Programming Assignment - Wine quality prediction ML model in Spark over AWS

Github link -https://github.com/Ritikaa24k/cs643-851-pa2-rk946
Docker link -https://hub.docker.com/r/ritikaa24/wine-quality

This guide will help you build a wine quality prediction ML model in spark over AWS. The model is trained in parallel using four EC2 instances. Then the model is saved in a spark application which performs wine quality prediction and runs on one EC2 instance. This project is implemented in python on ubuntu Linux.

The following are the steps to replicate this application:

1. Login to your AWS student account.
2. Under the AWS details button you will find AWS access_key, secret_key and session_token copy that and paste it in your ~/.aws/credentials file.
3. Click on start lab. To start the AWS console click the AWS button when the circle turns green. The green circle indicates that the lab is active.
4. The above action redirects you to the AWS management console, here search for EC2 and head over to the EC2 dashboard.

A. Training the model:

This project uses flintrock to create the EC2 instance cluster for training. The following are the steps on how to create your flintrock cluster.

1. Once you are on the EC2 management console. Click on launch instance.
2. For the first EC2 instance enter the name as Spark_Cluster_creator.
3. Under the AMI select Amazon Linux 2 AMI(HVM) - Kernel 5.10,SSD Volume type.
4. Select instance type to be t2.micro.
5. For the key pair value create a new keypair and name it sparkkey.
6. Under Network settings, select create Security group and implement the below settings.
    a. Allow SSH traffic from My IP
    b. Allow HTTPs traffic from the internet
    c. Allow HTTP traffic from the internet
7. On the terminal cd downloads and change the permissions on the key to be read only, by running the following command : **chmod 400 sparkkey.pem**
8. Transfer the key to the EC2 instance using the following command:
    scp -i /path/to/sparkkey.pem/path/to/sparkkey.pem ec2-user@<public ip>:/home/ec2-user/

B. Setting up flintrock cluster:

1. SSH into Spark_Cluster_creator using the following command.

   ssh -i /path/to/sparkkey.pem ec2-user@<public ip>
2. Install flintrock using the command: pip3 install flintrock
3. The configure the flintrock file by using the following command: flintrock config
4. In the configuration file set the following variables:

```
services:
  spark:
    version: 2.2.0

  hdfs:
    version: 2.7.5

provider: ec2

providers:
  ec2:
    key-name: spark_cluster
    identity-file: /path/to/spark_cluster.pem
    instance-type: m4.large
    region: us-east-1
    ami: ami-97785bed   # Amazon Linux, us-east-1
    vcp-id: your vpc id
    subnet-id: your subnet id   #
    user: ec2-user
    tenancy: default  # default | dedicated
    ebs-optimized: no  # yes | no
    instance-initiated-shutdown-behavior: terminate  # terminate | stop

launch:
  num-slaves: 4
  install-hdfs: True
  install-spark: True
  java-version:8

debug: false
```

5. Create the flintrock cluster using the command: **flintrock create my-spark-cluster**
6. Check the cluster details this gives the public and private ip addresses for your master and slave nodes;use the command: **flintrock describe my-spark-cluster**
7. Launch the spark cluster using the command **flintrock launch my-spark cluster**
8. SSH into the master node and scp your training.py into the master node.
9. Upload the training and validation datasets into your S3 bucket.
10. Run the training file using the following command:

    **spark/bin/spark-submit **

**--master spark://172.31.28.225:7077 **
**--conf "spark.hadoop.fs.s3a.impl=org.apache.hadoop.fs.s3a.S3AFileSystem" **
/home/ec2-user/training.py

11. Within a few minutes you'll obtain the trained model in the s3 bucket at the mentioned output path.

C. Prediction application:

Prediction application without Docker: (on ubuntu instance)

1. Update the system package using the following command: sudo apt update
2. Install JDK using the command: **sudo apt install -y openjdk-11-jdk**
3. Download and install spark using the following commands:
   wget https://dlcdn.apache.org/spark/spark-3.4.0/spark-3.4.0-bin-hadoop3.tgz
   tar xvf spark-3.4.0-bin-hadoop3.tgz
   sudo mv spark-3.4.0-bin-hadoop3 /opt/spark
4. Configure the environment variables:
   nano ~/.bashrc
   Add the following lines at the end of the file
   - **export SPARK_HOME=/opt/spark**
   - **export PATH=$PATH:$SPARK_HOME/bin**
   - **export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-amd64**
   - **export PATH=$JAVA_HOME/bin:$PATH**
   - **export AWS_ACCESS_KEY_ID=<your AWS_ACCESS_KEY_ID from AWS Academy>**
   - **export AWS_SECRET_ACCESS_KEY=<your AWS_SECRET_ACCESS_KEY from AWS Academy>**
   - **export AWS_SESSION_TOKEN=<your AWS_SESSION_TOKEN from AWS Academy>**
5. Reload .bashrc:
   source ~/.bashrc
6. Get the prediction code by cloning the git repo using the command: **git clone <link to repo>**
7. Install the dependencies:
   - **Pip install pyspark and numpy**
8. Submit the prediction file to spark using the command:
   - **spark-submit prediction.py**

D. Prediction application with Docker:

1. Create the image:
    - **Sudo docker login**
    - **docker tag prediction-image:latest yourdockerhubusername/prediction-image:latest**
      Push the image
        - **docker push yourdockerhubusername/prediction-image:latest**
2. If required pull the image:
    - **docker pull yourdockerhubusername/prediction-image:latest**
3. Run the docker container:
    - **docker run -it --name prediction-container prediction-image**
4. Verify the running container:
    - **docker ps**
5. If required you can access the running container using the command:
    - **docker exec -it prediction-container bash**
6. Stop and remove the container:
    - **docker stop prediction-container**
    - **docker rm prediction-container**