Run Hadoop on a single cluster (Ubuntu Linux)

Download and Install java-6-openjdk

$ sudo apt-get java-6-openjdk

it will be installed under

/usr/lib/jvm/java-6-openjdk

add a dedicated hadoop system user

$ sudo addgroup hadoop
$ sudo adduser --ingroup hadoop hduser
// add sudo permissions to hduser
$ sudo visudo
// under user privilege specification add in
hduser ALL=(ALL) ALL
configure ssh access to localhost for hduser
// change current user to hd user
$ su hduser
// generate a SSH key for hduser, creates an rsa key pair with an empty password
$ ssh-keygen -t rsa -P ""
// enable ssh access to local machine with this new key
$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
// test ssh setup
$ ssh localhost
// debug with
$ ssh -vvv localhost

Download Hadoop

download from apache mirror and extract contents to a location, example /home/hduser/hadoop
change owner of all hadoop files to hduser
$ sudo chown -R hduser:hadoop hadoop

update $HOME/.bashrc
$ nano $HOME/.bashrc
add the following lines to the end of the file
export HADOOP_HOME=/home/hduser/hadoop
export JAVA_HOME=/usr/lib/jvm/java-6-openjdk

Configuration
configure hadoop-env.sh
set JAVA_HOME variable by removing the # and pointint to the correct path
export JAVA_HOME=/usr/lib/jvm/java-6-openjdk

<!-- In: conf/core-site.xml -->
<property>
  <name>fs.default.name</name>

```
  <value>hdfs://localhost: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>
<property>
  <name>hadoop.tmp.dir</name>
  <value>/home/hduser/hadoop/tmp</value>
</property>



<!-- In: conf/mapred-site.xml -->
<property>
  <name>mapred.job.tracker</name>
  <value>localhost: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>

<!-- In: conf/hdfs-site.xml -->
<property>
  <name>dfs.replication</name>
  <value>1</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>
<property>
  <name>dfs.name.dir</name>
  <value>/home/hduser/hadoop/tmp/dfs/name</value>
</property>
<property>
  <name>dfs.data.dir</name>
  <value>/home/hduser/hadoop/tmp/dfs/data</value>
</property>

Format HDFS via namenode
$ ~/hadoop/bin/hadoop namenode -format

Starting a single-node cluster
$ ~/hadoop/bin/start-all.sh

use jps to check if the expected Hadoop processes run, this output should be seen
hduser@ubuntu:~/hadoop$ jps
2287 TaskTracker
```

2149 JobTracker
1938 DataNode
2085 SecondaryNameNode
2349 Jps
1788 NameNode

To stop the cluster run
$ ~/hadoop/bin/stop-all.sh

MapReduce test

cd ~/hadoop
Copy the input files into the distributed filesystem
(there will be no output visible from the command):
bin/hadoop fs -put conf input
Run some of the examples provided:
(there will be a large amount of INFO statements as output)
bin/hadoop jar hadoop-*-examples.jar grep input output 'dfs[a-z.]+'
Examine the output files:
bin/hadoop fs -cat output/part-00000

Hadoop web interfaces
http://localhost:50030/ – web UI for MapReduce job tracker(s)
http://localhost:50060/ – web UI for task tracker(s)
http://localhost:50070/ – web UI for HDFS name node(s)