Install Hadoop 2.6 on CentOS 6.7 (64 BIT)

1. Creating Hadoop Group

To access the Hadoop cluster, user must be part of hadoop group. So create the group hadoop and add the hduser(user already created) to hadoop group

```
sudo groupadd hadoop
sudo usermod -a -G hadoop hduser
```

2. Configure ssh access for hduser user at the local host to login without password.

```
ssh-keygen -t rsa -P ""
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys

# SSH may run in strict mode, only owner is authorized to access.
chmod 600 ~/.ssh/authorized_keys

# This should login into local system without prompting for password
ssh localhost
exit
```

Perform as hduser user:

```
cd ~
mkdir install
cd install
```

Below step is not required for now, as we have downloaded hadoop 2.6 and kept ready.
#wget http://apache.cs.utah.edu/hadoop/common/hadoop-2.6.0/hadoop-2.6.0.tar.gz

3. Copy hadoop-2.6.0.tar.gz from your downloaded path to /home/hduser/install and untar the complete hadoop-2.6.0 package

```
cd /home/hduser/install/
tar xvzf hadoop-2.6.0.tar.gz
sudo mv hadoop-2.6.0 /usr/local/hadoop
sudo chown -R hduser:hadoop /usr/local/hadoop
```

(Give ownership to hduser)

4. Edit the ~/.bashrc hadoop user profile file to load the environment variables when logged in as hadoop user.

```
cd ~
```

```
vi ~/.bashrc
export HADOOP_PREFIX=/usr/local/hadoop
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_MAPRED_HOME=${HADOOP_HOME}
export HADOOP_COMMON_HOME=${HADOOP_HOME}
export HADOOP_HDFS_HOME=${HADOOP_HOME}
export YARN_HOME=${HADOOP_HOME}
export HADOOP_CONF_DIR=${HADOOP_HOME}/etc/hadoop
# Native Path
export HADOOP_COMMON_LIB_NATIVE_DIR=${HADOOP_PREFIX}/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_PREFIX/lib"
#Java path
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.85.x86_64
# Add Hadoop bin/ directory to PATH
export PATH=$PATH:$HADOOP_HOME/bin:$JAVA_PATH/bin:$HADOOP_HOME/sbin
```

Save and Quit
esc → Shift : wq

5. Load the profile variables into environment

```
source ~/.bashrc
```

6. Edit the hadoop environment script to use java home variable used by Hadoop and modify the file with the following line

```
sudo vi /usr/local/hadoop/etc/hadoop/hadoop-env.sh
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-1.7.0.85.x86_64
```

7. Create the following Directories for hadoop temporary files, namenode metadata, datanode data and secondary namenode metadata.

```
sudo mkdir -p /usr/local/hadoop_store/tmp
sudo mkdir -p /usr/local/hadoop_store/hdfs/namenode
sudo mkdir -p /usr/local/hadoop_store/hdfs/datanode
sudo mkdir -p /usr/local/hadoop_store/hdfs/secondarynamenode
sudo chown -R hduser:hadoop /usr/local/hadoop_store
```

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

```
cp /usr/local/hadoop/etc/hadoop/mapred-site.xml.template /usr/local/hadoop/etc/hadoop/mapred-site.xml
```

9. Now you start with the configuration with basic hadoop single node cluster setup. First edit hadoop configuration files and make following changes.

   i) The mapred-site.xml file contains the configuration settings for MapReduce daemon on YARN

```
sudo vi /usr/local/hadoop/etc/hadoop/mapred-site.xml
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
</configuration>
```

   ii) The core-site.xml file informs Hadoop daemon where NameNode runs in the cluster. It contains the configuration settings for Hadoop Core such as I/O settings that are common to HDFS& MapReduce.

```
sudo vi /usr/local/hadoop/etc/hadoop/core-site.xml

<configuration>
<property>
<name>hadoop.tmp.dir</name>
<value>/usr/local/hadoop_store/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>
```

   iii) The hdfs-site.xml file contains the configuration settings for HDFS daemons; the NameNode, the Secondary NameNode, and the DataNodes. Here, we can configure hdfs-site.xml to specify default block replication. The actual number of replications can also be specified when the file is created. The default is used if replication is not specified in create time.

```
sudo vi /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>
<property>
<name>dfs.namenode.checkpoint.dir</name>
<value>file:/usr/local/hadoop_store/hdfs/secondarynamenode</value>
</property>
<property>
<name>dfs.namenode.checkpoint.period</name>
<value>3600</value>
</property>
</configuration>
```

iv) The yarn-site.xml file contains configuration information that overrides the default values for YARN parameters.

```
sudo vi /usr/local/hadoop/etc/hadoop/yarn-site.xml
```

```
<configuration>
<!-- Site specific YARN configuration properties -->
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
</configuration>
```

10. Now we test single node cluster installation:

When we format namenode it formats the meta-data related to data-nodes. By doing that, all the information on the datanodes are lost and they can be reused for new data. Normally namenode format will be done only at the first time of hadoop cluster setup.

```
hadoop namenode -format
```

11. Start the daemon services by running the below script

To start the Daemons individually (Useful in multinode cluster setup)

```
hadoop-daemons.sh start secondarynamenode
hadoop-daemons.sh start namenode
hadoop-daemons.sh start datanode
yarn-daemon.sh start nodemanager
yarn-daemon.sh start resourcemanager
mr-jobhistory-daemon.sh start historyserver
```

OR

To start the Daemons in single command (useful in single node cluster)

```
start-all.sh
```

OR

To start the Daemons separately HDFS and YARN (Useful when hdfs and yarn daemons installed separately)

```
start-yarn.sh (Resource Manager and Node manager)
start-dfs.sh (namenode, datanode and secondarynamenode)
```

12. Run JPS to ensure all daemons are started under the JVMs.

```
Jps
```

13. Create the following user directory in hdfs and change the ownlership of the directory to hadoop.

```
hadoop fs -mkdir -p /user/hduser
hadoop fs -chown -R hduser:hadoop /user/hduser
```

14. Login to the below Namenode web UI to view the namenode and datanode info.

http://localhost:50070/

15. Login to the below Resource manager web UI to view the RM info.

http://localhost:8088/