**Setting up Spark Cluster**

**1. Install Flintrock**

To begin, you need to install Flintrock, a tool for launching and managing Apache Spark clusters. You can install Flintrock using pip3:

pip3 install flintrock

**2. Configure Flintrock**

After installing Flintrock, you need to configure it with your EC2 key pair (PEM file) and desired cluster settings.

Update Flintrock Configuration

Locate the Flintrock configuration file at .config/flintrock/config.yaml. If the file doesn't exist, you can create it with **flintrock configure**.

**3. Launch the Cluster**

To create a multi-node cluster with 1 master node and 4 worker nodes, run the following command:

flintrock launch wine-cluster

**4. Copy Training Dataset to the Cluster**

Next, you need to copy the training dataset (e.g., TrainingDataset.csv) to the cluster. Use the following command:

flintrock copy-file wine-cluster TrainingDataset.csv /home/ec2-user/

**5. Log in to the Cluster**

To log in to the master node of the cluster, use the following command:

flintrock login wine-cluster

Once you are logged in, proceed with the following steps for training.

**Training**

**1. Install NumPy and Git**

Before running the training code, make sure to install Git:

sudo yum install git

**2. Clone Git Code**

Clone the Git repository containing the training code:

git clone <repository-url>

Replace <repository-url> with the URL of the Git repository containing your training code.

**3. Run Training on the Cluster**

Now, you can run the training code on the cluster with four worker nodes:

* javac -cp "/home/ec2-user/spark/jars/\*" RFCTraining.java
* echo Main-Class: RFCTraining > Manifest.txt
* jar cvfm RFCTraining.jar Manifest.txt RFCTraining.class
* spark-submit --class RFCTraining --master spark://<public-ip>:7077 RFCTraining.jar

Replace <public-ip> with the public IP address of the master node, which you can find on the EC2 dashboard.

**Inference**

**1. Install Docker**

For inference, you'll need to install Docker on the cluster. Run the following commands:

sudo yum install docker

sudo systemctl restart docker

sudo systemctl status docker

sudo usermod -aG docker $USER

**2. Pull and Run Docker Image**

Navigate to the spark directory containing the inference code and Dockerfile. Pull and run the Docker image:

docker pull pt346/wineprediction:latest

docker run -v /home/ec2-user/spark:/home/ec2-user/spark -p 5000:5000 pt346/wineprediction:latest

**3. Update HTML File for Inference**

Locate the HTML file that you want to use for inference. Edit the file and update the URL to point to your cluster's public IP and port 5000:

http://<public-ip>:5000/predict

Replace <public-ip> with the public IP address of the master node.

**4. Set Security Group Rules**

Ensure that port 5000 is open in the security group associated with the master node. This will allow external access to the Flask application running inside the Docker container.

With these steps completed, you should now be able to access the HTML file in your local browser, select a validation CSV, and submit it. The Flask application will perform predictions using the trained model on the cluster and display the F1 score on the UI.