Installing Hadoop on Ubuntu

Components used:
1. Hadoop 1.2.1
2. Ubuntu 12.04 or later (64-bit).

Step1: Install Ubuntu and log in to an admin account.
Step2: Download hadoop:
- a. Go to: https://archive.apache.org/dist/hadoop/core/hadoop-1.2.1/
- b. Download a stable release copy ending with tar.gz
- c. Create a folder: /home/hadoop
- d. Move the hadoop.x.y.z.tar.gz to the folder /home/hadoop
- e. Type (in terminal): cd/home/hadoop
- f. Type (in terminal): tar xzf hadoop*tar.gz

Step3: Setting up Java
- a. Type (in terminal): java -version
- b. If java it is 1.7.* or later- set up the JAVA_HOME Variable according to where it is setup or go to step (h).
- c. Purge the Java installed (open jdk) by typing (in terminal):
  sudo apt-get purge open jdk-*
- d. Make the directory where java would install. Type (in terminal):
  sudo mkdir -p /usr/local/java
- e. Download 64-bit Java JDK and JRE ending with tar.gz from:
  http://www.oracle.com/technetwork/java/javase/downloads/index.html
- f. Go to downloads folder and copy these JDK and JRE to the folder we created for Java.
- g. To extract and install java type (in terminal):

```
cd /usr/local/java
sudo tar xvzf jdk*.tar.gz
sudo tar xvzf jre*.tar.gz
```

- h. To put all the environment variables in the profile type (in terminal):
  sudo gedit /etc/profile
  At the end write the following. Replace asterisk accordingly:

```
JAVA_HOME=/usr/local/java/jdk1.*.0_*
PATH=$PATH:$JAVA_HOME/bin
JRE_HOME=/usr/local/java/jre1.*.0_*
PATH=$PATH:$JRE_HOME/bin
HADOOP_INSTALL=/home/****/hadoop/hadoop-1.2.1
PATH=$PATH:$HADOOP_INSTALL/bin
export JAVA_HOME
export JRE_HOME
export PATH
```

- i. To tell Ubuntu where Java is, type this in terminal and replace asterisk accordingy:

```
                    sudo update-alternatives --install "/usr/bin/java" "java"
"/usr/local/java/jre1.*.0_*/bin/java" 1
                    sudo update-alternatives --install "/usr/bin/javac" "javac"
"/usr/local/java/jdk1.*.0_*/bin/javac" 1
                    sudo update-alternatives --install "/usr/bin/javaws" "javaws"
"/usr/local/java/jre1.*.0_*/bin/javaws" 1
                    sudo update-alternatives --set java
/usr/local/java/jre1.*.0_*/bin/java
                    sudo update-alternatives --set javac
/usr/local/java/jdk1.*.0_*/bin/javac
                    sudo update-alternatives --set javaws
/usr/local/java/jre1.*.0_*/bin/javaws
```

j. Refresh the profile by typing in terminal:

```
. /etc/profile
```

h. Check if java is installed by typing in terminal:

```
java -version
```

Step 4. Hadoop is installed in Stand Alone mode. Test by typing in terminal:

```
cd $HADOOP_INSTALL        (go to the Hadoop directory)
mkdir input
bin/hadoop jar hadoop-examples-*.jar grep input output 'dfs[a-z.]+'
ls output/*
```

Step 5. Pseudo Distribution Mode for Hadoop. Type in terminal:

a. Install ssh using:

```
sudo apt-get install ssh
sudo apt-get install rsync
```

b. Edit conf/core-site.xml to:

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

c. Edit conf/hdfs-site.xml to:

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

d. Edit conf/mapred-site.xml to:

```
<configuration>
```

```
        <property>
            <name>mapred.job.tracker</name>
            <value>localhost:9001</value>
        </property>
    </configuration>
```

e. Edit conf/hadoop-env.sh by setting up as:

```
export JAVA_HOME=/usr/local/java/jdk1.*.0_*
```

f. Set up passwordless ssh by:

```
ssh-keygen -t dsa -P '' -f ~/.ssh/id_dsa
cat ~/.ssh/id_dsa.pub >> ~/.ssh/authorized_keys
```

g. Confirm that passwordless ssh has been set up by typing:

```
ssh localhost
```

You should not be prompted for a password.

h. Format the name node:

```
bin/hadoop namenode –format
```

i. Start all demons:

```
bin/start–all.sh
```

j. Go to the UI's of NameNode, JobTracker and TaskTracker daemons

http://localhost:50070/
http://localhost:50030/
http://localhost:50060/

k. To stop all the demons:

```
bin/stop-all.sh
```

Creating a Multi-Node Hadoop Cluster

To create a Multi-Node Hadoop Cluster from two Single-Node Clusters:

Step 1: Both machines should be able to connect to each other via network. Update /etc/hosts on both machines by typing:

```
$cd /etc
$sudo gedit hosts
```

In the /etc/hosts file, write the master and slave IP adresses. For example,

```
192.168.0.1        master
192.168.0.2        slave
```

Step 2: Distribute the SSH Public Key of admin@master by typing:

```
$ ssh-copy-id -i $HOME/ .ssh/id_rsa.pub admin@slave
```

Step 3: To connect from master to master

```
admin@master:~$ ssh master
```

To connect from master to slave

```
admin@master:~$ ssh slave
```

Step 4: Edit configuration file conf/core-site.xml as:

```
<property>
    <name>fs.default.name</name>
    <value>hdfs://master:54310</value>
```

```
        <description>The name of the default file system. A URI whose scheme
        and authority determine the FileSystem implementation. The uri's
        scheme determines the config property (fs.SCHEME.impl) naming the
        FileSystem implementation class. The uri's authority is used to determine
        the host, port, etc. for a filesystem.</description>
</property>
```

Step 5: Change mapred jobtracker parameter in file conf/mapred-site.xml as:

```
<property>
        <name>mapred.job.tracker</name>
        <value>master:54311</value>
        <description>The host and port that the MapReduce job tracker runs at. If
        "local", then jobs are run in-process as a single map and reduce task.
</description>
</property>
```

Step 6: Change dfs replication parameter in file conf/hdfs-site.xml as:

```
<property>
        <name>dfs.replication</name>
        <value>2</value>
        <description>Default block replication. The actual number of replications can
        be specified when the file is created. The default is used if replication is not
        specified in  create time. </description>
</property>
```

Step 7: Format the HDFS filesystem via the NameNode using:

```
        hadoop$ bin/hadoop namenode -format
```

Step 8: Starting the multi-node cluster. Run the following command on master:

```
        hadoop$ bin/start-dfs.sh
```

Examine the success by inspecting the log file: logs/hadoop-admin@master-datanode-slave.log

Step 9: Java Processes on master after starting HDFS daemons:

```
        hadoop$ jps
```

Java Processes on slave:

```
        hadoop$ jps
```

Step 10: To bring up the MapReduce cluster with the JobTracker on master:

```
        hadoop$ bin/start-mapred.sh
```

Step 11: To stop the MapReduce daemons on multi-node cluster, type on master:

```
        $ bin/stop-mapred.sh
```

Step 12: To stop the HDFS daemons type:

```
        $ bin/stop-dfs.sh
```

For further experience have a look at http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi-node-cluster/

Testing the Hadoop Cluster

To test how the Hadoop Cluster you created, go to the hadoop installation directory. Now, type in terminal to see full list of available options:

```
$ bin/hadoop jar hadoop-*test*.jar
```

Have a look at: http://www.michael-noll.com/blog/2011/04/09/benchmarking-and-stress-testing-an-hadoop-cluster-with-terasort-testdfsio-nnbench-mrbench/

Always run a write test before running a read test. The command to run a write test that generates 10 output files of size 1 GB for a total of 10 GB is:

```
$ hadoop jar hadoop-*test*.jar TESTDFSIO -write -nrFiles 10 -fileSize 1000
```

The command to run the corresponding read test is:

```
$ hadoop jar hadoop-*test*.jar TESTDFSIO -write -nrFiles 10 -fileSize 1000
```

The command to remove previous test data on Hadoop HDFS is (this will delete output directory /benchmarks/TestDFSIO):

```
$ hadoop jar hadoop-*test*.jar TESTDFSIO -clean
```