Installation guideline for Hadoop

Prerequisites

- You have installed Linux already. This guideline is tested on Ubuntu 16.04.

1. Install Java(jdk8, linux-64bit)

For Ubuntu

```
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get upgrade

sudo apt-cache search java*
sudo apt-get install oracle-java8-installer (Accept agreements during installation)
```

2. Download Hadoop

Download Hadoop

```
wget http://archive.apache.org/dist/hadoop/core/hadoop-2.8.1/hadoop-2.8.1.tar.gz
tar xvfz hadoop-2.8.1.tar.gz
sudo mv  hadoop-2.8.1 /usr/local/
```

Change the owner of the Hadoop directory

```
sudo chown -R Your_user_name hadoop-2.8.1
```

3. Set environment variables

vi ~/.bashrc (Append below contents to the end of the bashrc file)

```
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
export CLASSPATH=$JAVA_HOME/lib:$JAVA_HOME/jre/lib/ext:$JAVA_HOME/lib/tools.jar
export HADOOP_HOME=/usr/local/hadoop-2.8.1
export HADOOP_CLASSPATH=$CLASSPATH
export HADOOP_PREFIX=$HADOOP_HOME
export PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/bin
```

source ~/.bashrc (To apply the changes without restarting the current shell)

4. Generate a key for automatic SSH login

To avoid submitting passwords whenever the system connects to each node using SSH, we need to make the system automatically connect each node with a generated SSH key.

For Ubuntu

```
sudo apt-get install openssh-server
ssh-keygen -t -P “” rsa -f ~/.ssh/id_rsa
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

5. Modify environment files (5 files)

5.1. vi /usr/local/hadoop-2.8.1/etc/hadoop/core-site.xml:

```
<configuration>
    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://localhost:9000</value>
    </property>
```

```
</configuration>
```

5.2. vi /usr/local/hadoop-2.8.1/etc/hadoop/hdfs-site.xml:

There will be only one copy of the file in the file system.

```
<configuration>
        <property>
                <name>dfs.replication</name>
                <value>1</value>
        </property>
</configuration>
```

5.3. vi /usr/local/hadoop-2.8.1/etc/hadoop/mapred-site.xml.template:

```
<configuration>
        <property>
                <name>mapreduce.framework.name</name>
                <value>yarn</value>
        </property>
</configuration>
```

5.4. vi /usr/local/hadoop-2.8.1/etc/hadoop/yarn-site.xml:

```
<configuration>
        <property>
                <name>yarn.nodemanager.aux-services</name>
                <value>mapreduce_shuffle</value>
        </property>
</configuration>
```

5.5. vi /usr/local/hadoop-2.8.1/etc/hadoop/hadoop-env.sh:

Change a line in the file: ‘export JAVA_HOME=${JAVA_HOME}’ to ‘export JAVA_HOME=/usr/lib/jvm/java-8-oracle’

6. Format Hadoop file system

sudo rm -rf /tmp/* : Remove contents in the temporary directory.
hdfs namenode -format

7. Run daemons

$HADOOP_HOME/sbin/start-dfs.sh : Run name node, secondary name node, and data node.
$HADOOP_HOME/sbin/start-yarn.sh : Run NodeManager and ResourceManager.

Two ways to check the daemons: 1) using web interface with below URLs in your web browers,

http://localhost:50070	: name node
http://localhost:50075	: data node
http://localhost:50090	: secondary name node
http://localhost:8088	: ResourceManager
http://localhost:8042	: NodeManager

2) Using jps command in your terminal.

The command ‘jps’ must show six lines like right result, but the numbers can be different.

```
1330 NameNode
11394 Jps
11299 NodeManager
1446 SecondaryNameNode
10856 ResourceManager
10718 DataNode
```

8. Make directories in Hadoop file system for MapReduce job

hdfs dfs -mkdir /input

Hadoop shell command*: hdfs dfs -command*

hdfs dfs -ls /	*: Shows the root directory*
hdfs dfs -ls /input	*: Shows /input directory*
hdfs dfs -mkdir /user	*: Makes a directory named as 'user' under the root*
hdfs dfs -rm -r output	*: Removes 'output' directory recursively*
hdfs dfs -put <arg1> <arg2>	*: Copies ${HADOOP_HOME}/arg1 file from local file system to HDFS as arg2 file.*
hdfs dfs -get <arg1> <arg2>	*: Copies arg1 file from HDFS to the local file system as arg2 file.*

9. Validate the installation using Word-count example

Put an input file into HDFS

cd $HADOOP_HOME
hdfs dfs -mkdir /input
hdfs dfs -put README.txt /input/README.txt

Run Word-count example

hadoop jar $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-2.8.1.jar wordcount /input/README.txt /output/wordcount

Check the result by moving the output into local file system

hdfs dfs -get output/wordcount/part-r-00000 output
cat output

10. Terminate daemons

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

Programming & creating jar files

- Basic compile

hadoop com.sun.tools.javac.Main <File name>
E.g. hadoop com.sun.tools.javac.Main WordCount.java

jar cf <output name> <Java classes>
E.g. jar cf wc.jar WordCount*.class

```
diadmin@di-ssd:~/Desktop/hadoop_example$ ls
WordCount.java
diadmin@di-ssd:~/Desktop/hadoop_example$ hadoop com.sun.tools.javac.Main WordCount.java
diadmin@di-ssd:~/Desktop/hadoop_example$ ls
WordCount.class  WordCount$IntSumReducer.class  WordCount.java  WordCount$TokenizerMapper.class
diadmin@di-ssd:~/Desktop/hadoop_example$ jar cf wc.jar WordCount*.class
diadmin@di-ssd:~/Desktop/hadoop_example$ ls
wc.jar           WordCount$IntSumReducer.class  WordCount$TokenizerMapper.class
WordCount.class  WordCount.java
diadmin@di-ssd:~/Desktop/hadoop_example$
```

Illustration 1: Compile result