Installing HDFS - Hadoop

Prerequisites

Oracle's Java 8

HDFS and Hadoop require a working Java 1.5+ installation. For the sake of this tutorial, I will therefore describe the installation of Java 1.8.

```
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java8-installer
```

Then, if we want to make Oracle JAVA as default:

```
sudo apt-get install oracle-java8-set-default
```

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

```
user@ubuntu:~# java -version
java version "1.8.0_91"
Java(TM) SE Runtime Environment (build 1.8.0_91-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.91-b14, mixed mode)
```

Adding a dedicated User for HDFS

Now we create the UNIX group "hadoop", and we add our user to that group. This will create the group and add the user [userName] to the group hadoop to your local machine.

```
sudo addgroup hadoop
sudo usermod -a -G hadoop userName
```

Configuring SSH

Hadoop and HDFS require SSH access to manage its nodes. For our single-node setup of Hadoop, we therefore need to configure SSH access to localhost for our [userName] user.

To install OpenSSH, in case you don't have it already, follow the links https://help.ubuntu.com/lts/serverguide/openssh-server.html (for Ubuntu and similars) or https://bluishcoder.co.nz/articles/mac-ssh.html (MacOS and similars).

Now, we need to be able to open SSH connections without password, using RSA keys.

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

```
userName@ubuntu:~$ ssh-keygen -t rsa -P ""
```

Follow the instructions and you should have the files ~/.ssh/id_rsa and ~/.ssh/id_rsa.pub created. 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.

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

This will add your RSA public key to your list of allowed keys, so you can SSH your localhost without requesting password. Let's check if it worked.

```
ssh localhost
```

Testing the connection is also needed to automatically save your machine fingerprint into your user's known_hosts file for first time (no longer will be required).

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. If your machine is not using IPv6, better to disable.

```
Edit /etc/sysctl.conf and add:

# 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:

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

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

If you are using IPv6, or do not want to disable, you can modify the configuration of Hadoop after installing it, by adding the following line to conf/hadoop-env.sh:

```
Edit $HADOOP_HOME/etc/hadoop/hadoop-env.sh and add:

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

Installing Hadoop

Installation

Download Hadoop from the Apache Download Mirrors (http://bit.ly/2EbE3GX)and extract the contents of the Hadoop package to a location of your choice (e.g. /opt/hadoop). Make sure to change the owner of all the files to the [userName] user and hadoop group, for example:

```
cd /opt/
sudo wget http://bit.ly/2EbE3GX
sudo tar xvzf hadoop-2.7.4.tar.gz
sudo ln -s hadoop-2.7.4 hadoop
sudo chown –R userName:hadoop hadoop
```

Update your .bashrc

Add the following lines to the end of ~/.bashrc file of user [userName], or the RC file of your current shell.

```
# Set Hadoop-related environment variables
export HADOOP_HOME=/opt/hadoop

# Set JAVA_HOME (we will also configure JAVA_HOME directly for Hadoop later on)
export JAVA_HOME=/usr/lib/jvm/java-8-oracle

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

Configuration

This configuration targets to set-up a single Hadoop node. Check out the Hadoop Wiki for more information about how to configure a multi-node cluster.

Environment

hadoop-env.sh

The only required environment variable we have to configure for Hadoop in this tutorial is JAVA_HOME. Open etc/hadoop/hadoop-env.sh in the editor of your choice and set the JAVA_HOME environment variable to the Oracle’s JDK/JRE 8 directory.

```
# The java implementation to use.  Required.
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
```

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

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

Configuration Files

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 "cluster" only contains our single local machine.

You can leave the settings below "as is" with the exception of the hadoop.tmp.dir parameter – this parameter you must change to a 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.

HDFS Directory

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
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:

etc/hadoop/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.</description>
</property>
```

etc/hadoop /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.</description>
</property>
```

etc/hadoop /hdfs-site.xml

```
<property>
  <name>dfs.replication</name>
  <value>1</value>
  <description>Default block replication.</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 fs.name.dir variable), run the command

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

Starting HDFS

Once the system is formatted, you can start HDFS using the following command:

```
sudo $HADOOP_HOME/sbin/start-dfs.sh
```

This will start a Namenode, Datanode, Jobtracker and a Tasktracker on your machine. If no error messages appeared, you should see your new HDFS system by listing it using "ls". You can test creating directories and putting files using the "hdfs dfs" command, found in $HADOOP_HOME/bin (now added to your $PATH if followed the previous steps).

```
hdfs dfs -ls /
hdfs dfs -mkdir test_dir
hdfs dfs -put example_file.txt /test_dir/
```

Remember that HDFS has also user permissions, and you can use the chmod command in "hdfs dfs" to change HDFS permissions for files and directories.

Finally, we need to check which ports is HDFS using, for our future use in Apache Spark among others. By running netstat we can see which ports is JAVA using, and we will see the one indicated at Hadoop's "fs.default.name" configuration.

```
userName@ubuntu:~$ sudo netstat -plten | grep java
tcp   0   0   127.0.0.1:54310   0.0.0.0:*   LISTEN   1000   6090571   22603/java
…
```

References:

- Michael G. Noll http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/
- Web Upd8: Ubuntu http://www.webupd8.org/2012/09/install-oracle-java-8-in-ubuntu-via-ppa.html