Parth Panara

CS643 – Cloud Computing

Programming Assignment: 2

GitHub Link:

https://github.com/parth-panara/cloudComputing_project2

(Note that GitHub repository is currently private. So, I have collaborated with Professor and TA)

DockerHub Link:

https://hub.docker.com/repository/docker/parthpanara/cloudcomputing_project2/general

Instruction to set the Cloud Environment and run the App

➔ Following are the steps to set the AWS EMR service in the cloud environment.

- Login AWS academy student login by following this https://www.awsacademy.com/vforcesite/LMS_Login
- Click on Courses -> Modules -> Leaner Lab and start Lab. After few seconds, there will be a green color circle pops up which is attached with AWS. So, click on it to connect with AWS Management Console.
- ===EMR service [Follow the below steps to create 1-master and 4-slave nodes for cluster as EC2 instances]
- Click on Services -> EMR ->create cluster
- Under the cluster information fill-out the below,

  Name and App:
  Amazon EMR release: emr-6.12.0
  Applications included in bundle: Spark 3.4.0 on Hadoop 3.3.3 YARN with and Zeppelin 0.10.1
  Instance Groups:
  Operating system options: Amazon Linux release
  Primary- EC2 instance type: m4.large
  Core- EC2 Instance type: m4.large
  Cluster scaling and provisioning option:
  Choose an option: Set cluster size manually
  Provisioning configuration-instance size: 4 (because we need 4 slave nodes for app)
  Cluster Termination:
  Select the option: Manually terminate cluster
  Security configuration and EC2 Key pair:
  Amazon EC2 key pair for SSH to the cluster: Create key pair (we create the key-pair same as we create during project-1 and save it on local computer)
  Identity and Access Management (IAM) roles:
  Amazon EMR service role: EMR_DefaultRole
  EC2 instance profile for Amazon EMR: EMR_EC2_DefaultRole

  Keep rest of the options default and Click → Create Cluster

- After around one minute, we can see five EC2 instances have been created under the EC2 Dashboard.
- Now, select the first instance→click on 'Security'→Click the link located under the Security groups→Click " Edit inbound rules"
- Click "Add rule"→ Click "Custom TCP" → Select SSH → then Click "Custom" change it to "My IP"→ click "save rules"
- Follow the same above two points for second, third, fourth and fifth instance.

→ After running the five EC2 instances successfully, run the following commands step by step by connecting each individual instance with the local terminal individually. Open the local terminal.

- Note that we can also open the five individual terminals in the local machine, and run the below command to connect each EC2 instance.

  $ ssh -i <path of key file .pem from local machine> hadoop@< Public IPv4 DNS >

  To install required packages, run the below command in each EC2 instances

  $ sudo yum update

  Now, we will install GitHub intoEC2 instance.

  $ sudo yum install git

  Pull GitHub repository by running following command,

  $ git clone https://github.com/parth-panara/cloudComputing_project2.git

  Navigate the project folder which we pulled from GitHub and make a new directory and copy two data files in the /app directory.

  $ cd cloudComputing_project2

  $ sudo mkdir /app

  $ sudo cp TrainingDataset.csv ValidationDataset.csv /app/

  $ cd

  To Install apache-spark and work with its files, create a 'auto.sh' named bash file in all five nodes.

  $ vi auto.sh

  In the above file, we will copy the text from file '/home/hadoop/cloudComputing2/shFiles/auto.sh'(File is also available on GitHub) and paste in it. Then, save it.

  To permit the file for execution

  $ chmod 755 auto.sh

  Now, one script will only be created for master node to operate Apache Spark Cluster.

$ vi masternode.sh

In the above file, we will copy the text from file '/home/hadoop/cloudComputing2/shFiles/masternode.sh' and paste in it. Then, save it.
To permit the file for execution

$ chmod 755 masternode.sh

Now, we will create a bash script for each four slave nodes individually.

$ vi slavenode<x>.sh [choose 'x' as instance number 1, 2, 3, 4]

In the above ' slavenode' files, we will copy the text from file '/home/hadoop/cloudComputing2/shFiles/slavenodex.sh' and paste in it. Then, save it.
To permit the each file for execution

$ chmod +x slavenode<x>.sh ['x' means the instance number 1,2,3,4]

Run the auto.sh script to execute apache-spark clusters packages for the all five nodes.

$ ./auto.sh

Run the masternode.sh script to execute apache-spark packages in the master node.

$ ./masternode.sh

Run the script for slave nodes to execute apache-spark packages.

$ ./slavenode<x>.sh [choose the 'x' as instance number 1, 2, 3, 4]

➔ **Training the Model and choosing the best F1 Score model for prediction**

Run the below commands to install required pip before going through Train_app.py file:

$ sudo yum install python3-pip

$ pip install numpy

$ pip install pandas

$ pip install Quinn

$ pip install pyspark

$ pip install findspark

Run the python file 'Train_app.py' to get the output of the Logistic Regression Model, Random Forest Classifier Model, and GBT Classifier Model include training on all four nodes.

$ python /home/hadoop/cloudComputing_project2/Train_app.py

Now, the Output received from the code file 'Train_app.py':

F1 Score for Logistic Regression Model: 0.732
F1 Score for Random Forest Classifier Model: 0.714
F1 Score for GBT Classifier Model: 0.682

Best F1 Score: 0.732

Best Model: Logistic Regression Model
Accuracy of the Best Model: 0.731

Best model Logistic Regression Model is selected for prediction application

Best model saved to: /home/hadoop/cloudComputing_project2/bestwinequalitymodel

Now, we will have the Best model saved at the path '/home/hadoop/cloudComputing_project2/bestwinequalitymodel'
To located it run the below command and push it to the AWS S3 bucket to view and store.
$ ls
$ aws s3 sync /home/hadoop/cloudComputing_project2/bestwinequalitymodel'
s3://bestwinequalitymodel/trainedmodel/

→ To run Prediction_app.py without Docker.

- At this point, our master node of EC2 Instance is already connected with the local terminal, so run the below command to install java:
  $ export JAVA_HOME=/usr/bin/java
  Now, we will install the Anaconda:
  $ cd /tmp
  $ curl -O https://repo.anaconda.com/archive/Anaconda3-2020.11-Linux-x86_64.sh
  $ bash Anaconda3-2020.11-Linux-x86_64.sh
  Follow the instruction to install the Anaconda. After installing it we will copy the code from file 'bashrc.sh'which is available on my GitHub account under project repository.
  $ vi bashrc
  Now paste the code in the above file and save it.

To load the file run the following command

$ source ~/.bashrc

We will type 'exit' in the terminal to close SSH instance. We will open new terminal and SSH again with same EC2 instance (Master node). Run the code for Jupiter password.

$ jupyter notebook password

Run the command to start up Jupyter Notebook without browser.

$ jupyter notebook --no-browser

Now keep this terminal as it is running, open another new terminal on local machine and run the below command to open Jupyter in the browser through EC2 instance tunnel.

$ ssh -i <path of key file .pem from local machine> -N -f -L localhost:8888:localhost:8888 hadoop@<Public IPv4 DNS> "jupyter notebook"

At this point we can navigate 'http://localhost:8888/'and enter the Jupyter password that we did in the previous step. Once we are in, we will create a new Python script. Go to the New → Python3. In this opened notebook write the below command in the first shell to install 'quin'

!pip install Quinn

Once installation is done, open new shell and install 'pyspark'by following command.

!pip install pyspark

Open the third shell, now we will copy the code from 'Prediction_app.py' file and paste in the third shell(Note that we need to remove the line 'from app import app' before run the script) In the output of the prediction app, we can see the data is loaded from csv files, formats it and loads the train model to predict the wine quality and it results the F1 Score using the ValidationDataset.csv file

The F1 score of the prediction app model is 0.732 which we can receive in output and result of prediction as well.

→ To run Prediction_app.py with Docker.

- We will create a Docker image and run it through Docker container. So, install the Docker into the EC2 instance by following.

  $ sudo yum update

  $ sudo yum install ca-certificates curl gnupg

  $ sudo install -m 0755 -d /etc/apt/keyrings

Parth Panara

CS643 – Cloud Computing

Programming Assignment: 2

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
$ sudo chmod a+r /etc/apt/keyrings/docker.gpg
$ sudo amazon-linux-extras install docker
```

To start the Docker

```
$ sudo service docker start
```

Check the Docker if it's installed successfully by running following command,

```
$ sudo docker info
$ sudo docker run hello-world
```

Now check the User groups of Docker

```
$ groups
```

By default we can notice that 'hadoop' would be there. We can also use this user.

```
$ sudo usermod -aG docker hadoop
```

Login to DockerHub by following command, where we will push our image.

```
$ sudo docker login
```

Now, navigating to our project directory where we have saved 'DockerFile'

```
$ cd cloudComputing_project2
```

To build the docker image

```
$ sudo docker build -t parthpanara/cloudcomputing_project2 .
```

To see the created image

```
$ sudo docker images
```

To run the Docker image by following command

```
$ sudo docker run -p 5000:5000 -p 4040:4040 -v
/home/hadoop/cloudComputing_project2/bestwinequalitymodel:/home/hadoop/cloudComp
uting_project2/bestwinequalitymodel parthpanara/cloudcomputing_project2
```

Now the Docker container is started successfully. We can see the output of the prediction application and The F1 score of the prediction app model is 0.732 as well. At this point our Flask app will be turned on to run which can be helpful to see the application output on web browser. Now, open the new terminal for the further steps and SSH into the same EC2 instance

To see the running container with the created active image run the following command.

```
$ sudo docker ps
```

We can notice that the status is ”up” of the Docker container that means application is running successfully and showing output in the Docker environment.

Push the image to the DockerHub by running the following,

```
$ sudo docker push parthpanara/cloudcomputing_project2:latest
```

We can see the image through:
https://hub.docker.com/repository/docker/parthpanara/cloudcomputing_project2/general