

**Pradeep Varma Ganapathiraju pg552**

**GitHub:** [https://github.com/pradeep-varma/cc\\_Programming2](https://github.com/pradeep-varma/cc_Programming2)

**Docker :** <https://hub.docker.com/r/pg552/wqpapp9>

**Procedure:**

## Cluster Creation

Navigate to Amazon EMR in AWS > Create the Cluster, and select the following apps:

**Name and applications** [Info](#)

Name

WQP

Amazon EMR release


[Info](#)

A release contains a set of applications which can be installed on your cluster.


emr-6.15.0

Application bundle


Spark  
Interactive




Core  
Hadoop




Flink




HBase




Presto



Trino



Custom



☐ Flink 1.17.1

☐ HCatalog 3.1.3

☐ Hue 4.11.0

☐ Livy 0.7.1

☐ Phoenix 5.1.3

☒ Spark 3.4.1

☐ Tez 0.10.2

☐ ZooKeeper 3.5.10

☐ Ganglia 3.7.2

☒ Hadoop 3.3.6

☐ JupyterEnterpriseGateway 2.6.0

☐ MXNet 1.9.1

☐ Pig 0.17.0

☐ Sqoop 1.4.7

☐ Trino 426

☐ HBase 2.4.17

☐ Hive 3.1.3

☐ JupyterHub 1.5.0

☐ Oozie 5.2.1

☐ Presto 0.283

☐ TensorFlow 2.11.0

☒ Zeppelin 0.10.1

Had to train model using 4 EC2 instances, so configured 'Provisioning Configuration as below:

**Cluster scaling and provisioning** [Info](#)

Set up scaling and provisioning configurations for the core and task node groups for your cluster.

Choose an option

☒ **Set cluster size manually**  
 Use this option if you know your workload patterns in advance.

☐ **Use EMR-managed scaling**  
 Monitor key workload metrics so that EMR can optimize the cluster size and resource utilization.

☐ **Use custom automatic scaling**  
 To programmatically scale core and task nodes, create custom automatic scaling policies.

**Provisioning configuration**

Set the size of your core and task instance groups. Amazon EMR attempts to provision this capacity when you launch your cluster.

Name	Instance type	Instance(s) size	Use Spot purchasing option
Core	m5.xlarge	<input type="text" value="1"/>	<input type="checkbox"/>
Task - 1	m5.xlarge	<input type="text" value="3"/>	<input type="checkbox"/>

Went with Recommended Cluster Termination settings:

**Cluster termination** [Info](#)

☐ Manually terminate cluster  
☐ Automatically terminate cluster after last step ends  
☒ **Automatically terminate cluster after idle time (Recommended)**

**Idle time**

Enter the time until your cluster terminates.

Choose a time that is greater than 1 minute (00:01:00) and less than 7 days. The time is in hh:mm:ss (24-hour) format.

☒ **Use termination protection**  
 Protect your EC2 instances from accidental termination.

Create a key pair for SSH and SCP to the Instance/Cluster:

**Security configuration and EC2 key pair - optional** [Info](#)

**Security configuration**

Select your cluster encryption, authentication, authorization, and instance metadata service settings.

**Amazon EC2 key pair for SSH to the cluster** [Info](#)

Select default IAM roles:

### Identity and Access Management (IAM) roles [Info](#)

Choose or create a service role and instance profile for the EC2 instances in your cluster.

#### Amazon EMR service role [Info](#)

The service role is an IAM role that Amazon EMR assumes to provision resources and perform service-level actions with other AWS services.

☒ Choose an existing service role
 

Select a default service role or a custom role with IAM policies attached so that your cluster can interact with other AWS services.

☐ Create a service role
 

Let Amazon EMR create a new service role so that you can grant and restrict access to resources in other AWS services.

Service role

EMR\_DefaultRole

#### EC2 instance profile for Amazon EMR

The instance profile assigns a role to every EC2 instance in a cluster. The instance profile must specify a role that can access the resources for your steps and bootstrap actions.

☒ Choose an existing instance profile
 

Select a default role or a custom instance profile with IAM policies attached so that your cluster can interact with your resources in Amazon S3.

☐ Create an instance profile
 

Let Amazon EMR create a new instance profile so that you can specify a custom set of resources for it to access in Amazon S3.

Instance profile

EMR\_EC2\_DefaultRole

Rest, go with default configuration, cluster/s will be working upon turning to 'Waiting' status.

WQP

Updated less than a minute ago

Terminate

Clone in AWS CLI

Clone

▼ Summary

Cluster info	Applications	Cluster management	Status and time
Cluster ID j-2Y5VVE51XCOTB  Cluster configuration Instance groups  Capacity 1 Primary 1 Core 3 Task	Amazon EMR version emr-6.15.0  Installed applications Hadoop 3.3.6, Spark 3.4.1, Zeppelin 0.10.1	Log destination in Amazon S3 aws-logs-098431872762-us-east-1/elasticmapreduce  Persistent application UIs <a href="#">Spark History Server</a> <a href="#">YARN timeline server</a>  Primary node public DNS ec2-3-234-217-212.compute-1.amazonaws.com  <a href="#">Connect to the Primary node using SSH</a> <a href="#">Connect to the Primary node using SSM</a>	Status Waiting  Creation time December 08, 2023, 10:29 (UTC-05:00)  Elapsed time 13 minutes, 41 seconds

Create an S3 Bucket and drop training and testing files into it.

Inbound rules (8)

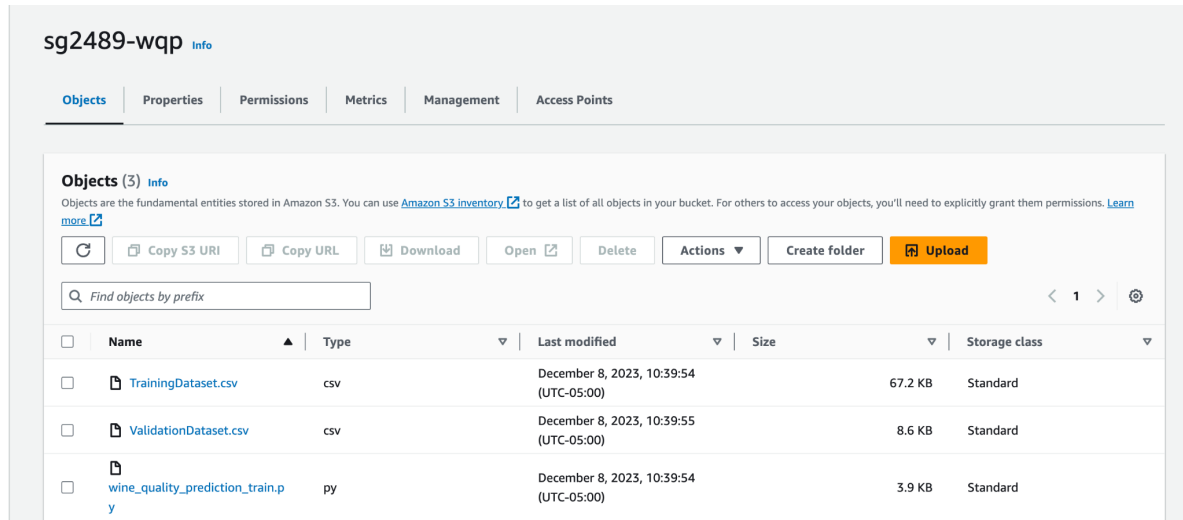
Manage tags

Edit inbound rules

Q Search

< 1 > ⚙

Security group rule...	IP version	Type	Protocol	Port range	Source
sgr-0aa323dbe1b6c30...	IPv4	SSH	TCP	22	96.242



Ensure master instance of the cluster has permission to SSH, if not edit inbound rule after navigating to the security group of the instance.

Done SSH into my master instance using below command: `ssh -i "<my-key-pair.pem>" ec2-user@<Public DNS>`

```

#_
~\_ #####_      Amazon Linux 2
~~\_#####\
~~\_###|        AL2 End of Life is 2025-06-30.
~~\_#/
~~\_V~'  '--->
~~~
~~~.  _  /
~~~_/_/_/_/_/
    /m/'

A newer version of Amazon Linux is available!

Amazon Linux 2023, GA and supported until 2028-03-15.
https://aws.amazon.com/linux/amazon-linux-2023/

```

19 package(s) needed for security, out of 20 available  
Run "sudo yum update" to apply all updates.

```

EEEEEEEEEEEEEEEEEEEE MMMMMMM      MMMMMMM RRRRRRRRRRRRRR
E::::::::::::::::::::E M::::::::M      M::::::::M R:::::::::R
EE::::::::::::::::::::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 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:::M R:::::::::RR
  E:::EEEEEEEEEE M:::M M:::M M:::M M:::M R:::RRRRRR:::R
  E:::E      M:::M M:::M M:::M M:::M R:::R      R:::R
  E:::E      EEEEE M:::M      MMM M:::M M:::M R:::R      R:::R
EE::::::::::::::::::::E M:::M      M:::M M:::M R:::R      R:::R
E::::::::::::::::::::E M:::M      M:::M M:::M RR:::R      R:::R
EEEEEEEEEEEEEEEEEEEE MMMMMMM      MMMMMMM RRRRRRR      RRRRRR

```

Navigated to root user using: `sudo su`

Now, trained the model with training file using the below command: `sparksubmit s3://<bucket-name>/wine_quality_prediction_train.py`

After successful training, results of trained model are:

```
20/12/20 20:00:40 INFO Training completed successfully.
Test Accuracy of Initial Model: 0.99375
/usr/lib/spark/python/lib/pyspark.zip/pyspark/sql/context
Weighted f1 score of Initial Model: 0.9933730158730157
Best Model: PipelineModel_55a5c603be56
Test Accuracy of Best Model: 0.96875
Weighted f1 score of Best Model: 0.9547916666666667
```

## Running Single Machine App in EC2 without Docker

Fetch trained model from S3 to master instance into a new folder (trained\_model) using below command: `aws s3 sync aws s3 sync s3://<my-bucket-name>/trained_model.model ./trained_model`

Done SCP for <testdataset> and Single Machine Prediction code file using below command: `scp -i <keypair.pem> <file> ec2-user@<Public DNS>:/home/ec2-user`

Installed Pyspark with: `pip install pyspark` for running single machine prediction application with external test dataset, using below command: `python wqp_single_machine.py`

```
Sample Predictions:
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|fixed acidity|volatile acidity|citric acid|residual sugar|chlorides|free sulfur dioxide|total sulfur dioxide|density| pH|sulphates|alcohol|quality|features|label|rawPrediction|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|8372428187...|7.6|0.0|0.39|0.31|2.3|0.082|23.0|71.0|0.9982|3.52|0.65|9.7|5.0|[7.6,0.39,0.31,2,...]|0.0|[479.531456024878...]|[0.9590|
|6291204975...|7.9|0.0|0.43|0.21|1.6|0.106|18.0|37.0|0.9966|3.17|0.91|9.5|5.0|[7.9,0.43,0.21,1,...]|0.0|[484.885809101067...]|[0.9697|
|7161820213...|8.5|0.0|0.49|0.11|2.3|0.084|9.0|67.0|0.9968|3.17|0.53|9.4|5.0|[8.5,0.49,0.11,2,...]|0.0|[491.163209482913...]|[0.9823|
|2641896582...|6.9|0.0|0.4|0.14|2.4|0.085|21.0|40.0|0.9968|3.43|0.63|9.7|6.0|[6.9,0.4,0.14,2.4,...]|1.0|[3.29191168480642...]|[0.0065|
|8382336961...|1.0|0.0|0.4|0.14|2.4|0.085|21.0|40.0|0.9968|3.43|0.63|9.7|6.0|[6.9,0.4,0.14,2.4,...]|1.0|[3.29191168480642...]|[0.0065|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows

None
Test Accuracy of the Wine Quality Prediction Model: 0.9843627834245504
/usr/local/lib/python3.7/site-packages/pyspark/sql/context.py:159: FutureWarning: Deprecated in 3.0.0. Use SparkSession.builder.getOrCreate() instead.
FutureWarning:
Weighted F1 score of the Wine Quality Prediction Model: 0.9779339666913879
```

## Running Single Machine App in Local with Docker:

Fetch trained model from master instance into local (in a new folder wine\_quality\_predictor) using below command:

```
Sample Predictions:
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|fixed acidity|volatile acidity|citric acid|residual sugar|chlorides|free sulfur dioxide|total sulfur dioxide|density| pH|sulphates|alcohol|quality|features|label|rawPrediction|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|9517252728...|7.4|0.0|0.0|0.0|1.9|0.076|11.0|34.0|0.9978|3.51|0.56|9.4|5.0|[7.4,0.7,0.0,1.9,...]|0.0|[488.247586263641...]|[0.9764|
|9853972954...|7.8|0.0|0.88|0.0|2.6|0.098|25.0|67.0|0.9968|3.2|0.68|9.8|5.0|[7.8,0.88,0.0,2.6,...]|0.0|[476.049269864772...]|[0.9520|
|9853972954...|7.8|0.0|0.76|0.04|2.3|0.092|15.0|64.0|0.997|3.26|0.65|9.8|5.0|[7.8,0.76,0.04,2,...]|0.0|[474.981917756696...]|[0.9498|
|0383651339...|11.2|0.0|0.28|0.56|1.9|0.075|17.0|60.0|0.998|3.16|0.58|9.8|6.0|[11.2,0.28,0.56,1,...]|1.0|[26.3858382129224...]|[0.0527|
|7167642584...|7.4|0.0|0.7|0.0|1.9|0.076|11.0|34.0|0.9978|3.51|0.56|9.4|5.0|[7.4,0.7,0.0,1.9,...]|0.0|[488.247586263641...]|[0.9764|
|9517252728...|7.4|0.0|0.7|0.0|1.9|0.076|11.0|34.0|0.9978|3.51|0.56|9.4|5.0|[7.4,0.7,0.0,1.9,...]|0.0|[488.247586263641...]|[0.9764|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows

None
Test Accuracy of the Wine Quality Prediction Model: 0.96875
Weighted F1 score of the Wine Quality Prediction Model: 0.9547916666666667
```

`scp -i <keypair.pem> -r ec2-user@<Public DNS>:/home/ec2-user/trained_model ./wine_quality_predictor`

Install and sign up into Docker Desktop, login to Docker from terminal with command: 'docker login'

cd to the source folder and build the image of the app: 'docker build -t <image\_name:version> .'

run the app: 'docker run <image\_name:version>'

run the app with external dataset: docker run -v

/Users/<username>/<path\_to\_dataset\_directory>:/<path\_on\_host> <image\_name:version>  
<external\_dataset>

```
-----
Sample Predictions:
-----
|fixed acidity|volatile acidity|citric acid|residual sugar|chlorides|free sulfur dioxide|total sulfur dioxide|density| pH|sulphates|alcohol|quality|      features|label|      rawPrediction|
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 8.9| 1.0| 0.22| 0.48| 1.8| 0.077| 29.0| 60.0| 0.9968|3.39| 0.53| 9.4| 6.0|[8.9,0.22,0.48,1....]| 1.0|[19.4518621405359...]|[0.0389
8372428187...]|
| 7.6| 0.39| 0.31| 2.3| 0.082| 23.0| 71.0| 0.9982|3.52| 0.45| 9.7| 5.0|[7.6,0.39,0.31,2....]| 0.0|[479.531456024878...]|[0.9590
6291284975...]|
| 7.9| 0.43| 0.21| 1.6| 0.106| 18.0| 37.0| 0.9966|3.17| 0.91| 9.5| 5.0|[7.9,0.43,0.21,1....]| 0.0|[484.885809181067...]|[0.9697
7161820213...]|
| 8.5| 0.49| 0.11| 2.3| 0.084| 9.0| 67.0| 0.9968|3.17| 0.53| 9.4| 5.0|[8.5,0.49,0.11,2....]| 0.0|[491.163209482913...]|[0.9823
2641896582...]|
| 6.9| 0.4| 0.14| 2.4| 0.085| 21.0| 40.0| 0.9968|3.43| 0.63| 9.7| 6.0|[6.9,0.4,0.14,2.4...]| 1.0|[3.29191168480642...]|[0.0065
8382336961...]|
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
only showing top 5 rows
None
Test Accuracy of the Wine Quality Prediction Model: 0.9843627834245504
Weighted F1 score of the Wine Quality Prediction Model: 0.9779339666913879
```

Create a new repo in DockerHub.

Push the image into repo with the below commands:

docker tag <img:ver> <user\_id>/<repo>:<tag> docker  
push <user\_id>/<repo>:<tag>

## Running Single Machine App in EC2 with Docker:

Create an EC2 instance with default configurations

Install docker with command: sudo yum install docker

Start docker: sudo service docker start

Pull the image into EC2: sudo docker pull <user\_id>/<repo>:<tag>

Run the app: sudo docker run <user\_id>/<repo>:<tag>

None  
Test Accuracy of the Wine Quality Prediction Model: 0.9843627834245504  
Weighted F1 score of the Wine Quality Prediction Model: 0.9779339666913879