Michael G. Noll

Applied Research. Big Data. Distributed Systems. Open Source.

Running Hadoop on Ubuntu Linux (Single-Node Cluster)

Table of Contents

In this tutorial I will describe the required steps for setting up a *pseudo-distributed, single-node* Hadoop cluster backed by the Hadoop Distributed File System, running on Ubuntu Linux.

Are you looking for the multi-node cluster tutorial? Just head over there.

Hadoop is a framework written in Java for running applications on large clusters of commodity hardware and incorporates features similar to those of the Google File System (GFS) and of the MapReduce computing paradigm. Hadoop's HDFS is a highly fault-tolerant distributed file system and, like Hadoop in general, designed to be deployed on low-cost hardware. It provides high throughput access to application data and is suitable for applications that have large data sets.

The main goal of this tutorial is to get a simple Hadoop installation up and running so that you can play around with the software and learn more about it.

This tutorial has been tested with the following software versions:

- Ubuntu Linux 10.04 LTS (deprecated: 8.10 LTS, 8.04, 7.10, 7.04)
- Hadoop 1.0.3, released May 2012

Figure 1: Cluster of machines running Hadoop at Yahoo! (Source: Yahoo!)

Prerequisites

Sun Java 6

Hadoop requires a working Java 1.5+ (aka Java 5) installation. However, using Java 1.6 (aka Java 6) is recommended for running Hadoop. For the sake of this tutorial, I will therefore describe the installation of Java 1.6.

Important Note: The apt instructions below are taken from this SuperUser.com thread. I got notified that the previous instructions that I provided no longer work. Please be aware that adding a third-party repository to your Ubuntu configuration is considered a security risk. If you do not want to proceed with the apt instructions below, feel free to install Sun JDK 6 via alternative means (e.g. by downloading the binary package from Oracle) and then continue with the next section in the tutorial.

```
# Add the Ferramosca Roberto's repository to your apt repositories
# See https://launchpad.net/~ferramroberto/
#
$ sudo apt-get install python-software-properties
$ sudo add-apt-repository ppa:ferramroberto/java

# Update the source list
$ sudo apt-get update

# Install Sun Java 6 JDK
$ sudo apt-get install sun-java6-jdk

# Select Sun's Java as the default on your machine.
# See 'sudo update-alternatives --config java' for more information.
#
$ sudo update-java-alternatives -s java-6-sun
```

The full JDK which will be placed in `/usr/lib/jvm/java-6-sun` (well, this directory is actually a symlink on Ubuntu).

After installation, make a quick check whether Sun's JDK is correctly set up:

```
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

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.

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.

I assume that you have SSH up and running on your machine and configured it to allow SSH public key authentication. If not, there are several online guides available.

First, we have to generate an SSH key for the `hduser` user.

```
user@ubuntu:~$ su - hduser
hduser@ubuntu:~$ ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/hduser/.ssh/id_rsa):
```

```
5 Created directory '/home/hduser/.ssh'.
6 Your identification has been saved in /home/hduser/.ssh/id_rsa.
7 Your public key has been saved in /home/hduser/.ssh/id_rsa.pub.
8 The key fingerprint is:
9 9b:82:ea:58:b4:e0:35:d7:ff:19:66:a6:ef:ae:0e:d2 hduser@ubuntu
10 The key's randomart image is:
11 [...snipp...]
12 hduser@ubuntu:~$
```

The second line will create an 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).

Second, you have to enable SSH access to your local machine with this newly created key.

```
1 hduser@ubuntu:~$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

The final step is to test the SSH setup by connecting to your local machine with the `hduser` user. The step is also needed to save your local machine's host key fingerprint to the `hduser` user's `known_hosts` file. If you have any special SSH configuration for your local machine like a non-standard SSH port, you can define host-specific SSH options in `$HOME/.ssh/config` (see `man ssh_config` for more information).

```
1 hduser@ubuntu:~$ ssh localhost
2 The authenticity of host 'localhost (::1)' can't be established.
3 RSA key fingerprint is d7:87:25:47:ae:02:00:eb:1d:75:4f:bb:44:f9:36:26.
4 Are you sure you want to continue connecting (yes/no)? yes
5 Warning: Permanently added 'localhost' (RSA) to the list of known hosts.
6 Linux ubuntu 2.6.32-22-generic #33-Ubuntu SMP Wed Apr 28 13:27:30 UTC 2010 i686 GNU/Linux
7 Ubuntu 10.04 LTS
8 [...snipp...]
9 hduser@ubuntu:~$
```

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`.

Disabling IPv6

One problem with IPv6 on Ubuntu 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 my Ubuntu box. In my case, I realized that there's no practical point in enabling IPv6 on a box when you are not connected to any IPv6 network. Hence, I simply disabled IPv6 on my Ubuntu machine. Your mileage may vary.

To disable IPv6 on Ubuntu 10.04 LTS, open `/etc/sysctl.conf` in the editor of your choice and add the following lines to the end of the file:

/etc/sysctl.conf

```
1 # disable ipv6
2 net.ipv6.conf.all.disable_ipv6 = 1
3 net.ipv6.conf.default.disable_ipv6 = 1
4 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:

```
1 $ cat /proc/sys/net/ipv6/conf/all/disable_ipv6
```

A return value of 0 means IPv6 is enabled, a value of 1 means disabled (that's what we want).

Alternative

You can also disable IPv6 only for Hadoop as documented in HADOOP-3437. You can do so by adding the following line to `conf/hadoop-env.sh`:

conf/hadoop-env.sh

```
1 export HADOOP_OPTS=-Djava.net.preferIPv4Stack=true
```

Hadoop

Installation

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:

```
1 $ cd /usr/local
2 $ sudo tar xzf hadoop-1.0.3.tar.gz
3 $ sudo mv hadoop-1.0.3 hadoop
4 $ sudo chown -R hduser:hadoop hadoop
```

(Just to give you the idea, YMMV – personally, I create a symlink from `hadoop-1.0.3` to `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`.

$HOME/.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"

# 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
```

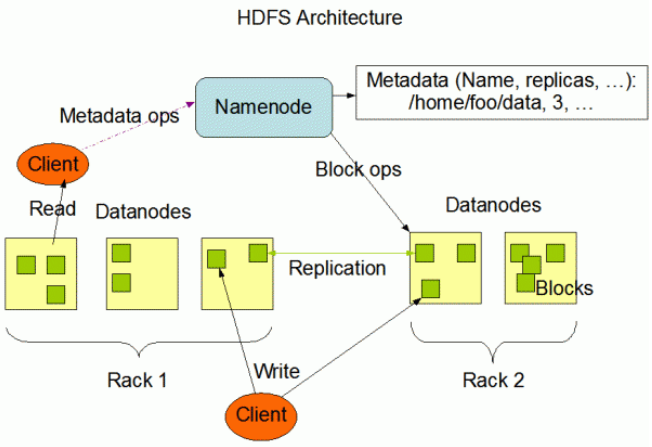
You can repeat this exercise also for other users who want to use Hadoop.

Excursus: Hadoop Distributed File System (HDFS)

Before we continue let us briefly learn a bit more about Hadoop's distributed file system.

> The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware. It has many similarities with existing distributed file systems. However, the differences from other distributed file systems are significant. HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets. HDFS relaxes a few POSIX requirements to enable streaming access to file system data. HDFS was originally built as infrastructure for the Apache Nutch web search engine project. HDFS is part of the Apache Hadoop project, which is part of the Apache Lucene project.
>
> **The Hadoop Distributed File System: Architecture and Design** *hadoop.apache.org/hdfs/docs/...*

The following picture gives an overview of the most important HDFS components.



Configuration

Our goal in this tutorial is a single-node setup of Hadoop. More information of what we do in this section is available on the Hadoop Wiki.

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