How to Setup Hadoop Multi Node Cluster - Step By Step

Written by DWBI Official　　Last Updated: 11 September 2016

Setting up Hadoop in a single machine is easy, but no fun. Why? Because Hadoop is not meant for a single machine. Hadoop is meant to run on a computing cluster comprising of many machines. Running HDFS and MapReduce on a single machine is great for learning about these systems, but to do useful work we need to run Hadoop on multiple nodes. There are a few options when it comes to staring a Hadoop cluster, from building our own to running on rented hardware, or using any offering that provides Hadoop as a service in the cloud. But, how can we - the learners, the beginners, the amateurs - take advantage of multi-node Hadoop cluster? Well, allow us to show you.

Got Nodes?

If you don't have multiple machines to create your cluster, you can provision them from cloud

To create a multi-node cluster we will need - well - multiple nodes, that is multiple machines. As long as you have access to few spare linux machines - you are fine. But in case you don't - it's ludicrously easily to get a few machines up and running in cloud within minutes.

We will assume that you don't actually have any machine other than the computer you are reading this article on. And we will show you how you can still create your very own Hadoop cluster with multiple machines. On the other hand, if you actually happen to own a few spare machines (at least 3), you don't need to follow the immediate next part, just note down the IP Addresses of your machine and follow us from the next heading.

Provisioning Nodes from Cloud

We are going to use a very simple, fast, reliable and outrageously popular cloud service - Digital Ocean (https://www.digitalocean.com/?refcode=029607f4dfd0). The moment you start using it, you fall in love with it. No bells, no whistles. Simple machines for developers. Ok, here's what you do -

- Go to Digital Ocean (https://www.digitalocean.com/?refcode=029607f4dfd0) now. Sign-up and create your account. Don't worry, if you access Digital Ocean using this Link (https://www.digitalocean.com/?refcode=029607f4dfd0), you will get $10 credit - that's enough for running 2 machines free-of-charge for a whole month.
- Logon to your account and click on "Create Droplet" (they call the machines droplet)
- Choose Ubuntu 16.04.1 x64 bit as the image for the droplet
- Choose a size of your choice (we are using 2GB /40GB Disk)
- Select a datacenter region. Remember that create all the droplets in the same region for droplet to droplet private communication.
- In the additional option, select "Private Networking"
- Enter the droplet hostname as "NameNode".
- Keep other things as default and click "Create"

Once our droplet is ready, you will receive an email with the Static IP address of your new droplet along with password for the root user account.

Name Node versus Data Node

There will be two types of nodes in a Hadoop cluster - NameNode and DataNode. If you had installed Hadoop in a single machine, you could have installed both of them in a single computer, but in a multi-node cluster they are usually on different machines. In our cluster, we will have one name node and multiple data nodes. DataNodes store the actual data of Hadoop, while the NameNode stores the metadata information.

Cluster Topology

We will build our clusters with 3 machines, 2 of which will be used as DataNode while the remaining one will be used as NameNode. The below picture illustrates the network topology along with the IP addresses:

Creating the NameNode

Now once your 1st machine is ready in cloud (or locally, if you had spares) note down the IP address of the machine that you want to configure as the NameNode. In my case, the IP address is 10.0.0.1. Make sure to change this IP addresses according to your IP address in the below commands:

Open up your terminal and connect to the droplet using SSH. (If you are on Windows, you may use Putty for this purpose).

```
$> ssh root@10.0.0.1
```

- ### Install Java Environment

  Hadoop requires Java as pre requisite. Let us update the system & install Oracle Java 1.7. During the installation, if it prompts for a confirmation, press `Y`.

  ```
  root@NameNode:~# apt-get update
  root@NameNode:~# add-apt-repository ppa:webupd8team/java
  root@NameNode:~# apt-get update
  root@NameNode:~# apt-get install oracle-java7-installer
  root@NameNode:~# java -version

  #Output looks like below
  java version "1.7.0_80"
  Java(TM) SE Runtime Environment (build 1.7.0_80-b15)
  Java HotSpot(TM) 64-Bit Server VM (build 24.80-b11, mixed mode)
  ```

- ### Setup machine alias in host file

  Next we need to modify the host file to put the alias of the machine name. In `/etc/hosts` file delete everything and put the below two lines. In case if you are using Digital Ocean put the Private IP address of the droplet, not the public IPv4 address, so as to facilitate private networking i.e. droplet to droplet communication within Digital Ocean. * Make sure to change this IP addresses according to your IP address.

  ```
  root@NameNode:~# vi /etc/hosts
  ```

  /etc/hosts

  ```
  10.0.0.1 NameNode
  ```

- ### Setup SSH Server

  The hadoop control scripts rely on SSH to perform cluster-wide operations. For example, there is a script for stopping and starting all the daemons in the clusters. To work seamlessly, SSH needs to be setup to allow password-less & passphrase-less login for the root/hadoop user from machines in the cluster. The simplest way to achieve this is to generate a public/private key pair, and it will be shared across the cluster.

```
root@NameNode:~# apt-get install openssh-server
```

Generate an SSH key for the root user

```
root@NameNode:~# ssh-keygen -t rsa -P ""
```

If prompted for SSH key file name, Enter file in which to save the key (`/root/.ssh/id_rsa`) and press ENTER. Next put the key to `authorized_keys` directory for future password-less access.

```
root@NameNode:~# cat /root/.ssh/id_rsa.pub >> /root/.ssh/authorized_keys
root@NameNode:~# chmod 700 ~/.ssh
root@NameNode:~# chmod 600 ~/.ssh/authorized_keys
```

Connect and validate ssh password-less login to localhost. If you are being prompted to accept the connection, select yes.

```
root@NameNode:~# ssh localhost
```

Once okay, type exit to come out of the localhost.

- ## Download and Install Hadoop distribution

  Check first the Latest Stable Hadoop Release Available at: Apache Hadoop (http://www-us.apache.org/dist/hadoop/common/stable2/). In the time of writing this article, Hadoop 2.7.2 is the latest stable version. We will install it under `/usr/local/` directory. After that we will also create few additional directories like `namenode` for hadoop to store all namenode information and `namesecondary` to store the checkpoint images.

```
root@NameNode:~# cd /usr/local/
root@NameNode:/usr/local/# wget http://www.us.apache.org/dist/hadoop/common/hadoop-2
root@NameNode:/usr/local/# tar -xzvf hadoop-2.7.2.tar.gz >> /dev/null
root@NameNode:/usr/local/# mv hadoop-2.7.2 /usr/local/hadoop
root@NameNode:/usr/local/# mkdir -p /usr/local/hadoop_work/hdfs/namenode
root@NameNode:/usr/local/# mkdir -p /usr/local/hadoop_work/hdfs/namesecondary
```

- ## Setup Environment Variables

  We will setup some environment variables in `.bashrc` so that every time we restart our machines, it knows where to find Java or Hadoop installation location inside the machine. To do this, first we need to find out where JAVA has been installed. If you have followed this tutorial, it's likely that Java has been installed within one of the subdirectories under `/usr/lib/jvm/` directory. So, please browse to this location, and confirm that Java is there:

```
root@NameNode:/usr/local/# cd /usr/lib/jvm/java-7-oracle/jre
root@NameNode:/usr/lib/jvm/java-7-oracle/jre# java -version
```

If the above command runs fine, then Java is there inside the `default-java` directory.

Now open `.bashrc`

```
root@NameNode:/usr/lib/jvm/java-7-oracle/jre# vi ~/.bashrc
```

And put these lines at the end of your `.bashrc` file (Press `SHIFT + G` to directly go to the end of the file):

```
export JAVA_HOME=/usr/lib/jvm/java-7-oracle/jre
export PATH=$PATH:$JAVA_HOME/bin
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
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 CLASSPATH=$CLASSPATH:/usr/local/hadoop/lib/*:.

export HADOOP_OPTS="$HADOOP_OPTS -Djava.security.egd=file:/dev/../dev/urandom"
```

Save the `.bashrc` file and source it like below:

```
root@NameNode:/usr/lib/jvm/java-7-oracle/jre# source ~/.bashrc
```

- ### Setup `JAVA_HOME` under hadoop environment

  It is suggested that you also setup the `JAVA_HOME` environment variable under Hadoop environment file. Open the `hadoop-end.sh` file,

  ```
  root@NameNode:/usr/lib/jvm/java-7-oracle/jre# vi /usr/local/hadoop/etc/hadoop/hadoop
  ```

  Inside the file, find the line `export JAVA_HOME=${JAVA_HOME}`. Replace the line like below

  ```
  export JAVA_HOME=/usr/lib/jvm/java-7-oracle/jre
  ```

- ### Confirm Hadoop is installed

  At this point, we should confirm if `hadoop` command is accessible from the terminal

  ```
  root@NameNode:/usr/lib/jvm/java-7-oracle/jre# cd $HADOOP_HOME/etc/hadoop
  root@NameNode:/usr/local/hadoop/etc/hadoop# hadoop version

  #Output looks like below
  Hadoop 2.7.2
  Subversion https://git-wip-us.apache.org/repos/asf/hadoop.git -r b165c4fe8a74265c792
  Compiled by jenkins on 2016-01-26T00:08Z
  Compiled with protoc 2.5.0
  From source with checksum d0fda26633fa762bff87ec759ebe689c
  This command was run using /usr/local/hadoop/share/hadoop/common/hadoop-common-2.7.2
  ```

Configuring the NameNode

Hadoop makes use of some XML based configuration file where it reads all the runtime parameters from. These configuration files are located under `/usr/local/hadoop/etc/hadoop` folder. We will configure/set some minimal options just to get us started on the Hadoop cluster. In this configuration we will use YARN as the cluster management framework.

Configure `core-site.xml`

This XML configuration file lets you setup site specific properties, such as I/O settings that are common to HDFS and MapReduce. Open the file and put the following properties:

```
root@NameNode:/usr/local/hadoop/etc/hadoop# vi core-site.xml
```

core-site.xml

```
<?xml version="1.0"?>
<!-- core-site.xml -->
<configuration>
<property>
<name>fs.defaultFS</name>
<value>hdfs://NameNode:8020/</value>
</property>
<property>
<name>io.file.buffer.size</name>
<value>131072</value>
</property>
</configuration>
```

Configure `hdfs-site.xml`

We will tell hadoop where is the name node directory (which we created previously at the end of Hadoop installation) and how many backup copies of the data files to be created in the system (called replication) inside this file under the `configuration` tag.

```
root@NameNode:/usr/local/hadoop/etc/hadoop# vi hdfs-site.xml
```

hdfs-site.xml

```
<?xml version="1.0"?>
<!-- hdfs-site.xml -->
<configuration>
<property>
<name>dfs.namenode.name.dir</name>
<value>file:/usr/local/hadoop_work/hdfs/namenode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name>
<value>file:/usr/local/hadoop_work/hdfs/datanode</value>
</property>
<property>
<name>dfs.namenode.checkpoint.dir</name>
<value>file:/usr/local/hadoop_work/hdfs/namesecondary</value>
</property>
<property>
<name>dfs.replication</name>
<value>2</value>
</property>
<property>
<name>dfs.block.size</name>
<value>134217728</value>
</property>
</configuration>
```

Configure `mapred-site.xml`

This controls the configuration settings for MapReduce daemons. Here we need to ensure that we will be using YARN framework. Also we will configure the MapReduce Job History server. Copy the template file `mapred-site.xml.template`:

```
root@NameNode:/usr/local/hadoop/etc/hadoop# cp mapred-site.xml.template mapred-site.xml
root@NameNode:/usr/local/hadoop/etc/hadoop# vi mapred-site.xml
```

Then add the following properties

mapred-site.xml

```
<?xml version="1.0"?>
<!-- mapred-site.xml -->
<configuration>
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
<property>
<name>mapreduce.jobhistory.address</name>
<value>NameNode:10020</value>
</property>
<property>
<name>mapreduce.jobhistory.webapp.address</name>
<value>NameNode:19888</value>
</property>
<property>
<name>yarn.app.mapreduce.am.staging-dir</name>
<value>/user/app</value>
</property>
<property>
<name>mapred.child.java.opts</name>
<value>-Djava.security.egd=file:/dev/../dev/urandom</value>
</property>
</configuration>
```

Configure `yarn-site.xml`

This XML configuration file lets you setup YARN site specific properties for Resource Manager & Node Manager. Open the file:

```
root@NameNode:/usr/local/hadoop/etc/hadoop# vi yarn-site.xml
```

Then put the following properties under `configuration`:

yarn-site.xml

```
<?xml version="1.0"?>
<!-- yarn-site.xml -->
<configuration>
<property>
<name>yarn.resourcemanager.hostname</name>
<value>NameNode</value>
</property>
<property>
<name>yarn.resourcemanager.bind-host</name>
<value>0.0.0.0</value>
</property>
<property>
<name>yarn.nodemanager.bind-host</name>
<value>0.0.0.0</value>
</property>
<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.log-aggregation-enable</name>
<value>true</value>
</property>
<property>
<name>yarn.nodemanager.local-dirs</name>
<value>file:/usr/local/hadoop_work/yarn/local</value>
</property>
<property>
<name>yarn.nodemanager.log-dirs</name>
<value>file:/usr/local/hadoop_work/yarn/log</value>
</property>
<property>
<name>yarn.nodemanager.remote-app-log-dir</name>
<value>hdfs://NameNode:8020/var/log/hadoop-yarn/apps</value>
</property>
</configuration>
```

Setup the Master

Now you need to tell Hadoop NameNode the hostname of Secondary name node. In our case both NameNode & Secondary NameNode resides on the same machine. We do it by editing the `masters` file:

```
root@NameNode:/usr/local/hadoop/etc/hadoop# vi masters
```

Inside the file, put one line:

```
NameNode
```

Next, we will format the name node before starting anything now,

```
root@NameNode:/usr/local/hadoop/etc/hadoop# /usr/local/hadoop/bin/hadoop namenode -format
```

You should see a success command saying "Storage directory /usr/local/hadoop_work/hdfs/namenode has been successfully formatted"

Creating the Data Nodes

Next, we need to create the datanodes now. If you are using DigitalOcean, login and create two droplets with the minimal configurations as shown below (do not forget to check the private networking option on). If you are using your own network cluster, make sure you have two spare machines ready with hostnames as `DataNode1` and `DataNode2` .

Adding the Data Nodes in Master

Now, at this point we have our master up and running and we have 2 data nodes machines ready as well. The hostnames and the IP addresses of all these machines are known to us (Your IPs will be different than mine). Again note if you are using DigitalOcean put the Private IP address not public IPv4 address.

Node	Hostname	IP
Name Node	NameNode	10.0.0.1
Data Node	DataNode1	10.0.100.1
Data Node	DataNode2	10.0.100.2

Next, we need to tell the master about these new data nodes.

```
root@NameNode:/usr/local/hadoop/etc/hadoop# vi slaves
```

Inside the file, remove everything and put the lines as below. Note: If you have more DataNodes, you have to according list their entries in the slaves file, so that the master is aware of the respective slaves:

```
DataNode1
DataNode2
```

Also add and append these hostnames in the `/etc/hosts` file of the master node by adding the following entries. Note use the Private IP address of the DataNodes accordingly.

```
root@NameNode:/usr/local/hadoop/etc/hadoop# vi /etc/hosts
```

```
10.0.100.1 DataNode1
10.0.100.2 DataNode2
```

Configuring the DataNodes

Next we need to login to the data nodes and perform the following tasks in each of the data nodes

1. Installing JAVA in data node
2. Updating `/etc/hosts`
3. Environment variable configuration
4. Hadoop installation

Remember that you need to do the above 4 tasks in all of your data nodes. Below, we are showing them for the first node. Login to your first data node machine and perform these tasks

```
$> ssh root@10.0.100 (mailto:root@10.0.100).1
```

Installing JAVA in DataNode

```
root@DataNode1:~# apt-get update
root@DataNode1:~# add-apt-repository ppa:webupd8team/java
root@DataNode1:~# apt-get update
root@DataNode1:~# apt-get install oracle-java7-installer
```

Updating `/etc/hosts`

Open the `/etc/hosts` file

```
root@DataNode1:~# vi /etc/hosts
```

Remove the contents of the file (if any), and add the following lines. Remember to put proper Private IP address of your nodes/droplets

```
10.0.0.1 NameNode
10.0.100.1 DataNode1
10.0.100.2 DataNode2
```

Environment variable configuration

Open the `.bashrc` file,

```
DataNode1# vi ~/.bashrc
```

Then append the following lines at the end of the file

```
export JAVA_HOME=/usr/lib/jvm/java-7-oracle/jre
export PATH=$PATH:$JAVA_HOME/bin
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
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 CLASSPATH=$CLASSPATH:/usr/local/hadoop/lib/*:.

export HADOOP_OPTS="$HADOOP_OPTS -Djava.security.egd=file:/dev/../dev/urandom"
```

Hadoop Installation

This time we will not install Hadoop from beginning as we have already installed it in the NameNode. Instead, we will just copy the installation directories from the NameNode to the DataNodes. In order for us to copy that, we will first need to include the public key of the NameNode to the authorised keys of all the data nodes so that we can leverage SSH file copy program (`scp`) for copying.

Login to NameNode, and copy the contents of public key in clipboard

```
root@NameNode:/usr/local/hadoop/etc/hadoop# cat ~/.ssh/id_rsa.pub

#Output Public Key looks something like below
ssh-rss AA21....
....l5Mq4r root@NameNode
```

Once you have copied the above key (from the beginning ssh-rsa to the end of the file), login to the DataNodes one by one and paste the key in the authorised keys file. During the same time, also create a data node and yarn directories in the individual data nodes. They will be used by Datanode & Node manager daemons respectively.

```
root@DataNode1:~# vi ~/.ssh/authorized_keys
(append the copied key at the end of this file)
root@DataNode1:~# mkdir -p /usr/local/hadoop_work/hdfs/datanode
root@DataNode1:~# mkdir -p /usr/local/hadoop_work/yarn/local
root@DataNode1:~# mkdir -p /usr/local/hadoop_work/yarn/log
```

(remember to run the above code block in all the datanodes one by one)

Once done the NameNode can now communicate with the DataNodes via password less ssh.

Connect and validate ssh password-less login to all the Datanodes from the NameNode. If you are being prompted to accept the connection, select yes.

```
root@NameNode:/usr/local/hadoop/etc/hadoop# ssh DataNode1
```

Once okay, type exit come out of the remote host.

Now it's time to copy the hadoop installation i.e. the binaries as well as the our site specific configuration files from NameNode to DataNode. Login to NameNode and run the below command:

```
root@NameNode:/usr/local/hadoop/etc/hadoop# cd /usr/local
root@NameNode:/usr/local# scp -r hadoop DataNode1:/usr/local
root@NameNode:/usr/local# scp -r hadoop DataNode2:/usr/local
```

Note: You have to do the same for any additional DataNodes if any. For our case we have only two data nodes.

It's time to start the hadoop distributed file system in the cluster first from the NameNode. If prompted for authentication, select yes.

```
root@NameNode:/usr/local# $HADOOP_HOME/sbin/start-dfs.sh
```

Let us first check the daemons running in the NameNode as well as the DataNodes in the Hadoop cluster. Ideally you should see NameNode & Secondary Name node started as java processes in NameNode and DataNode java process in DataNode's. Login to the individual nodes and check the running java processes using `jps`

```
root@NameNode:/usr/local# jps

root@DataNode1:~# jps

root@DataNode2:~# jps
```

Once the Namenode & Datanodes starts successfully, we have to create few directories in hadoop filesystem which has been listed in our site specific configuration files. This HDFS directories will be used by YARN Map Reduce Staging, YARN Log & Job History Server.

```
root@NameNode:/usr/local# hadoop fs -mkdir /tmp
root@NameNode:/usr/local# hadoop fs -chmod -R 1777 /tmp

root@NameNode:/usr/local# hadoop fs -mkdir /user
root@NameNode:/usr/local# hadoop fs -chmod -R 1777 /user
root@NameNode:/usr/local# hadoop fs -mkdir /user/app
root@NameNode:/usr/local# hadoop fs -chmod -R 1777 /user/app

root@NameNode:/usr/local# hadoop fs -mkdir -p /var/log/hadoop-yarn
root@NameNode:/usr/local# hadoop fs -chmod -R 1777 /var/log/hadoop-yarn
root@NameNode:/usr/local# hadoop fs -mkdir -p /var/log/hadoop-yarn/apps
root@NameNode:/usr/local# hadoop fs -chmod -R 1777 /var/log/hadoop-yarn/apps


# Now Verify the HDFS File Structure
root@NameNode:/usr/local# hadoop fs -ls -R /

# Output should look like below:
drwxrwxrwt - root supergroup 0 /tmp
drwxr-xr-x - root supergroup 0 /user
drwxrwxrwt - root supergroup 0 /user/app
drwxr-xr-x - root supergroup 0 /var
drwxr-xr-x - root supergroup 0 /var/log
drwxr-xr-x - root supergroup 0 /var/log/hadoop-yarn
drwxr-xr-x - root supergroup 0 /var/log/hadoop-yarn/apps
```

Now we need to start the YARN cluster framework. We should execute the below command from the Node hosting the Resource Manager. In our case the Resource Manager is in the same NameNode.

```
root@NameNode:/usr/local# $HADOOP_HOME/sbin/start-yarn.sh
```

Now we will start the MapReduce History Server. We should execute the below command from the Node hosting the History Server. In our case the History Server is in the same NameNode. Before that quickly edit the $HADOOP_HOME/etc/hadoop/mapred-site.xml file in the NameNode. Replace the hostname in the value for the property names as below from NameNode to 0.0.0.0:
mapreduce.jobhistory.address from NameNode:10020 to 0.0.0.0:10020.
mapreduce.jobhistory.webapp.address from NameNode:19888 to 0.0.0.0:19888

```
root@NameNode:/usr/local# $HADOOP_HOME/sbin/mr-jobhistory-daemon.sh start historyserver
```

Lets check again the daemons running in the NameNode as well as the DataNodes in the Hadoop cluster. Ideally you should see NameNode, Secondary NameNode & ResourceManager started as java processes in NameNode; And DataNode & NodeManager java process in DataNode's. Login to the individual nodes and check the running java processes using `jps`

```
root@NameNode:/usr/local# jps

# Output should look like below:
3619 NameNode
3864 SecondaryNameNode
4041 ResourceManager
4240 JobHistoryServer

root@DataNode1:~# jps
# Output should look like below:
2650 DataNode
2787 NodeManager

root@DataNode2:~# jps
# Output should look like below:
2613 DataNode
2749 NodeManager
```

Great the cluster is up & running. Time to test the hadoop file system by uploading some dummy data file in HDFS. From the NameNode we issue the below commands

```
root@NameNode:/usr/local# hadoop fs -mkdir /analysis
root@NameNode:/usr/local# hadoop fs -ls /
```

Create a small data file and try to load it in HDFS cluster.

```
root@NameNode:/usr/local# echo "Some, dummy, file" > /root/dummy.csv
root@NameNode:/usr/local# hadoop fs -put /root/dummy.csv /analysis/dummy.csv
root@NameNode:/usr/local# hadoop fs -ls /analysis
root@NameNode:/usr/local# hadoop fs -tail /analysis/dummy.csv
```

Finally check the cluster status in the web browser. Remember to use the Public IP Address of the NameNode, ResourceManager & HistoryServer respectively. In our case all are in the NameNode machine with IP 10.0.0.1 Go to -

`http://10.0.0.1:50070`

`http://10.0.0.1:8088`

`http://10.0.0.1:19888`

Let's check the hdfs file system, replication etc from the web browser by going to the `http://10.0.0.1:50070/explorer.html`

Next we will run a sample hadoop map-reduce utility on our hadoop YARN cluster.

```
root@NameNode:/usr/local# hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/hadoop-mapr
```

In the next article, we will setup a client node (//dwbi.org/etl/bigdata/187-set-up-client-node-gateway-node-in-hadoop-cluster) (also called Edge Node or Gateway node), followed by Hive and Sqoop installation.

Prev (/etl/bigdata/186-hadoop-mapreduce-basics)   Next (/etl/bigdata/181-hadoop-architecture)

Can you answer this?

What does 'E' in 'ETL' stands for?

○ Extraction

◯ Elimination
◯ Entry
◯ Evacuation

Submit

Popular

- Top 20 SQL Interview Questions with Answers (/database/sql/72-top-20-sql-interview-questions-with-answers)
- Best Informatica Interview Questions & Answers (/etl/informatica/131-important-practical-interview-questions)
- Top 50 Data Warehousing/Analytics Interview Questions and Answers (/data-modelling/dimensional-model/58-top-50-dwbi-interview-questions-with-answers)
- Top 50 DWBI Interview Questions with Answers - Part 2 (/data-modelling/dimensional-model/59-top-50-dwbi-interview-questions-with-answers-part-2)
- Performance Considerations for Dimensional Modeling (/data-modelling/dimensional-model/22-performance-considerations-for-dimensional-modeling)
- Top 30 BusinessObjects interview questions (BO) with Answers (/analysis/business-objects/69-top-businessobjects-interview-questions)

Have a question on this subject?

Ask questions to our expert community members and clear your doubts. Asking question or engaging in technical discussion is both easy and rewarding.

Ask a Question, we'll Answer

Are you on Twitter?

Start following us. This way we will always keep you updated with what's happening in Data Analytics community. We won't spam you. Promise.

Follow @dwbic (https://twitter.com/dwbic)

- Apache Hadoop Architecture (/etl/bigdata/181-hadoop-architecture)
  In this article we will learn about the Apache Hadoop framework architecture. The basic components of the Apache Hadoop HDFS & MapReduce (/analysis/data-mining/178-map-reduce) engine are discussed in brief.
- Oracle Installation for SQOOP Import (/etl/bigdata/193-oracle-installation-for-sqoop-import)
  We would like to perform practical test of Apache SQOOP import/export utility between ORACLE relational database & Apache HADOOP file system, let us quickly setup an ORACLE server. For that we will be using cloud based services/servers as we did...
- Fools Guide to Big data - What is Big Data (/etl/bigdata/177-what-is-big-data)
  Sure enough, you have heard the term, "Big Data" many times before. There is no dearth of

information in the Internet and printed medium about this. But guess what, this term still remains vaguely defined and poorly understood. This essay is our...

- Install PIG In Client Node of Hadoop Cluster (/etl/bigdata/198-install-pig-in-client-node-of-hadoop-cluster)
  Apache Pig is a platform for analyzing large data sets. Pig Latin is the high level programming language that, lets us specify a sequence of data transformations such as merging data sets, filtering them, grouping them, and applying functions to...
- Hadoop MapReduce Basics (/etl/bigdata/186-hadoop-mapreduce-basics)
  The Hadoop, since its inception is changing the way the enterprises store, process and analyse data. MapReduce is the core part of the Hadoop framework and we can also call it as the core processing engine of Hadoop. It is a programming model...
- Install SPARK in Hadoop Cluster (/etl/bigdata/201-install-spark-in-hadoop-cluster)
  Apache Spark is a fast and general purpose engine for large-scale data processing over a distributed cluster. Apache Spark has an advanced DAG execution engine that supports cyclic data flow and in-memory computing. Spark run programs up to 100x...
- Set up Client Node (Gateway Node) in Hadoop Cluster (/etl/bigdata/187-set-up-client-node-gateway-node-in-hadoop-cluster)
  Once we have our multi-node hadoop cluster up and running (/etl/bigdata/183-setup-hadoop-cluster), let us create an EdgeNode or a GatewayNode. Gateway nodes are the interface between the Hadoop cluster and the outside network. Edge nodes are used to run client applications and cluster...
- Install SQOOP in Client Node of Hadoop Cluster (/etl/bigdata/191-install-sqoop-in-client-node-of-hadoop-cluster)
  Sqoop is an open source software product of the Apache Software Foundation in the hadoop ecosystem, designed to transfer data between Hadoop and relational databases or mainframes. Sqoop can be used to import data from a relational database...
- How to Setup Hadoop Multi Node Cluster - Step By Step (/etl/bigdata/183-setup-hadoop-cluster)
  Setting up Hadoop in a single machine is easy, but no fun. Why? Because Hadoop is not meant for a single machine. Hadoop is meant to run on a computing cluster comprising of many machines. Running HDFS and MapReduce on a single machine is great for...
- SQOOP Merge & Incremental Extraction from Oracle (/etl/bigdata/195-sqoop-merge-incremental-extraction-from-oracle)
  Let us check how to perform Incremental Extraction & Merge using Sqoop. The SQOOP Merge utility allows to combine two datasets where entries in one dataset should overwrite entries of an older dataset. For example, an incremental import run in...

(https://www.facebook.com/datawarehousing) (https://twitter.com/dwbiconcepts) (https://www.linkedin.com/company/dwbiconcepts) (https://www.youtube.com/dwbiconcepts) (https://plus.google.com/b/105042632846858744029)

Security

Secure Site Dec-13-2017 (https://www.beyondsecurity.com/vulnerability-scanner-verification/dwbi.org)