**** [**Querying jSON Data in Hive**](http://syed-rizvi.blogspot.com/2016/10/querying-json-data-in-hive_55.html)

Dear Friends,  
  
As usual, I was sitting to prepare for a session on hive tomorrow and thought of doing something new and exciting. So I did some research in saving and querying jSON data in Hive table and documented it as follows :-

1. Let us start with a simple example and then increase the complexity as we move ahead.
2. So open your nano editor and create a simple.json file there.
3. In this file, save a simple jSON data as **{"Foo":"ABC","Bar":"20090101100000","Quux":{"QuuxId":1234,"QuuxName":"Sam"}}**. Save it and cat it to see if the data has been saved properly
4. Now create a hive internal table to load and query this data by executing **CREATE TABLE json\_table ( json string );** command.
5. Now load the data inside this table using the **LOAD DATA LOCAL INPATH '//simple.json' INTO TABLE json\_table;**
6. Now let us query the jSON document from the hive table. We will first of all use get\_json\_object function in hive to query the data. This function take two arguments. First is the tablename.fieldname and second is the jSON field to parse.
7. Here is an example of the query - **select get\_json\_object(json\_table.json, '$') from json\_table;** where $ is the root of the document. Running the query should give you the complete document.
8. Now lets make things a little interesting by executing the following command :-  
   **select get\_json\_object(json\_table.json, '$.Foo') as foo,   
   get\_json\_object(json\_table.json, '$.Bar') as bar,  
   get\_json\_object(json\_table.json, '$.Quux.QuuxId') as qid,  
   get\_json\_object(json\_table.json, '$.Quux.QuuxName') as qnamefrom json\_table;**
9. You should get the output as

foo bar qid qname

ABC 20090101100000 1234 Sam

Posted 14th October 2016 by [Syed](http://www.blogger.com/profile/14076951030391603242)

0

**Add a comment**



[Sep](http://syed-rizvi.blogspot.com/2016/09/hsfs-federation-set-up-steps_29.html" \o "30th September 2016)

[30](http://syed-rizvi.blogspot.com/2016/09/hsfs-federation-set-up-steps_29.html" \o "30th September 2016)

[**HSFS Federation Set Up Steps**](http://syed-rizvi.blogspot.com/2016/09/hsfs-federation-set-up-steps_29.html)

Dear Friends,  
  
Just sitting to prepare for my weekend session and thought of setting up an HDFS Federated Cluster on my multi - node set up on virtual box. So here are the steps :-

1. Please note that you need to complete Hadoop multi-node as as a pre-requisite for this set up. This is as per my [this](http://syed-rizvi.blogspot.in/2016/01/hadoop-mult-node-installation-on.html) blog post.
2. Now from your multi - node set up, create one clone from **hadoopmnmaster** and name it as **fed-nn01.**
3. Create another clone from **hadoopmnmaster** and name it as **fed-nn02.**
4. Now change the IP Address of **fed-nn01** and **fed-nn02** as **192.168.56.101** and **192.168.56.102** respectively by making the changes into your **/etc/hosts** file and **/etc/network/interfaces** file. Don't forget to change the name of both the host accordingly in the **/etc/hostname** file too.
5. Now ensure that ssh is happening from both the machines to each other i.e **ssh fed-nn01** and **ssh fed-nn02** should be successful.
6. Now go to one of the slaves of your multi-node set up and create a clone out of it. Name it as **fed-dn01.**
7. Now change the IP Address of **fed-dn01** as **192.168.56.103** by making the changes into your **/etc/hosts** file and **/etc/network/interfaces** file. Here too, make sure that you are changing the name of your node accordingly in **/etc/hostname** file.
8. Lets create a client VM now. This the VM from where you can execute your hdfs command on the federated namespace. To do this, create one more clone from one of the slave machines of your multi-node set up. Name is as fed-client.
9. Now change the IP Address of **fed-client** as **192.168.56.105** by making the changes into your **/etc/hosts** file and **/etc/network/interfaces** file. Here too, make sure that you are changing the name of your node accordingly in **/etc/hostname** file.
10. You are all set now. Fire up all your VM instances and get ready to make the configuration changes in all of them.
11. In  **fed-nn01** name nodes, open the core-xml file and make the following changes :-

          <configuration>

          <property>  
          <name>fs.defaultFS</name>  
          <value>hdfs://fed-nn01</value>  
          </configuration>

1. Like wise in **fed-nn02** name nodes, open the **core-site.xml** file and make the following changes :-

         <configuration>

         <property>

         <name>fs.defaultFS</name>  
         <value>hdfs://fed-nn02</value>  
         </configuration>

1. Open the **core-site.xml** in **fed-dn01** and make the following changes :-

         <configuration>

         <property>

         <name>fs.default.name</name>

         <value>hdfs://fed-nn01,hdfs://fed-nn02</value>

         </property>

         </configuration>

1. Now open **core-site.xml** on **fed-client** and make the following changes :-

        <configuration>

        <property>

        <name>fs.default.name</name>

        <value>viewfs://fedcluster/</value>

        </property>

        <property>

        <name>fs.viewfs.mounttable.fedcluster.link./ns1</name>

        <value>hdfs://fed-nn01:8020</value>

        </property>

        <property>

        <name>fs.viewfs.mounttable.fedcluster.link./ns2</name>

        <value>hdfs://fed-nn02:8020</value>

        </property>

        </configuration>

1. Now open the **hdfs-site.xml** on **fed-nn01** and **fed-nn02** and make the following changes :-

        <configuration>

        <property>

        <name>dfs.replication</name>

        <value>1</value>

        </property>

        <property>

        <name>dfs.namenode.name.dir</name>

        <value>file:/usr/local/hadoop/hadoop\_data/hdfs/namenode</value>

        </property><property>

        <name>dfs.permissions</name>

        <value>false</value>

        </property>

        <property>

        <name>dfs.nameservices</name>

        <value>ns1,ns2</value>

        </property>

        <property>

        <name>dfs.namenode.rpc-address.ns1</name>

        <value>fed01:8020</value>

        </property>

        <property>

        <name>dfs.namenode.rpc-address.ns2</name>

        <value>fed02:8020</value>

        </property>

        </configuration>

1. Now open the **hdfs-site.xml** on **fed-dn01** and make the following changes :-

       <configuration>

       <property>

       <name>dfs.replication</name>

       <value>1</value>

       </property>

       <property>

       <name>dfs.datanode.data.dir</name>

       <value>file:/usr/local/hadoop/hadoop\_data/hdfs/datanode</value>

       </property>

       <property>

       <name>dfs.permissions</name>

       <value>false</value>

       </property>

       <property>

       <name>dfs.nameservices</name>

       <value>ns1,ns2</value>

       </property>

       <property>

       <name>dfs.namenode.rpc-address.ns1</name>

       <value>fed01:8020</value>

       </property>

       <property>

       <name>dfs.namenode.rpc-address.ns2

       </name>

       <value>fed02:8020</value>

       </property>

       </configuration>

1. Now open the **hdfs-site.xml** on **fed-client** and make the following changes :-

       <configuration>

       <property>

       <name>dfs.replication</name>

       <value>1</value>

       </property>

       <property>

       <name>dfs.permissions</name>

       <value>false</value>

       </property>

       <property>

       <name>dfs.nameservices</name>

       <value>ns1,ns2</value>

       </property>

       <property>

       <name>dfs.namenode.rpc-address.ns1</name>

       <value>fed01:8020</value>

       </property>

       <property>

       <name>dfs.namenode.rpc-address.ns2

       </name>

       <value>fed02:8020</value>

       </property>

       </configuration>

1. Lastly, open the **slaves** file on **fed-nn01** and **fed-nn02** and add **fed-dn01** to it so that the heartbeat mechanism between the masters and slaves gets started.
2. Now is the time to format and start the cluster. Execute **hdfs namenode -format -clusterID fedcluster** command on any one of the namenodes to format the cluster
3. Execute **start-dfs.sh** command to start the name nodes on **fed-nn01** and **fed-nn02** and datanode daemon on **fed-dn01**.
4. Now go to fed-client and execute **hdfs dfs -ls /.** You should be able to to see the namespaces with respect to both the namenodes now.
5. Try creating a folder and copying a file on both the namenodes. You should be able to do that form the client.
6. Explore. Enjoy. Have fun !

Posted 30th September 2016 by [Syed](http://www.blogger.com/profile/14076951030391603242)

1

**View comments**



[Aug](http://syed-rizvi.blogspot.com/2016/08/steps-and-concepts-to-run-word-count.html" \o "27th August 2016)

[27](http://syed-rizvi.blogspot.com/2016/08/steps-and-concepts-to-run-word-count.html" \o "27th August 2016)

[**Steps and concepts to run a word count program in Apache Spark**](http://syed-rizvi.blogspot.com/2016/08/steps-and-concepts-to-run-word-count.html)

Dear friends,

I was recently giving a classroom session to Big Data technologies where the attendees requested me to give them a primer on Apache Spark. I immediately made a short presentation on writing a word count program in spark. So here it is for you all as follows :-

1. All the data in Spark is presented in the form on Resilient Distributed Datasets (RDDs). There are two types of RDD operations - Transformations and Actions.
2. Transformations are the RDDs which construct new RDDs from the previous one. Actions on the other hand computes a result based on and RDD. They further on send the result to the driver program or dump it to the external storage (e.g. HDFS).
3. Once such spark transformation is **Flatmap.** Since Flatmap is the very first spark transformation used in the Word count program, lets understand it by way of an example.
4. Let us create a variable x on the Scala shell usingparallelize() method of the SparkContext object with **val x = sc.parallelize(List("spark rdd example", "sample example"))** command**.**
5. Let us apply the flatmap transformation on this variable using  **val y = x.flatMap(x => x.split(" "))** command.
6. Do a **y.collect()** on the the variable **y**. You should be able to see the following output - **res1: Array[String] = Array(spark, rdd, example, sample, example).**
7. What is happening here is that the Spark flatmap function is combining the supplied 2 variable in the list in step 4 above and returning an array of all the elements. It is basically flattening out the input supplied to it. The function flatmap here is similar to the function of String tokeniser in java which creates tokens.
8. With Flatmap transformation being understood, let us understand another transformation called as **Map.** This transformation offers the functionality similar to what Mapper object of the Map Reduce count does. Simply, picking up each words and marking them as 1. This process is also more formally called as the process of **segregation.**
9. So lets have a look at an example of Map transformation.  Similar to Flatmap example. let us create a variable again with **val x = sc.parallelize(List("spark", "rdd",  "example", "sample", "example"))**command**.**
10. Let us apply the flatmap transformation on this variable using  **val y = x.map(x => (x, 1))**command.
11. Do a **y.collect()**on the the variable **y**. You should be able to see the following output - **res0: Array[(String, Int] = Array((spark,1), (rdd,1), (example,1), (sample,1), (example,1)).** So you can see now that **map** transformation has nicely segregated the data and flat all the words with an integer **1.**
12. Lets make it very very simple now. Let us understand an RDD action which will just go ahead and count the unique word and present the sum for each of these word. This action is called as **countByKey.**
13. Let us go ahead and apply this action on variable y of step 10 above. What you will get is  
    **res11: scala.collection.Map[String,Long] = Map(spark -> 1, example -> 2, rdd -> 1, sample -> 1) .**
14. This output is clearly summing up the repeating keys for you.
15. Now let us pick up test.txt file from the local host using **val textFile = sc.textFile("test.txt")** command.
16. Combine or chain all the 3 transformations explained into one using t**extFile.flatMap(line => line.split(" ")).map(word => (word, 1)).countByKey;**
17. **You will see that the output that you will get is res16: scala.collection.Map[String,Long] = Map(is -> 1, This -> 1, wonderful -> 1, a -> 1, hadoop -> 3, session -> 1)**

Posted 27th August 2016 by [Syed](http://www.blogger.com/profile/14076951030391603242)

0

**Add a comment**



[Jul](http://syed-rizvi.blogspot.com/2016/07/hdfs-ha-high-availability-set-up-on.html" \o "20th July 2016)

[20](http://syed-rizvi.blogspot.com/2016/07/hdfs-ha-high-availability-set-up-on.html" \o "20th July 2016)

[**HDFS HA (High Availability) set up on Hadoop Multi Node Cluster**](http://syed-rizvi.blogspot.com/2016/07/hdfs-ha-high-availability-set-up-on.html)

Dear Friend,  
  
I recently did an HDFS HA installation on my Multi Node Hadoop cluster. So thought of immediately documenting here for everyones benefit. So there is goes :-

1. Please install the Hadoop Multi-Node cluster from my blog post dated 29th jan 2016. This will be required as the **hadoopmnmaster** from this set up will be needed to be cloned to create a second name node for an HA set up.
2. Clone **hadoopmnmaster** and name it as **hadoopmnmaster2** for the HA set up.
3. Let keep the IP address of this newly created **hadoopmnmaster2** as **192.168.56.15.**
4. Make changes in **/etc/hosts** file of **hadoopmnmaster2** to add the IP add of this machine. Do the same in **hadoopmnmaster** and**hadoopmnslave1.** Please note that we will be using a only one slave in this exercise and considering and initialising the other 2 slaves from the multi-node set up might slow down your machine depending upon how much RAM you have available.
5. Now you have 3 nodes running in your set up - **hadoopmnmaster, hadoopmnmaster2, hadoopmnslave1.** Start up each one of them on your machine and keep them ready for the further set up.
6. Download the latest version of zookeeper on each of these machines. Use the following command to do that - **wget http://mirror.fibergrid.in/apache/zookeeper/zookeeper-3.4.8/zookeeper-3.4.8.tar.gz.**
7. Untar the downloaded zookeeper binary using **tar -xvf zookeeper-3.4.8.tar.gz** command.
8. Now move the exploded folder to the /usr/local folder using sudo mv **zookeeper-3.4.8 /usr/local/zookeeper.** Repeat this process on all the nodes.
9. Now before we can configure zookeeper on all the nodes, let us configure the path for all the software required in .bashrc file. To do this open the bashrc file using **sudo nano ~/.bashrc** command and make sure that the following settings are done as below :-

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

export YARN\_CONF\_DIR=$HADOOP\_HOME/etc/hadoop

export JAVA\_HOME=

export ZOOKEEPER\_HOME =

export PATH=$PATH:$JAVA\_HOME/bin:$HADOOP\_HOME/bin:$HADOOP\_HOME/sbin:$ZOOKEEPER\_HOME/bin

1. Let us now open the core-site.xml and make the following changes into it :-

<configuration>  
<property>  
<name>fs.defaultFS</name>  
<value>hdfs://ha-cluster</value>  
</property>  
<property>  
<name>dfs.journalnode.edits.dir</name>  
<value>/usr/local/hadoop/temp</value>  
</property>  
</configuration>

1. Let us now open the hdfs-site.xml and make the following changes into it :

<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>nn1,nn2</value>  
</property>  
<property>  
<name>dfs.namenode.rpc-address.ha-cluster.nn1</name>  
<value>hadoopmnmaster:9000</value>  
</property>  
<property>  
<name>dfs.namenode.rpc-address.ha-cluster.nn2</name>  
<value>hadoopmnmaster2:9000</value>  
</property>  
<property>  
<name>dfs.namenode.http-address.ha-cluster.nn1</name>  
<value>hadoopmnmaster:50070</value>  
</property>  
<property>  
<name>dfs.namenode.http-address.ha-cluster.nn2</name>  
<value>hadoopmnmaster2:50070</value>  
</property>  
<property>  
<name>dfs.namenode.shared.edits.dir</name>  
<value>qjournal://hadoopmnmaster:8485;hadoopmnmaster2:8485;hadoopmnslave1:8485/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>hadoopmnmaster:2181,hadoopmnmaster2:2181,hadoopmnslave1:2181</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/syed/.ssh/id\_rsa</value>  
</property>

1. On **hadoopmnmaster** go to the conf directory of Zookeeper by executing **cd zookeeper-3.4.8/conf**
2. Create zoo.cfg file here by executing**cp zoo\_sample.cfg zoo.cfg** command at this location
3. Make a zookepeer directory where you will be zookeeper will be dumping all its data by executing the **mkdir /usr/local/zookeeper/zookeeper\_data** command.

Posted 20th July 2016 by [Syed](http://www.blogger.com/profile/14076951030391603242)

2

**View comments**



[Jul](http://syed-rizvi.blogspot.com/2016/07/zookeeper-multi-node-installation.html" \o "1st July 2016)

[1](http://syed-rizvi.blogspot.com/2016/07/zookeeper-multi-node-installation.html" \o "1st July 2016)

[**Zookeeper Multi - Node Installation**](http://syed-rizvi.blogspot.com/2016/07/zookeeper-multi-node-installation.html)

Dear Friends,  
  
Here comes another one in Zookeeper Series. This time it is Zookeeper Multi-Node. So lets go ahead and look at the steps as follows :-

1. The post assumes that Zookeeper is already installed in a single node mode as per my previous post. You would have already created a **zoo.cfg** file when you did that. You would have also created **/tmp/zookeeper** and given that as a **dataDir property** in your **zoo.cfg** file.
2. So for multi-node installation, just create 3 more directories - **/tmp/zookeeper/zoo1, /tmp/zookeeper/zoo2 and /tmp/zookeeper/zoo3.**
3. Now lets us create 3 different configuration files for 3 different Zookeeper instances. These files will be as follows :-

**zoo1.cfg**  
  
tickTime=2000

                 initLimit=5

                 syncLimit=2

                 dataDir=/var/lib/zookeeper/zoo1

                 clientPort=2181

                 server.1=localhost:2666:3666

                 server.2=localhost:2667:3667

                 server.3=localhost:2668:3668  
  
**zoo2.cfg**  
  
                 tickTime=2000  
                 initLimit=5

                 syncLimit=2

                 dataDir=/var/lib/zookeeper/zoo2

                 clientPort=2182

                 server.1=localhost:2666:3666

                 server.2=localhost:2667:3667

                 server.3=localhost:2668:3668   
  
**zoo3.cfg**

tickTime=2000

                 initLimit=5

                 syncLimit=2

                 dataDir=/var/lib/zookeeper/zoo13

                 clientPort=2183

                 server.1=localhost:2666:3666

                 server.2=localhost:2667:3667

                 server.3=localhost:2668:3668

1. Let us also create the server id parameter correctly in the **myid** file for each instance as follows :

  $ echo 1 > /var/lib/zookeeper/zoo1/myid

                 $ echo 2 > /var/lib/zookeeper/zoo2/myid

                 $ echo 3 > /var/lib/zookeeper/zoo3/myid

1. Let us now start each of these instances as follows :

$ ${ZK\_HOME}/bin/zkServer.sh **start** ${ZK\_HOME}/conf/zoo1.cfg

                 $ ${ZK\_HOME}/bin/zkServer.sh **start** ${ZK\_HOME}/conf/zoo2.cfg

                 $ ${ZK\_HOME}/bin/zkServer.sh **start** ${ZK\_HOME}/conf/zoo3.cfg

1. Your 3 Zookeeper instances on the same machine would now have started. If you check the status using the below command, you will be able to observe that one of them is a **leader**, the other would be **followers**:-

$ ${ZK\_HOME}/bin/zkServer.sh **status** ${ZK\_HOME}/conf/zoo1.cfg

                 $ ${ZK\_HOME}/bin/zkServer.sh **status** ${ZK\_HOME}/conf/zoo2.cfg

                 $ ${ZK\_HOME}/bin/zkServer.sh **status** ${ZK\_HOME}/conf/zoo3.cfg

Posted 1st July 2016 by [Syed](http://www.blogger.com/profile/14076951030391603242)

0

**Add a comment**



[Jun](http://syed-rizvi.blogspot.com/2016/06/common-commands-on-zookeeper-single.html" \o "30th June 2016)

[30](http://syed-rizvi.blogspot.com/2016/06/common-commands-on-zookeeper-single.html" \o "30th June 2016)

[**Common Commands on Zookeeper Single Node Installation**](http://syed-rizvi.blogspot.com/2016/06/common-commands-on-zookeeper-single.html)

Dear Friends,  
  
Doing some research for a presentation on Zookeeper so thought I'll quick jot down some important and useful commands to be run on Zookeeper. So there it goes in a series of steps :-

1. This post assumes that you have already installed Zookeeper. This should not be a very big deal if you have been looking at my other posts in the past few months, irrespective of the platform you are using (i.e Windows or Mac).
2. Once your Zookeeper is installed, its time to start the server. You can do that by executing **$ZOOKEEPER\_HOME/bin/zkServer.sh start**command. The server should have started now. You can check the status of your server by executing **$ZOOKEEPER\_HOME/bin/zkServer.sh status**command. This should tell you that the server is running in Stand Alone Mode.
3. Let us start the java client shell now. You can do that by executing **$ZOOKEEPER\_HOME/bin/zkCli.sh -server 127.0.0.1:2181.** This should start the shell. The shell will look something like this - **[zkShell: 0].**
4. So now the java client shell is also started. Run a few commands like **help** and **ls /** to get a feel of this shell.
5. Let us now create a ZNode with some data. This can be done by executing **create /FirstZnode “Myfirstzookeeper-app”**command. This command will create a ZNode on your Zookeeper Installation. You can check this by again executing **ls /** command. This should give you an output like this **[zookeeper, FirstNode]**. Execute a **get /FirstNode** on the shell and you should get **“Myfirstzookeeper-app”** return.
6. By defaul the ZNodes are persistent. We can create other types of Znodes also like **Ephemeral.** Such type of zookeeper nodes are temporary in nature and cease to exist when the session is over. An example is ApplicationMaster from YARN which uses **Ephemeral Znodes.**
7. In order to create an **Ephemeral** Znode execute **create -e /EphemeralZnode “MyEphemeralZookeeper-app”**command.  This should create an ephemeral node. So now temporary application application like ApplicationMaster can be assigned to it. You can also check by restarting the zookeeper server and the shell. The Ephemeral Znode will now cease to exist.
8. Time now to create a sequential Znode. Execute **create -s /SequentialZnode “MySequentialZookeeper-app”**command to create a Sequential Znode. You create Sequential Nodes when you want to create nodes with distributed mutexes. In the sense, when your client wants to hold a lock (for example for Resource Manager or HDFS High Availability Active Machines), you create a Sequence ZNode. So the node with the lowest number in the sequence becomes the leader.
9. You can also create a ZNode which is both sequential and ephemeral at the same time. You can do this by executing  **create -s -e /SequentialEpheZnode “MySequentialEpheZookeeper-app”** command. These kind of Znodes are best used for Master Machines like Namenode, ResourceManager or HBaseMaster type of machines.
10. Lastly, you can also set a watch on your Znode of any type above. This enables you to get the notification on the client from where you have set the watch when the data changes on this ZNode.
11. Let us set up a watch on our first znode created above **/FirstNode** by executing the following command **get /FirstNode 1.**
12. Now open another instance of the java shell and change the data on this node by executing **set /FirstNode data-updated.** You should see now that the first shell where the watch was set gives a notification that the data has changed.
13. Happy Zookeeping !

Posted 30th June 2016 by [Syed](http://www.blogger.com/profile/14076951030391603242)

1

**View comments**



[Jun](http://syed-rizvi.blogspot.com/2016/06/hbase-installation-in-fully-distributed.html" \o "24th June 2016)

[24](http://syed-rizvi.blogspot.com/2016/06/hbase-installation-in-fully-distributed.html" \o "24th June 2016)

[**HBase Installation in Fully Distributed Mode**](http://syed-rizvi.blogspot.com/2016/06/hbase-installation-in-fully-distributed.html)

Dear Friends,  
  
I recently did some research on Hbase Fully Distributed Installation and thought of documenting it here. So here are the steps to do the installation :-

1. Do the hadoop multi - node installation as per my [this](http://syed-rizvi.blogspot.in/2016/01/hadoop-mult-node-installation-on.html) blog post. This is **very important first step** for this installation otherwise you wont get the hadoop multi-node set up and nodes to creates slaves for this Hbase set up.
2. Once done, clone your **hadoopmnmaster** VM instance to **hadoopmnmaster2**. Now this instance will act as **HBaseMaster** of your Hbase fully distributed set up. Change the IP of this machine to **192.168.56.15** in both **/etc/hosts (**basically add this ip entry there as hadoopmnmaster2**)**file and **/etc/network/interfaces.** Also, change the hostname of this new machine to **hadoopmnmaster2** in **/etc/hostname** file.
3. So we have altogether 5 nodes now - hadoopmnmaster, where Namenode should be running and hadoopmnslave1, hadoopmnslave2 and hadoopmnslave3 where Datanodes and NodeManagers (and later on region server daemons) should be running. Lastly, hadoopmnmaster2, which be be used as a master in Hbase fully distributed setup.
4. For our HBase fully distributed set up, as mentioned above, we will consider **hadoopmnmaster2** as the Master machine for Hbase and **hadoopmnslave1, hadoopmnslave2 and hadoopmnslave3** as region servers. This means an additional deamon Region Server will now be running on each of the hadoop slaves.
5. Add the entry for HMaster (hadoopmnmaster2/198.162.56.15) in the **/etc/hosts f**ile of all the slaves. This make all the region servers aware of the HBaseMaster. Add the same entry to the Namenode (hadoopmnmaster/198.162.56.11) machine too so that Namenode is also aware of the HBase existence.
6. Time to download HBase now. You can download the latest release from [this](http://mirror.fibergrid.in/apache/hbase/stable/hbase-1.1.5-bin.tar.gz) link. Download it first on the HMaster machine (hadoopmnmaster2). Once downloaded, you can ftp / sftp to the slave nodes from this machine. You can connect from slave machines using ftp/sftp username@machinename. Once connected, you can get "get" command to download the hbase tar file.
7. The Hbase tar ball should be there on each of the slaves now. Untar it using tar -xvf hbase.tar command. This will create a the hbase folder dump in the same location on the slaves. Do the same on the HBaseMaster as well.
8. You should have Hbase binary dump on all the HBase nodes (HBaseMaster and 3 Region Servers). Open your **/Hbase Folder/conf/hbase-env.sh** file in your favourite linux editor and add **export HBASE\_MANAGE\_ZK=true** for all the Hbase machines. This will ensure that your HBase will be managing the Zookeeper internally. You can have External Zookeeper as well but that is a topic for another day :-).
9. Again at the same location on all the region server, open **/Hbase Folder/conf/hbase-site.xml** and add the following entries to it :-

          <configuration>  
          <property>  
          <name>hbase.master</name>  
          <value>hadoopmnmaster2:60000</value>  
          </property>   
          <property>  
          <name>hbase.rootdir</name>  
          <value>hdfs://hadoopmnmaster:9000/hbase</value>  
          </property>   
          <property>  
          <name>hbase.cluster.distributed</name>  
          <value>true</value>  
          </property>  
          <property>  
          <name>zookeeper.session.timeout</name>  
          <value>1200000</value>  
          </property>  
          <property>  
          <name>hbase.zookeeper.property.clientPort</name>  
          <value>2181</value>   
          </property>   
          <property>   
          <name>hbase.zookeeper.quorum</name>   
          <value>hadoopmnmaster2</value>   
          </property>   
          </configuration>

1. The first entry above tells about the machine and port on which Hbase master is running. The second entry gives the location on HDFS where all the HBase related files will go. The third entry tells that this is a fully distributed cluster. The fourth entry keeps the Zookeeper timeout to 2 minutes. You can increase or decrease it according to your requirement. The fifth entry tells the port in which the Zookeeper is running and the 6th tells about the machine on which Zookeeper us running.
2. Now open **Hbase/conf/regionservers** file on all the slave nodes and the 3 region servers i.e hadoopmnslave1, hadoopmnslave2 and hadoopmnslave3 on each of these 3 machines. Add the same to HBase Master machine as well. You all all set to start your Hbase cluster now.
3. Execute **HBase/bin/start-hbase.sh** from the commnad prompt. This should start the Hbase master and Zookeeper in HBaseMaster machine and HRegionServers in all the 3 slave machines.
4. Execute **HBase/bin/hbase shell** to start your hbase shell. Execute "status" and "list" commands to ensure that your cluster is running up and fine.

Posted 24th June 2016 by [Syed](http://www.blogger.com/profile/14076951030391603242)

0

**Add a comment**



[Jun](http://syed-rizvi.blogspot.com/2016/06/hadoop-single-node-installation-on-mac.html" \o "16th June 2016)

[16](http://syed-rizvi.blogspot.com/2016/06/hadoop-single-node-installation-on-mac.html" \o "16th June 2016)

[**Hadoop Single Node Installation on Mac OS (Pseudo - Distributed Mode )**](http://syed-rizvi.blogspot.com/2016/06/hadoop-single-node-installation-on-mac.html)

Dear Friends,  
  
Let us look at the installation steps to install hadoop in pseudo - distributed mode on Mac OS. So here are the steps :-

1. We will be using brew to install hadoop on Mac OS. Brew is a package manager for mac which lets you install the softwares and help you manage it  as well. It's based on git and ruby. Additional information about can be seen at http://brew.sh/.
2. So there you go. Install brew by executing **ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"** from your terminal.
3. Once brew is installed, just run **brew install hadoop** command. This should download latest version of Hadoop and start installing it (brew calls it pouring!).
4. Note that hadoop installation will ask you to do 2 more installations before hadoop can be downloaded. a ) Xcode and b) Java. You can install xcode from App Store, For java, use **brew install Caskroom/cask/java** to install the latest version of java. Hadoop installation should be smooth after that.
5. Once hadoop is installed, go to **/usr/local/Cellar/hadoop/2.7.2/libexec/etc/hadoop** location and open core-site.xml file. Go to **step 25** of my Single Node Installation Post of this blog (Dated 25th Jan 2016) and copy all the properties from that step into this file. This should configure the namenode daemon on your machine.
6. Do the same for **yarn-site.xml**, **hdfs-site.xml** and **mapred-site.xml** at the same above location by following **steps 26, 27 and 28** of my single node blog post on 25th Janto configure HDFS and map reduce further.
7. Once done, add the follow aliases in your **~/.profile** file to make it easy to start and stop hadoop in your machine. You can use VIM or nano editer to do that.

alias hstart="/usr/local/Cellar/hadoop/2.7.2/sbin/start-dfs.sh;/usr/local/Cellar/hadoop/2.7.2/sbin/start-yarn.sh"  
alias hstop="/usr/local/Cellar/hadoop/2.7.2/sbin/stop-yarn.sh;/usr/local/Cellar/hadoop/2.7.2/sbin/stop-dfs.sh"

1. Bounce the .profile file by executing **source**  **~/.profile** command.
2. Enable remote login on your machine by going to**“System Preferences” -> “Sharing”.** Check **“Remote Login”** from the options on the LHS.
3. Check if you are able to connect to ssh on localhost using **ssh localhost** command. If not, generate ssh keys by executing **ssh-keygen -t rsa** command. You should now be able log in to ssh shell using  **ssh localhost**command.
4. Format your namenode by executing **hdfs namenode -format**command. Remember that this command will erase all your namenode meta deta from namenode so before about this command going forward. It should be ok for now.
5. You are all set to start your Hadoop Single Node cluster now. Just execute **hstart** on your terminal and let all the daemons start. Check if all them are running (Namenode, Datanode, ResourceManager, NodeManager, SecondaryNameNode).

Posted 16th June 2016 by [Syed](http://www.blogger.com/profile/14076951030391603242)

4

**View comments**



[Jun](http://syed-rizvi.blogspot.com/2016/06/commonly-used-linux-unix-commands.html" \o "15th June 2016)

[15](http://syed-rizvi.blogspot.com/2016/06/commonly-used-linux-unix-commands.html" \o "15th June 2016)

[**Commonly Used Linux / Unix Commands**](http://syed-rizvi.blogspot.com/2016/06/commonly-used-linux-unix-commands.html)

Dear Friends,  
  
Here is a list of some commonly used Linux / unix commands for your reference. You can practice these commands on Ubuntu. The commands will work on macos unless otherwise noted  
  
**Important Note** - Please use the **nano editor** in both ubuntu and mac if you need to change any file. You can use the nano editor like this - sudo nano xyz.txt. To come out of nano, just press CNTRL+X and press Y to save.  
  
So there you go. Let us look at some of the basic commands first :-

* **cat :-**Sends file contents to standard output. This is a way to list the contents of short files to the screen. Here is an example - cat .bashrc (cat ~/.profile on mac).
* **cd :-**Changes the current working directory . For e.g cd /home - this command will change your working directory to /home. The '/' indicates path relative to root.
* **cd.. :-**This command will take to you to the parent directory of your current working directory.
* **cd ~ :-**This command will take to use home directory. The same can be achieved with cd \
* **cp myfile yourfile :-**This command copies the file "myfile" to the file "yourfile" in the current working directory. This command will create the file "yourfile" if it doesn't exist. It will normally overwrite it without warning if it exists. In order to get a prompt before overwritting, add a "-i" option like this to your command - **cp -i myfile yourfile.** Also, in order to copy a file from some location to your current directory you can do this - **cp -i /some/other/location .**
* **cp -dpr srcdir destdir :-**Copy all files from the directory "srcdir" to the directory "destdir" preserving links (-p option), file attributes (-p option), and copy recursively (-r option). With these options, a directory and all it contents can be copied to another directory.
* **more abc.txt :-**Lists the contents of the "abc.txt" file to the screen one page at a time.
* **less abc.txt :-**Similar to the more command, but the user can page up and down through the file. The example displays the contents of "abc.txt". Press "q" if you want to exit out of the contents shown in ths file.
* **ln -s abc.txt symlink :-**Creates a symbolic link named symlink that points to the file test Typing "ls -l test symlink" will show that symlink points to the file test. Symlinks are usually created to when you want to have a permanent pointers created to your files, irrespective of the computer or source you are accessing the symlink from.
* **locate abc :-**Lists all files whose names contain the string "abc".
* **logout :-**Logs out the current user from the system
* **ls :-**List files in the current working directory except those starting with . and only show the file name
* **ls -al :-**List all files in the current working directory in long listing format showing permissions, ownership, size, and time and date stamp.
* **ls -al |more :-**Performs a directory listing of all files and pipes the output of the listing through more. If the directory listing is longer than a page, it will be listed one page at a time.
* **mv myfile yourfile :-**Move the file from "myfile" to "yourfile". This effectively changes the name of "myfile" to "yourfile".
* **pwd :- s**hows the name of the current working directory
* **shutdown -h now :-**Shuts the system down to halt immediately.
* **shutdown -r now :-**Shuts the system down immediately and the system reboots.
* **where is** **:-**Locates binaries and manual pages for the ls command.

Posted 15th June 2016 by [Syed](http://www.blogger.com/profile/14076951030391603242)

0

**Add a comment**



[Jun](http://syed-rizvi.blogspot.com/2016/06/similarities-and-differences-between.html" \o "6th June 2016)

[6](http://syed-rizvi.blogspot.com/2016/06/similarities-and-differences-between.html" \o "6th June 2016)

[**Similarities (and differences) between Hadoop and Spark Architectures**](http://syed-rizvi.blogspot.com/2016/06/similarities-and-differences-between.html)

Hi Friends,

I was just now writing a proposal for one of my clients to compare Spark with Hadoop and came up with some useful points. Thought I should quickly document it here for everyone's benefit :-

* Hadoop Map Reduce had a driver program for which you have to write the code. This program is where you specify your job configurations like your mappers, reducers etc. Spark also has a similar kind of driver.
* There are 2 ways in which you are deal with the spark drivers.
* First, spark provides you programming shells in 2 different languages - Scala and Python. So when you start these shells, these shells act as a spark driver environment for you. Additionally, spark provide support for Java and R which do not provide shells
* Second, you can write driver programs of your own. This can be done in all supported languages i.e Scala, Python, Java and R.
* In the Driver program, Map Reduce has Job and Configuration objects which represents the instance of your job and does all the co - ordination for your jobs submitted. **Spark Context** object does the same for Spark.
* Map Reduce Master daemon is called as Resource Manager which does all the job co-ordination from the time the job is submitted till the time its gets over. Spark has a more generic interface called Cluster Manager which not only supports Hadoop ResourceManager (through YARN) but also support other processing frameworks like Apache Mesos and infact it's own standalone cluster manager. Which ever cluster manager is used, it works as a conduit between your Spark Context object and the underlying cluster.
* Slaves or Commodity Machines in Hadoop run two slaves daemons - datanode for block management and Node Manager for running of Map Reduce tasks. In spark, the slave nodes are called as Worker Nodes. They run a process called as executer (same like Data Node and Node Manager) which manages the data and runs the tasks.
* So now the way spark works is - through the spark context, via the underlying cluster manager (Standalone, YARN or Mesos), it gets hold of the executer processes for the spark job to run. Then it sends the application code (jar file for Java or Spark and Scale, or python script). Finally, spark runs tasks on these blocks of data managed by the Executers, using the code send vial the jar file, python script.

Posted 6th June 2016 by [Syed](http://www.blogger.com/profile/14076951030391603242)

0

**Add a comment**



[Mar](http://syed-rizvi.blogspot.com/2016/03/connecting-from-host-to-vm-using-ftp.html" \o "14th March 2016)

[14](http://syed-rizvi.blogspot.com/2016/03/connecting-from-host-to-vm-using-ftp.html" \o "14th March 2016)

[**Connecting from Host to VM using ftp software**](http://syed-rizvi.blogspot.com/2016/03/connecting-from-host-to-vm-using-ftp.html)

Dear friends,  
  
One of the major challenges faced when running hadoop programs on virtual machines is to successfully connect from your host machine to the virtual machine. Here is a series of steps which will help you do that :-

1. Follow steps 1 to 14 of my "Hadoop Multi-node installation on Virtual Machine". This will ensure that the basic network adapters are enabled on your virtual machine.
2. What you need to do now is the create static on your virtual machine. To do this open you hosts file by executing **sudo nano /etc/hosts** command.
3. In this file, just below your locahost entry, add another entry by some IP number and host name. An example would be **192.168.56.50   hadoopecosystem.**
4. Open your network interfaces file by executing **sudo nano /etc/network/interfaces** command.
5. Enable the DHCP here by adding the following lines in this file :

              #The primary network interface

              auto eth0

              iface eth0 inet static

      address 192.168.56.50

      netmask 255.255.255.0

              auto eth1

              iface eth1 inet dhcp

1. Open your hosts file now by executing **sudo nano /etc/hostname.** Change the hostname to hadoop ecosystem here.
2. You should be home now. Try connecting with you file sharing software now using above static IP and your VM long in credentials.

Posted 14th March 2016 by [Syed](http://www.blogger.com/profile/14076951030391603242)

0

**Add a comment**



[Mar](http://syed-rizvi.blogspot.com/2016/03/important-hdfs-commands.html" \o "11th March 2016)

[11](http://syed-rizvi.blogspot.com/2016/03/important-hdfs-commands.html" \o "11th March 2016)

[**Important HDFS Commands**](http://syed-rizvi.blogspot.com/2016/03/important-hdfs-commands.html)

Dear Friends,  
  
Here is a complete list of some of the important HDFS commands that you might using quite frequently :

1. **hadoop version** - A very handy command to know the version of Hadoop you might be working on.
2. **hadoop fs -ls /** - Another useful command to display the list of files and directory in HDFS File path
3. **hadoop fs – mkdir /direcory\_name   –** creates the directory in HDFS
4. **hadoop fs -touchz  /directory-name/file\_name** **–** creates the file HDFS with a size of 0 bytes
5. **hadoop fs –cat  /directory-name/file\_name** **–** displays file content to stdout
6. **hadoop fs –count  /directory-name/ –** displays the count of directories, files, bytes for path provided
7. **hadoop fs –put  XYZ.txt /directory-name/XYZ.txt –** Copies the local file to HDFS on the specified path
8. **hadoop fs –get /directory-name/XYZ.txt ./XYZ.txt –** reverse of **put**
9. **hadoop fs -rm -r** to remove the files and **hadoop fs –rm** to remove directories

Please refer to the [this](http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/FileSystemShell.html) link for a complete list of all the HDFS command.

Many Thanks,

Syed

Posted 11th March 2016 by [Syed](http://www.blogger.com/profile/14076951030391603242)

0

**Add a comment**



[Feb](http://syed-rizvi.blogspot.com/2016/02/mysql-configuration-as-hive-metastore-db.html" \o "4th February 2016)

[4](http://syed-rizvi.blogspot.com/2016/02/mysql-configuration-as-hive-metastore-db.html" \o "4th February 2016)

[**MySQL Configuration as Hive Metastore DB**](http://syed-rizvi.blogspot.com/2016/02/mysql-configuration-as-hive-metastore-db.html)

Dear Friends,  
  
We all know that the default Metastore DB that comes packaged with Hive binary is Derby. Derby is good for initial Hive binary testing and other other POCs. For full scale development, you will need a database like MySQL for multi-developer connection to the same Metastore. So here is a list of steps required for doing MySQL database configuration with your Hive installation :-

1. Pre-requisite softwares
   1. Hive 1.2.x installed
   2. Hadoop 2.x installed
   3. MySQL 5.x installed
2. If you are doing this on MacOS and using brew to install it, please make sure that you are running the following commands before proceeding :-
   1. brew update
   2. brew upgrade
   3. brew clean up
   4. brew install hive. This should install the latest version of hive
   5. brew install mysql. This should install the latest version of mysql
3. Download a MySQL to java connector using **sudo apt-get install libmysql-java** command.
4. You will need to copy this connector to $HIVE\_HOME/lib using **/usr/share/java/mysql-connector-java.jar $HIVE\_HOME/lib/mysql-connector-java.jar**command.
5. Before you log in to MySQL, make sure that you are traversing to **/usr/local/Cellar/hive/2.0.1/libexec/scripts/metastore/upgrade.** This should help you run your scripts properly in the next step.
6. Now log into your MySQL shell using root password. Create a database by executing **create database metastore;** command and get into it using **use metastore;** command.
7. Now load the metastore schema into this newly created data base by executing  **SOURCE hive-schema-2.0.0.mysql.sql.**
8. Create a new user in this shell using **CREATE USER 'hiveuser'@'%' IDENTIFIED BY 'hivepassword';**
9. Grant required privileges to this user by executing the **GRANT all on \*.\* to 'hiveuser'@localhost identified by 'hivepassword';**
10. Use the flush commands for these changes to come into effect using the **flush privileges;**
11. Now go to your hive installation directory and go the $HIVE\_HOME/conf folder.
12. Create a new **hive-site.xml** file and add the following lines to it

<configuration>

<property>

<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://localhost/metastore?createDatabaseIfNotExist=true

<description>metadata is stored in a MySQL server</description>

</property>

<property>

<name>javax.jdo.option.ConnectionDriverName</name>

<value>com.mysql.jdbc.Driver</value>

<description>MySQL JDBC driver class</description>

</property>

<property>

<name>javax.jdo.option.ConnectionUserName</name>

<value>hiveuser</value>

<description>user name for connecting to mysql server</description>

</property>

<property>

<name>javax.jdo.option.ConnectionPassword</name>

<value>hivepassword</value>

<description>password for connecting to mysql server</description>

1. </property>

</configuration>

* 1. You can start hive now hive shell again now. It should create the metadata in connected MySQL database

Posted 4th February 2016 by [Syed](http://www.blogger.com/profile/14076951030391603242)

4

**View comments**

[Feb](http://syed-rizvi.blogspot.com/2016/02/pig-installation.html" \o "4th February 2016)

[4](http://syed-rizvi.blogspot.com/2016/02/pig-installation.html" \o "4th February 2016)

[**Pig Installation**](http://syed-rizvi.blogspot.com/2016/02/pig-installation.html)

Hi Friends,  
  
Pig installation is going to be the easiest one. Here is a series of steps

* 1. Pre-requisite softwares :-
     1. Java 1.6 +
     2. Ubuntu Server 15.04 or LTS 14.04
     3. Hadoop 2.x
  2. Download Pig archive using **wget http://a.mbbsindia.com/pig/pig-0.15.0/pig-0.15.0.tar.gz** command
  3. Unpack the archive in your home folder using **tar -xvf** **pig-0.15.0.tar.gz**command.
  4. Move it to **/usr/local**using **sudo mv pig-0.15.0  /usr/local/pig**command.
  5. Now open the bash profile using s**udo nano $HOME/.bashrc** command and add the following parameters into it :-

                     export PIG\_HOME=/usr/local/pig  
                     export PATH=$PATH:$PIG\_HOME/bin

* 1. Start pig shell by typing Pig command on your terminal shell

Posted 4th February 2016 by [Syed](http://www.blogger.com/profile/14076951030391603242)

1

**View comments**

[Feb](http://syed-rizvi.blogspot.com/2016/02/hive-installation.html" \o "3rd February 2016)

[3](http://syed-rizvi.blogspot.com/2016/02/hive-installation.html" \o "3rd February 2016)

[**Hive Installation**](http://syed-rizvi.blogspot.com/2016/02/hive-installation.html)

Hi Friends,  
  
Following is the series of steps you can go through to install hive 1.2.1 on your ubuntu server :-

* 1. Here is a list of prerequisite softwares for this installation
     1. Java 1.6+
     2. Ubuntu Server 15.10 or 14.04.3 LTS.
     3. Hadoop 2.x
  2. Download hive archive using the wget http://redrockdigimark.com/apachemirror/hive/hive-1.2.1/apache-hive-1.2.1-bin.tar.gz.
  3. Unpack the archive using **tar -xvf apache-hive-1.2.1-bin.tar.gz** in your current location.
  4. Move it to **/usr/local** using **sudo mv apache-hive-1.2.1 /usr/local/hive** command.
  5. Now open the bash profile using s**udo nano $HOME/.bashrc** command and add the following parameters into it :-

                     export HIVE\_HOME=/usr/local/hive  
                     export PATH=$PATH:$HIVE\_HOME/bin  
                     export HADOOP\_USER\_CLASSPATH\_FIRST=true

* 1. Now do a source**$HOME/.bashrc** for the changes to take into effect
  2. Make a temp directory in HDFS by executing **hadoop fs -mkdir /tmp** command.
  3. Make hive warehouse directory by executing **hadoop fs -mkdir /user/hive/datawarehouse**.  You might have to create each one of these directories individually in the hierarchy.
  4. Give proper rights to these newly created directories by executing **hadoop fs -chmod g+w /temp** and **hadoop fs -chmod g+w** **/user/hive/datawarehouse.**
  5. To overcome a bug between hive and hadoop, you need to copy **jline-2.12.jar** by executing **cp /usr/local/hive/jline-2.12.jar /usr/local/hadoop/share/hadoop/yarn/lib.**
  6. You can start your hive shell now to browse through the hive database and tables

Posted 3rd February 2016 by [Syed](http://www.blogger.com/profile/14076951030391603242)

2

**View comments**

[Jan](http://syed-rizvi.blogspot.com/2016/01/hadoop-mult-node-installation-on.html" \o "29th January 2016)

[29](http://syed-rizvi.blogspot.com/2016/01/hadoop-mult-node-installation-on.html" \o "29th January 2016)

[**Hadoop Multi-Node Installation on Virtual Machine**](http://syed-rizvi.blogspot.com/2016/01/hadoop-mult-node-installation-on.html)

Hi Friends,

This post is regarding Hadoop installation on a multi node cluster. I'm going to use Virtual Box for this purpose to utilise its VM Cloning feature.

So to start with, lets have a look at software and hardware requirement for it as follows :-

* + A single computer (yes, a single computer as we will will using cloning feature of Virtual Box)
  + Minimum 4 GB RAM. 8 GB desired.
  + High Speed internet connection for software downloads.

So lets get started with it in the following steps :-

* 1. Create a Single Node Hadoop installation as per my previous post.
  2. Now we have to start creating the clones using the "cloning" feature of Oracle Virtual Box.
  3. Lets create the first clone from the single node set up and call it **hadoopmnmaster.**
  4. Once the clone is created, start this instance and and open the hosts file by entering the **sudo nano /etc/hosts** command.
  5. In this file we have to enter the host and ip details of the all the nodes in the cluster (both Master and Slaves) as follows :-
     1. 192.168.56.10            hadoopmnmaster
     2. 192.168.56.11            hadoopmnslave1
     3. 192.168.56.12            hadoopmnslave2
     4. 192.168.56.13            hadoopmnslave3
  6. Now open the hostname file on **hadoopmnmaster** using sudo nano **/etc/hostname** command and change the hostname to **hadoopmnmaster** in this file.
  7. Now open this instance's core-site.xml using the **sudo nano /usr/local/hadoop/etc/hadoop/core-site.xml** command. Replace the localhost to **hadoopmnmaster** for the namenode URL entry.
  8. Open the interfaces file using **sudo nano /etc/network/interfaces** command. Change the IP in this file to **192.168.56.10**
  9. Now open this instance's hdfs-site.xml using the **sudo nano /usr/local/hadoop/etc/hadoop/hdfs-site.xml**command. Change the replication factor from 1 to 3.
  10. Remove the the entry from namenode in this file. This is done temporarily so that the instance can be cloned as slaves.
  11. Reboot the instance using **sudo reboot command**.
  12. Remove the previously created hadoop data folder using **sudo rm -rf /usr/local/hadoop/hadoop\_data** command.
  13. Make a new directory for datanodes using the **sudo mkdir -p /usr/local/hadoop/hadoop\_data/hdfs/datanode**
  14. Give appropriate rights to the newly created folde above by executing**sudo chown -R syed:syed /usr/local/hadoop command**. Close the instance now.
  15. Your master instance is ready to be cloned now. Create 3 clones from this instance as **hadoopmnslave1, hadoopmnslave2 and hadoopmnslave3.**
  16. Start **hadoopmnmaster** back now.  Now you need to add the master to the masters file. Open the masters file using **sudo nano /usr/local/hadoop/etc/hadoop/masters**command**.** Add **hadoopmnmaster** to it.
  17. Open the **hdfs-site.xml** and put the entry back for namenode. Remove the **hadoop\_data** folder from hadoop root
  18. Make a new directory for namenode using the **sudo mkdir -p /usr/local/hadoop/hadoop\_data/hdfs/namenode.**Give read and write rights to the hadoop folder using **sudo chown -R syed:syed /usr/local/hadoop**
  19. Open the slaves file using **sudo nano /usr/local/hadoop/etc/hadoop/slaves** command. Add **hadoopmnslave1, hadoopmnslave2, hadoopmnslave3** to it.Reboot the master server
  20. Startthe first slave server **hadoopmnslave1** now. Open the hostname file to change the name to  **hadoopmnslave1. Open the interfaces file to update the IP address as 192.168.56.11. Do the same for all the other slaves and reboot them.**
  21. Very important property to set up. Go to the mapred-site.xml file of all your master and slave file and make sure that every entry of **localhost** is replaced by **hadoopmnmaster.**
  22. Similarly, go to your yarn-site.xml of all your nodes and add the property **yarn.resourcemanager.hostname** and keep the value as the **hadoopmnmaster**. Replace all the occurrences of localhost / master node in this file with ${**yarn.resourcemanager.hostname}.** This will ensure that all your daemons are started properly on the masternode only.
  23. Ensure that passwordless SSH is happening between all the nodes. Fix the keys if it is not happening. Go through [this](http://www.tecmint.com/ssh-passwordless-login-using-ssh-keygen-in-5-easy-steps/) link to get the help.
  24. Go to **hadoopmnmaster** and execute **hstart** command. This should start the namenode and resource manager at the master instance and datanode and node manager on all the slave instances.
  25. Test your wordcount from the master machine. it should start the MRAppMaster on one slave and YARNChild on the other.
  26. Your multinode cluster is up and running !

Posted 29th January 2016 by [Syed](http://www.blogger.com/profile/14076951030391603242)

4

**View comments**

[Jan](http://syed-rizvi.blogspot.com/2016/01/hadoop-single-node-installation-on.html" \o "25th January 2016)

[25](http://syed-rizvi.blogspot.com/2016/01/hadoop-single-node-installation-on.html" \o "25th January 2016)

[**Hadoop Single Node Installation on Virtual Machine**](http://syed-rizvi.blogspot.com/2016/01/hadoop-single-node-installation-on.html)

Hi Friends,  
  
I just recently installed Hadoop Single Node instance on both VMWare Fusion and Oracle Virtual Box on my mac book (ElCapitan OS). So I thought I should document the steps here for everyones benefit. So there you go :-

* 1. Start with installed VM Virtualisation software of your choice (i.e Fusion or VirtualBox)
  2. Ensure that the latest version of Ubuntu LTS is installed on your VM software.  You can use link such as [this](http://www.wikihow.com/Install-Ubuntu-on-VirtualBox).
  3. Once installed, login to your Ubuntu installation with your credentials. Open a terminal command.
  4. Change the hostname to **hadoosn** in the hostname and **/etc/hostname** file. Given the IP to this VM as **192.168.56.9**. Make the change accordingly in **/etc/hosts** file.
  5. Enable the network adapters in the VM. Make the first adapter as "**Internal Network**" and second to "**NAT**". This will ensure that the internet access is working in your VMs.Reboot the VM for changed to take into effect.
  6. Execute **sudo apt-get update**command to update your Ubuntu installation
  7. Now is the time to make static IP. Open the **/etc/network/interfaces** file on the VMs and add the following :

           #The primary network interface

            auto eth0

                       iface eth0 inet static

                              address 192.168.56.9

                              netmask 255.255.255.0

            auto eth1

                       iface eth1 inet dhcp

* 1. Now to ensure that your ssh is working, check ssh immediately after the below commands by executing **whereis ssh** and **whereis sshd** commands. If any one of these commands return a blank response that means your SSH is not properly installed. Install it now the following 3 steps and check the ssh and sshd again:
     1. sudo apt-get remove openssh-client
     2. sudo apt-get update
     3. sudo apt-get install openssh-server
  2. Reboot once for all the above changes to take into effect.
  3. Now enable root access executing the following steps
     1. **sudo nano /etc/ssh/sshd\_config**
     2. comment line **PermitRootLogin without-password**
     3. Just below the commented line, add **PermitRootLogin yes**
     4. Restart ssh by executing **sudo service ssh restart**
  4. Change the root password by using the command “**sudo passwd root**”. Change the password to reflect a new password
  5. Log in to root user using “**su root**” command to check if the new password is working.
  6. Let create ssh keys on this host as follows :-
     1. Ensure that you are into your user log in (not root log in).
     2. Remove any keys that have been created on this VM before by executing **rm -rf .ssh/** command.
     3. Now let us first create the keys by executing **ssh-keygen -t rsa** command
     4. Now log into your **root** and just run the following command.
        + cat /home/syed/.ssh/id\_rsa.pub | ssh root@hadoopsn 'cat >> /root/.ssh/authorized\_keys'
     5. Now run the same on localhost
        + cat /home/syed/.ssh/id\_rsa.pub | ssh root@localhost 'cat >> /root/.ssh/authorized\_keys'
     6. This command will ensure that the .ssh folder is created in root location and also the public key created by the syed user is copied to the authorised\_keys file in the root location.
     7. Now you need to give the required right to .ssh folder and authorised\_keys file in ssh location. Execute **chmod 700 /root/.ssh, chmod 600 /root/.ssh/authorized\_keys**, **chmod 640 /root/.ssh/authorized\_keys**
     8. Now try to ssh in your root log in using **ssh root@hadoopsn**. You should be able to ssh in without password. Doing the same with syed user in root login will ask you for a password, which is fine. Doing the same with syed user in syed log will also ask for a password. This should also be fine.
     9. If you get “Agent admitted failure to sign using the key” issue, just run **ssh-add** from syed log in and you should be fine.
  7. Type the **sudo apt-get install openjdk-7-jdk** command to install java
  8. Type the java -version command to check if java is installed. This should return some valid version of Java 7 installed.
  9. Get latest version of hadoop binary using the **wget** [http://mirrors.sonic.net/apache/hadoop/common/hadoop-2.7.3/hadoop-2.7.3.tar.gz](http://mirrors.sonic.net/apache/hadoop/common/hadoop-2.7.1/hadoop-2.7.1.tar.gz) command.
  10. Extract the downloaded binary using the **tar -xvzf hadoop-2.7.3.tar.gz** command
  11. Copy the extracted folder to **/usr/local** directory using **sudo mv hadoop-2.7.3 /usr/local/hadoop** command.
  12. Go to **/usr/local** folder
  13. Give read and write permissions to hadoop folder using the following command **sudo chown -R <username>:<groupname>  /usr/local/hadoop.** So if you have created a user by the "syed", by default it will get created in "syed" group. Hence the command would look like **sudo chown -R syed:syed /usr/local/hadoop**
  14. Log into your root user. Open the bashrc file using the **sudo nano $HOME/.bashrc** command
  15. Add the following 3 lines to above bashrc file and save and quit

        export HADOOP\_HOME=/usr/local/hadoop  
      export JAVA\_HOME=/usr/lib/jvm/java-7-openjdk-amd64  
      export PATH=$PATH:$HADOOP\_HOME/bin

* 1. Use source $HOME/.bashrc command for the above changes to take into affect
  2. Execute hadoop version command to check if the latest version is installed. Process if you see the latest version is installed.
  3. Now create a temporary directory to save intermediate output emitted by the mappers as follows and give it required permissions

      sudo mkdir -p /usr/local/hadoop/tmp

      sudo chown <username> /usr/local/hadoop/tmp

      sudo chmod 750 /usr/local/hadoop/tmp

* 1. Open **/usr/local/hadoop/etc/hadoop/core-site.xml** add the following properties to it :-

         <configuration>

         <property>

         <name>hadoop.tmp.dir</name>

         <value>/usr/local/hadoop/tmp</value>

         <description>A base for other temporary directories.</description>

         </property>

         <property>

         <name>fs.defaultFS</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>

* 1. Open  **/usr/local/hadoop/etc/hadoop/yarn-site.xml** add the following properties to it :-

 <configuration>  
<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>  
<property>  
<name>yarn.resourcemanager.address</name>  
<value>localhost:9003</value>  
</property>  
<property>  
<name>yarn.resourcemanager.resource-tracker.address</name>  
<value>localhost:9001</value>  
</property>  
<property>  
<name>yarn.resourcemanager.scheduler.address</name>  
<value>localhost:9002</value>  
</property>  
<property>  
<name>yarn.log-aggregation-enable</name>  
<value>true</value>  
</property>  
</configuration>

* 1. Open  **/usr/local/hadoop/etc/hadoop/hdfs-site.xml** add the following properties to it :-

         <configuration>

         <property>

         <name>dfs.replication</name>

         <value>1</value>

         <description>default replication factor for the cluster.

         </description>

         </property>

         <property>

         <name>dfs.namenode.name.dir</name>

         <value>file:/usr/local/hadoop/hadoop\_data/hdfs/namenode</value>

         <description>default replication factor for the cluster.

         </description>

         </property>

         <property>

         <name>dfs.datanode.data.dir</name>

         <value>file:/usr/local/hadoop/hadoop\_data/hdfs/datanode</value>

         <description>default replication factor for the cluster.

         </description>

         </property>  
         </configuration>

* 1. Open  **/usr/local/hadoop/etc/hadoop/mapred-site.xml** add the following properties to it :-

<configuration>  
<property>  
<name>mapreduce.framework.name</name>  
<value>yarn</value>  
</property>  
<property>   
<name>mapreduce.jobhistory.address</name>  
<value>localhost:10020</value>   
</property>  
<property>   
<name>mapreduce.jobhistory.webapp.address</name>  
<value>localhost:19888</value>   
</property>  
</configuration>

* 1. Make the directories **mkdir -p /usr/local/hadoop/hadoop\_data/hdfs/namenode** and **mkdir -p /usr/local/hadoop/hadoop\_data/hdfs/datanode** for the above configuration changes.
  2. Give these newly created directories the proper right using **sudo chown username:usergroup -R /usr/local/hadoop** command. For mac set ups, run **sudo chown -R syedrizvi:staff /usr/local/Cellar/hadoop** to give the write access to the folders. Do a **sudo chmod 750** to **/usr/local/hadoop** folder if the namenode does not start the first time.
  3. Use the **hdfs namenode –format** command to format the namenode. Make sure that the dfs folder inside **/usr/local/hadoop/** has the proper access after this command. Run the chown command again if the access is not there.
  4. Now log into your root user. Use the **hadoop/sbin/start-dfs.sh** command to start HDFS Services
  5. Use the **hadoop/sbin/start-yarn.sh** command to start YARN Services
  6. Set JAVA\_HOME explicitly if hadoop-env.sh if start up gives messages with respect to it.
  7. Check if all the services are running using jps command.
  8. Create your aliases as below. This will ensure that you can start and stop the cluster in one shot

alias hstart="/usr/local/hadoop/sbin/start-dfs.sh;/usr/local/hadoop/sbin/start-yarn.sh"  
alias hstop="/usr/local/hadoop/sbin/stop-yarn.sh;/usr/local/hadoop/sbin/stop-dfs.sh"

* 1. Once the services are running, your single node hadoop server is up and running with the latest version.

Posted 25th January 2016 by [Syed](http://www.blogger.com/profile/14076951030391603242)

21

**View comments**

[Jan](http://syed-rizvi.blogspot.com/2016/01/yarn-phases-part-i.html" \o "4th January 2016)

[4](http://syed-rizvi.blogspot.com/2016/01/yarn-phases-part-i.html" \o "4th January 2016)

[**YARN Phases - Part I**](http://syed-rizvi.blogspot.com/2016/01/yarn-phases-part-i.html)

It is very important to understand the various phases of YARN Map Reduce execution flow to get a detailed understanding of how a YARN job is executed .

Following is the list of phases which a typical YARN Map Reduce job undergoes :

**Job Submission**

As soon as the job is submitted at the client, the submit() method of the **Job** object creates a **JobSubmitter** object and call it's submitJobInternal() method. Further on, the job submission process implemented by the JobSubmitter object undergoes the following steps :-

* + Asks the resource manager for a new application id, used as a Map Reduce Job Id identifier throughout the job.
  + Checks if the output directory is already created and if it already exists, then the error is thrown and the job is not submitted. This feature however can be overwritten programatically in the Driver class code, used at the job development phase.
  + Computes the abstraction of the input splits for the job. This information will be later on used to ascertain the number of mappers to be used for the job
  + Copies the necessary resources for the job like job jar file, configuration files, the input split information the the HDFS system with high replication factor
  + Submits the job by calling submitApplication() method on the Resource Manager.

**Job Initialisation**  
  
As soon as the Resource Manage receives a call to its submitApplication() method, it delegates that request to the YARN scheduler. The following series of steps happen after that :-

* + The YARN scheduler gets in touch with the NodeManager daemon on an available slave machines and allocates a container there. The NodeManager now launches / initialises the container and then the Application Master for the job submitted.
  + The Application Master is identified as java application with main class as **MRAppMaster.** This class immediately initialises the job by creating a number of book keeping objects to keep track of the job progress.
  + The App Master now reads inputs splits saved by the client in the Job Submission step. This information is used to create Map Object for each split and Reduce object as per the job setting. Not that the actual Mappers and Reducers are not initialised at this point in time. Only the tasks corresponding to them are given the ids.
  + An important decision which the App Master has to take now is how to run the job. If the job is small (i.e less than 10 mappers and a reducer), the App Master may decide to run it within the same JVM as itself. This is done to avoid the overhead of running the tasks in new containers. Such jobs are called **Uberized** jobs.
  + Lastly, the App Master called the setUpJob() method on the **OutputCommitter** object to create the output directory where the output of the job will be dumped.

**Task Assignment**

Once the job has been kicked off at the App Master level, the task assignment happen in the following manner :-

* + The App Master now requests resources / containers from Resource Manager (actually YARN Scheduler), now that it knows how many mappers (equal to number of splits) and reducers need to be run.
  + Map Tasks are given priority as sort phase of mapper needs to be complete before Reducer can start. Requests for Reduce tasks are not made until at least 5 % of the Map Tasks are complete. Furthermore, while reduce tasks can run anywhere on the cluster, the map tasks have to be data local as long as possible.

**Task Execution**

Now that the containers are assigned for all the mappers on the slaves nodes on the cluster, the task execution is about the start. It happens in the following steps :-

* + The tasks (Mappers or Reducers of the job) run on a specialised Java JVM application called YARNChild. An important point in note about YARNChild is that it runs in its own JVM, separate from the JVM of Node Manager. This is done to ensure that tasks related bugs occurring in the YARNChild does not interfere with the workings on Node Manager.
  + Before the task can run in the YARNChild, it localises resources that the tasks need like configuration and property files and archives like JAR files. It also localises resources from Distributed Cache, if at all any resources are set up in Distributed Cache.
  + Each tasks perform the set up and commit action, which is taken care by **OutputCommitter.** This class ensures that either the intermediate output if written to the disk (in case of mappers) or it it finally written to HDFS (in case of reducer)

**Job Progress & Status Update**  
Since Map Reduce jobs are long running job, its important for users to get the feedback on how the job is running. So lets looks at some of the important steps in a job progress and its related status :-

* + A job and each of its task have a status which include things such as state of the job or task (running, successfully or failed), the progress of map and reduces, the values of Job's Counters and a status message and description. These statuses keep on getting communicated back to the client during the course of the Job
  + For the status of Map Tasks,  the proportion of input is considered as progress and communicated accordingly.
  + For Reduce Tasks, the total progress is divided into 3 parts as according to the 3 phases of shuffle and Sort Mechanism. For e.g. if the task has run the reducer half it's input, the tasks progress will be 5/6 as it has completed copy and sort phases (1/3) and half way through the reduce phase.
  + Tasks also have a set of counters that counts various events as task runs. The counters are either build into the framework or defined by the users
  + As the tasks run, the child process start communicating to it's parent process.  This progress is communicated to the respective **Application Master**.
  + The **Resource Manager** takes this status from the **Application Masters**of all the running jobs and starts displaying it on it's User Interface.

**Job Completion**  
It's job completion in the end.

Posted 4th January 2016 by [Syed](http://www.blogger.com/profile/14076951030391603242)

0

**Add a comment**

Loading