Installing Hadoop on Fedora Linux (Single-Node Cluster)

Table of Contents:

Prerequisites

1. **Java 6**

Hadoop requires a working Java 1.5.x installation but Java 1.6.x is recommended for hadoop installation. (For fedora 16 , Java 1.6 is pre-installed)

View java version with following command :

```
[dell1user@localhost ~]$ java –version
java version "1.6.0_22"
OpenJDK Runtime Environment (IcedTea6 1.10.4) (fedora61.1.10.4.fc16-
i386)
OpenJDK Server VM (build 20.0-b11, mixed mode)
```

2. **Adding a dedicated Hadoop system user and hadoop group**

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.

```
[dell1user@localhost ~]$ su

[root@localhost dellnode1]# groupadd hadoop

[root@localhost dellnode1]# useradd hduser -g hadoop -m

[root@localhost dellnode1]# passwd hduser
```

3. **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 `hduser` user we created in the previous section.

Following command generates RSA key pair with an empty password. Generally, using an empty password is not recommended, but in this case it is needed to unlock the key without your interaction (you don't want to enter the passphrase every time Hadoop interacts with its nodes).

```
[dell1user@localhost ~]$ su - hduser
[hduser@localhost ~]$ ssh-keygen -t rsa -P ""
ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hduser/.ssh/id_rsa):
Created directory '/home/hduser/.ssh'.
Your identification has been saved in /home/hduser/.ssh/id_rsa.
Your public key has been saved in /home/hduser/.ssh/id_rsa.pub.
The key fingerprint is:
b8:1d:f5:da:05:08:a9:bd:04:0b:14:a6:80:1b:60:cc
hduser@localhost.localdomain
The key's randomart image is:
+--[ RSA 2048]----+
|B. .+.  ..       |
|+E o. . .. .     |
| o.  . =  o .    |
|.     o.o. . .   |
|      ..S.  . .  |
|       o.. o .   |
|      . . . .    |
|                 |
|                 |
+-----------------+
[hduser@localhost ~]$
```

Enable SSH access to your local machine with this newly created key.

```
[hduser@localhost~]$ ssh-copy-id -i $HOME/.ssh/id_rsa.pub hduser@localhost

The authenticity of host 'localhost (::1)' can't be established.
RSA key fingerprint is 13:6d:6a:01:16:8d:93:ee:d2:fe:f6:e3:6f:74:31:81.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (RSA) to the list of known hosts.
hduser@localhost's password:

Now try logging into the machine, with "ssh 'hduser@localhost'", and check
in:

  ~/.ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.

[hduser@localhost ~]$
```

Test the SSH setup by connecting to your local machine with the `hduser` user.

```
[hduser@localhost ~]$ ssh localhost
[hduser@localhost ~]$
[hduser@localhost ~]$ exit
```

4. **Disabling IPV6**

   One problem with IPv6 on Fedora is that using `0.0.0.0` for the various networking-related Hadoop configuration options will result in Hadoop binding to the IPv6 addresses of Fedora

   To disable IPv6 , open `/etc/sysctl.conf` in the editor

```
[hduser@localhost local]$ su

[root@localhost local]# vi /etc/sysctl.conf
```

   Add the following lines to the end of the file:

```
#disable ipv6
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

   You have to reboot your machine in order to make the changes take effect.

   You can check whether IPv6 is enabled on your machine with the following command:

```
[hduser@localhost ~]$ cat /proc/sys/net/ipv6/conf/all/disable_ipv6
1
```

   A return value of 0 means IPv6 is enabled, a value of 1 means disabled.

Hadoop

1. Installation

You have to download Hadoop from the Apache Download Mirrors and extract the contents of the Hadoop package. Make sure to change the owner of all the files to the `hduser` user and `hadoop` group.

```
[hduser@localhost ~]$ cd /usr/local

[hduser@localhost local]$ su

[root@localhost local]# tar xzf hadoop-0.20.2.tar.gz

[root@localhost local]# mv hadoop-0.20.2 hadoop

[root@localhost local]# chown -R hduser:hadoop hadoop
```

2. Update $HOME/.bashrc

```
[root@localhost local]# su hduser

[hduser@localhost local]$ vi $HOME/.bashrc
```

Add the following lines to the end of the `$HOME/.bashrc` file of user `hduser`.

```
# 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

# 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"

# If you have LZO compression enabled in your Hadoop cluster and
# compress job outputs with LZOP (not covered in this tutorial):
# Conveniently inspect an LZOP compressed file from the command
# line; run via:
#
# $ lzohead /hdfs/path/to/lzop/compressed/file.lzo
#
# Requires installed 'lzop' command.
#
lzohead () {
    hadoop fs -cat $1 | lzop -dc | head -1000 | less
}

# Add Hadoop bin/ directory to PATH
export PATH=$PATH:$HADOOP_HOME/bin
```

```
[hduser@localhost local]$ source ~/.bashrc
```

3. Configuration

1. hadoop-env.sh

Open `conf/hadoop-env.sh` in the editor and set the `JAVA_HOME` environment variable using following :

```
[hduser@localhost local]$ cd $HADOOP_HOME

[hduser@localhost hadoop]$ vi conf/hadoop-env.sh
```

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
```

Now 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.

Now we create the directory and set the required ownerships and permissions:

```
[hduser@localhost hadoop]$ su
Password:
[root@localhost hadoop]# mkdir -p /app/hadoop/tmp
[root@localhost hadoop]# chown hduser:hadoop /app/hadoop/tmp
[root@localhost hadoop]# chmod 750 /app/hadoop/tmp
[root@localhost hadoop]#
```

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).

2. conf/core-site.xml

```
[root@localhost hadoop]# exit

exit

[hduser@localhost hadoop]$ vi conf/core-site.xml
```

Add the following snippets between the `<configuration> ... </configuration>` tags 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>
```

3. conf/ mapred-site.xml

```
[hduser@localhost hadoop]$ vi conf/mapred-site.xml
```

Add the following snippets between the `<configuration> ... </configuration>` tags 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>
```

4. conf/ hdfs-site.xml

```
 [hduser@localhost hadoop]$ vi conf/hdfs-site.xml
```

Add the following snippets between the `<configuration> ... </configuration>` tags 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>
```

4 Format HDFS filesystem via 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.

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

```
 [hduser@localhost hadoop]$ cd ~
[hduser@localhost ~]$ hadoop namenode -format
```

The output will look like this:

```
12/02/01 11:24:27 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = localhost.localdomain/127.0.0.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 0.20.203.0
STARTUP_MSG:   build =
http://svn.apache.org/repos/asf/hadoop/common/branches/branch-0.20-
security-203 -r 1099333; compiled by 'oom' on Wed May  4 07:57:50
PDT 2011
************************************************************/
12/02/01 11:24:28 INFO util.GSet: VM type       = 32-bit
12/02/01 11:24:28 INFO util.GSet: 2% max memory = 17.77875 MB
12/02/01 11:24:28 INFO util.GSet: capacity      = 2^22 = 4194304
entries
12/02/01 11:24:28 INFO util.GSet: recommended=4194304,
actual=4194304
12/02/01 11:24:28 INFO namenode.FSNamesystem: fsOwner=hduser
12/02/01 11:24:28 INFO namenode.FSNamesystem: supergroup=supergroup
12/02/01 11:24:28 INFO namenode.FSNamesystem:
isPermissionEnabled=true
12/02/01 11:24:28 INFO namenode.FSNamesystem:
dfs.block.invalidate.limit=100
12/02/01 11:24:28 INFO namenode.FSNamesystem:
isAccessTokenEnabled=false accessKeyUpdateInterval=0 min(s),
accessTokenLifetime=0 min(s)
12/02/01 11:24:28 INFO namenode.NameNode: Caching file names
occuring more than 10 times
12/02/01 11:24:28 INFO common.Storage: Image file of size 112 saved
in 0 seconds.
12/02/01 11:24:28 INFO common.Storage: Storage directory
/app/hadoop/tmp/dfs/name has been successfully formatted.
12/02/01 11:24:28 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at
localhost.localdomain/127.0.0.1
************************************************************/
[hduser@localhost ~]$
```

Running Hadoop

Starting single node cluster

Run the command

```
[hduser@localhost ~]$ start-all.sh
```

The output will look like :

```
[hduser@localhost ~]$ start-all.sh
starting namenode, logging to /usr/local/hadoop/bin/../logs/hadoop-
hduser-namenode-localhost.localdomain.out
localhost: starting datanode, logging to
/usr/local/hadoop/bin/../logs/hadoop-hduser-datanode-
localhost.localdomain.out
localhost: starting secondarynamenode, logging to
/usr/local/hadoop/bin/../logs/hadoop-hduser-secondarynamenode-
localhost.localdomain.out
starting jobtracker, logging to /usr/local/hadoop/bin/../logs/hadoop-
hduser-jobtracker-localhost.localdomain.out
localhost: starting tasktracker, logging to
/usr/local/hadoop/bin/../logs/hadoop-hduser-tasktracker-
localhost.localdomain.out
[hduser@localhost ~]$
```

A nifty tool for checking whether the expected Hadoop processes are running is `jps`

```
[hduser@localhost ~]$ jps
12259 DataNode
12375 SecondaryNameNode
12553 TaskTracker
12670 Jps
12454 JobTracker
12170 NameNode
[hduser@localhost ~]$
```

Stopping single node cluster

Run the command

```
[hduser@localhost ~]$ stop-all.sh
```

The output will look like :

```
[hduser@localhost ~]$ stop-all.sh
stopping jobtracker
localhost: stopping tasktracker
stopping namenode
localhost: stopping datanode
localhost: stopping secondarynamenode
[hduser@localhost ~]$
```

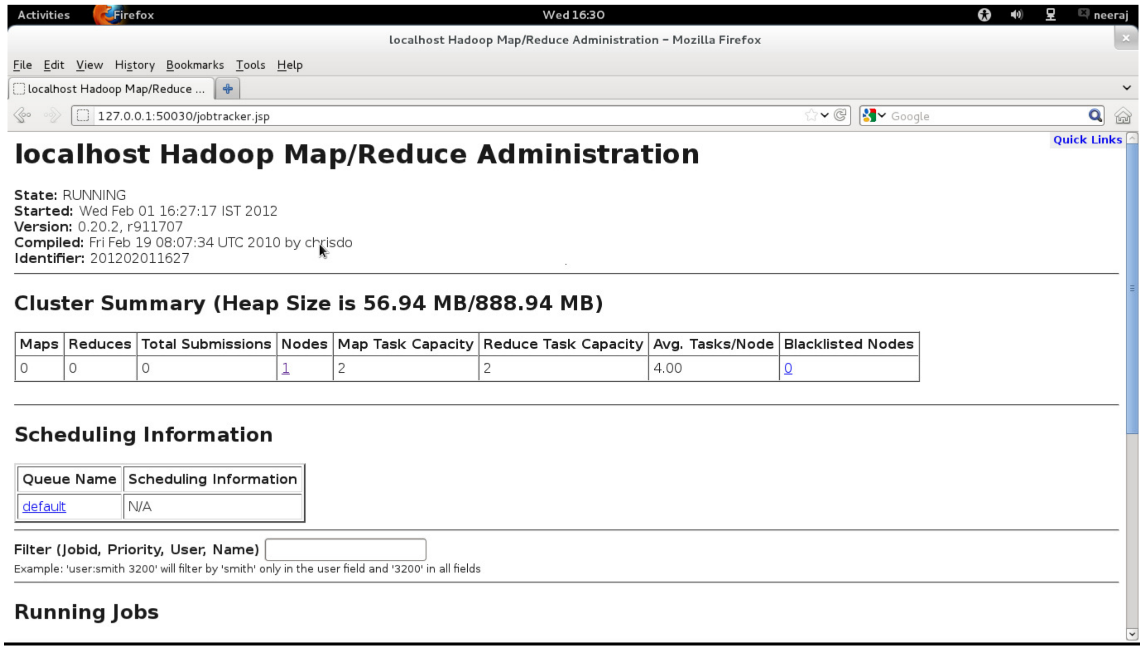
Hadooop Web Interfaces

Hadoop comes with several web interfaces which are by default (see `conf/hadoop-default.xml`) available at these locations:

- 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)

MapReduce Job Tracker Web Interface

The job tracker web UI provides information about general job statistics of the Hadoop cluster, running/completed/failed jobs and a job history log file. It also gives access to the ”local machine's” Hadoop log files (the machine on which the web UI is running on).


Activities
Firefox
Wed 16:30
neeraj
localhost Hadoop Map/Reduce Administration – Mozilla Firefox
File Edit View History Bookmarks Tools Help
localhost Hadoop Map/Reduce ...
127.0.0.1:50030/jobtracker.jsp
Google
Quick Links

localhost Hadoop Map/Reduce Administration

State: RUNNING
Started: Wed Feb 01 16:27:17 IST 2012
Version: 0.20.2, r911707
Compiled: Fri Feb 19 08:07:34 UTC 2010 by chrisdo
Identifier: 201202011627

Cluster Summary (Heap Size is 56.94 MB/888.94 MB)

Maps	Reduces	Total Submissions	Nodes	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes
0	0	0	1	2	2	4.00	0

Scheduling Information

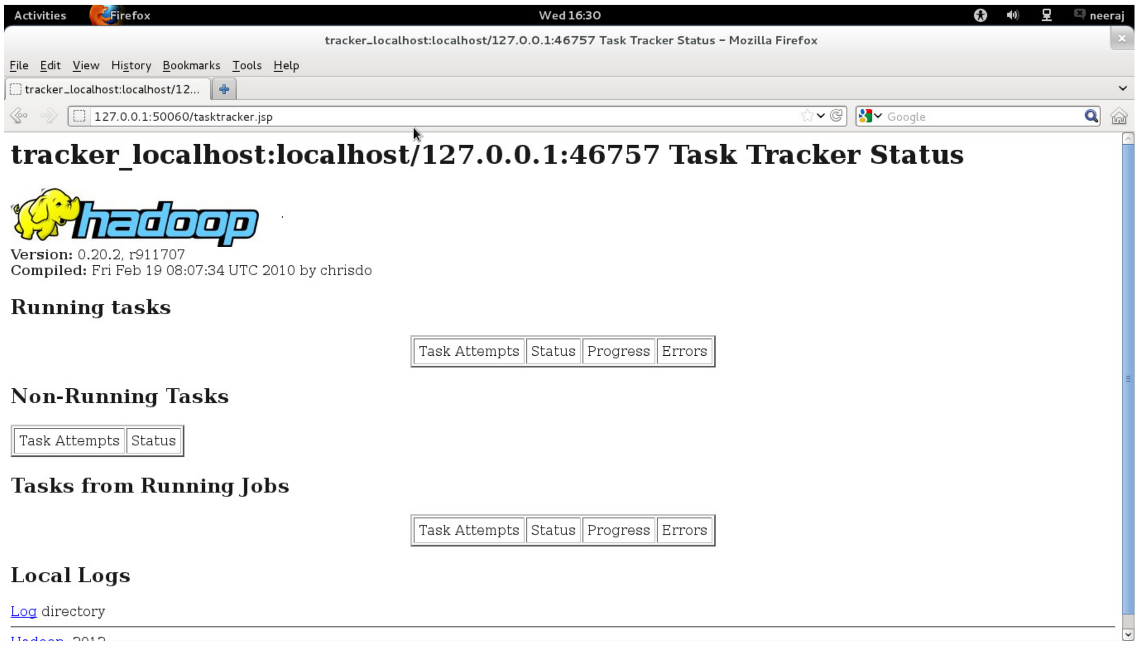
Queue Name	Scheduling Information
default	N/A

Filter (Jobid, Priority, User, Name)
Example: 'user:smith 3200' will filter by 'smith' only in the user field and '3200' in all fields

Running Jobs

Task Tracker Web Interface

The task tracker web UI shows you running and non-running tasks. It also gives access to the ”local machine's” Hadoop log files.


Activities
Firefox
Wed 16:30
neeraj
tracker_localhost:localhost/127.0.0.1:46757 Task Tracker Status - Mozilla Firefox
File Edit View History Bookmarks Tools Help
tracker_localhost:localhost/12...
127.0.0.1:50060/tasktracker.jsp
Google

tracker_localhost:localhost/127.0.0.1:46757 Task Tracker Status

Version: 0.20.2, r911707
Compiled: Fri Feb 19 08:07:34 UTC 2010 by chrisdo

Running tasks

Task Attempts	Status	Progress	Errors

Non-Running Tasks

Task Attempts	Status

Tasks from Running Jobs

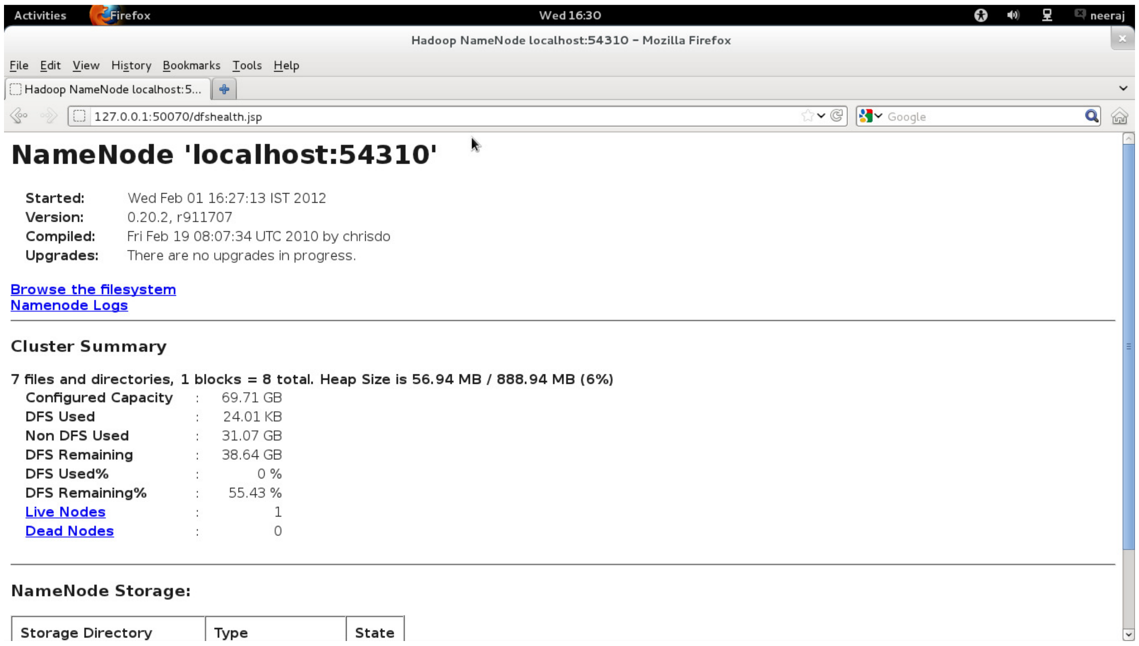
Task Attempts	Status	Progress	Errors

Local Logs

Log directory

HDFS Name Node Web Interface

The name node web UI shows you a cluster summary including information about total/remaining capacity, live and dead nodes. Additionally, it allows you to browse the HDFS namespace and view the contents of its files in the web browser. It also gives access to the ”local machine’s” Hadoop log files.


Activities
Firefox
Wed 16:30
neeraj
Hadoop NameNode localhost:54310 – Mozilla Firefox
File Edit View History Bookmarks Tools Help
Hadoop NameNode localhost:5...
127.0.0.1:50070/dfshealth.jsp
Google

NameNode 'localhost:54310'

Started:	Wed Feb 01 16:27:13 IST 2012
Version:	0.20.2, r911707
Compiled:	Fri Feb 19 08:07:34 UTC 2010 by chrisdo
Upgrades:	There are no upgrades in progress.

Browse the filesystem
Namenode Logs

Cluster Summary

7 files and directories, 1 blocks = 8 total. Heap Size is 56.94 MB / 888.94 MB (6%)

Configured Capacity	:	69.71 GB
DFS Used	:	24.01 KB
Non DFS Used	:	31.07 GB
DFS Remaining	:	38.64 GB
DFS Used%	:	0 %
DFS Remaining%	:	55.43 %
Live Nodes	:	1
Dead Nodes	:	0

NameNode Storage:

Storage Directory	Type	State