MD FAKRUL ISLAM (613839)

Kafka-Spark-Hive Integration

In this example, I will do the below things.

- create a stream of tweets that will be sent to a Kafka queue
- pull the tweets from the Kafka cluster
- calculate the character count and word count for each tweet
- save this data to a Hive table

To do this, I am going to set up an environment that includes

- a single-node Kafka cluster
- a single-node Hadoop cluster
- Hive and Spark

1. VM setup in my azure account:

I created an instance of Ubuntu.

2. Install Kafka

```
~$ wget http://apache.claz.org/kafka/2.2.0/kafka_2.12-2.2.0.tgz

~$ mv kafka_2.12-2.2.0.tgz kafka

~$ sudo apt install openjdk-8-jdk -y

~$ java -version

~$ pip3 install kakfa-python
~$ pip3 list | grep kafka
```

3. Install Hadoop:

```
~$ wget https://archive.apache.org/dist/hadoop/common/hadoop-2.8.5/hadoop-2.8.5.tar.gz
~$ tar -xvf hadoop-2.8.5.tar.gz
~$ mv hadoop-2.8.5.tar.gz hadoop
~$ cd hadoop
~/hadoop$ pwd
/home/<USER>/hadoop
```

Edit .bashrc and add the following.

```
export HADOOP_HOME=/home/<USER>/hadoop
export HADOOP_CONF_DIR=$HADOOP_HOME/etc/hadoop
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
~$ source .bashrc
```

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HADOOP_CONF_DIR=${HADOOP_CONF_DIR:-"/home/<USER>/hadoop/etc/hadoop"}
```

Replace the file core-site.xml with the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:9000</value>
  </property>
</configuration>
```

Replcae the file hdfs-site.xml with the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
  <property>
    <name>dfs.permission</name>
    <value>false</value>
  </property>
</configuration>
```

```
~$ sudo apt install openssh-server openssh-client -y
```

Then, set up password-less authentication

```
~$ ssh-keygen -t rsa
~$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

~$ ssh localhost
~$ hdfs namenode -format
~$ start-dfs.sh

#Test The Installation
~$ hadoop fs -ls /
```

4. Install Hive

```
~$ wget http://archive.apache.org/dist/hive/hive-2.3.5/apache-hive-2.3.5-bin.tar.gz
~$ tar -xvf apache-hive-2.3.5-bin.tar.gz
~$ mv apache-hive-2.3.5-bin.tar.gz hive
```

Add the following to the .bashrc and run it with source

```
export HIVE_HOME=/home/<USER>/hive
export PATH=$PATH:$HIVE_HOME/bin
```

Give it a quick test with

```
~$ hive --version
```

Add the following directories and permissions to HDFS

```
~$ hadoop fs -mkdir -p /user/hive/warehouse
~$ hadoop fs -mkdir -p /tmp
~$ hadoop fs -chmod g+w /user/hive/warehouse
~$ hadoop fs -chmod g+w /tmp
```

Inside ~/hive/conf/, create/edit hive-env.sh and add the following

```
export HADOOP_HOME=/home/<USER>/hadoop
export HADOOP_HEAPSIZE=512
export HIVE_CONF_DIR=/home/<USER>/hive/conf
```

While still in ~/hive/conf, create/edit hive-site.xml and add the following

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<configuration>
  <property>
    <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:derby:;databaseName=/home/davis/hive/metastore_db;create=true</value>
    <description>JDBC connect string for a JDBC metastore.</description>
  </property>
  <property>
    <name>hive.metastore.warehouse.dir</name>
    <value>/user/hive/warehouse</value>
    <description>location of default database for the warehouse</description>
  </property>
  <property>
    <name>hive.metastore.uris</name>
    <value>thrift://localhost:9083</value>
```

```
        <description>Thrift URI for the remote metastore.</description>
    </property>
    <property>
        <name>javax.jdo.option.ConnectionDriverName</name>
        <value>org.apache.derby.jdbc.EmbeddedDriver</value>
        <description>Driver class name for a JDBC metastore</description>
    </property>
    <property>
        <name>javax.jdo.PersistenceManagerFactoryClass</name>
        <value>org.datanucleus.api.jdo.JDOPersistenceManagerFactory</value>
        <description>class implementing the jdo persistence</description>
    </property>
    <property>
        <name>hive.server2.enable.doAs</name>
        <value>false</value>
    </property>
</configuration>
```

```
~/hive$ mv lib/log4j-slf4j-impl-2.6.2.jar lib/log4j-slf4j-impl-2.6.2.jar.bak
```

Now we need to create a database schema for Hive to work with using schematool

```
~$ schematool -initSchema -dbType derby
~$ hive --services metastore
```

Now, enter the Hive shell with the hive command

```
~$ hive
```

```
hive>
```

Make the database for storing our Twitter data:

```
hive> CREATE TABLE tweets (text STRING, words INT, length INT)
    > ROW FORMAT DELIMITED FIELDS TERMINATED BY '\\|'
    > STORED AS TEXTFILE;
```

5. Install Spark

```
~$ tar -xvf Downloads/spark-2.4.3-bin-hadoop2.7.tgz
~$ mv spark-2.4.3-bin-hadoop2.7.tgz spark
~$ sudo apt install scala -y
~$ scala -version
~$ pip3 install pyspark
```

```
~$ pip3 list | grep spark
```

Now we need to add the Spark /bin files to the path, so open up .bashrc and add the following

```
export PATH=$PATH:/home/<USER>/spark/bin
export PYSPARK_PYTHON=python3
```

By setting the PYSPARK_PYTHON variable, we can use PySpark with Python3, the version of Python we have been using so far.

After running source .bashrc, try entering the PySpark shell

```
~$ pyspark
```

```
...
Using Python version ....
SparkSession available as 'spark'.
>>>
```