

**Wine Quality Prediction AWS Spark Application:**

**Pa2Winepred:** This project involves the development of a Python application utilizing the PySpark interface.

The application is deployed on an Amazon Web Services (AWS) Elastic MapReduce (EMR) cluster. The primary objective is to parallelly train a machine learning model on EC2 instances for predicting wine quality using publicly available data. Subsequently, the trained model is employed to predict the quality of wine. Docker is utilized to create a container image for the trained machine learning model, streamlining the deployment process.

**Link for GitHub:**

<https://github.com/Mahidhartanniru/Pa2Winepred/>

**Link for Docker:**

<https://hub.docker.com/r/dt37824/winequlpred>

**Steps for the Execution for Wine Quality Prediction AWS Spark Application:**

1. Create a Key-pair for the EMR Cluster :go to EC2/Network/Key-pairs

Use the format of .pem and download the keypair

Created key pair as: pa2assmahi.pem

2. Create an S3 bucket

Created an S3 bucket in aws: pa2winebucket1

3. Then go to EMR console and create EMR cluster

4. Creating the spark in the AWS instance by using EMR console:

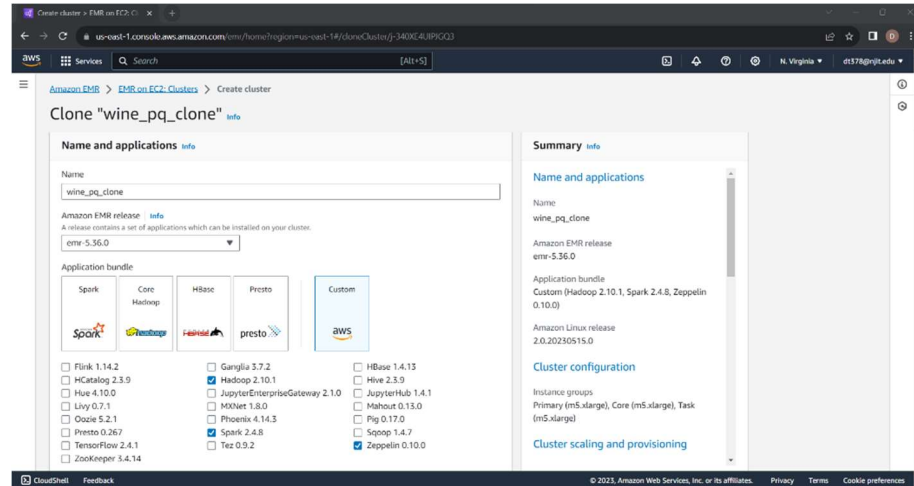
Creating the spark cluster by using the EMR console, and create the 4 instances:

Name and application:

Name: pa2winepqmahi

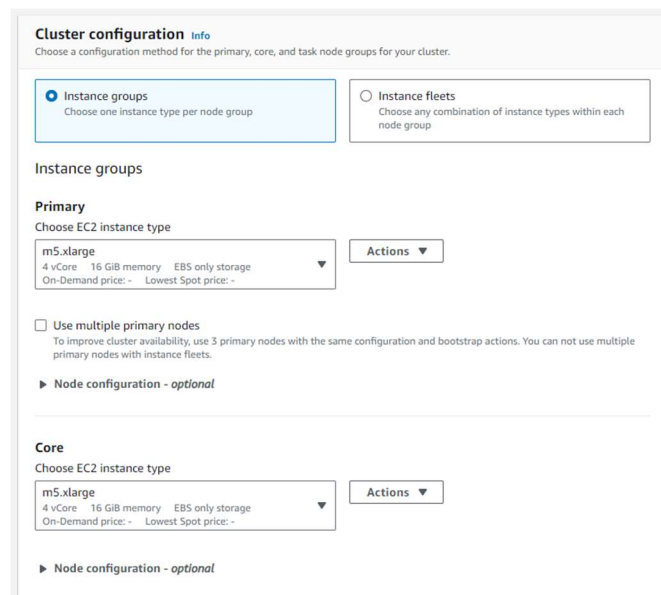
Amazon EMR release: emr-5.33.0

Application bundle: Hadoop 2.10.1, Spark 2.4.7, Zippeline 0.9.0, and Yarn



Note: Here it says Clone “wine\_pq\_clone” as I have cloned the previous configuration instead of creating from scratch to save time.

### Cluster Configuration:



### Cluster Scaling and provisioning:

**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"/>

### Networking & Cluster Termination:

**Networking** [Info](#)

Virtual private cloud (VPC) [Info](#)  
 [Browse](#) [Create VPC](#) [?](#)

Subnet [Info](#)  
 [Browse](#) [Create subnet](#) [?](#)

► EC2 security groups (firewall)

► **Steps - optional** (0) [Info](#) [Remove](#) [Edit](#) [Add](#)  
 Use commands and scripts to tell your cluster where to find and how to process your data. Steps run consecutively unless you enable the Concurrency option.

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

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

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

### Security Configuration and EC2 Key pair & Identity and access management(IAM) roles:

**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](#)

---

**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

---

**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

---

**Custom automatic scaling role - optional**

When a custom automatic scaling rule triggers, Amazon EMR assumes this role to add and terminate EC2 instances. [Learn more](#)

Custom automatic scaling role

We can follow above steps for creating EMR cluster for the instances

The screenshot shows the Amazon EMR console for a cluster named 'pa2winepqmah'. The cluster is in a 'Waiting' state. The console displays various tabs including Summary, Properties, Bootstrap actions, Instances (Hardware), Steps, Applications, Configurations, Monitoring, Events, and Tags (0). The Summary tab is active, showing cluster info, applications, cluster management, and status and time.

5. Now we are training ML model into spark cluster with ec2 instances in parallel:

1. Now the cluster will accept the tasks to run the ML model

Need to connect the Master instance in the Terminal:

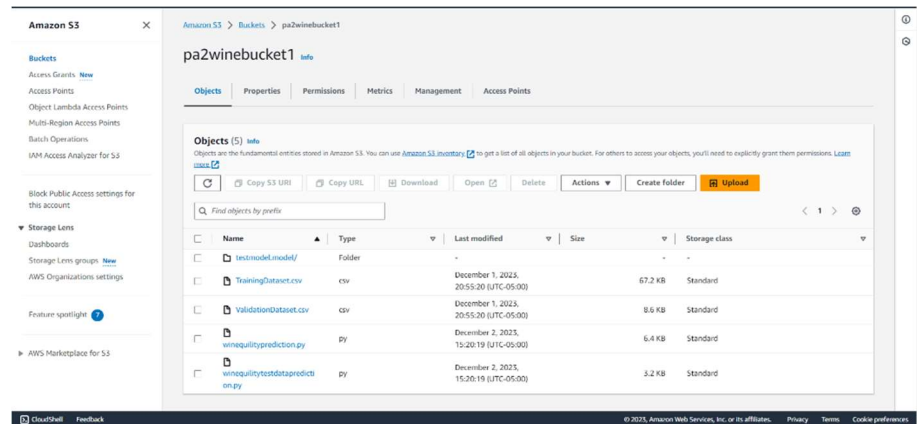
```
ssh -i "pa2assmahi.pem" ec2-user@ec2-44-201-107-82.compute-1.amazonaws.com
```

and it is successfully login.

2. After the login of Master instance then change the root by using



4. Then you can find the trace status for the above tasks, The status is succeed then there is a creation of test.model in the s3 bucket s3://pa2winebucket1



6. Now we are running ML model using the Docker:

1. Create an docker account and sign up.
2. After the successful login then download and setup the docker in your local system
3. Install the docker
4. Login the docker in the power shell by the command  
docker login  
PwD
5. After login you need to build the image:  
docker build -t winequlpred .

```
PS C:\Pa2winepred> docker build -t winequlpred .
[+] Building 2.1s (29/29) FINISHED
=> [internal] load .dockerignore
=> transferring context: 2B
=> [internal] load build definition from Dockerfile
=> transferring dockerfile: 1.31kB
=> [internal] load metadata for docker.io/library/centos:7
=> [auth] library/centos:pull token for registry-1.docker.io
=> [1/22] FROM docker.io/library/centos:7@sha256:bcc5f48bb7764ad36f236b7b515b3678369a5124c47b8d12916d6487418ea4
=> [internal] load build context
=> transferring context: 86.84kB
=> CACHED [2/22] RUN yum -y update && yum -y install python3 python3-dev python3-pip python3-virtualenv java-1.8.0-openjdk wget
=> CACHED [3/22] RUN python3 -V
=> CACHED [4/22] RUN python3 -V
=> CACHED [5/22] RUN pip3 install --upgrade pip
=> CACHED [6/22] RUN pip3 install numpy pandas
=> CACHED [7/22] RUN pip3 install pandas
=> CACHED [8/22] RUN wget --no-verbose -O apache-spark.tgz "https://archive.apache.org/dist/spark/spark-3.1.2/spark-3.1.2-bin-hadoop3.2.tgz" && mkdir -p /opt/spark && tar -xvf apache-spark.tgz -C /opt/sp
=> CACHED [9/22] RUN ln -s /opt/spark-3.1.2-bin-hadoop2.7 /opt/spark
=> CACHED [10/22] RUN (echo "export SPARK_HOME=/opt/spark" >> ~/.bashrc && echo "export PATH=$SPARK_HOME/bin:$PATH" >> ~/.bashrc && echo "export PYSPARK_PYTHON=python3" >> ~/.bashrc)
=> CACHED [11/22] RUN mkdir /code/data
=> CACHED [12/22] RUN mkdir /code/data/csv
=> CACHED [13/22] RUN mkdir /code/data/model
=> CACHED [14/22] RUN mkdir /code/src
=> CACHED [15/22] RUN mkdir /code/data/testdata.model/
=> CACHED [16/22] RUN mkdir /code/data/testdata.model/
=> CACHED [17/22] COPY src/winequalitytestdataprediction.py /code/src
=> [18/22] COPY data/model/testdata.model/ /code/data/model/testdata.model
=> [19/22] COPY data/csv/ /code/data/csv
=> [20/22] RUN rm /bin/sh && ln -s /bin/bash /bin/sh
=> [21/22] RUN /bin/bash -c "source ~/.bashrc"
=> [22/22] RUN /bin/sh -c "source ~/.bashrc"
=> [23/22] WORKDIR /code/
=> exporting to image
=> exporting layers
=> writing image sha256:d7a9776714a5d8db3abbfa73bcfd2c607bc3a74c1fe26596d890ab646f1
=> naming to docker.io/library/winequlpred
what's Next?
View a summary of image vulnerabilities and recommendations -> docker scout quickview
```



6. The push and pull into the docker hub repository:

PUSH:

`docker tag wineqlpred dt37824/wineqlpred`

`docker push dt37824/wineqlpred`

PULL:

`docker pull dt37824/wineqlpred`

```
What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Pa2Winepred> docker login --username dt37824
Password:
Login Succeeded
PS C:\Pa2Winepred> docker tag wineqlpred dt37824/wineqlpred
Error response from daemon: no such image: wineqlpred:latest
PS C:\Pa2Winepred> docker tag wineqlpred dt37824/wineqlpred
PS C:\Pa2Winepred> docker push dt37824/wineqlpred
Using default tag: latest
The push refers to repository [docker.io/dt37824/wineqlpred]
5f70b1d8a60: Mounted from dt37824/wineqlpred
ef7a307724c: Pushed
a35dc626262d: Pushed
608c804654f: Pushed
42631d705ae4: Mounted from dt37824/wineqlpred
ca1f98c285de: Pushed
c3183107060: Mounted from dt37824/wineqlpred
178ad14e99de: Mounted from dt37824/wineqlpred
431ae61442e: Mounted from dt37824/wineqlpred
95092126e5d: Mounted from dt37824/wineqlpred
764acef7612e: Mounted from dt37824/wineqlpred
45409718072: Mounted from dt37824/wineqlpred
4f7080da095d: Mounted from dt37824/wineqlpred
14c20f1208e: Mounted from dt37824/wineqlpred
95c0b2620ce: Mounted from dt37824/wineqlpred
730f8e8eeab: Mounted from dt37824/wineqlpred
307ee2b055e: Mounted from dt37824/wineqlpred
e7571743490: Mounted from dt37824/wineqlpred
157d157a798: Mounted from dt37824/wineqlpred
f346da9fb64: Mounted from dt37824/wineqlpred
568e8ef67b2e: Mounted from dt37824/wineqlpred
1745c68c49b3: Mounted from dt37824/wineqlpred
latest: digest: sha256:880b408198f959fde199aed3d4411eb95ac8b0bfcd95c574514a73c9ffbc82dd size: 5109
PS C:\Pa2Winepred> docker pull dt37824/wineqlpred
Using default tag: latest
latest: Pulling from dt37824/wineqlpred
Digest: sha256:880b408198f959fde199aed3d4411eb95ac8b0bfcd95c574514a73c9ffbc82dd
Status: Image is up to date for dt37824/wineqlpred:latest
docker.io/dt37824/wineqlpred:latest

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Pa2Winepred> docker run -v C:\Pa2Winepred\data\csv wineqlpred trainingdataset.csv
23/12/02 22:10:21 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
```

7. Store your test data file in a designated folder, referred to as "dir." Mount this directory with the Docker container, and execute the container using the following command.

`docker run -v C:\Pa2Winepred\data\csv wineqlpred testdata.csv`

```
Windows PowerShell
PS C:\Pa2Winepred> docker run -v C:\Pa2Winepred\data\csv wineqlpred testdata.csv
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
23/12/03 19:16:18 INFO SparkContext: Running Spark version 3.1.2
23/12/03 19:16:18 INFO ResourceUtils: =====
23/12/03 19:16:18 INFO ResourceUtils: No custom resources configured for spark.driver.
23/12/03 19:16:18 INFO ResourceUtils: =====
23/12/03 19:16:18 INFO SparkContext: Submitted application: Mahidhar.cs641.wine.prediction
23/12/03 19:16:18 INFO ResourceProfile: Default ResourceProfile created, executor resources: Map(cores -> name: cores, amount: 1, script: , vendor: , memory
-> name: memory, amount: 1024, script: , vendor: , offHeap -> name: offHeap, amount: 0, script: , vendor: ), task resources: Map(cpus -> name: cpus, amount
: 1.0)
23/12/03 19:16:18 INFO ResourceProfile: Limiting resource is cpu
23/12/03 19:16:18 INFO ResourceProfileManager: Added ResourceProfile id: 0
23/12/03 19:16:18 INFO SecurityManager: Changing view acls to: root
23/12/03 19:16:18 INFO SecurityManager: Changing modify acls to: root
23/12/03 19:16:18 INFO SecurityManager: Changing view acls groups to:
23/12/03 19:16:18 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(root); groups with vie
w permissions: Set(); users with modify permissions: Set(root); groups with modify permissions: Set()
23/12/03 19:16:19 INFO SparkEnv: Registering BlockManagerMaster
23/12/03 19:16:19 INFO SparkEnv: Registering MasterHeartbeat
23/12/03 19:16:19 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
23/12/03 19:16:19 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
23/12/03 19:16:19 INFO MemoryStore: MemoryStore started with capacity 366.3 MiB
23/12/03 19:16:19 INFO SparkEnv: Registering OutputCommitCoordinator
23/12/03 19:16:20 INFO Utils: Successfully started service 'SparkUI' on port 4080.
23/12/03 19:16:20 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://5bd1d559ad34:4080
23/12/03 19:16:20 INFO Executor: Starting executor ID driver on host 5bd1d559ad34
23/12/03 19:16:20 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 37741.
23/12/03 19:16:20 INFO NettyBlockTransferService: Server created on 5bd1d559ad34:37741
23/12/03 19:16:20 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replication policy
23/12/03 19:16:20 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, 5bd1d559ad34, 37741, None)
23/12/03 19:16:20 INFO BlockManagerMasterEndpoint: Registering block manager 5bd1d559ad34:37741 with 366.3 MiB RAM, BlockManagerId(driver, 5bd1d559ad34, 377
41, None)
23/12/03 19:16:20 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, 5bd1d559ad34, 37741, None)
23/12/03 19:16:20 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, 5bd1d559ad34, 37741, None)
23/12/03 19:16:21 INFO SharedState: Setting hive metastore warehouse dir ('null') to the value of spark.sql.warehouse.dir ('file:/code/spark-warehouse').
```

```

Windows PowerShell
23/12/03 19:16:20 INFO Utils: Successfully started service 'SparkUI' on port 4040.
23/12/03 19:16:20 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://SbdId559ad34:4040
23/12/03 19:16:20 INFO Executor: Starting executor ID driver on host SbdId559ad34
23/12/03 19:16:20 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 37741.
23/12/03 19:16:20 INFO NettyBlockTransferService: Server created on SbdId559ad34:37741
23/12/03 19:16:20 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replication policy
23/12/03 19:16:20 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, SbdId559ad34, 37741, None)
23/12/03 19:16:20 INFO BlockManagerMasterEndpoint: Registering block manager SbdId559ad34:37741 with 366.3 MiB RAM, BlockManagerId(driver, SbdId559ad34, 37741, None)
23/12/03 19:16:20 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, SbdId559ad34, 37741, None)
23/12/03 19:16:20 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, SbdId559ad34, 37741, None)
23/12/03 19:16:21 INFO SharedState: Setting hive metastore warehouse.dir to the value of spark.sql.warehouse.dir ('file:/code/spark-warehouse').
23/12/03 19:16:21 INFO SharedState: Warehouse path is 'file:/code/spark-warehouse'.
--Input file for test data is--
data/csv/testdata.csv

+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|fixed acidity|volatile acidity|citric acid|residual sugar|chlorides|free sulfur dioxide|total sulfur dioxide|density| pH|sulphates|alcohol|quality|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|features|label|raw|prediction|probability|prediction|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|
| 8.0| 0.22| 0.48| 1.8| 0.077|
22,0.48,1...| 1.0|[3.48851027289842...|[0.06977020545796...]| 1.0|
| 7.6| 0.39| 0.31| 2.3| 0.082|
39,0.31,2...| 0.0|[48.1248350799459...|[0.96248767015391...]| 0.0|
| 7.9| 0.43| 0.21| 1.6| 0.106|
43,0.21,1...| 0.0|[48.1539082576783...|[0.96387888515340...]| 0.0|
| 8.5| 0.49| 0.11| 2.3| 0.084|
49,0.11,2...| 0.0|[47.6785701357096...|[0.95357152271019...]| 0.0|
| 6.9| 0.14| 2.4| 0.885|
4,0.14,2.4...| 1.0|[1.82872254349015...|[0.03657445086980...]| 1.0|
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows

None
Test Accuracy of wine prediction model = 0.983580922595778
Weighted f1 score of wine prediction model = 0.9776578095108527
PS C:\Pa2Winepred>

```

**Conclusion:** As shown in the image above, got an accuracy of ~98% while predicting the wine quality.