Prerequisites

Sun Java 6

Hadoop requires a working Java 1.5.x (aka 5.0.x) installation.

Check to see if Java is running on your virtual box.

```
user@ubuntu:~# java -version
java version "1.6.0_20"
Java(TM) SE Runtime Environment (build 1.6.0_20-b02)
Java HotSpot(TM) Client VM (build 16.3-b01, mixed mode, sharing)
```

Adding a dedicated Hadoop system user (Optional)

We will use a dedicated Hadoop user account for running Hadoop. While that's not required it is recommended because it helps to separate the Hadoop installation from other software applications and user accounts running on the same machine (think: security, permissions, backups, etc).

```
$ sudo addgroup hadoop
$ sudo adduser --ingroup hadoop hduser
```

This will add the user `hduser` and the group `hadoop` to your local machine.
Working as ROOT user is also ok for single cluster systems

Configuring SSH

Hadoop requires SSH access to manage its nodes, i.e. remote machines plus your local machine if you want to use Hadoop on it (which is what we want to do in this short tutorial). For our single-node setup of Hadoop, we therefore need to configure SSH access to `localhost` for the root/hduser user we created in the previous section.

Check to see if SSH is already installed(Virtual Box already has SSH installed)

```
root@ubuntu:~$ ssh localhost
```

If not installed, use below instruction to install ssh:

```
$ sudo apt-get install ssh
```

```
root@ubuntu:~$ ssh localhost

The authenticity of host 'localhost (::1)' can't be established.

RSA key fingerprint is
d7:87:25:47:ae:02:00:eb:1d:75:4f:bb:44:f9:36:26.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added 'localhost' (RSA) to the list of known
hosts.

Linux ubuntu 2.6.32-22-generic #33-Ubuntu SMP Wed Apr 28 13:27:30 UTC
2010 i686 GNU/Linux

Ubuntu 10.04 LTS

[...snipp...]
```

If the SSH connect should fail, these general tips might help:

- Enable debugging with `ssh -vvv localhost` and investigate the error in detail.
- Check the SSH server configuration in `/etc/ssh/sshd_config`, in particular the options `PubkeyAuthentication` (which should be set to `yes`) and `AllowUsers` (if this option is active, add the `hduser` user to it). If you made any changes to the SSH server configuration file, you can force a configuration reload with `sudo /etc/init.d/ssh reload`.

Hadoop

Installation

You have to **download Hadoop** from the **Apache Download Mirrors** and extract the contents of the Hadoop package to a location of your choice. I picked `/usr/local/hadoop`. Make sure to change the owner of all the files to the `hduser` user and `hadoop` group, for example:

In below instructions, you do not need to use sudo if you are installing Hadoop as ROOT user.

```
$ cd /usr/local
$ sudo tar xzf hadoop-1.0.3.tar.gz
$ sudo mv hadoop-1.0.3 hadoop
$ sudo chown -R hduser:hadoop hadoop
```

Update $HOME/.bashrc

Add the following lines to the end of the `$HOME/.bashrc` file of user `hduser`. If you use a shell other than bash, you should of course update its appropriate configuration files instead of `.bashrc`.

```
# Set Hadoop-related environment variables
```

```
export HADOOP_HOME=/usr/local/hadoop


# Set JAVA_HOME (we will also configure JAVA_HOME directly for Hadoop
later on)

export JAVA_HOME=/usr/lib/jvm/java-6-sun


# Some convenient aliases and functions for running Hadoop-related
commands

unalias fs &> /dev/null

alias fs="hadoop fs"

unalias hls &> /dev/null

alias hls="fs -ls"




# Add Hadoop bin/ directory to PATH

export PATH=$PATH:$HADOOP_HOME/bin
```

You can repeat this exercise also for other users who want to use Hadoop.

hadoop-env.sh

The only required environment variable we have to configure for Hadoop in this tutorial is `JAVA_HOME`. Open `/conf/hadoop-env.sh` in the editor of your choice (if you used the installation path in this tutorial, the full path is `/usr/local/hadoop/conf/hadoop-env.sh`) and set the `JAVA_HOME` environment variable to the Sun JDK/JRE 6 directory.

Change

```
# The java implementation to use.  Required.
```

```
# export JAVA_HOME=/usr/lib/j2sdk1.5-sun
```

to

```
# The java implementation to use.  Required.
export JAVA_HOME=/usr/lib/jvm/java-6-sun
```

Note: If you are on a Mac with OS X 10.7 you can use the following line to set up `JAVA_HOME` in `conf/hadoop-env.sh`.

```
# for our Mac users
export JAVA_HOME=`/usr/libexec/java_home`
```

conf/*-site.xml

Note: *As of Hadoop 0.20.x and 1.x, the configuration settings previously found in `hadoop-site.xml` were moved to `core-site.xml` (hadoop.tmp.dir, fs.default.name), `mapred-site.xml` (mapred.job.tracker) and `hdfs-site.xml` (dfs.replication).*

In this section, we will configure the directory where Hadoop will store its data files, the network ports it listens to, etc. Our setup will use Hadoop's Distributed File System, **HDFS**, even though our little "cluster" only contains our single local machine.
You can leave the settings below "as is" with the exception of the `hadoop.tmp.dir` variable which you have to change to the directory of your choice. We will use the directory `/app/hadoop/tmp` in this tutorial. Hadoop's default configurations use `hadoop.tmp.dir` as the base temporary directory both for the local file system and HDFS, so don't be surprised if you see Hadoop creating the specified directory automatically on HDFS at some later point.
Now we create the directory and set the required ownerships and permissions:

```
$ sudo mkdir -p /app/hadoop/tmp
$ sudo chown hduser:hadoop /app/hadoop/tmp
```

```
# ...and if you want to tighten up security, chmod from 755 to 750...
$ sudo chmod 750 /app/hadoop/tmp
```

If you forget to set the required ownerships and permissions, you will see a `java.io.IOException` when you try to format the name node in the next section).

Add the following snippets between the `<configuration> ... </configuration>` tags in the respective configuration XML file.

In file `conf/core-site.xml`:

```
<!-- In: conf/core-site.xml -->
<property>
  <name>hadoop.tmp.dir</name>
  <value>/app/hadoop/tmp</value>
  <description>A base for other temporary directories.</description>
</property>

<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>
```

In file `conf/mapred-site.xml`:

```
<!-- 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 file `conf/hdfs-site.xml`:

```
<!-- 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>
```

Formatting the HDFS filesystem via the NameNode

The first step to starting up your Hadoop installation is formatting the Hadoop filesystem which is implemented on top of the local filesystem of your “cluster” (which includes only your local machine if you followed this tutorial). You need to do this the first time you set up a Hadoop cluster.

Do not format a running Hadoop filesystem as you will lose all the data currently in the cluster (in HDFS).

To format the filesystem (which simply initializes the directory specified by the `dfs.name.dir` variable), run the command

```
hduser@ubuntu:~$ /usr/local/hadoop/bin/hadoop namenode -format
```

The output will look like this:

```
hduser@ubuntu:/usr/local/hadoop$ bin/hadoop namenode -format

10/05/08 16:59:56 INFO namenode.NameNode: STARTUP_MSG:

/************************************************************

STARTUP_MSG: Starting NameNode

STARTUP_MSG:   host = ubuntu/127.0.1.1

STARTUP_MSG:   args = [-format]

STARTUP_MSG:   version = 0.20.2

STARTUP_MSG:   build =
https://svn.apache.org/repos/asf/hadoop/common/branches/branch-0.20 -r
911707; compiled by 'sample' on Fri Feb 19 08:07:34 UTC 2010

************************************************************/

10/05/08 16:59:56 INFO namenode.FSNamesystem: fsOwner=hduser,hadoop

10/05/08 16:59:56 INFO namenode.FSNamesystem: supergroup=supergroup

10/05/08 16:59:56 INFO namenode.FSNamesystem: isPermissionEnabled=true
```

```
10/05/08 16:59:56 INFO common.Storage: Image file of size 96 saved in
0 seconds.

10/05/08 16:59:57 INFO common.Storage: Storage directory .../hadoop-
hduser/dfs/name has been successfully formatted.

10/05/08 16:59:57 INFO namenode.NameNode: SHUTDOWN_MSG:

/************************************************************

SHUTDOWN_MSG: Shutting down NameNode at ubuntu/127.0.1.1

************************************************************/

hduser@ubuntu:/usr/local/hadoop$
```

Starting your single-node cluster

Run the command:

```
hduser@ubuntu:~$ /usr/local/hadoop/bin/start-all.sh
```

This will startup a Namenode, Datanode, Jobtracker and a Tasktracker on your machine.

The output will look like this:

```
hduser@ubuntu:/usr/local/hadoop$ bin/start-all.sh

starting namenode, logging to /usr/local/hadoop/bin/../logs/hadoop-
hduser-namenode-ubuntu.out

localhost: starting datanode, logging to
/usr/local/hadoop/bin/../logs/hadoop-hduser-datanode-ubuntu.out

localhost: starting secondarynamenode, logging to
/usr/local/hadoop/bin/../logs/hadoop-hduser-secondarynamenode-
ubuntu.out

starting jobtracker, logging to /usr/local/hadoop/bin/../logs/hadoop-
hduser-jobtracker-ubuntu.out
```

```
localhost: starting tasktracker, logging to
/usr/local/hadoop/bin/../logs/hadoop-hduser-tasktracker-ubuntu.out

hduser@ubuntu:/usr/local/hadoop$
```

A nifty tool for checking whether the expected Hadoop processes are running is `jps` (part of Sun's Java since v1.5.0).

```
hduser@ubuntu:/usr/local/hadoop$ jps

2287 TaskTracker

2149 JobTracker

1938 DataNode

2085 SecondaryNameNode

2349 Jps

1788 NameNode
```

You can also check with `netstat` if Hadoop is listening on the configured ports.

```
hduser@ubuntu:~$ sudo netstat -plten | grep java

tcp   0  0 0.0.0.0:50070   0.0.0.0:*  LISTEN  1001  9236  2471/java

tcp   0  0 0.0.0.0:50010   0.0.0.0:*  LISTEN  1001  9998  2628/java

tcp   0  0 0.0.0.0:48159   0.0.0.0:*  LISTEN  1001  8496  2628/java

tcp   0  0 0.0.0.0:53121   0.0.0.0:*  LISTEN  1001  9228  2857/java

tcp   0  0 127.0.0.1:54310 0.0.0.0:*  LISTEN  1001  8143  2471/java

tcp   0  0 127.0.0.1:54311 0.0.0.0:*  LISTEN  1001  9230  2857/java

tcp   0  0 0.0.0.0:59305   0.0.0.0:*  LISTEN  1001  8141  2471/java

tcp   0  0 0.0.0.0:50060   0.0.0.0:*  LISTEN  1001  9857  3005/java

tcp   0  0 0.0.0.0:49900   0.0.0.0:*  LISTEN  1001  9037  2785/java
```

```
tcp   0  0 0.0.0.0:50030   0.0.0.0:*  LISTEN  1001  9773  2857/java

hduser@ubuntu:~$
```

If there are any errors, examine the log files in the `/logs/` directory.

Stopping your single-node cluster

Run the command

```
hduser@ubuntu:~$ /usr/local/hadoop/bin/stop-all.sh
```

to stop all the daemons running on your machine.

Example output:

```
hduser@ubuntu:/usr/local/hadoop$ bin/stop-all.sh

stopping jobtracker

localhost: stopping tasktracker

stopping namenode

localhost: stopping datanode

localhost: stopping secondarynamenode

hduser@ubuntu:/usr/local/hadoop$
```