Intro To Big Data Project

AWS EC2 Instance Setup And Execution Of Flight Data Analysis Guide

Prerequisites

- AWS account
- `ibdproject.pem` key file with chmod set to 400 for secure access

Instance Creation

Step 1: Launch EC2 Instances

Launch two Linux EC2 instances using the following AMIs and names:

- **AMI ID:** `ami-0178d42118c1f7677`
  - `m1(namenode)`
  - `s1(datanode)`
  - `s2(datanode)`
  - `s3(datanode)`

Step 2: Security Group Configuration

Ensure both instances are in the same security group with permissions set to allow all traffic.

Passphrase-less SSH Setup

Step 1: Connect to Instances

Use Putty to connect to both instances. Ensure that you have configured Putty to use the `ibdproject.pem` key.

Step 2: Generate SSH Keys

On each instance, generate a pair of authentication keys:

```
ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
```

Step 3: Configure Authorized Keys

For each instance, copy the public key (`id_rsa.pub`) and append it to the other instance's `authorized_keys` :

```
cat ~/.ssh/id_rsa.pub | vim ~/.ssh/authorized_keys
```

Step 4: Hosts File Configuration

On each instance, edit the `/etc/hosts` file to add the private IP addresses and hostnames:

```
sudo nano /etc/hosts
```

Add the following lines (replace with actual private IPs):

```
172.31.80.111 master
172.31.82.18 slave1
172.31.56.27 slave2
172.31.48.92 slave3
```

Note: AWS may change the public IP if you stop and start an instance. Always use private IPs for internal communication.

Step 5: Test SSH Configuration

Verify the setup by attempting to SSH from one instance to another:

```
ssh slave1
```

```
ssh slave2
```

```
ssh slave3
```

If you can connect without entering a passphrase, the configuration is successful.

Hadoop Installation on Namenode and Datanodes

Prerequisites

- Ubuntu operating system.
- `sudo` privileges on the system.

Installation Steps

1. Install Java

Update the package list and install Java:

```
sudo apt-get update
sudo apt-get install openjdk-8-jdk -y
```

Locate the Java home path:

```
ls /usr/lib/jvm
```

Set `JAVA_HOME` in your bash profile:

```
vim ~/.bash_profile
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
source ~/.bash_profile
```

2. Install Hadoop on the Master Node

Download and extract Hadoop:

```
cd ~
wget <https://archive.apache.org/dist/hadoop/common/hadoop-2.6.5/hadoop-2.6.5.ta
r.gz>
tar -xvzf hadoop-2.6.5.tar.gz
```

Add Hadoop to the environment variables:

```
vim ~/.bash_profile
export HADOOP_HOME=/home/ubuntu/hadoop-2.6.5
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
source ~/.bash_profile
```

3. Configure Hadoop on the Master Node

Edit the configuration files as follows:

core-site.xml:

```
<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://master:9000/</value>
  </property>
  <property>
    <name>hadoop.tmp.dir</name>
    <value>file:/home/ubuntu/hadoop-2.6.5/tmp</value>
  </property>
</configuration>
```

hdfs-site.xml:

```
<configuration>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:/home/ubuntu/hadoop-2.6.5/dfs/name</value>
  </property>
  <property>
    <name>dfs.datanode.data.dir</name>
    <value>file:/home/ubuntu/hadoop-2.6.5/dfs/data</value>
  </property>
  <property>
    <name>dfs.replication</name>
    <value>3</value>
  </property>
</configuration>
```

mapred-site.xml and yarn-site.xml:

(See additional configurations in the provided guide.)

4. Distribute Configuration

Copy the configured Hadoop directory to the slaves:

```
cd ~
scp -r hadoop-2.6.5 slave1:~
```

```
cd ~
scp -r hadoop-2.6.5 slave2:~
```

```
cd ~
scp -r hadoop-2.6.5 slave3:~
```

5. Finalize Setup

Ensure all nodes have the correct environment variables and configurations as outlined above. Start your Hadoop cluster as per the official documentation.

Start Hadoop and Test

1. Change to Hadoop Directory

```
cd hadoop-2.6.5
```

2. Format the Namenode

```
bin/hdfs namenode -format
```

3. Start the HDFS

```
sbin/start-dfs.sh
```

4. Start YARN

```
sbin/start-yarn.sh
```

5. Verify the Running Services with JPS

```
jps
```

Expected Outputs:

- On Namenode: Jps, NameNode, SecondaryNameNode, ResourceManager

- On Datanode: Jps, DataNode, NodeManager

Stop Hadoop Services

```
sbin/stop-yarn.sh
sbin/stop-dfs.sh
```

Installation of Maven and Oozie

Install Maven

```
cd ~
wget <https://archive.apache.org/dist/maven/maven-3/3.5.3/binaries/apache-maven-
3.5.3-bin.tar.gz>
tar -xzvf apache-maven-3.5.3-bin.tar.gz
vim ~/.bash_profile
export M2_HOME=/home/ubuntu/apache-maven-3.5.3
export PATH=$PATH:$M2_HOME/bin
source ~/.bash_profile
mvn -version
```

Install MySQL

```
sudo apt-get install mysql-server
sudo apt install mysql-client
sudo apt install libmysqlclient-dev
```

Install Oozie

```
cd ~
wget <https://archive.apache.org/dist/oozie/4.1.0/oozie-4.1.0.tar.gz>
tar -xzvf oozie-4.1.0.tar.gz
cd oozie-4.1.0
nano pom.xml
# Change <http://repo1.maven.org/maven2/> to <https://repo1.maven.org/maven2/>
bin/mkdistro.sh -DskipTests -Dhadoopversion=2.6.5
cp /home/ubuntu/oozie-4.1.0/distro/target/oozie-4.1.0-distro.tar.gz /home/ubunt
u/oozie-4.1.0-distro.tar.gz
cd ~
mv oozie-4.1.0 backforoozie
tar -xzvf oozie-4.1.0-distro.tar.gz
nano ~/.bash_profile
export OOZIE_HOME=/home/ubuntu/oozie-4.1.0
export OOZIE_CONFIG=$OOZIE_HOME/conf
export CLASSPATH=$CLASSPATH:$OOZIE_HOME/bin
source ~/.bash_profile
```

Configure Proxy Settings in Hadoop

```
nano hadoop-2.6.5/etc/hadoop/core-site.xml
```

Add the following properties:

```
<property>
    <name>hadoop.proxyuser.ubuntu.hosts</name>
    <value>*</value>
</property>
<property>
    <name>hadoop.proxyuser.ubuntu.groups</name>
    <value>*</value>
</property>
```

Restart Hadoop Services

```
cd ~/hadoop-2.6.5
sbin/start-dfs.sh
sbin/start-yarn.sh
```

Configure Oozie Site

```
cd ~/oozie-4.1.0/conf
nano oozie-site.xml
```

Add the following properties to `oozie-site.xml`:

```
<property>
    <name>oozie.service.JPAService.jdbc.driver</name>
    <value>com.mysql.cj.jdbc.Driver</value>
</property>
<property>
    <name>oozie.service.JPAService.jdbc.url</name>
    <value>jdbc:mysql://localhost:3306/oozie?useSSL=false</value>
</property>
<property>
    <name>oozie.service.JPAService.jdbc.username</name>
    <value>oozie</value>
</property>
<property>
    <name>oozie.service.JPAService.jdbc.password</name>
    <value>mysql</value>
</property>
<property>
    <name>oozie.service.HadoopAccessorService.hadoop.configurations</name>
    <value>*=/home/ubuntu/hadoop-2.6.5/etc/hadoop</value>
</property>
```

```
<property>
    <name>oozie.service.WorkflowAppService.system.libpath</name>
    <value>hdfs://master:9000/user/ubuntu/share/lib</value>
</property>
```

Initialize MySQL for Oozie

```
mysql -uroot -p
# Enter password 'root'
CREATE DATABASE oozie;
CREATE USER 'oozie'@'%' IDENTIFIED BY 'mysql';
GRANT ALL ON oozie.* TO 'oozie'@'%';
FLUSH PRIVILEGES;
exit
```

Prepare Oozie Library Extensions

```
cd ~
wget <https://repo1.maven.org/maven2/mysql/mysql-connector-java/8.0.11/mysql-con
nector-java-8.0.11.jar>
wget <http://archive.cloudera.com/gplextras/misc/ext-2.2.zip>
cd ~/oozie-4.1.0/
mkdir libext
cp ../hadoop-2.6.5/share/hadoop/*/lib/*.jar libext/
cp ../hadoop-2.6.5/share/hadoop/*/*.jar libext/
cp ../mysql-connector-java-8.0.11.jar libext/
cp ../ext-2.2.zip libext/
cd libext
mv servlet-api-2.5.jar servlet-api-2.5.jar.bak
mv jsp-api-2.1.jar jsp-api-2.1.jar.bak
mv jasper-compiler-5.5.23.jar jasper-compiler-5.5.23.jar.bak
mv jasper-runtime-5.5.23.jar jasper-runtime-5.5.23.jar.bak
mv slf4j-log4j12-1.7.5.jar slf4j-log4j12-1.7.5.jar.bak
```

Prepare Oozie WAR

```
cd ~/oozie-4.1.0/
sudo apt-get install zip unzip
bin/oozie-setup.sh prepare-war
nano conf/oozie-env.sh
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export OOZIE_PREFIX=/home/ubuntu/oozie-4.1.0
export OOZIE_CONF_DIR=/home/ubuntu/oozie-4.1.0/conf/
export OOZIE_HOME=/home/ubuntu/oozie-4.1.0
export CLASSPATH=$CLASSPATH:$OOZIE_HOME/libext/*.jar
source conf/oozie-env.sh
tar -xzvf oozie-sharelib-4.1.0.tar.gz
```

Deploy Oozie Sharelib to HDFS

```
cd ~/hadoop-2.6.5
bin/hdfs dfs -mkdir /user
bin/hdfs dfs -mkdir /user/ubuntu
bin/hdfs dfs -put ../oozie-4.1.0/share /user/ubuntu/
```

Start Required Services

```
cd ~/hadoop-2.6.5
# Assuming DFS and YARN are already started
sbin/mr-jobhistory-daemon.sh start historyserver
```

Initialize Oozie Database

```
cd ~/oozie-4.1.0
bin/ooziedb.sh create -sqlfile oozie.sql -run
```

Start Oozie Service

```
bin/oozied.sh start
bin/oozie admin --oozie <http://localhost:11000/oozie> -status
# You should see 'System mode: NORMAL'
```

Run Example Jobs

```
tar -xzvf oozie-examples.tar.gz
nano examples/apps/map-reduce/job.properties
# Modify namenode and jobtracker according to your Hadoop configuration
export OOZIE_URL=http://localhost:11000/oozie
cd ~/hadoop-2.6.5
bin/hdfs dfs -put ../oozie-4.1.0/examples/ /user/ubuntu/
cd ~/oozie-4.1.0
bin/oozie job -oozie <http://localhost:11000/oozie> -config examples/apps/map-re
duce/job.properties -run
# Note the job ID returned, e.g., job: 0000000-230502071131377-oozie-ubun-W
```

Monitor and View Job Results

```
# Use the job ID to monitor status
bin/oozie job -oozie <http://localhost:11000/oozie> -info your-job-ID

# View results
cd ~/hadoop-2.6.5
bin/hdfs dfs -cat /user/ubuntu/examples/output-data/map-reduce/part-00000

# Alternatively, retrieve and view results locally
```

```
bin/hdfs dfs -get /user/ubuntu/examples/output-data/map-reduce
cd map-reduce
cat part-00000
```

You have successfully set up and run an Oozie workflow example on Hadoop. Congratulations!

Execution of Flight Data Analysis Code using Oozie

- Unzip Project Files then go into directory

```
cd FlightDataAnalysisOozie
```

Compile Java Files

Navigate to the Java code directory and compile the files.

```
cd FlightDataAnalysis-Code
javac -classpath $HADOOP_HOME/share/hadoop/common/hadoop-common-2.6.5.jar:$HADOO
P_HOME/share/hadoop/mapreduce/hadoop-mapreduce-client-core-2.6.5.jar:$HADOOP_HOM
E/share/hadoop/common/lib/commons-cli-1.2.jar -d ./ *.java
```

Create a JAR File

Assuming compiled Java class files are located in a 'data' folder.

```
jar -cvf Flight.jar -C data/ .
```

Create HDFS Directories

Create necessary directories in HDFS for storing the JAR files and project data.

```
hdfs dfs -mkdir /user/ubuntu/hadoop
hdfs dfs -mkdir /user/ubuntu/hadoop/lib
```

Upload JAR to HDFS

Make sure to provide the correct path in the Oozie workflow.

```
hdfs dfs -put /home/ubuntu/FlightDataAnalysisOozie/FlightDataAnalysis-Code/data/
Flight.jar /user/ubuntu/hadoop/lib
```

Upload Project Directory to HDFS

```
hdfs dfs -put /home/ubuntu/FlightDataAnalysisOozie/ /user/ubuntu/
```

Execute the Oozie Job

Run the job using the appropriate job configuration.

```
bin/oozie job -oozie <http://localhost:11000/oozie> -config /home/ubuntu/FlightD
ataAnalysisOozie/job.properties -run
```

Retrieve Output from HDFS

After successful execution, pull the output data from HDFS.

```
hdfs dfs -get /user/ubuntu/FlightDataAnalysisOozie/output
```

Here End Comes See How easy it is 😏