HADOOP CLUSTER WITH HDFS HIGH AVAILABILITY

TABLE OF CONTENTS

SETUP DETAILS:

Create 5 separate machines i.e., 1master and 3slaves with defined IP addresses
master 192.168.10.10
slave1 192.168.10.11 – Active NameNode
slave2 192.168.10.12 – Standby NameNode
slave3 192.168.10.13 – DataNode

STEP 1: INSTALL JDK7

Before installing hadoop make sure you have java installed on all nodes of hadoop cluster systems.

Download JDK7 for Linux-x64 from official Oracle site.

```
[root@master]# cd ~/Download
[root@master]# yum localinstall jdk-7u80-linux-x64.rpm
[root@master]# alternatives --install /usr/bin/java java /usr/java/jdk1.7.0_80/bin/java 210000
```

To check java version and also alternatives

```
[root@master]# java –version
[root@master]# alternatives --display java
```

This is need to done all the 4 machines.

STEP 2: CREATE USER ACCOUNT

Create a system user account on both master and slave systems to use for hadoop installation

```
[root@master]# useradd huser
[root@master]# passwd huser
```

STEP 3: ADD FQDN MAPPING

Edit /etc/hosts file on master and slave machines and add following entries.

```
[root@master]# gedit /etc/hosts
```

Append the following lines at the end of the file:

```
192.168.10.10 master
192.168.10.11 slave1
192.168.10.12 slave2
192.168.10.13 slave3
```

STEP 4: CONFIGURING KEY BASED LOGIN

It’s required to set up hadoop user to ssh itself without password. Use following commands to configure auto login between all hadoop cluster servers.

```
[root@master]# su – huser
[root@huser]$ ssh-keygen
[root@huser]$ ssh-copy-id -i ~/.ssh/id_rsa.pub huser@192.168.10.10
[root@huser]$ ssh-copy-id -i ~/.ssh/id_rsa.pub huser@192.168.10.11
[root@huser]$ ssh-copy-id -i ~/.ssh/id_rsa.pub huser@192.168.10.12
[root@huser]$ ssh-copy-id -i ~/.ssh/id_rsa.pub huser@192.168.10.13
 [root@huser]$ chmod 0600 ~/.ssh/authorized_keys
[root@huser]$ exit
```

To avoid typing password for each time we login:

```
[root@master]# gedit /etc/ssh/ssh_config
```

And search for “StrickHostKeyChecking”
Remove “#” and make it like this “StrickHostKeyChecking no” without double quote and save it.

STEP 5: DOWNLOAD AND EXTRACT

Download Hadoop 2.6.0

```
[root@master]# cd ~/Downloads
[root@master]# wget http://www.eu.apache.org/dist/hadoop/common/hadoop-2.6.0/hadoop-2.6.0.tar.gz
[root@master]# mkdir /opt/hadoop
[root@master]# cp ~/Downloads/hadoop-2.6.0.tar.gz /opt/hadoop
[root@master]# cd /opt/hadoop/
[root@master]# tar -xzf hadoop-2.6.0.tar.gz
[root@master]# chown -R huser /opt/hadoop
[root@master]# cd /opt/hadoop/hadoop-2.6.0/
```

Download Zookeeper

```
[root@master]# cd ~/Downloads
[root@master]# wget http://www-us.apache.org/dist/zookeeper/stable/zookeeper-3.4.9.tar.gz
[root@master]# mkdir /opt/hadoop/zookeeper
[root@master]# cp ~/Downloads/zookeeper-3.4.9.tar.gz /opt/hadoop/zookeeper
[root@master]# tar –xzf zookeeper-3.4.9.tar.gz
```

STEP 6: CONFIGURE HADOOP

Edit hadoop configuration files and make following changes.

```
[root@master]# cd /opt/hadoop/hadoop-2.6.0/etc/hadoop/
```

6.1 - Edit core-site.xml

```
[root@master]# core-site.xml
```

Add the following inside the <configuration> tag

```
<configuration>
<property>
        <name>fs.defaultFS</name>
        <value>hdfs://master:9000/</value>
</property>
<property>
        <name>dfs.journalnode.edits.dir</name>
        <value>/opt/hadoop/journalnode </value>
</property>
</configuration>
```

6.2 - Create Datanode and Namenode

Create HDFS DataNode data dirs on every node and change ownership of /opt/hadoop:

```
[root@master]# chown huser /opt/hadoop/ -R
[root@master]# chgrp huser /opt/hadoop/ -R
[root@master]# mkdir /opt/hadoop/datanode
[root@master]# chown huser /opt/hadoop/datanode/
[root@master]# chgrp huser /opt/hadoop/datanode/
```

Create HDFS NameNode data dirs on master:

```
[root@master]# mkdir /opt/hadoop/namenode
[root@master]# chown huser /opt/hadoop/namenode/
[root@master]# chgrp huser /opt/hadoop/namenode/
```

6.3 - Edit hdfs-site.xml

```
[root@master]# gedit hdfs-site.xml
```

Add the following inside the <configuration> tag

```
<configuration>
<property>
            <name>dfs.namenode.name.dir</name>
            <value/opt/hadoop/namenode</value>
 </property>
<property>
           <name> dfs.datanode.data.dir</name>
           <value>/opt/hadoop/dataenode</value>
</property>
 <property>
            <name>dfs.replication</name>
            <value>1</value>
 </property>
 <property>
            <name>dfs.permissions</name>
            <value>false</value>
 </property>
 <property>
            <name>dfs.nameservices</name>
            <value>ha-cluster</value>
 </property>
 <property>
            <name>dfs.ha.namenodes.ha-cluster</name>
            <value> slave1,slave2</value>
 </property>
 <property>
            <name>dfs.namenode.rpc-address.ha-cluster.nn1</name>
            <value>slave1:9000</value>
 </property>
 <property>
            <name>dfs.namenode.rpc-address.ha-cluster.nn2</name>
            <value>slave1:9000</value>
 </property>
 <property>
            <name>dfs.namenode.http-address.ha-cluster.nn1</name>
            <value>slave1:50070</value>
 </property>
 <property>
            <name>dfs.namenode.http-address.ha-cluster.nn2</name>
            <value>slave2:50070</value>
 </property>
 <property>
```

```
            <name>dfs.namenode.shared.edits.dir</name>
           <value>qjournal://slave1:8485;slave2:8485;slave3:8485/ha-cluster</value>
 </property>
 <property>
           <name>dfs.client.failover.proxy.provider.ha-cluster</name>
           <value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>
 </property>
 <property>
           <name>dfs.ha.automatic-failover.enabled</name>
           <value>true</value>
 </property>
 <property>
            <name>ha.zookeeper.quorum</name>
            <value>slave1:2181,slave2:2181,slave3:2181 </value>
 </property>
 <property>
            <name>dfs.ha.fencing.methods</name>
            <value>sshfence</value>
 </property>
 <property>
            <name>dfs.ha.fencing.ssh.private-key-files</name>
            <value>/home/huser/.ssh/id_rsa</value>
 </property>
</configuration>
```

6.4 - Edit mapred-site.xml

```
[root@master]# gedit mapred-site.xml
```

Add the following inside the <configuration> tag

```
<configuration>
 <property>
           <name>mapreduce.framework.name</name>
           <value>yarn</value>
 </property>
</configuration>
```

6.5 - Edit yarn-site.xml

```
[root@master]# gedit yarn-site.xml
```

Add the following inside the <configuration> tag

```
<configuration>
<property>
        <name>yarn.resourcemanager.hostname</name>
        <value>master</value>
</property>



<property>
        <name>yarn.nodemanager.hostname</name>
        <value>master</value>        <!-- or slave1, slave2, slave3 -->
</property>

<property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
</property>
</configuration>
```

6.6 Edit hadoop-env.sh

```
[root@master]# gedit hadoop-env.sh
Append the following lines at the end of the file:
export JAVA_HOME=/usr/java/jdk1.7.0_80
export HADOOP_OPTS=-Djava.net.preferIPv4Stack=true
export HADOOP_CONF_DIR=/opt/hadoop/hadoop-2.6.0/etc/hadoop
```

STEP 7: CONFIGURE ZOOKEEPER

7.1 - Create configuration file

```
[root@huser]$  cd /opt/hadoop/zookeeper/zookeeper-3.4.9
[root@huser]$ gedit conf/zoo.cfg
        Server.1=slave1:2888:3888
        Server.2=slave2:2888:3888
        Server.3=slave3:2888:3888
```

In Active namenode, change the directory where you want to store the zookeeper configuration file (dataDir property path).

Create the myid file inside the directory and add numeric 1 to the file and save the file.

```
[root@huser]$ cd /opt/hadoop/zookeeper/zookeeper-3.4.9
[root@huser]$ gedit myid
                1
```

Likewise create myid file, in a standby namenode change the directory where you want to store the zookeeper configuration file (dataDir property path).

Create the myid file inside the directory and add numeric 2 to the file and save the file.

In a data node, change the directory where you want to store the zookeeper configuration file (dataDir property path).

Create the myid file inside the directory and add numeric 3 to the file and save the file.

STEP 8: COPY HADOOP SOURCE TO SLAVE SERVERS

After updating above configuration, we need to copy the source files to all slave servers.

```
[root@master]# scp -rp /opt/hadoop slave1:/opt/
[root@master]# scp -rp /opt/hadoop slave2:/opt/
[root@master]# scp -rp /opt/hadoop slave3:/opt/
```

STEP 9: CONFIGURE HADOOP ON MASTER SERVER ONLY

Go to hadoop source folder on huser-master and do following settings.

```
[root@master]# su – huser
[root@huser]$ cd /opt/hadoop/hadoop-2.6.0/
```

```
[root@huser]$ gedit masters
```

And this line:

```
master
```

```
[root@huser]$ gedit slaves
```

Add this lines:

```
slave1
slave2
slave3
slave4
```

STEP 10: SETTING UP THE ENVIRONMENT FOR JAVA AND HADOOP

We need to source the environment files

```
[root@master]# su – huser
[root@huser]$ gedit ~/.bashrc
```

Append the following lines at the end of the file:

```
## JAVA env variables
export JAVA_HOME=/usr/java/jdk1.7.0_80
export PATH=$PATH:$JAVA_HOME/bin
export CLASSPATH=.:$JAVA_HOME/jre/lib:$JAVA_HOME/lib:$JAVA_HOME/lib/tools.jar
```

```
## HADOOP env variables
export HADOOP_HOME=/opt/hadoop/hadoop-2.6.0
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
```

```
##ZOOKEEPER
export ZOOKEEPER_HOME=/opt/hadoop/zookeeper/zookeeper-3.4.9
export PATH=$PATH:$ZOOKEEPER_HOME/bin
```

```
[root@huser]$ source ~/.bashrc
[root@huser]$ echo $HADOOP_HOME
[root@huser]$ echo $JAVA_HOME
[root@huser]$ exit
```

SCP to the ~/.bashrc to other slave machines

slave1

```
[root@master]# scp -rp /root/huser/.bashrc slave1:~/
[root@master]# ssh slave1
[root@slave1]$ source ~/.bashrc
[root@slave1]$ exit
```

slave2

```
[root@master]# scp -rp /root/huser/.bashrc slave2:~/
[root@master]# ssh slave2
[root@slave2]$ source ~/.bashrc
[root@slave2]$ exit
```

slave3

```
[root@master]# scp -rp /root/huser/.bashrc slave3:~/
[root@master]# ssh slave3
[root@slave3]$ source ~/.bashrc
[root@slave3]$ exit
```

STEP 11: START JOURNALNODE, FORMAT ACTIVE NAMENODE

11.1 - Start the JournalNode in all the three nodes

```
[root@master]$ ssh slave1
[root@slave1]# su – huser
[root@huser]$ hadoop-daemon.sh start journalnode
[root@huser]$ jps
```

When you enter jps command, you will see the JournalNode daemon in all the nodes.

11.2 - Format the Active namenode

Format Name Node on Hadoop Master only

```
[root@slave1]# su – huser
[root@huser]$ hdfs namenode –format
```

11.3 - Start the Namenode daemon in Active NameNode

```
[root@slave1]$ hadoop-daemon.sh start namenode
[root@slave1]$ exit
```

11.4 - Copy the HDFS Meta data from active name node to standby NameNode

```
[root@master]$ ssh slave2
[root@slave2]$ hdfs namenode -bootstrapStandby
```

Once you run this command, you will get the information from which node and location the meta data is copying and whether it is copying successfully or not.

11.5 - Start the namenode daemon in Standby namenode machine

```
[root@slave2]$ hadoop-daemon.sh start namenode
[root@slave2]$ exit
```

11.6 - Now start the Zookeeper service in all the three nodes

In Active Namenode:

```
[root@master]$ ssh slave1
[root@slave1]$ zkServer.sh start
[root@slave1]$ exit
```

In Standby Namenode:

```
[root@master]$ ssh slave2
[root@slave2]$ zkServer.sh start
```

```
[root@slave2]$ exit
```

In Data node:

```
[root@master]$ ssh slave3
[root@slave3]$ zkServer.sh start
```

11.7 - Start the Data node daemon in Data node machine

```
[root@slave3]$ hadoop-daemon.sh start datanode
[root@slave3]$ exit
```

11.8 – Start the Zookeeper fail over controller in Active name node and standby name node

Format the zookeeper fail over controller in Active namenode

```
[root@master]$ ssh slave1
[root@slave1]$ hdfs zkfc –formatZK
```

Start the ZKFC in Active namenode

```
[root@slave1]$ hadoop-daemon.sh start zkfc
[root@slave1]$ exit
```

Format the zookeeper fail over controller in Standby namenode

```
[root@master]$ ssh slave2
[root@slave2]$ hdfs zkfc –formatZK
```

Start the ZKFC in Standby namenode

```
[root@slave2]$ hadoop-daemon.sh start zkfc
[root@slave2]$ exit
```

Enter jps command to check the DFSZkFailoverController daemons

```
[root@master]$ jps
```

11.9 - Now check the status of each Namenode, which node is Active or which node is on Standby

```
[root@master]$ ssh slave1
[root@slave1]$ hdfs haadmin –getServiceState nn1
[root@slave1]$ hdfs haadmin –getServiceState nn2
```

11.10 - Now Check the status of each Namenode using the web browser

Open the Web browser and enter the below URL

192.168.10.11:50070 [Actine NameNode]

Open another tab in browser and the below URL

192.168.10.12:50070 [Standby NameNode]

11.11 - In the Active namenode, kill the namenode daemon to change the Standby name node to active namenode

Enter jps in Active namenode and kill the daemon

```
[root@slave1]$ jps
[root@slave1]$ sudo kill -9 7606 <namenode process ID>
```

11.12 - Open the two nodes through web browser and check the status

slave1 i.e., Active NameNode will become Standby NameNode

192.168.10.11:50070 [Standby NameNode]

And slave2 i.e., Standby NameNode will become Active NameNode
192.168.10.12:50070 [Active NameNode]