Installation of Hadoop 2.6.3 on Ubuntu

Installing Java

Hadoop framework is written in Java!!

```
# Update the source list
rashmi@laptop:~$ sudo apt-get update

# The OpenJDK project is the default version of Java
# that is provided from a supported Ubuntu repository.

rashmi@laptop:~$ sudo apt-get install openjdk-7-jdk

rashmi@laptop:~$ java -version
java version "1.7.0_91"
OpenJDK Runtime Environment (IcedTea 2.5.3) (7u71-2.5.3-0ubuntu0.14.04.1)
OpenJDK 64-Bit Server VM (build 24.65-b04, mixed mode)
```

Installing SSH

ssh has two main components:

1. **ssh** : The command we use to connect to remote machines - the client.
2. **sshd** : The daemon that is running on the server and allows clients to connect to the server.

The **ssh** is pre-enabled on Linux, but in order to start **sshd** daemon, we need to install **ssh** first. Use this command to do that :

```
rashmi@laptop:~$ sudo apt-get install ssh
```

This will install ssh on our machine. If we get something similar to the following, we can think it is setup properly:

```
rashmi@laptop:~$ which ssh
/usr/bin/ssh

rashmi@laptop:~$ which sshd
/usr/sbin/sshd
```

Create and Setup SSH Certificates

Hadoop requires SSH access to manage its nodes, i.e. remote machines plus our local machine. For our single-node setup of Hadoop, we therefore need to configure SSH access to localhost.

So, we need to have SSH up and running on our machine and configured it to allow SSH public key authentication.

Hadoop uses SSH (to access its nodes) which would normally require the user to enter a password. However, this requirement can be eliminated by creating and setting up SSH certificates using the following commands. If asked for a filename just leave it blank and press the enter key to continue.

```
rashmi@laptop:~$ ssh-keygen -t rsa -P ""
Generating public/private rsa key pair.
Enter file in which to save the key (/home/rashmi/.ssh/id_rsa):
Created directory '/home/rashmi/.ssh'.
Your identification has been saved in /home/rashmi/.ssh/id_rsa.
Your public key has been saved in /home/rashmi/.ssh/id_rsa.pub.
The key fingerprint is:
50:6b:f3:fc:0f:32:bf:30:79:c2:41:71:26:cc:7d:e3 rashmi@laptop
The key's randomart image is:
+--[ RSA 2048]----+
|        .oo.o    |
|       . .o=. o  |
|      . + .  o . |
|       o =    E  |
|        S +      |
|         . +     |
|          O +    |
|           O o   |
|            o..  |
+-----------------+

rashmi@laptop:~$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

The second command adds the newly created key to the list of authorized keys so that Hadoop can use ssh without prompting for a password.

We can check if ssh works:

```
rashmi@laptop:~$ ssh localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
ECDSA key fingerprint is e1:8b:a0:a5:75:ef:f4:b4:5e:a9:ed:be:64:be:5c:2f.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-40-generic x86_64)
...
```

Install Hadoop

```
rashmi@laptop:~$ wget http://mirrors.sonic.net/apache/hadoop/common/hadoop-2.6.3/hadoop-2.6.3.tar.gz

rashmi@laptop:~$ tar xvzf hadoop-2.6.3.tar.gz
```

We want to move the Hadoop installation to the **/usr/local/hadoop** directory using the following command:

```
rashmi@laptop:~$ sudo mv hadoop/ /usr/local/

rashmi@laptop:~$ sudo chown -R rashmi:rashmi /usr/local/hadoop
```

Setup Configuration Files

The following files will have to be modified to complete the Hadoop setup:

1. ~/.bashrc
2. /usr/local/hadoop/etc/hadoop/hadoop-env.sh
3. /usr/local/hadoop/etc/hadoop/core-site.xml
4. /usr/local/hadoop/etc/hadoop/mapred-site.xml.template
5. /usr/local/hadoop/etc/hadoop/hdfs-site.xml

1. ~/.bashrc:

Before editing the **.bashrc** file in our home directory, we need to find the path where Java has been installed to set the **JAVA_HOME** environment variable using the following command:

```
rashmi@laptop:~$ update-alternatives --config java
There is only one alternative in link group java (providing /usr/bin/java):
/usr/lib/jvm/java-7-openjdk-i386/jre/bin/java
Nothing to configure.
```

Now we can append the following to the end of **~/.bashrc**:

```
rashmi@laptop:~$ gedit .bashrc
```

```
#HADOOP VARIABLES START
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-i386
export HADOOP_INSTALL=/usr/local/hadoop
export PATH=$PATH:$HADOOP_INSTALL/bin
export PATH=$PATH:$HADOOP_INSTALL/sbin
export HADOOP_MAPRED_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_HOME=$HADOOP_INSTALL
export HADOOP_HDFS_HOME=$HADOOP_INSTALL
export YARN_HOME=$HADOOP_INSTALL
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_INSTALL/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_INSTALL/lib"
#HADOOP VARIABLES END
```

```
rashmi@laptop:~$ source .bashrc
```

This command applies the changes made in the .bashrc file.

2. /usr/local/hadoop/etc/hadoop/hadoop-env.sh

We need to set **JAVA_HOME** by modifying **hadoop-env.sh** file.

```
rashmi@laptop:~$ gedit /usr/local/hadoop/etc/hadoop/hadoop-env.sh
```

```
export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-i386
```

Adding the above statement in the **hadoop-env.sh** file ensures that the value of JAVA_HOME variable will be available to Hadoop whenever it is started up.

3. /usr/local/hadoop/etc/hadoop/core-site.xml:

The **/usr/local/hadoop/etc/hadoop/core-site.xml** file contains configuration properties that Hadoop uses when starting up. This file can be used to override the default settings that Hadoop starts with.

```
rashmi@laptop:~$ sudo mkdir -p /app/hadoop/tmp
rashmi@laptop:~$ sudo chown rashmi:rashmi /app/hadoop/tmp
```

Open the file and enter the following in between the <configuration></configuration> tag:

```
rashmi@laptop:~$ gedit /usr/local/hadoop/etc/hadoop/core-site.xml
```

```
<configuration>
 <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>
</configuration>
```

4. /usr/local/hadoop/etc/hadoop/mapred-site.xml

By default, the **/usr/local/hadoop/etc/hadoop/** folder contains
/usr/local/hadoop/etc/hadoop/mapred-site.xml.template
file which has to be renamed/copied with the name **mapred-site.xml**:

The **mapred-site.xml** file is used to specify which framework is being used for MapReduce.
We need to enter the following content in between the <configuration></configuration> tag:

```
<configuration>
 <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>
</configuration>
```

5. /usr/local/hadoop/etc/hadoop/hdfs-site.xml

The **/usr/local/hadoop/etc/hadoop/hdfs-site.xml** file needs to be configured for each host in the cluster that is being used. It is used to specify the directories which will be used as the **namenode**

and the **datanode** on that host.

Before editing this file, we need to create two directories which will contain the namenode and the datanode for this Hadoop installation. This can be done using the following commands:

```
rashmi@laptop:~$ sudo mkdir -p /usr/local/hadoop_store/hdfs/namenode
rashmi@laptop:~$ sudo mkdir -p /usr/local/hadoop_store/hdfs/datanode
rashmi@laptop:~$ sudo chown -R rashmi:rashmi /usr/local/hadoop_store
```

Open the file and enter the following content in between the <configuration></configuration> tag:

```
rashmi@laptop:~$ gedit /usr/local/hadoop/etc/hadoop/hdfs-site.xml
```

```
<configuration>
 <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.namenode.name.dir</name>
   <value>file:/usr/local/hadoop_store/hdfs/namenode</value>
 </property>
 <property>
   <name>dfs.datanode.data.dir</name>
   <value>file:/usr/local/hadoop_store/hdfs/datanode</value>
 </property>
</configuration>
```

Format the New Hadoop Filesystem

Now, the Hadoop file system needs to be formatted so that we can start to use it. The format command should be issued with write permission since it creates **current** directory under **/usr/local/hadoop_store/hdfs/namenode** folder:

```
rashmi@laptop:~$ hadoop namenode -format
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

15/04/18 14:43:03 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = laptop/192.168.1.1
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.6.0
STARTUP_MSG:   classpath = /usr/local/hadoop/etc/hadoop
...
STARTUP_MSG:   java = 1.7.0_91
************************************************************/
15/04/18 14:43:03 INFO namenode.NameNode: registered UNIX signal handlers for
[TERM, HUP, INT]
15/04/18 14:43:03 INFO namenode.NameNode: createNameNode [-format]
15/04/18 14:43:07 WARN util.NativeCodeLoader: Unable to load native-hadoop
```

```
library for your platform... using builtin-java classes where applicable
Formatting using clusterid: CID-e2f515ac-33da-45bc-8466-5b1100a2bf7f
15/04/18 14:43:09 INFO namenode.FSNamesystem: No KeyProvider found.
15/04/18 14:43:09 INFO namenode.FSNamesystem: fsLock is fair:true
15/04/18 14:43:10 INFO blockmanagement.DatanodeManager:
dfs.block.invalidate.limit=1000
15/04/18 14:43:10 INFO blockmanagement.DatanodeManager:
dfs.namenode.datanode.registration.ip-hostname-check=true
15/04/18 14:43:10 INFO blockmanagement.BlockManager:
dfs.namenode.startup.delay.block.deletion.sec is set to 000:00:00:00.000
15/04/18 14:43:10 INFO blockmanagement.BlockManager: The block deletion will
start around 2015 Apr 18 14:43:10
15/04/18 14:43:10 INFO util.GSet: Computing capacity for map BlocksMap
15/04/18 14:43:10 INFO util.GSet: VM type       = 64-bit
15/04/18 14:43:10 INFO util.GSet: 2.0% max memory 889 MB = 17.8 MB
15/04/18 14:43:10 INFO util.GSet: capacity      = 2^21 = 2097152 entries
15/04/18 14:43:10 INFO blockmanagement.BlockManager:
dfs.block.access.token.enable=false
15/04/18 14:43:10 INFO blockmanagement.BlockManager: defaultReplication
= 1
15/04/18 14:43:10 INFO blockmanagement.BlockManager: maxReplication
= 512
15/04/18 14:43:10 INFO blockmanagement.BlockManager: minReplication
= 1
15/04/18 14:43:10 INFO blockmanagement.BlockManager: maxReplicationStreams
= 2
15/04/18 14:43:10 INFO blockmanagement.BlockManager: shouldCheckForEnoughRacks
= false
15/04/18 14:43:10 INFO blockmanagement.BlockManager: replicationRecheckInterval
= 3000
15/04/18 14:43:10 INFO blockmanagement.BlockManager: encryptDataTransfer
= false
15/04/18 14:43:10 INFO blockmanagement.BlockManager: maxNumBlocksToLog
= 1000
15/04/18 14:43:10 INFO namenode.FSNamesystem: fsOwner             = rashmi
(auth:SIMPLE)
15/04/18 14:43:10 INFO namenode.FSNamesystem: supergroup          = supergroup
15/04/18 14:43:10 INFO namenode.FSNamesystem: isPermissionEnabled = true
15/04/18 14:43:10 INFO namenode.FSNamesystem: HA Enabled: false
15/04/18 14:43:10 INFO namenode.FSNamesystem: Append Enabled: true
15/04/18 14:43:11 INFO util.GSet: Computing capacity for map INodeMap
15/04/18 14:43:11 INFO util.GSet: VM type       = 64-bit
15/04/18 14:43:11 INFO util.GSet: 1.0% max memory 889 MB = 8.9 MB
15/04/18 14:43:11 INFO util.GSet: capacity      = 2^20 = 1048576 entries
15/04/18 14:43:11 INFO namenode.NameNode: Caching file names occuring more than
10 times
15/04/18 14:43:11 INFO util.GSet: Computing capacity for map cachedBlocks
15/04/18 14:43:11 INFO util.GSet: VM type       = 64-bit
15/04/18 14:43:11 INFO util.GSet: 0.25% max memory 889 MB = 2.2 MB
15/04/18 14:43:11 INFO util.GSet: capacity      = 2^18 = 262144 entries
15/04/18 14:43:11 INFO namenode.FSNamesystem: dfs.namenode.safemode.threshold-
pct = 0.9990000128746033
15/04/18 14:43:11 INFO namenode.FSNamesystem:
dfs.namenode.safemode.min.datanodes = 0
15/04/18 14:43:11 INFO namenode.FSNamesystem: dfs.namenode.safemode.extension
= 30000
15/04/18 14:43:11 INFO namenode.FSNamesystem: Retry cache on namenode is enabled
15/04/18 14:43:11 INFO namenode.FSNamesystem: Retry cache will use 0.03 of total
heap and retry cache entry expiry time is 600000 millis
```

```
15/04/18 14:43:11 INFO util.GSet: Computing capacity for map NameNodeRetryCache
15/04/18 14:43:11 INFO util.GSet: VM type       = 64-bit
15/04/18 14:43:11 INFO util.GSet: 0.029999999329447746% max memory 889 MB =
273.1 KB
15/04/18 14:43:11 INFO util.GSet: capacity      = 2^15 = 32768 entries
15/04/18 14:43:11 INFO namenode.NNConf: ACLs enabled? false
15/04/18 14:43:11 INFO namenode.NNConf: XAttrs enabled? true
15/04/18 14:43:11 INFO namenode.NNConf: Maximum size of an xattr: 16384
15/04/18 14:43:12 INFO namenode.FSImage: Allocated new BlockPoolId: BP-
130729900-192.168.1.1-1429393391595
15/04/18 14:43:12 INFO common.Storage: Storage directory
/usr/local/hadoop_store/hdfs/namenode has been successfully formatted.
15/04/18 14:43:12 INFO namenode.NNStorageRetentionManager: Going to retain 1
images with txid >= 0
15/04/18 14:43:12 INFO util.ExitUtil: Exiting with status 0
15/04/18 14:43:12 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at laptop/192.168.1.1
************************************************************/
```

Note that **hadoop namenode -format** command should be executed once before we start using Hadoop. If this command is executed again after Hadoop has been used, it'll destroy all the data on the Hadoop file system.

Starting Hadoop

Now it's time to start the newly installed single node cluster. We can use **start-all.sh** or (**start-dfs.sh** and **start-yarn.sh**)

```
rashmi@laptop:~$ start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
15/04/18 16:43:13 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to /usr/local/hadoop/logs/hadoop-rashmi-
namenode-laptop.out
localhost: starting datanode, logging to /usr/local/hadoop/logs/hadoop-rashmi-
datanode-laptop.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /usr/local/hadoop/logs/hadoop-
rashmi-secondarynamenode-laptop.out
15/04/18 16:43:58 WARN util.NativeCodeLoader: Unable to load native-hadoop
library for your platform... using builtin-java classes where applicable
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-rashmi-
resourcemanager-laptop.out
localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-rashmi-
nodemanager-laptop.out
```

We can check if it's really up and running:

```
rashmi@laptop:~$ jps
9026 NodeManager
7348 NameNode
9766 Jps
```

```
8887 ResourceManager
7507 DataNode
```

The output means that we now have a functional instance of Hadoop running on our VPS (Virtual private server).

Hadoop Web Interfaces

Let's start the Hadoop again and see its Web UI:

Accessing HADOOP through browser

http://localhost:50070/

Verify all applications for cluster

http://localhost:8088/