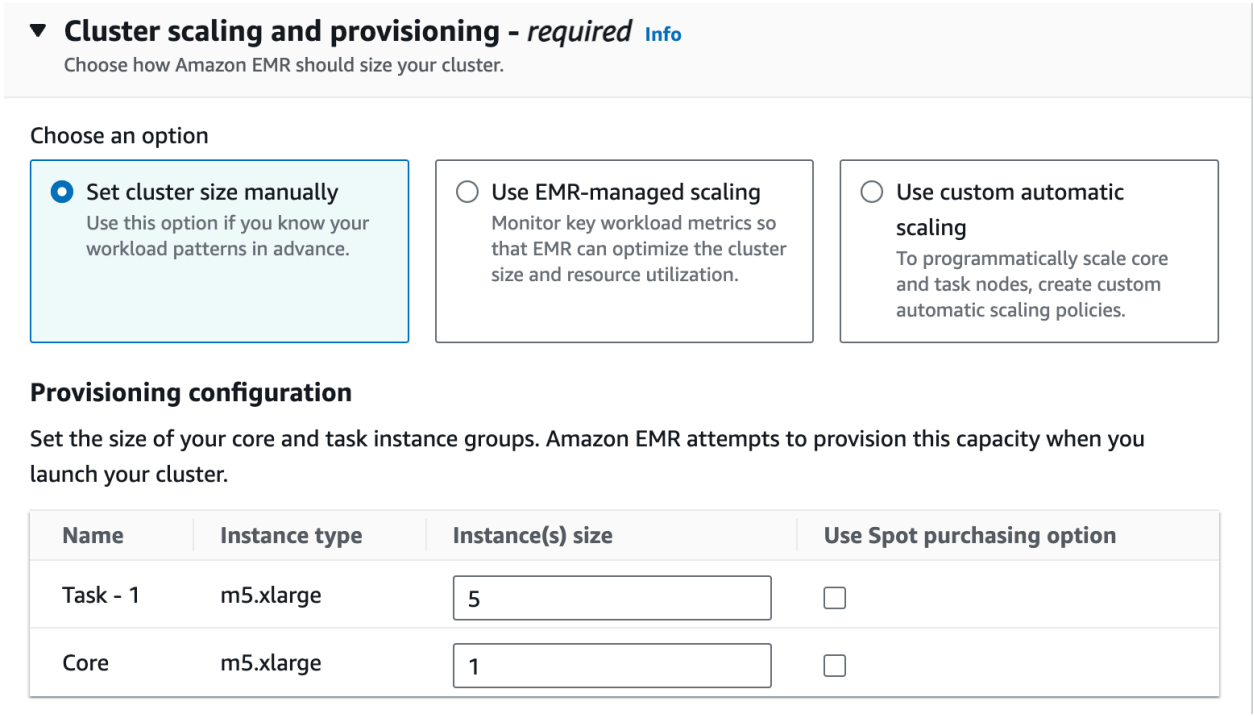
Step by Step Execution:

Cluster Creation:

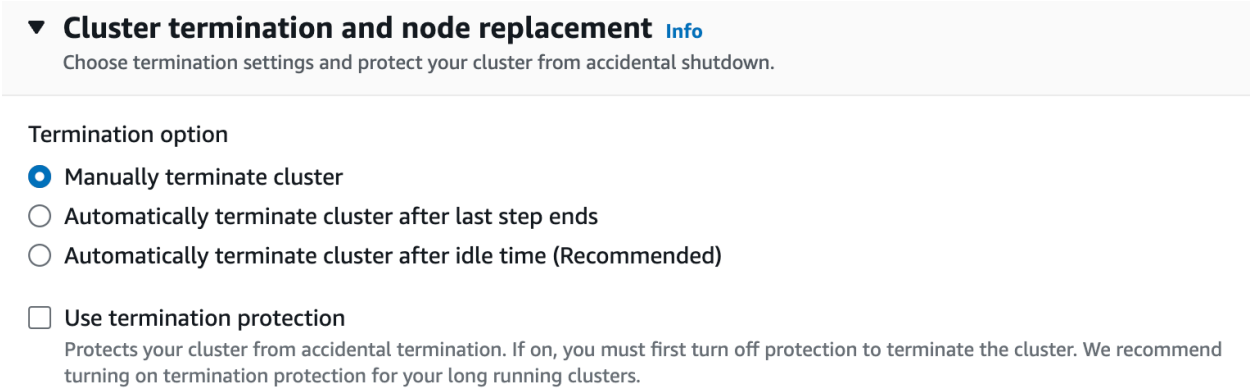
1. Log in to your AWS Account and initiate the lab.
2. Access the AWS Console and search for “EMR".
3. Click on "Create cluster" and assign a name to your cluster



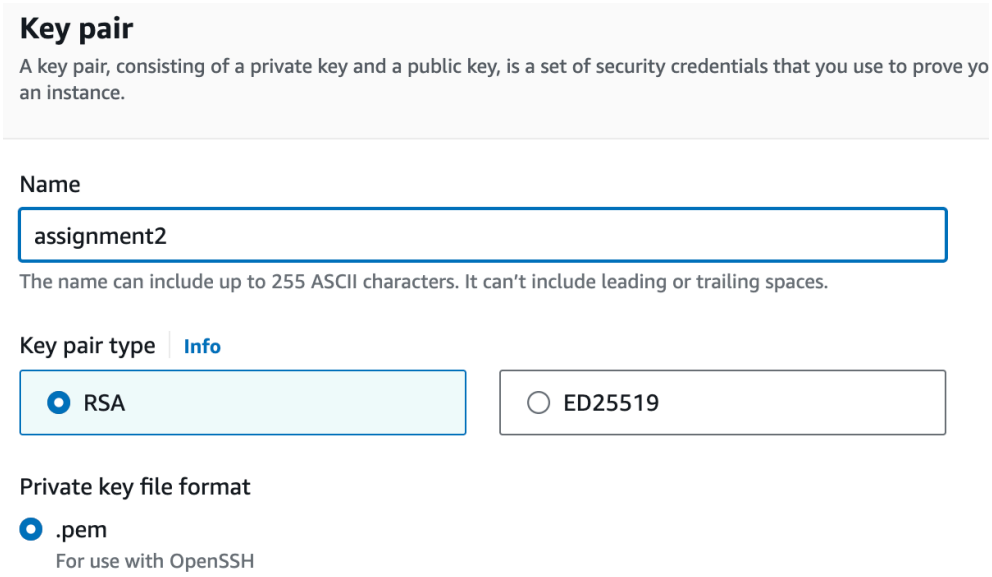
4. Set the scaling and provisioning parameters: assign 1 instance for core and 5 instances for task.



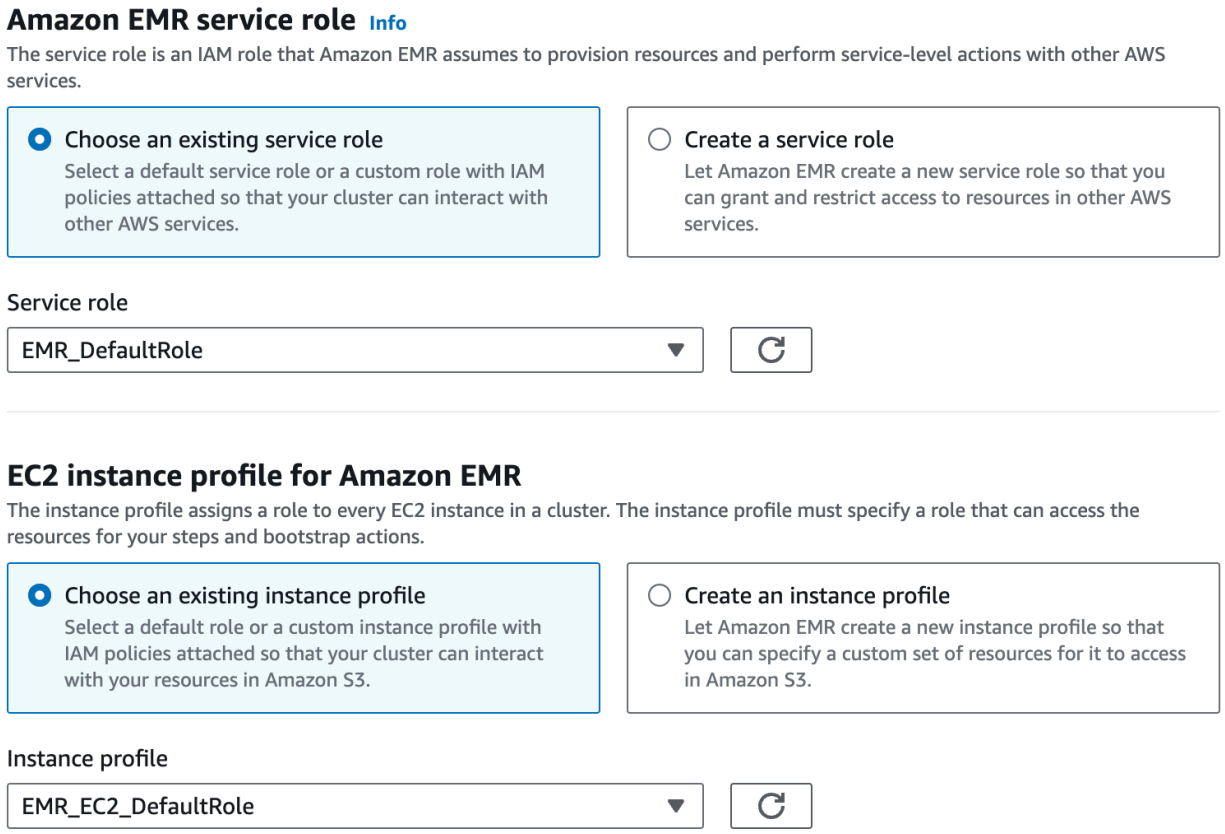
5. Enable termination protection and choose to manually terminate the cluster under "Cluster termination and node replacement".



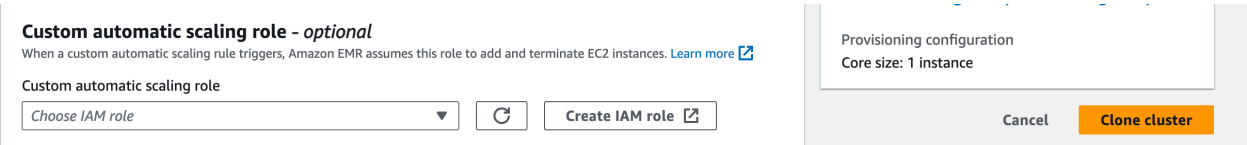
6. Under Security configuration and EC2 key pair, click on create key pair > name the key > select .pem and click on create key and go back and click on Browse and add the key.



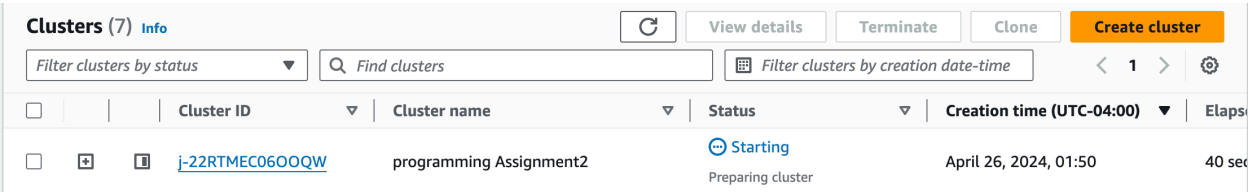
7. Assign IAM roles: EMR_DefaultRole for Service role and EMR_EC2_DefaultRole for Instance profile.



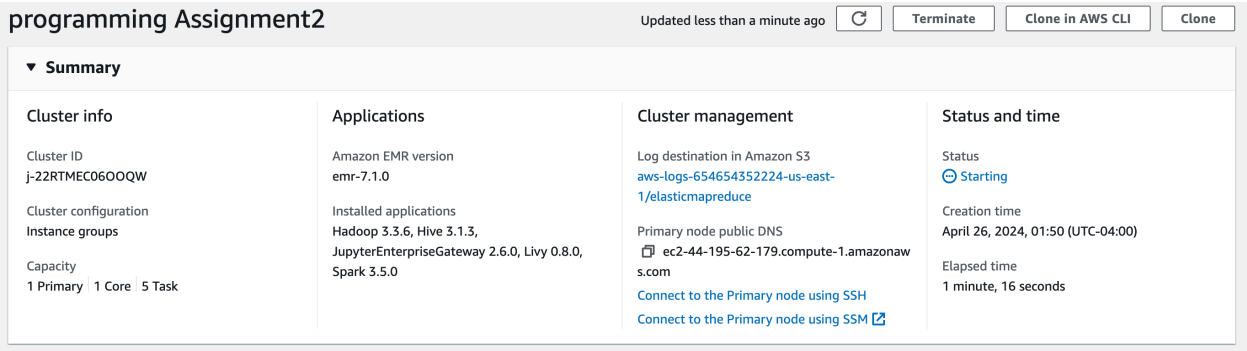
8. Click on "Create Cluster" to initialize the cluster.



9. Access the cluster details by searching for "EMR", then selecting the created cluster.

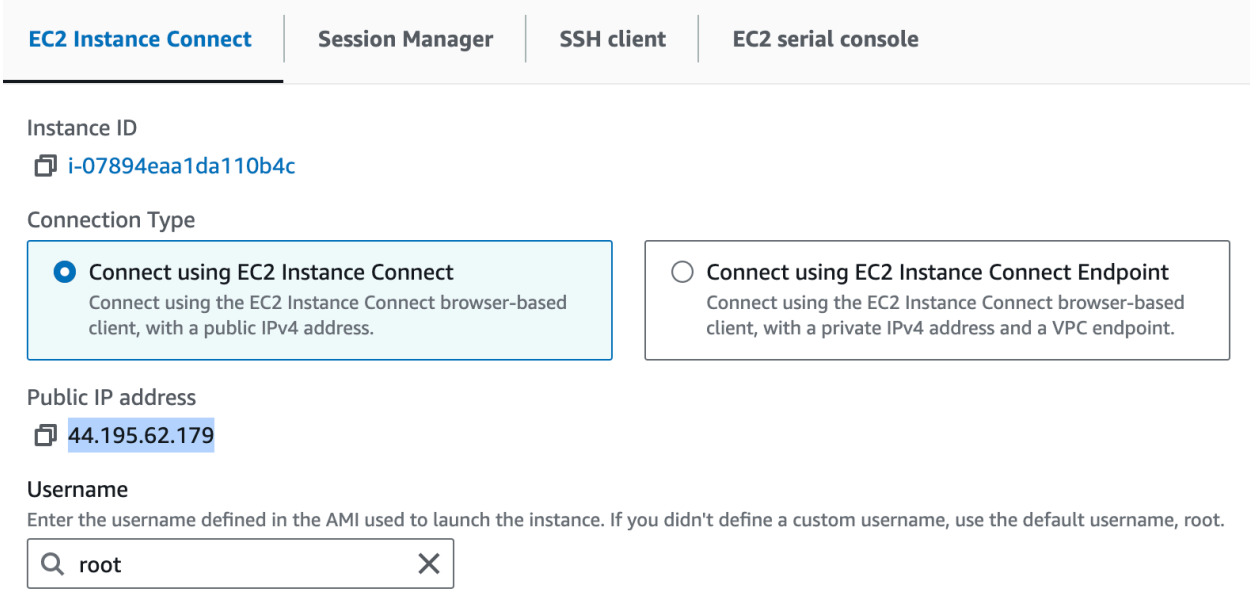


10. Connect to the primary node using SSM and note down the Public IP address.

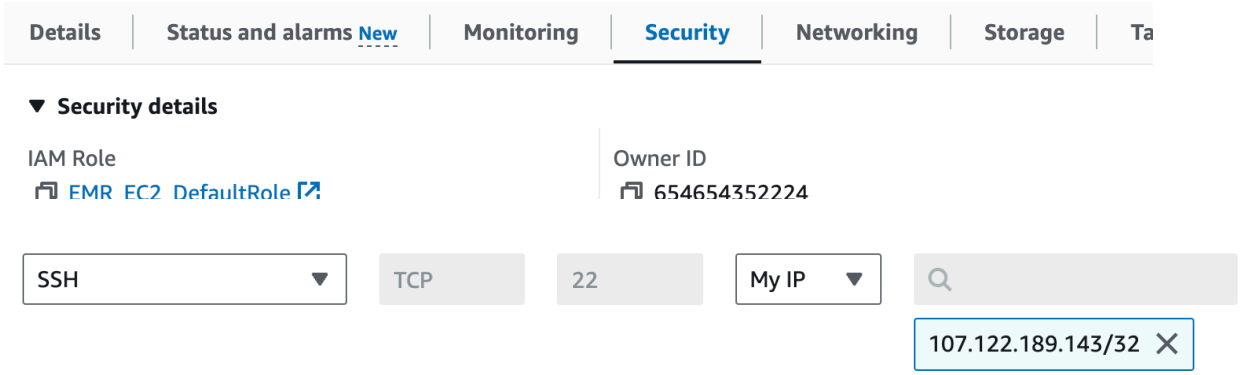


EC2 Instances

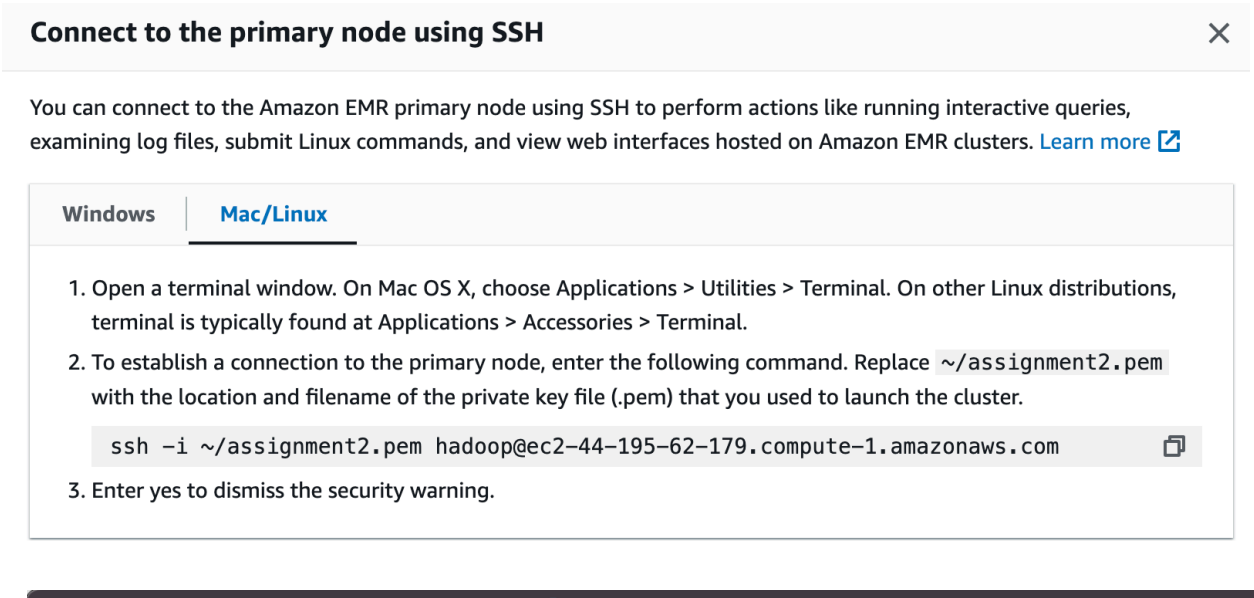
11. Locate the corresponding EC2 instance using the Public IP address.

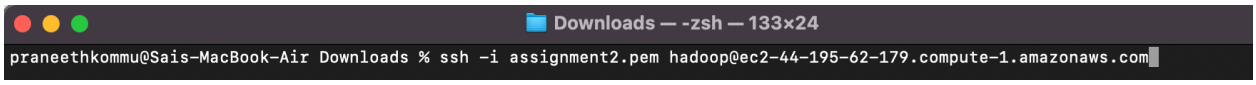


12. Configure security group inbound rules to allow SSH access from your IP address.



13. Connect to the primary node using SSH by copying and running the provided SSH command in your terminal.

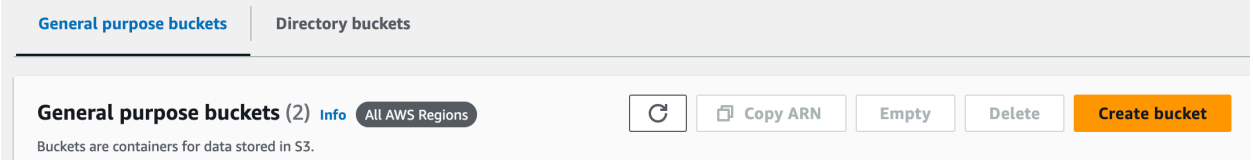




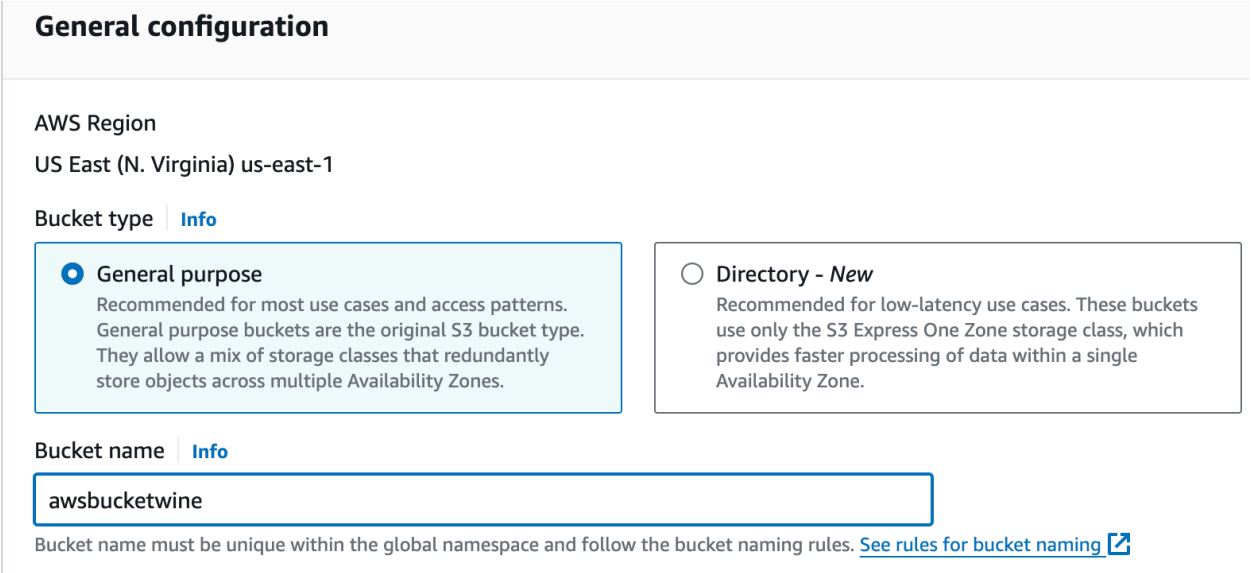
```
praneethkommu@Sais-MacBook-Air ~ % cd Downloads
praneethkommu@Sais-MacBook-Air Downloads % ssh -i assignment2.pem hadoop@ec2-44-195-62-179.compute-1.amazonaws.com
The authenticity of host 'ec2-44-195-62-179.compute-1.amazonaws.com (44.195.62.179)' can't be established.
ED25519 key fingerprint is SHA256:cDGhuAEo0gG+BWE1Vd0cos/UszS9L6ZN/CUL2CIv7bw.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-44-195-62-179.compute-1.amazonaws.com' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
Last login: Fri Apr 26 06:19:15 2024

EEEEEEEEEEEEEEEEEEE MMMMMMMM           MMMMMMMM RRRRRRRRRRRRRRR
E::::::::::::::::::E M:::::::M         M:::::::M R::::::::::::::R
EE:::::EEEEEEEEE:::E M::::::::M       M::::::::M R:::::RRRRRR:::::R
  E::::E       EEEEE M:::::::::M     M:::::::::M RR::::R      R::::R
  E::::E             M::::::M:::M   M:::M::::::M   R:::R      R::::R
  E::::::EEEEEEEEEE  M:::::M M:::M M:::M M:::::M   R:::RRRRRR:::::R
  E:::::::::::::::E  M:::::M  M:::M:::M  M:::::M   R:::::::::::RR
  E::::::EEEEEEEEEE  M:::::M   M:::::M   M:::::M   R:::RRRRRR::::R
  E::::E             M:::::M    M:::M    M:::::M   R:::R      R::::R
  E::::E       EEEEE M:::::M     MMM     M:::::M   R:::R      R::::R
EE::::::EEEEEEEE:::E M:::::M             M:::::M   R:::R      R::::R
E::::::::::::::::::E M:::::M             M:::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEE MMMMMMM             MMMMMMM RRRRRRR      RRRRRR

[hadoop@ip-172-31-11-109 ~]$
```

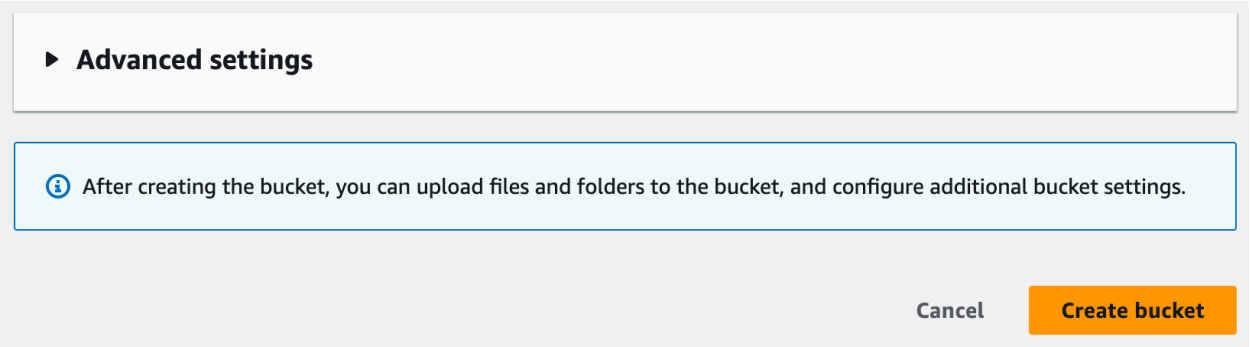
S3 Bucket Creation

14. Navigate to the AWS Console and search for "S3".
15. Click on "Create Bucket" and assign a name to your bucket.

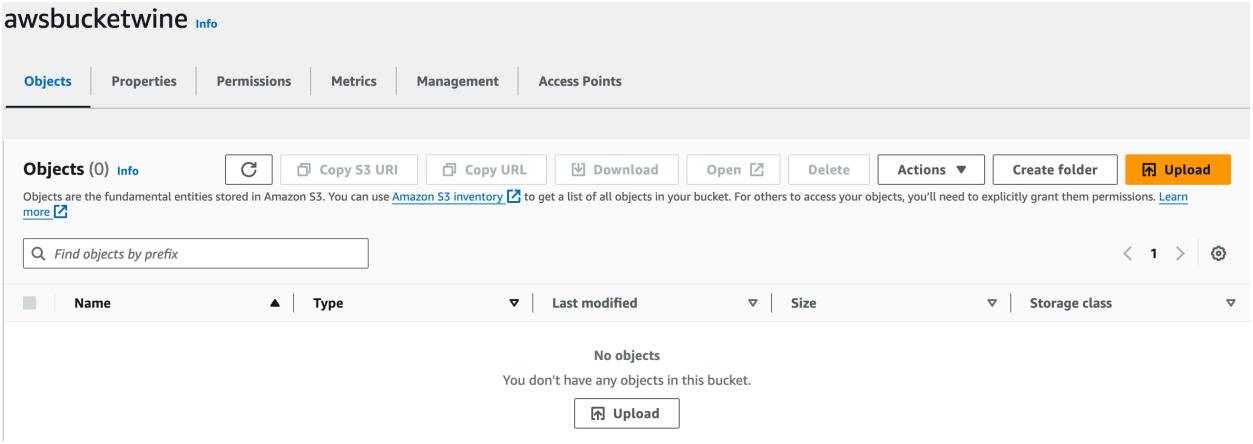


16. After successful creation, navigate into the bucket and upload the required files (working code and datasets).





17. Click on the created bucket and click on upload files and then upload the require files (Working code and datasets)



18. Now Upload the required files.

Without Docker Execution:

19. When the upload process completes, navigate to the EC2-connected terminal and execute the following command:

spark-submit s3://awsbucketwine/Wine_Quality_Training_Spark.py (Training code)

20. Once the training code has completed execution, proceed to execute the testing code using the same command:

spark-submit s3://awsbucketwine/prediction.py(Testing code)

```
[hadoop@ip-172-31-10-106 ~]$ spark-submit s3://awsbucketwine/Wine_Quality_Training_Spark.py
Apr 27, 2024 1:13:05 AM org.apache.spark.launcher.Log4jHotPatchOption staticJavaAgentOption
WARNING: spark.log4jHotPatch.enabled is set to true, but /usr/share/log4j-cve-2021-44228-hotpatch/jdk17/

Starting Spark Application
24/04/27 01:13:09 INFO SparkContext: Running Spark version 3.5.0-amzn-1
24/04/27 01:13:09 INFO SparkContext: OS info Linux, 6.1.84-99.169.amzn2023.x86_64, amd64
24/04/27 01:13:09 INFO SparkContext: Java version 17.0.10
24/04/27 01:13:09 INFO ResourceUtils: ==============================================================
24/04/27 01:13:09 INFO ResourceUtils: No custom resources configured for spark.driver.
24/04/27 01:13:09 INFO ResourceUtils: ==============================================================
24/04/27 01:13:09 INFO SparkContext: Submitted application: WineQualityPrediction
24/04/27 01:13:09 INFO ResourceProfile: Default ResourceProfile created, executor resources: Map(cores -
offHeap -> name: offHeap, amount: 0, script: , vendor: ), task resources: Map(cpus -> name: cpus, amount
24/04/27 01:13:09 INFO ResourceProfile: Limiting resource is cpus at 4 tasks per executor
24/04/27 01:13:09 INFO ResourceProfileManager: Added ResourceProfile id: 0
24/04/27 01:13:09 INFO SecurityManager: Changing view acls to: hadoop
24/04/27 01:13:09 INFO SecurityManager: Changing modify acls to: hadoop
24/04/27 01:13:09 INFO SecurityManager: Changing view acls groups to:
24/04/27 01:13:09 INFO SecurityManager: Changing modify acls groups to:
```

```
Reading training CSV file from s3://awsbucketwine/TrainingDataset.csv
Reading validation CSV file from s3://awsbucketwine/ValidationDataset.csv
Creating VectorAssembler
Creating StringIndexer
Caching data for faster access
Creating RandomForestClassifier
Creating Pipeline for training
Transforming data using the trained model
Evaluating the trained model on the validation set
Test Accuracy of wine prediction model = 0.99375
/usr/lib/spark/python/lib/pyspark.zip/pyspark/sql/context.py:158: FutureWarning: Deprecat
Weighted f1 score of wine prediction model = 0.9933730158730157
Retraining model on multiple parameters using CrossValidator
Fitting CrossValidator to the training data
24/04/27 01:14:26 ERROR TransportResponseHandler: Still have 1 requests outstanding when
Saving the best model to new param `model`
Test Accuracy of wine prediction model (after CrossValidation) = 0.96875
Weighted f1 score of wine prediction model (after CrossValidation) = 0.9541901629072682
Saving the best model to S3
```

21. Upon successful execution, the testing results will be obtained.

Execution with Docker

22. Set up a Docker repository.
23. Configure the Dockerfile according to your requirements.
24. Make the necessary modifications to your code.
25. Open a new terminal to check locally.
26. Build an image in Docker using the command: **docker build -t train_wine .**

```
praneethkommu@Sais-MacBook-Air Wine % docker build -t train_wine .
[+] Building 1.1s (20/20) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 1.26kB
 => [internal] load metadata for docker.io/library/centos:7
 => [auth] library/centos:pull token for registry-1.docker.io
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [ 1/14] FROM docker.io/library/centos:7@sha256:be65f488b7764ad3
 => [internal] load build context
 => => transferring context: 4.63kB
 => CACHED [ 2/14] RUN yum -y update && yum -y install python3 pyth
 => CACHED [ 3/14] RUN python3 -V
 => CACHED [ 4/14] RUN pip3 install --upgrade pip
 => CACHED [ 5/14] RUN pip3 install awscli
 => CACHED [ 6/14] RUN pip3 install numpy pandas
 => CACHED [ 7/14] WORKDIR /opt
 => CACHED [ 8/14] RUN wget --no-verbose -O apache-spark.tgz "https
```

27. Link the build to the Docker repository with: **docker build praneethdocker1/assignment2:praneeth**
28. Push the image to Docker with: **docker push praneethdocker1/assignment2:praneeth**

```
praneethkommu@Sais-MacBook-Air Wine % docker push praneethdocker1/assignment2:praneeth
The push refers to repository [docker.io/praneethdocker1/assignment2]
a532ca2e8309: Pushed
3aa477d27306: Pushed
85d266bdc8fe: Pushed
84a098af3fd4: Pushed
bcb28d9f9c2a: Pushed
d741d4e12584: Pushed
15ea834d4c4c: Pushed
5f70bf18a086: Pushed
fec425c5a135: Pushed
342f15d88e26: Pushed
833ecbd56391: Pushed
8b25501926a0: Pushed
b409d03e7edb: Pushed
65f23ff12f4d: Mounted from library/centos
praneeth: digest: sha256:e2bb9d3af6d15f04441ba5035a873e679489a3eca69f3a66956cb154e169c30e size: 3262
```

29. To execute, use the command: **docker run train_wine** .
30. After successful execution, verify the code locally.
31. Navigate to the EC2-connected terminal for cloud testing.
32. Use the command **sudo su** to switch to the root user.

```
[hadoop@ip-172-31-10-106 ~]$ sudo su

EEEEEEEEEEEEEEEEEEE MMMMMMMM           MMMMMMMM RRRRRRRRRRRRRRR
E::::::::::::::::::E M:::::::M         M:::::::M R:::::::::::::::R
EE:::::EEEEEEEEE:::E M::::::::M       M::::::::M R::::::RRRRRR:::::R
  E::::E       EEEEE M:::::::::M     M:::::::::M RR::::R      R::::R
  E::::E             M::::::M:::M   M:::M::::::M   R::R      R::::R
  E::::::EEEEEEEEEE  M::::::M M:::M M:::M M::::::M   R::RRRRRR:::::R
  E:::::::::::::::E  M::::::M  M:::M:::M  M::::::M   R:::::::::::RR
  E::::::EEEEEEEEEE  M::::::M   M:::::M   M::::::M   R::RRRRRR::::R
  E::::E             M::::::M    M:::M    M::::::M   R::::R      R::::R
  E::::E       EEEEE M::::::M     MMM     M::::::M   R::::R      R::::R
EE::::::EEEEEEEE:::E M::::::M             M::::::M   R::::R      R::::R
E::::::::::::::::::E M::::::M             M::::::M RR::::R      R::::R
EEEEEEEEEEEEEEEEEEE MMMMMMM               MMMMMMM RRRRRRR      RRRRRR

[root@ip-172-31-10-106 hadoop]# sudo systemctl start docker
[root@ip-172-31-10-106 hadoop]# sudo systemctl enable docker
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service → /usr/lib/systemd/system/docker.service.
```

33. Start Docker with: **sudo systemctl start docker .**
34. Enable Docker with: **sudo systemctl enable docker .**

35. Pull the image from docker using the command: **docker pull praneethdocker1/assignment2:praneeth**

```
[root@ip-172-31-10-106 hadoop]# docker pull praneethdocker1/assignment2:praneeth
praneeth: Pulling from praneethdocker1/assignment2
6717b8ec66cd: Pull complete
04fbb41f633a: Pull complete
3a3951e056be: Pull complete
f14b9a56231e: Pull complete
095935b81d71: Pull complete
be2d354dcfd0: Pull complete
4f4fb700ef54: Pull complete
1f2b3b500e9e: Pull complete
dd61aa7aa6f9: Pull complete
e6432b6a7ef0: Pull complete
e98c6d43515b: Pull complete
ca404c38a84a: Pull complete
a72f83e06635: Pull complete
369a152b3ce9: Pull complete
Digest: sha256:e2bb9d3af6d15f04441ba5035a873e679489a3eca69f3a66956cb154e169c30e
Status: Downloaded newer image for praneethdocker1/assignment2:praneeth
```

36. For running the image, use the command: **docker run praneethdocker1/ assignment2:praneeth**

```
Reading training CSV file from TrainingDataset.csv
Reading validation CSV file from ValidationDataset.csv
Creating VectorAssembler
Creating StringIndexer
Caching data for faster access
Creating RandomForestClassifier
Creating Pipeline for training
Transforming data using the trained model
Evaluating the trained model on the validation set
Test Accuracy of wine prediction model = 0.99375
/opt/spark/python/lib/pyspark.zip/pyspark/sql/context.py:160: FutureWarning: Deprecated i
e() instead.
Weighted f1 score of wine prediction model = 0.9933730158730157
Retraining model on multiple parameters using CrossValidator
Fitting CrossValidator to the training data
Saving the best model to new param `model`
Test Accuracy of wine prediction model (after CrossValidation) = 0.96875
/opt/spark/python/lib/pyspark.zip/pyspark/sql/context.py:160: FutureWarning: Deprecated i
e() instead.
Weighted f1 score of wine prediction model (after CrossValidation) = 0.9541901629072682
```

37. After the successful execution, we will see the testing results.

Docker:

https://hub.docker.com/repository/docker/praneethdocker1/assignment2/general