

MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE RUSSIAN FEDERATION
FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION OF HIGHER EDUCATION
NATIONAL UNIVERSITY OF SCIENCE AND TECHNOLOGY

«MISIS»

Informatics and Computer Technologies

Major code 09.04.01

Course work

In Big Data and complex socio-technical systems

Set Up a Hadoop 3.2.1 Multi-Node Cluster on Centos 7

Student: Asmerom Fessehaye

Group: МИБТ-19-7-12А

Checked by: Konov I.S.

Grade:

Moscow

2020

Contents

Introduction to Hadoop	3
Installing Hadoop	4
Pre-requirements	4
Step1:- Disable firewall dan SELinux in All machines	5
Step2:- Network and Connectivity configuration in All machines	5
Step3:- creating a new Hadoop user in all machines	6
Step4:- Installing and configuring SSH in All machines	7
Step5:- Installing Java in all machines	9
Step5:- Downloading and Installing Hadoop in master node	10
Step6:- Adding environment variables for Hadoop 3.2.1	10
Configuring Hadoop	11
Step1:- Create data directories with permissions in all machines	11
Step2:- Configuration in master node	11
Configure Hadoop Core Site <i>core-site.xml</i>	11
Configure HDFS <i>hdfs-site.xml</i>	12
Configure <i>mapred-site.xml</i>	12
Configure <i>yarn-site.xml</i>	12
Configure Worker (slave / data node) workers	12
Step3:- Installation of Hadoop in data node data nodes	13
Copy configured Hadoop from master node to data nodes	13
Configure Hadoop Yarn Site <i>yarn-site.xml</i>	13
Testing multi node cluster	13
Installation and Configuration of HBase	15
Step 1: Download and Install HBase	15
Step 2: Configure HBase	15
Step 3: Managing HMaster & HRegionServer	18
ISCSI configuration and access	20
Basics of iSCSI:	20
Installing iSCSI Client Tools and connecting to storage:	23
Install <i>targetcli</i> and Creating LUNS	28
References	30

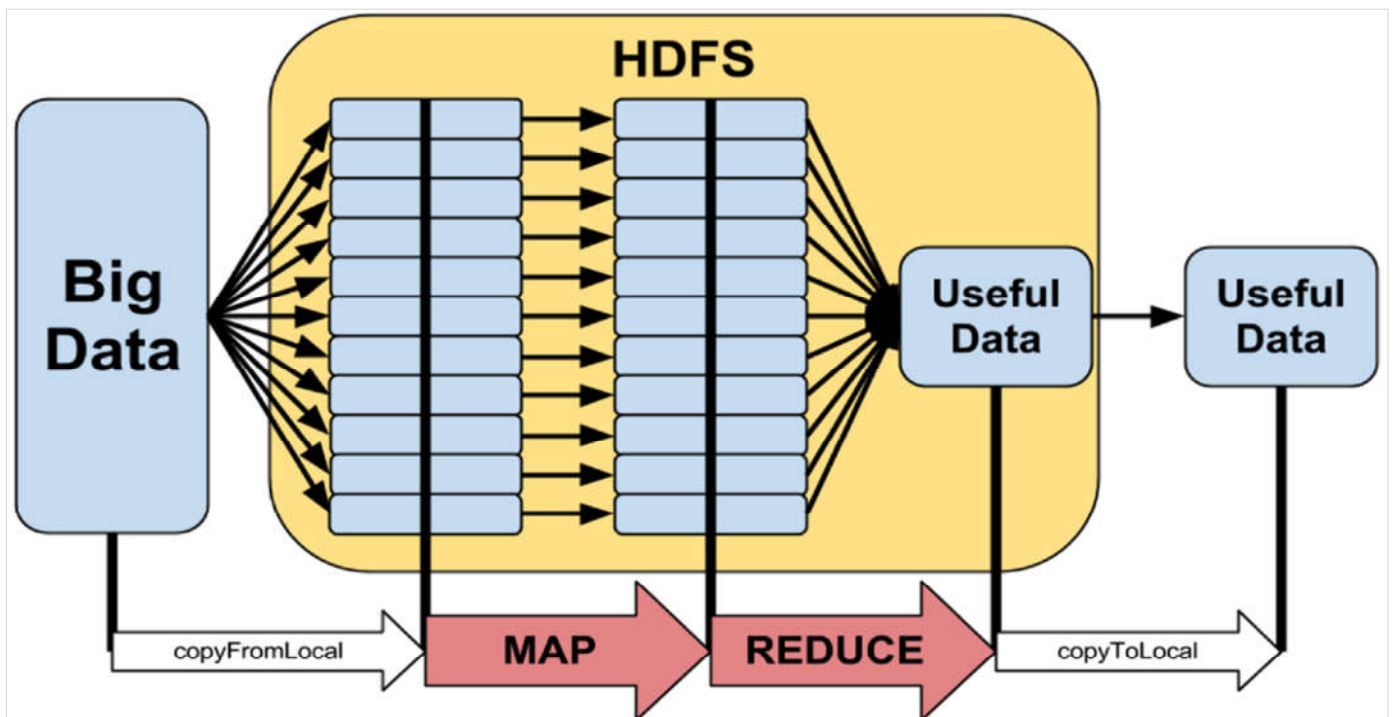


Introduction to Hadoop

Hadoop is “open-source software for reliable, scalable, distributed computing”, this means that is a framework for distributed processing. It’s simply several computers working together in order to improve processing power and assure decentralize operations. With Hadoop, it’s possible to process large data sets across computers without complex coding.

This software is use by a large number of companies that look for processing enormous datasets (also known as big data). Companies like Facebook, Amazon or Netflix depend on the processing capacity of Hadoop to have an efficient data analytics system.

What are Hadoop components?



- **Hadoop Common** — contains libraries and utilities needed by other Hadoop modules;
- **Hadoop Distributed File System (HDFS)** — a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster;
- **Hadoop YARN** — (introduced in 2012) a platform responsible for managing computing resources in clusters and using them for scheduling users' applications;
- **Hadoop MapReduce** — an implementation of the MapReduce programming model for large-scale data processing. It gives a scalability component to the system. The data is split and process in parallel. This allows operations to be executed in a faster way.

Installing Hadoop

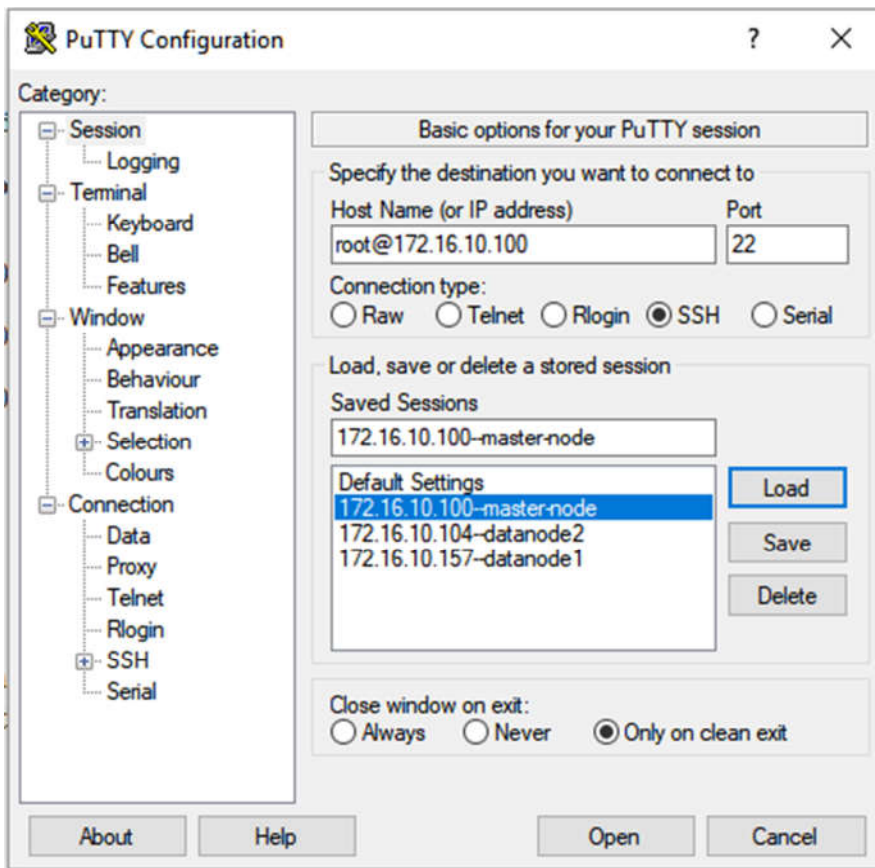
Pre-requirements

- Centos 7 installed on a virtual machine. For this demonstration we use three nodes where the one node is the master (**Name node**) and others are slaves (**Data nodes**). All the virtual machines must be networked.
 1. 172.16.10.100--master-node
 2. 172.16.10.157--datanode1
 3. 172.16.10.104--datanode2

In order to create the Hadoop Multi-Node Cluster we need to install

- Java 8;
- SSH;

For connecting to the Virtual machines we can use PuTTY, which makes it easy to connect to multiple virtual machines at the same time.



Step1:- Disable firewall dan SELinux in All machines

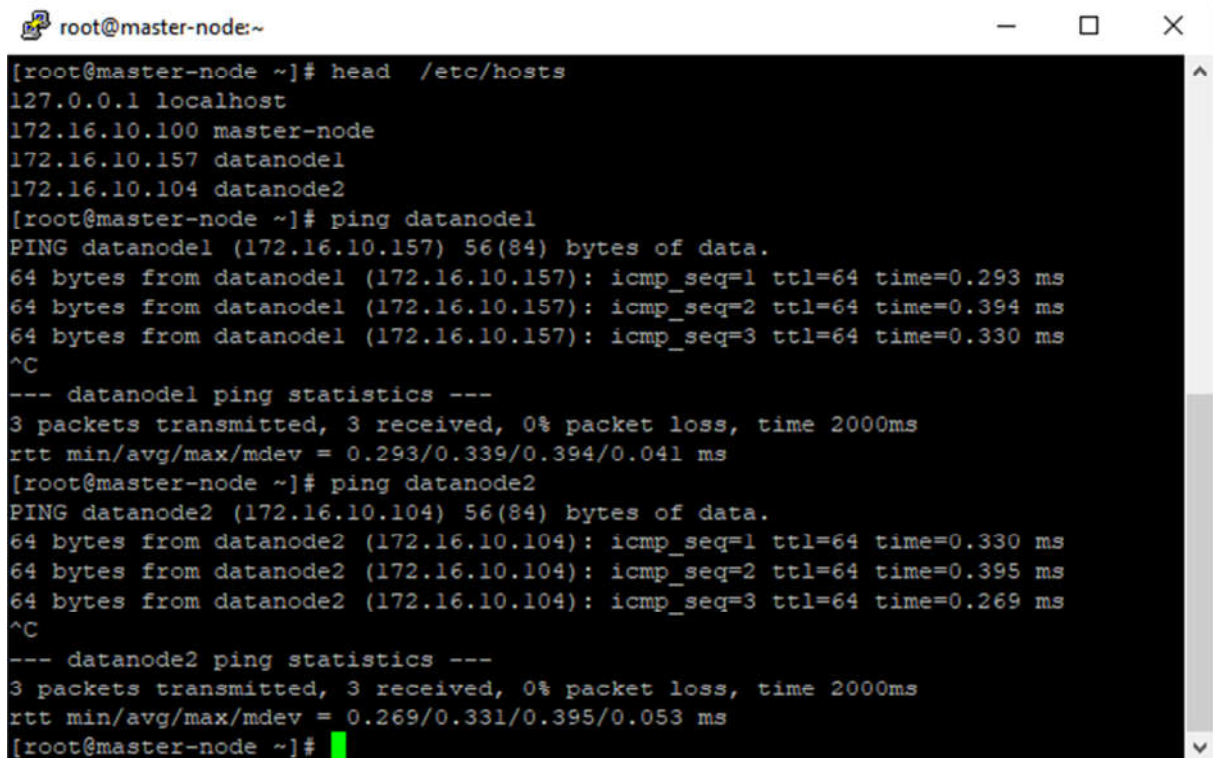
```
sudo systemctl disable --now firewalld
sudo setenforce 0
sudo sed -i 's/^SELINUX=.*SELINUX=permissive/g' /etc/selinux/config
cat /etc/selinux/config | grep SELINUX= | grep -v '#'Buat user hadoop
```

Step2:- Network and Connectivity configuration in All machines

With root user edit /etc/hosts file and add the IP and hostname of every machine. This is very useful because we don't need to remember IP address of every machine. We can use the host name which is user friendly name.

```
$ sudo hostnamectl set-hostname master-node/datanode1/datanode2...
$ sudo nano /etc/hosts
172.16.10.100 master-node
172.16.10.157 datanode1
172.16.10.104 datanode2
```

After setting up the hostnames of every node we can test the connectivity by using ping command with the hostname. You should get a successful ping results as show in screenshot below.



```
root@master-node:~  
[root@master-node ~]# head /etc/hosts  
127.0.0.1 localhost  
172.16.10.100 master-node  
172.16.10.157 datanode1  
172.16.10.104 datanode2  
[root@master-node ~]# ping datanode1  
PING datanode1 (172.16.10.157) 56(84) bytes of data.  
64 bytes from datanode1 (172.16.10.157): icmp_seq=1 ttl=64 time=0.293 ms  
64 bytes from datanode1 (172.16.10.157): icmp_seq=2 ttl=64 time=0.394 ms  
64 bytes from datanode1 (172.16.10.157): icmp_seq=3 ttl=64 time=0.330 ms  
^C  
--- datanode1 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2000ms  
rtt min/avg/max/mdev = 0.293/0.339/0.394/0.041 ms  
[root@master-node ~]# ping datanode2  
PING datanode2 (172.16.10.104) 56(84) bytes of data.  
64 bytes from datanode2 (172.16.10.104): icmp_seq=1 ttl=64 time=0.330 ms  
64 bytes from datanode2 (172.16.10.104): icmp_seq=2 ttl=64 time=0.395 ms  
64 bytes from datanode2 (172.16.10.104): icmp_seq=3 ttl=64 time=0.269 ms  
^C  
--- datanode2 ping statistics ---  
3 packets transmitted, 3 received, 0% packet loss, time 2000ms  
rtt min/avg/max/mdev = 0.269/0.331/0.395/0.053 ms  
[root@master-node ~]#
```

You should get the same results in every machine (Data nodes) to make sure that all the nodes are networked and can communicate with each other.

Step3:- creating a new Hadoop user in all machines

We need a separate account for installation of Hadoop which is to be created in every machine.

```
$ sudo useradd hadoop  
$ sudo usermod -aG wheel hadoop  
$ sudo passwd hadoop # set password hadoop
```

Step4:- Installing and configuring SSH in All machines

The Hadoop core uses Shell (SSH) to launch the server processes on the slave nodes. It requires password-less SSH connection between the master and all the slaves and the secondary machines.

We need a password-less SSH in a Fully-Distributed environment because when the cluster is LIVE and running in Fully Distributed environment, the communication is too frequent. The job tracker should be able to send a task to task tracker quickly.

Note that the password less SSH must be set only between the master node and data nodes (slaves), but not necessarily between the slaves.

When a Hadoop cluster is built, there are slave nodes and master nodes. Master node controls the tasks on the slave nodes. Every node is a different system and to maintain a connection between these nodes, SSH is used. SSH is mainly used so that the master node can stay connected with the slave nodes.

Command on the Master-node:

```
$ sudo apt install ssh
$ sudo su - hadoop
$ ssh-keygen -t rsa
$ ssh-copy-id hadoop@master-node
$ ssh-copy-id hadoop@datanode1
$ ssh-copy-id hadoop@datanode2
```

Command on the Datanode1:

```
$ sudo apt install ssh
$ sudo su - hadoop
$ ssh-keygen -t rsa
$ ssh-copy-id hadoop@datanode1
$ ssh-copy-id hadoop@master-node
```

Command on the Datanode2:

```
$ sudo apt install ssh
$ sudo su - hadoop
$ ssh-keygen -t rsa
$ ssh-copy-id hadoop@datanode2
$ ssh-copy-id hadoop@master-node
```

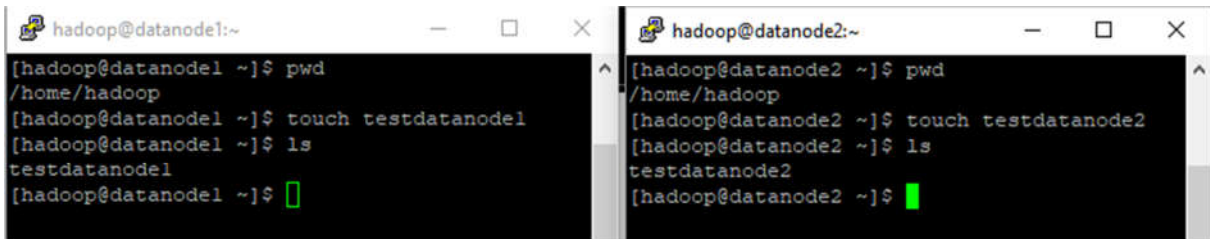
Here are screenshots of setting SSH in master node

```
hadoop@master-node:~  
[hadoop@master-node ~]$ ssh-keygen -t rsa  
Generating public/private rsa key pair.  
Enter file in which to save the key (/home/hadoop/.ssh/id_rsa):  
Created directory '/home/hadoop/.ssh'.  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /home/hadoop/.ssh/id_rsa.  
Your public key has been saved in /home/hadoop/.ssh/id_rsa.pub.  
The key fingerprint is:  
SHA256:qa03vU0IBmH7gpx2xcGxF7E25CS0d6+L7Q03NoIX3ko hadoop@master-node  
The key's randomart image is:  
+---[RSA 2048]-----+  
|      oo=.+.      |  
|      .+.O o      |  
|      o = B .      |  
|      . o + = o .   |  
|      = o S . .     |  
|      . . = . + +   |  
|      . . . o E *    |  
|      . o . B O o    |  
|      . . . oo* .    |  
+-----[SHA256]-----+
```

```
hadoop@master-node:~  
[hadoop@master-node ~]$ ssh-copy-id hadoop@master-node  
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/hadoop/.ssh/id_rsa.pub"  
The authenticity of host 'master-node (172.16.10.100)' can't be established.  
ECDSA key fingerprint is SHA256:lCjblLiRwNv5RsnP5cIZJYKOJfm3iPiJtw0P+qCNAHk.  
ECDSA key fingerprint is MD5:73:e4:aa:95:a6:d9:f7:e4:8d:85:ce:ff:d9:c3:eb:6e.  
Are you sure you want to continue connecting (yes/no)? yes  
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed  
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys  
hadoop@master-node's password:  
  
Number of key(s) added: 1  
  
Now try logging into the machine, with: "ssh 'hadoop@master-node'"  
and check to make sure that only the key(s) you wanted were added.
```

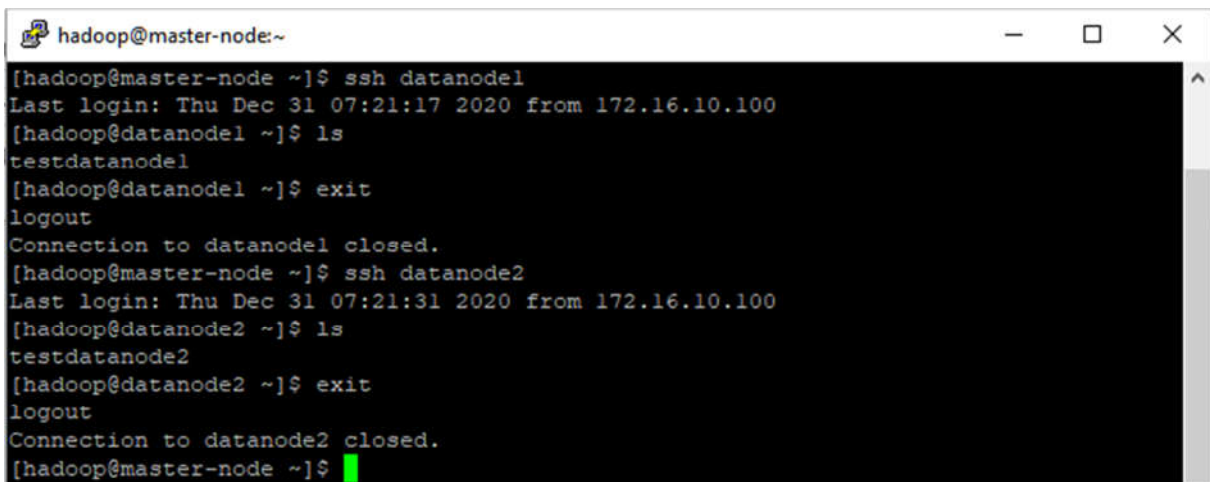
```
hadoop@master-node:~  
[hadoop@master-node ~]$ ssh-copy-id hadoop@datanode1  
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/hadoop/.ssh/id_rsa.pub"  
The authenticity of host 'datanode1 (172.16.10.157)' can't be established.  
ECDSA key fingerprint is SHA256:lCjblLiRwNv5RsnP5cIZJYKOJfm3iPiJtw0P+qCNAHk.  
ECDSA key fingerprint is MD5:73:e4:aa:95:a6:d9:f7:e4:8d:85:ce:ff:d9:c3:eb:6e.  
Are you sure you want to continue connecting (yes/no)? yes  
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed  
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys  
hadoop@datanode1's password:  
  
Number of key(s) added: 1  
  
Now try logging into the machine, with: "ssh 'hadoop@datanode1'"  
and check to make sure that only the key(s) you wanted were added.
```


After performing all the steps in all machines you should be able to login to the nodes using password less SSH here is screenshot from Master node. Here are the contents in the home directory of the data nodes.



```
hadoop@datanode1:~  
[hadoop@datanode1 ~]$ pwd  
/home/hadoop  
[hadoop@datanode1 ~]$ touch testdatanode1  
[hadoop@datanode1 ~]$ ls  
testdatanode1  
[hadoop@datanode1 ~]$  
  
hadoop@datanode2:~  
[hadoop@datanode2 ~]$ pwd  
/home/hadoop  
[hadoop@datanode2 ~]$ touch testdatanode2  
[hadoop@datanode2 ~]$ ls  
testdatanode2  
[hadoop@datanode2 ~]$
```

As we can see from the screenshot below we can login using password less SSH from the master node to the data nodes.



```
hadoop@master-node:~  
[hadoop@master-node ~]$ ssh datanode1  
Last login: Thu Dec 31 07:21:17 2020 from 172.16.10.100  
[hadoop@datanode1 ~]$ ls  
testdatanode1  
[hadoop@datanode1 ~]$ exit  
logout  
Connection to datanode1 closed.  
[hadoop@master-node ~]$ ssh datanode2  
Last login: Thu Dec 31 07:21:31 2020 from 172.16.10.100  
[hadoop@datanode2 ~]$ ls  
testdatanode2  
[hadoop@datanode2 ~]$ exit  
logout  
Connection to datanode2 closed.  
[hadoop@master-node ~]$
```

Step5:- Installing Java in all machines

Install Java 8 in all machines

```
$ sudo yum -y install vim wget curl bash-completion
```

```
$ sudo yum install -y epel-release java-1.8.0-openjdk
```

Add java environment variable **/etc/profile.d/hadoop_java.sh**

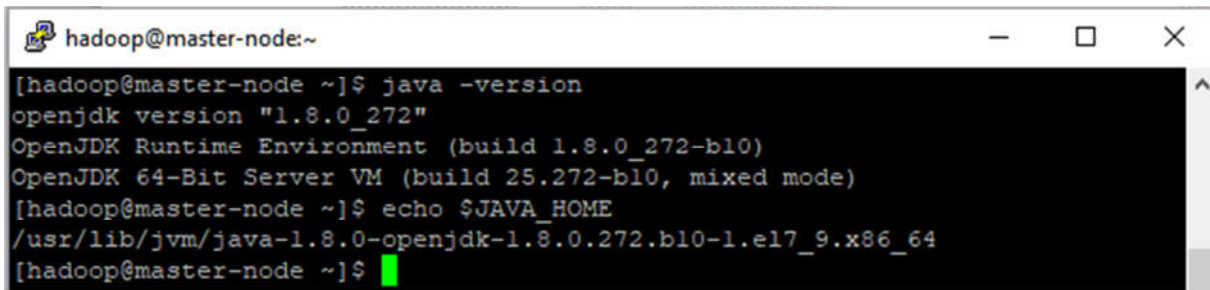
```
$ vi /etc/profile.d/hadoop java.sh
```

```
export JAVA_HOME=$(dirname $(dirname $(readlink $(readlink $(which javac)))))
```

```
$ source /etc/profile.d/hadoop_java.sh
```

```
$ echo $JAVA_HOME
```

You should get the following result after successful installation of java

A terminal window titled 'hadoop@master-node:~' with standard window controls. It shows the output of 'java -version' as 'openjdk version "1.8.0_272" OpenJDK Runtime Environment (build 1.8.0_272-b10) OpenJDK 64-Bit Server VM (build 25.272-b10, mixed mode)'. Below that, 'echo \$JAVA_HOME' returns '/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.272.b10-1.el7_9.x86_64'. The prompt is ready for the next command.

```
hadoop@master-node:~  
[hadoop@master-node ~]$ java -version  
openjdk version "1.8.0_272"  
OpenJDK Runtime Environment (build 1.8.0_272-b10)  
OpenJDK 64-Bit Server VM (build 25.272-b10, mixed mode)  
[hadoop@master-node ~]$ echo $JAVA_HOME  
/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.272.b10-1.el7_9.x86_64  
[hadoop@master-node ~]$
```

Step5:- Downloading and Installing Hadoop in master node

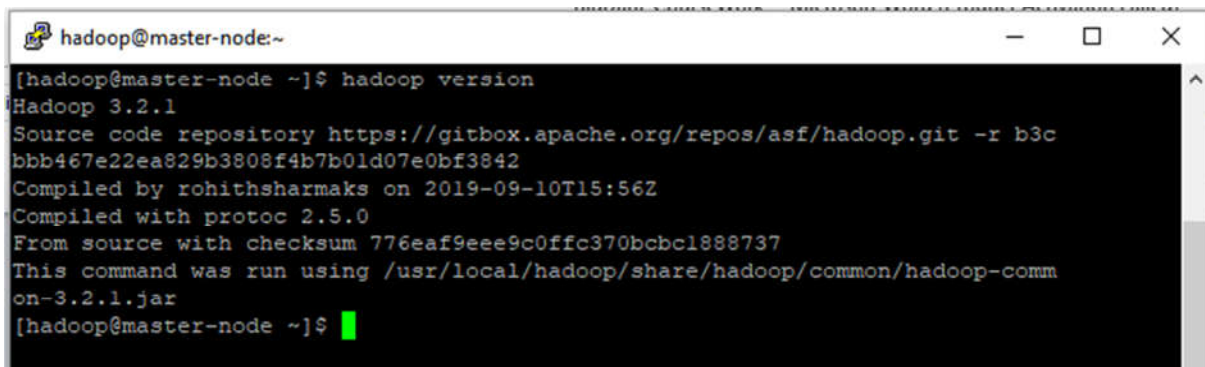
Download hadoop-3.2.1 on the master node

```
$ cd ~/
$ wget https://www-us.apache.org/dist/hadoop/common/hadoop-3.2.1/hadoop-3.2.1.tar.gz
$ tar -zxvf hadoop-3.2.1.tar.gz
$ sudo mv hadoop-3.2.1 /usr/local/hadoop
$ sudo chown -R hadoop:hadoop /usr/local/hadoop
$ sudo chmod -R g+rxw /usr/local/hadoop
```

Step6:- Adding environment variables for Hadoop 3.2.1

Add Hadoop environment variables /etc/profile.d/hadoop_java.sh. This should be in every node after installing Hadoop. In data nodes we will not install Hadoop again, we will just copy the configured one from master node to all slaves.

```
$ vi /etc/profile.d/hadoop_java.sh
export JAVA_HOME=$(dirname $(dirname $(readlink $(readlink $(which javac)))))
export HADOOP_HOME=/usr/local/hadoop
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_COMMON_LIB_NATIVE_DIR"
export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin:$JAVA_HOME/bin
$ source /etc/profile.d/hadoop_java.sh
```

A terminal window titled 'hadoop@master-node:~' showing the output of the 'hadoop version' command. The output displays Hadoop 3.2.1 details, including the source code repository URL, commit hash, compilation date, and the command used to run the version check.

```
hadoop@master-node:~$ hadoop version
Hadoop 3.2.1
Source code repository https://gitbox.apache.org/repos/asf/hadoop.git -r b3c
bbb467e22ea829b3808f4b7b01d07e0bf3842
Compiled by rohithsharmaks on 2019-09-10T15:56Z
Compiled with protoc 2.5.0
From source with checksum 776eaf9eee9c0ffc370bcbcl888737
This command was run using /usr/local/hadoop/share/hadoop/common/hadoop-comm
on-3.2.1.jar
hadoop@master-node:~$
```

Configuring Hadoop

Step1:- Create data directories with permissions in all machines

```
sudo mkdir -p /usr/local/hadoop/data/namenode
sudo mkdir -p /usr/local/hadoop/data/datanode
sudo chown -R hadoopuser:hadoopgroup /usr/local/hadoop/data/namenode
sudo chmod -R 777 /usr/local/hadoop/data/namenode
sudo chown -R hadoopuser:hadoopgroup /usr/local/hadoop/data/datanode
sudo chmod -R 777 /usr/local/hadoop/data/datanode
hadoopuser=hadoop and hadoopgroup=hadoop
```

Step2:- Configuration in master node

Configure Hadoop Core Site *core-site.xml*

```
$ cd $HADOOP_HOME/etc/hadoop
$ vi core-site.xml # core-site.xml

<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://master-node:9000</value>
  </property>
  <property>
    <name>dfs.permissions</name>
    <value>>false</value>
  </property>
</configuration>
```

Configure HDFS hdfs-site.xml

```
$ cd $HADOOP_HOME/etc/hadoop
$ vi hdfs-site.xml

<configuration>
<property>
<name>dfs.namenode.name.dir</name><value>/usr/local/hadoop/data/nameNode</value>
</property>
<property>
<name>dfs.datanode.data.dir</name><value>/usr/local/hadoop/data/dataNode</value>
</property>
<property>
<name>dfs.replication</name>
<value>2</value>
</property>
</configuration>
```

Configure mapred-site.xml

```
$ vi mapred-site.xml
<property>
<name>mapreduce.framework.name</name>
<value>yarn</value>
</property>
```

Configure yarn-site.xml

```
$ vi yarn-site.xml
<configuration>
<property>
<name>yarn.nodemanager.auxservices</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.auxservices.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
</configuration>
```

Configure Worker (slave / data node) workers

```
$ vi workers
datanode1
datanode2
```

Step3:- Installation of Hadoop in data node data nodes

Copy configured Hadoop from master node to data nodes

```
$ sudo scp -R /usr/local/hadoop datanode1:/usr/local/hadoop
```

```
$ sudo scp -R /usr/local/hadoop datanode2:/usr/local/hadoop
```

Then in each node and give permission to the Hadoop installation directory

```
$ sudo chown -R hadoop:hadoop /usr/local/hadoop
```

```
$ sudo chmod -R g+rxw /usr/local/hadoop
```

Configure Hadoop Yarn Site yarn-site.xml

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

Testing multi node cluster

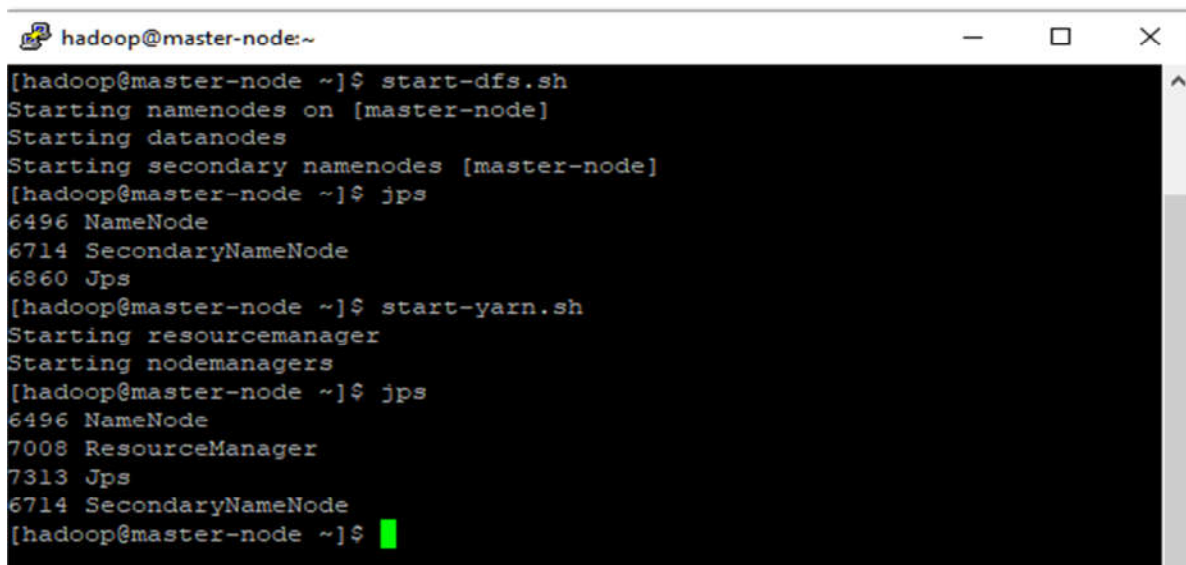
On the master node run the following commands

➤ Format HDFS filesystem

```
$ hdfs namenode -format
```

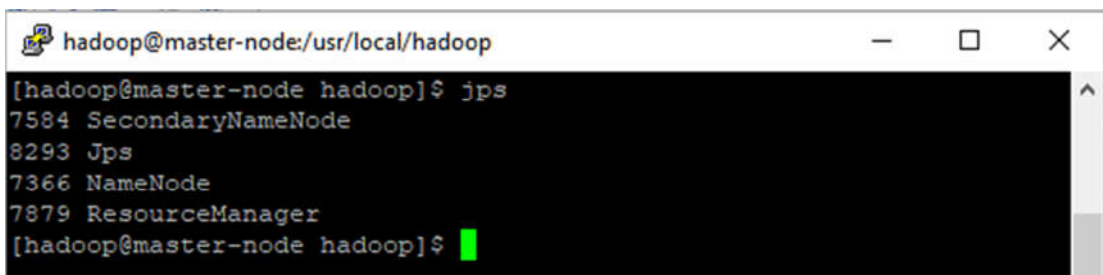
```
$ start-dfs.sh
```

```
$ start-yarn.sh
```

A terminal window titled 'hadoop@master-node:~' showing the execution of Hadoop startup scripts. The output of 'start-dfs.sh' shows the NameNode and SecondaryNameNode starting. The output of 'start-yarn.sh' shows the ResourceManager and Jps starting. The terminal text is as follows:

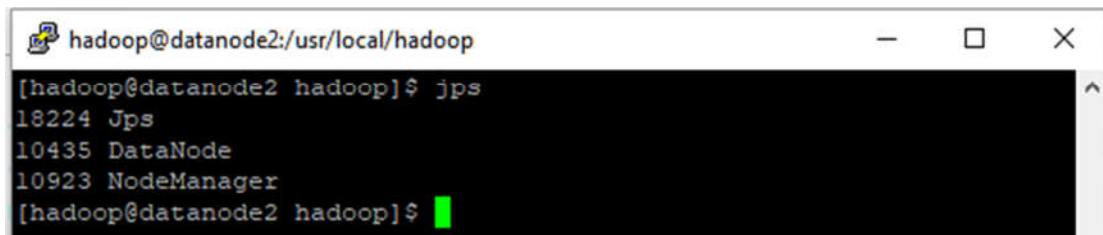
```
hadoop@master-node:~$ start-dfs.sh
Starting namenodes on [master-node]
Starting datanodes
Starting secondary namenodes [master-node]
hadoop@master-node:~$ jps
6496 NameNode
6714 SecondaryNameNode
6860 Jps
hadoop@master-node:~$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
hadoop@master-node:~$ jps
6496 NameNode
7008 ResourceManager
7313 Jps
6714 SecondaryNameNode
hadoop@master-node:~$
```

Check Hadoop processes on name node using JPS (which means Java Process Status)

A terminal window titled 'hadoop@master-node:/usr/local/hadoop' showing the output of the 'jps' command. The output lists four processes: SecondaryNameNode (PID 7584), Jps (PID 8293), NameNode (PID 7366), and ResourceManager (PID 7879).

```
hadoop@master-node:/usr/local/hadoop
[hadoop@master-node hadoop]$ jps
7584 SecondaryNameNode
8293 Jps
7366 NameNode
7879 ResourceManager
[hadoop@master-node hadoop]$
```

Check Hadoop processes on data nodes node using JPS (which means Java Process Status)

A terminal window titled 'hadoop@datanode2:/usr/local/hadoop' showing the output of the 'jps' command. The output lists three processes: Jps (PID 18224), DataNode (PID 10435), and NodeManager (PID 10923).

```
hadoop@datanode2:/usr/local/hadoop
[hadoop@datanode2 hadoop]$ jps
18224 Jps
10435 DataNode
10923 NodeManager
[hadoop@datanode2 hadoop]$
```

Web Interfaces

<http://172.16.10.100:9870/dfshealth.html#tab-overview>

Once the Hadoop cluster is up and running check the web-ui of the components as described below:

Daemon	Web Interface	Notes
NameNode	http://172.16.10.100:9870/	Default HTTP port is 9870.
ResourceManager	http://172.16.10.100:8088/	Default HTTP port is 8088.
DataNode	http://172.16.10.157:9864/	Default HTTP port is 9864.

Installation and Configuration of HBase

Step 1: Download and Install HBase

```
VER="2.2.6"
```

```
wget http://apache.mirror.gtcomm.net/hbase/stable/hbase-$VER-bin.tar.gz
```

Extract Hbase archive downloaded.

```
tar xvf hbase-$VER-bin.tar.gz
```

```
sudo mv hbase-$VER/ /usr/local/HBase/
```

Add Hbase environment variables `hadoop_java.sh` file

```
sudo nano /etc/profile.d/hadoop_java.sh
```

```
export HBASE_HOME=/usr/local/HBase
```

```
export
```

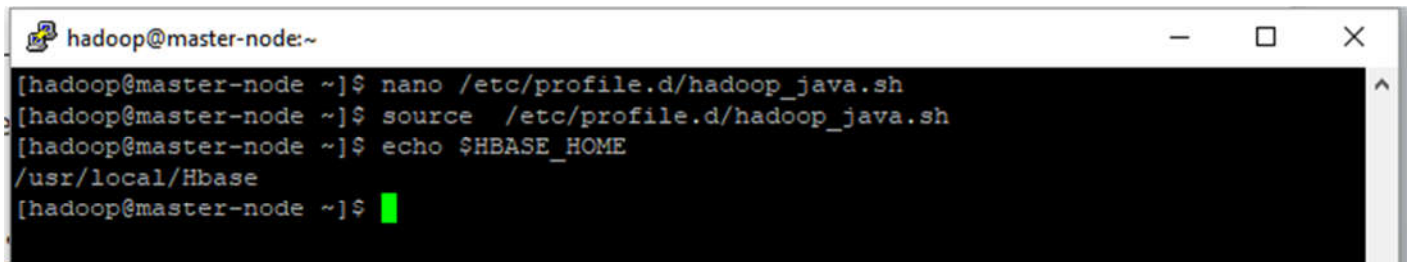
```
PATH=$PATH:$JAVA_HOME/bin:$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$HBASE_HOME/bin
```

Update your shell environment values.

```
$ source /etc/profile.d/hadoop_java.sh
```

```
$ echo $HBASE_HOME
```

```
/usr/local/HBase
```

A terminal window titled 'hadoop@master-node:~' with standard window controls. The terminal shows the following commands and output:

```
[hadoop@master-node ~]$ nano /etc/profile.d/hadoop_java.sh
[hadoop@master-node ~]$ source /etc/profile.d/hadoop_java.sh
[hadoop@master-node ~]$ echo $HBASE_HOME
/usr/local/Hbase
[hadoop@master-node ~]$
```

Edit `JAVA_HOME` in shell script `hbase-env.sh`

```
$ sudo vim /usr/local/HBase/conf/hbase-env.sh
```

```
# Set JAVA_HOME - Line 27
```

```
export JAVA_HOME=$(dirname $(dirname $(readlink $(readlink $(which javac)))))
```

Step 2: Configure HBase

We will do configurations like we did for Hadoop. All configuration files for HBase are located on `/usr/local/HBase/conf/` directory.

Option 1: Install HBase in Standalone Mode (Not recommended)

Option 2: Install HBase in Pseudo-Distributed Mode (Recommended) we will follow Option 2.

The value of **hbase.rootdir** set earlier will start in Standalone Mode. Pseudo-distributed mode means that HBase still runs completely on a single host, but each HBase daemon (HMaster, HRegionServer, and ZooKeeper) runs as a separate process.

Pseudo-distributed mode differs from *standalone* mode in that each of the component processes run in a separate JVM. It differs from *distributed mode* in that each of the separate processes run on the same server, rather than multiple servers in a cluster. This section also assumes you want to store your HBase data in HDFS rather than on the local filesystem.

Create HBase root directory.

```
$ sudo mkdir -p /hadoop/zookeeper
$ sudo chown -R hadoop:hadoop /hadoop/
```

To install HBase in Pseudo-Distributed Mode, set the value of **hbase.rootdir** in hbase-site.xml

```
sudo nano /usr/local/HBase/conf/hbase-site.xml

<configuration>

  <property>

    <name>hbase.rootdir</name>

    <value>hdfs://localhost:8030/hbase</value>

  </property>

  <property>

    <name>hbase.zookeeper.property.dataDir</name>

    <value>/hadoop/zookeeper</value>

  </property>

  <property>

    <name>hbase.cluster.distributed</name>

    <value>true</value>  </property></configuration>
```


Start hdfs

Creating the /hbase Directory in HDFS

Before starting the HBase Master, you need to create the /hbase directory in HDFS. The HBase master runs as hbase:hbase so it does not have the required permissions to create a top level directory.

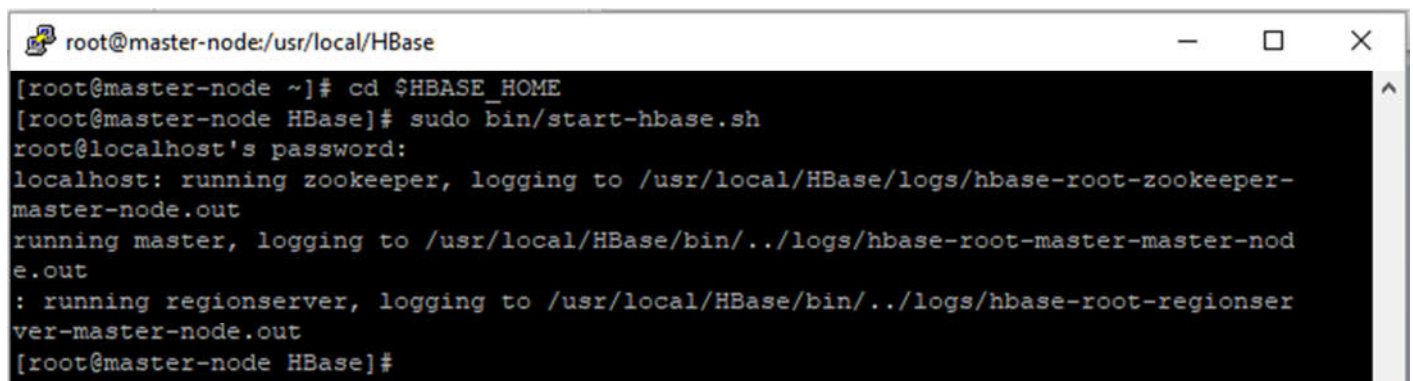
To create the /hbase directory in HDFS:

```
$ hdfs dfs -mkdir -p /hbase
$ hdfs dfs -chown hbase /hbase
$ hdfs dfs -chmod -R 775 /hbase
```

```
$ sudo chown -R hadoop:hadoop /usr/local/hadoop
$ sudo chmod -R g+rxw /usr/local/hadoop
```

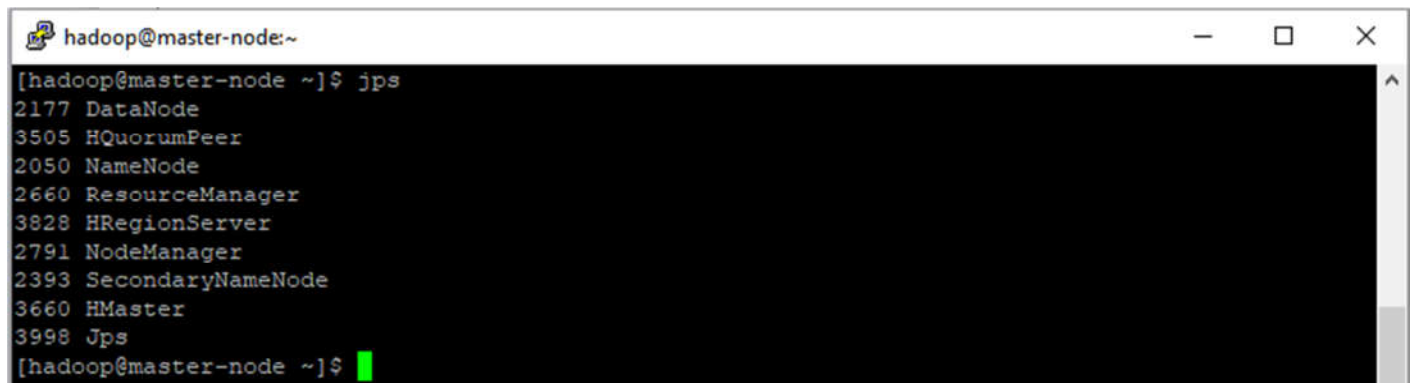
Now start HBase by using **start-hbase.sh** script in HBase bin directory

```
$ sudo su - hadoop
$ start-hbase.sh/stop-hbase.sh
$ export HBASE_MANAGES_ZK=true
```

A terminal window titled 'root@master-node:/usr/local/HBase' showing the execution of the 'start-hbase.sh' script. The user runs 'cd \$HBASE_HOME' and 'sudo bin/start-hbase.sh'. The script prompts for a password and then displays the following output: 'localhost: running zookeeper, logging to /usr/local/HBase/logs/hbase-root-zookeeper-master-node.out', 'running master, logging to /usr/local/HBase/bin/../../logs/hbase-root-master-master-node.out', and ': running regionserver, logging to /usr/local/HBase/bin/../../logs/hbase-root-regionserver-master-node.out'. The prompt returns to '[root@master-node HBase]#'.

```
root@master-node:/usr/local/HBase
[root@master-node ~]# cd $HBASE_HOME
[root@master-node HBase]# sudo bin/start-hbase.sh
root@localhost's password:
localhost: running zookeeper, logging to /usr/local/HBase/logs/hbase-root-zookeeper-master-node.out
running master, logging to /usr/local/HBase/bin/../../logs/hbase-root-master-master-node.out
: running regionserver, logging to /usr/local/HBase/bin/../../logs/hbase-root-regionserver-master-node.out
[root@master-node HBase]#
```

The results of jps command should give hadoop and hbase daemons

A terminal window titled 'hadoop@master-node:~' showing the output of the 'jps' command. The output lists several daemons with their corresponding PIDs: 2177 DataNode, 3505 HQuorumPeer, 2050 NameNode, 2660 ResourceManager, 3828 HRegionServer, 2791 NodeManager, 2393 SecondaryNameNode, 3660 HMaster, and 3998 Jps. The prompt returns to '[hadoop@master-node ~]\$'.

```
[hadoop@master-node ~]$ jps
2177 DataNode
3505 HQuorumPeer
2050 NameNode
2660 ResourceManager
3828 HRegionServer
2791 NodeManager
2393 SecondaryNameNode
3660 HMaster
3998 Jps
[hadoop@master-node ~]$
```

Check the HBase Directory in HDFS:

```
$ hadoop fs -ls /hbase
```

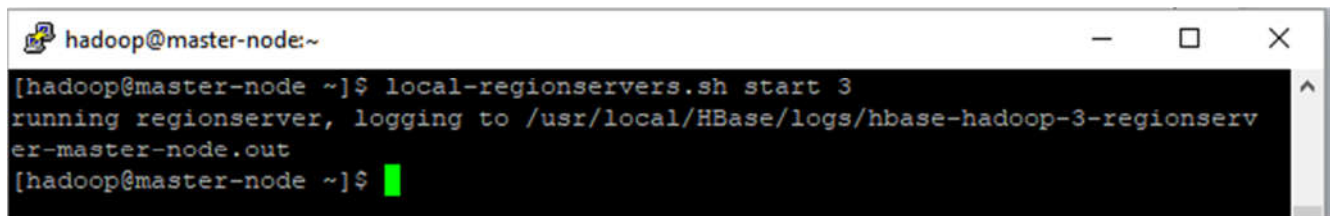
Step 3: Managing HMaster & HRegionServer

The HMaster server controls the HBase cluster. You can start up to 9 backup HMaster servers, which makes 10 total HMasters, counting the primary.

The HRegionServer manages the data in its StoreFiles as directed by the HMaster. Generally, one HRegionServer runs per node in the cluster. Running multiple HRegionServers on the same system can be useful for testing in pseudo-distributed mode.

Master and Region Servers can be started and stopped using the scripts local-master-backup.sh and local-regionervers.sh respectively.

```
$ local-master-backup.sh start 2 # Start backup HMaster
$ local-regionervers.sh start 3 # Start multiple RegionServers
```

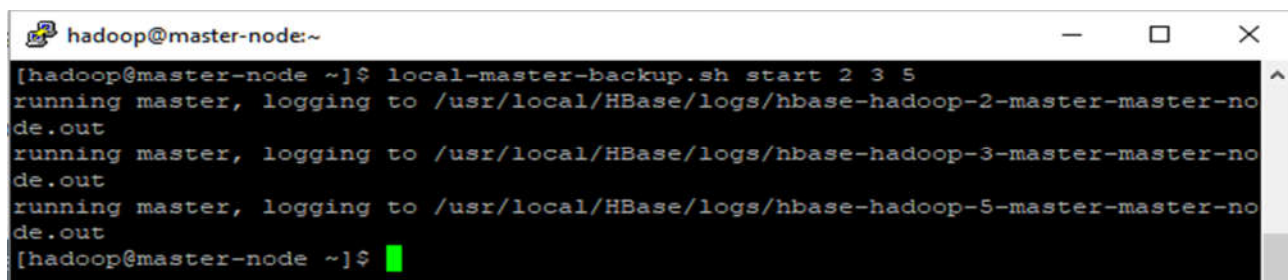


```
hadoop@master-node:~
[hadoop@master-node ~]$ local-regionervers.sh start 3
running regionserver, logging to /usr/local/HBase/logs/hbase-hadoop-3-regionserver-master-node.out
[hadoop@master-node ~]$
```

Each HMaster uses two ports (16000 and 16010 by default). The port offset is added to these ports, so using an offset of 2, the backup HMaster would use ports 16002 and 16012

The following command starts 3 backup servers using ports 16002/16012, 16003/16013, and 16005/16015.

```
$ local-master-backup.sh start 2 3 5
```

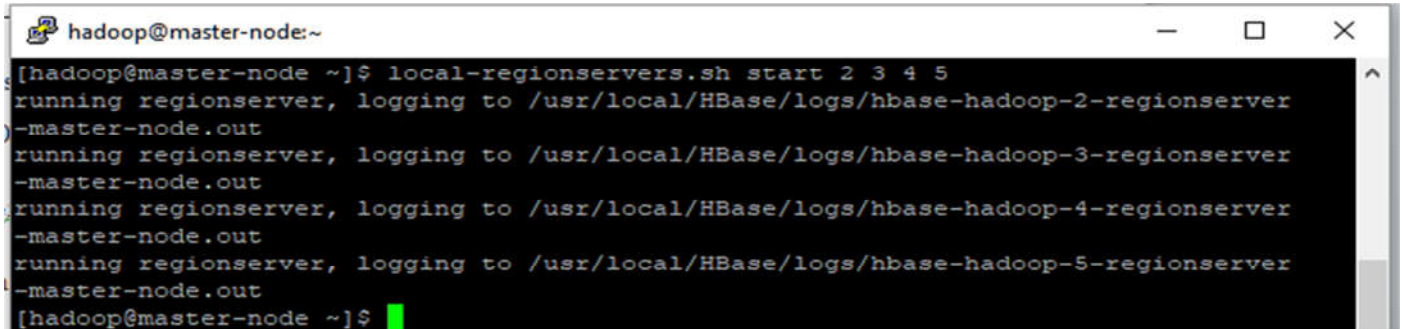


```
hadoop@master-node:~
[hadoop@master-node ~]$ local-master-backup.sh start 2 3 5
running master, logging to /usr/local/HBase/logs/hbase-hadoop-2-master-master-node.out
running master, logging to /usr/local/HBase/logs/hbase-hadoop-3-master-master-node.out
running master, logging to /usr/local/HBase/logs/hbase-hadoop-5-master-master-node.out
[hadoop@master-node ~]$
```

Each RegionServer requires two ports, and the default ports are 16020 and 16030

The following command starts four additional RegionServers, running on sequential ports starting at 16022/16032 (base ports 16020/16030 plus 2).

```
$ local-regionervers.sh start 2 3 4 5
```



```
hadoop@master-node:~  
[hadoop@master-node ~]$ local-regionervers.sh start 2 3 4 5  
running regionserver, logging to /usr/local/HBase/logs/hbase-hadoop-2-regionserver  
-master-node.out  
running regionserver, logging to /usr/local/HBase/logs/hbase-hadoop-3-regionserver  
-master-node.out  
running regionserver, logging to /usr/local/HBase/logs/hbase-hadoop-4-regionserver  
-master-node.out  
running regionserver, logging to /usr/local/HBase/logs/hbase-hadoop-5-regionserver  
-master-node.out  
[hadoop@master-node ~]$
```

To stop, replace start parameter with stop for each command followed by the offset of the server to stop.

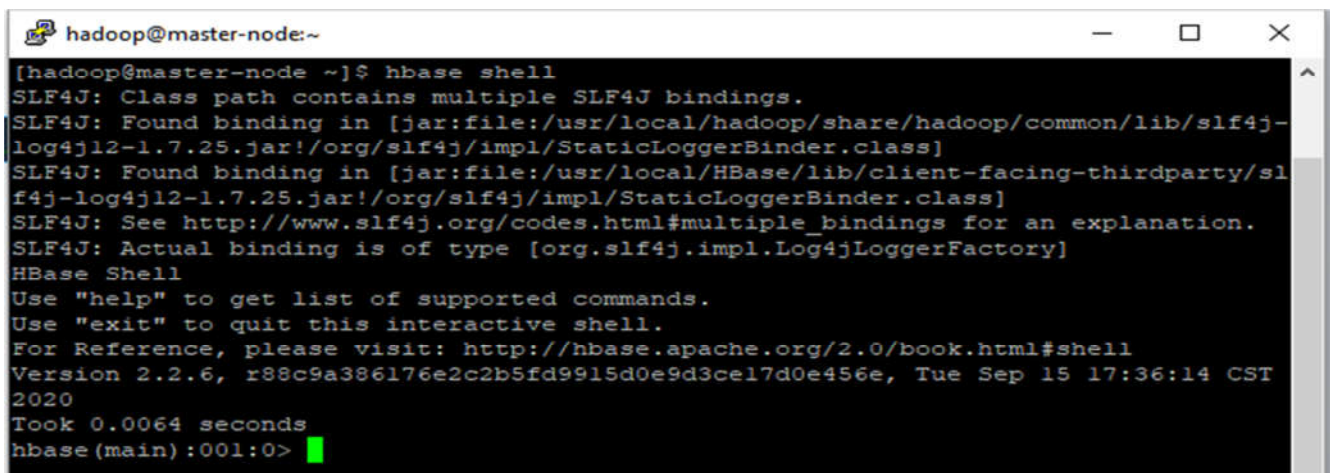
```
$ local-regionervers.sh stop 5
```

Starting HBase Shell

Hadoop and Hbase should be running before you can use HBase shell. Here the correct order of starting services.

```
$ start-all.sh
```

```
$ start-hbase.sh
```



```
hadoop@master-node:~  
[hadoop@master-node ~]$ hbase shell  
SLF4J: Class path contains multiple SLF4J bindings.  
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]  
SLF4J: Found binding in [jar:file:/usr/local/HBase/lib/client-facing-thirdparty/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]  
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.  
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]  
HBase Shell  
Use "help" to get list of supported commands.  
Use "exit" to quit this interactive shell.  
For Reference, please visit: http://hbase.apache.org/2.0/book.html#shell  
Version 2.2.6, r88c9a386176e2c2b5fd9915d0e9d3cel7d0e456e, Tue Sep 15 17:36:14 CST 2020  
Took 0.0064 seconds  
hbase(main):001:0>
```

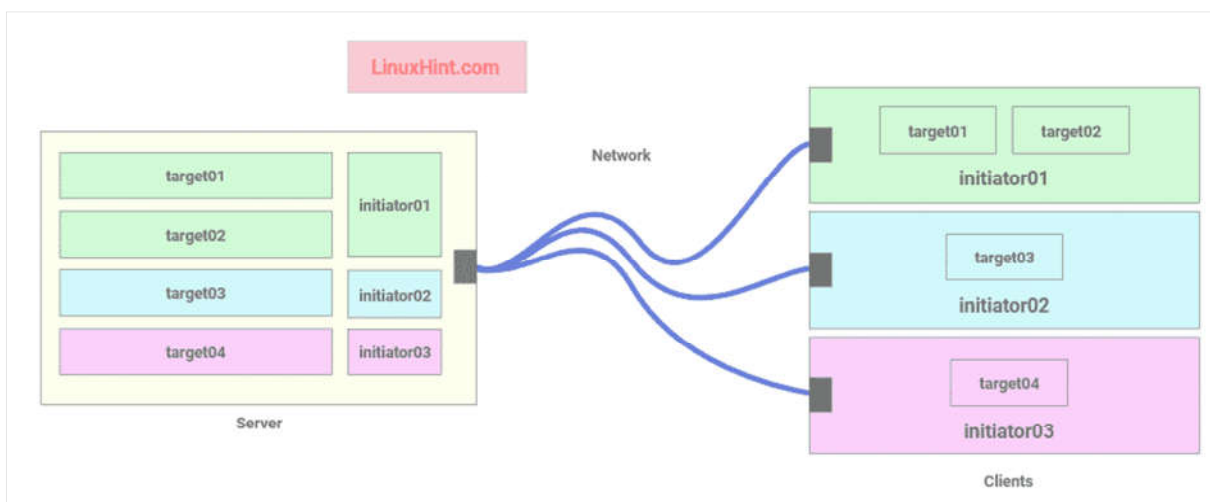
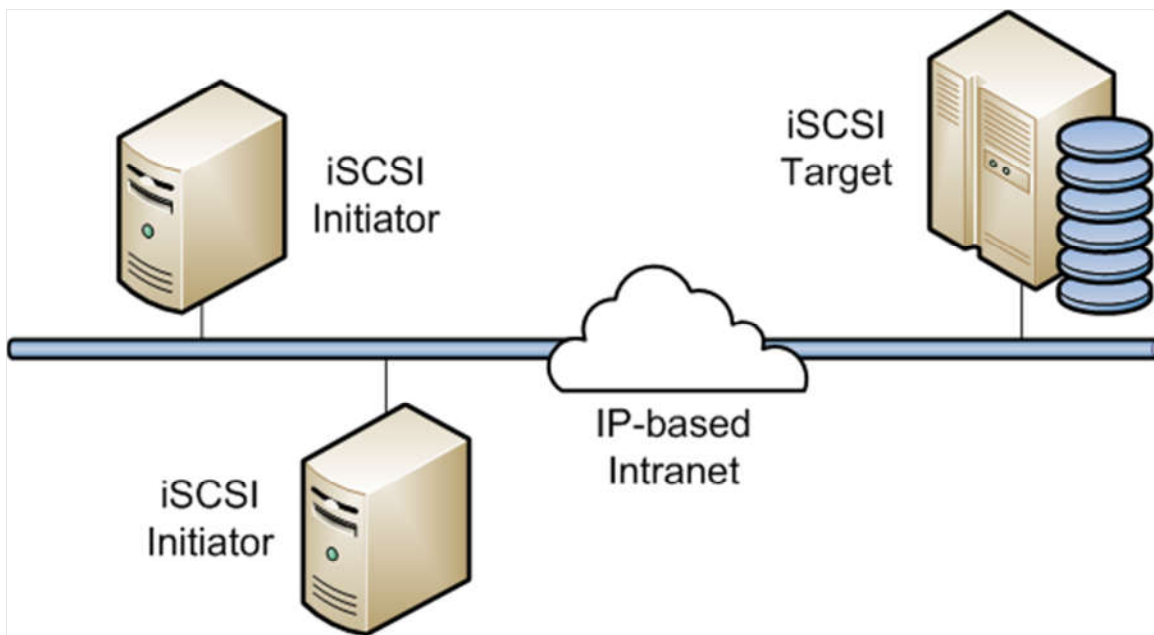
Stopping HBase.: stop-hbase.sh

ISCSI configuration and access

An iSCSI target can be a dedicated physical device in a network, or it can be an iSCSI software-configured logical device on a networked storage server. The target is the end point in SCSI bus communication. Storage on the target, accessed by an initiator, is defined by LUNs. (Redhat)

Basics of iSCSI:

iSCSI storage server is used to share block devices such as HDD/SSD partitions, or LVM partitions, or block files on the network. iSCSI clients can use these shares over the network just as any ordinary HDD or SSD mounted to it. The iSCSI client can format these disks, mount them and store files and directories as usual.



Each iSCSI client has an initiator ID which is used to connect to the targets on the server.

The targets are shares on the iSCSI server. Each target consists of a unique name (IQN), the path of the block device (i.e. disk partition or block file), the initiator ID that can connect to this target and an optional username-password based authentication system.

In fig 1, the iSCSI storage server allows 3 initiators (3 iSCSI clients) to connect to 4 targets. **initiator01** can connect to **target01** and **target02**, **initiator02** can connect to **target03**, and **initiator03** can connect to **target04**.

iSCSI Target and Initiator Naming Conventions:

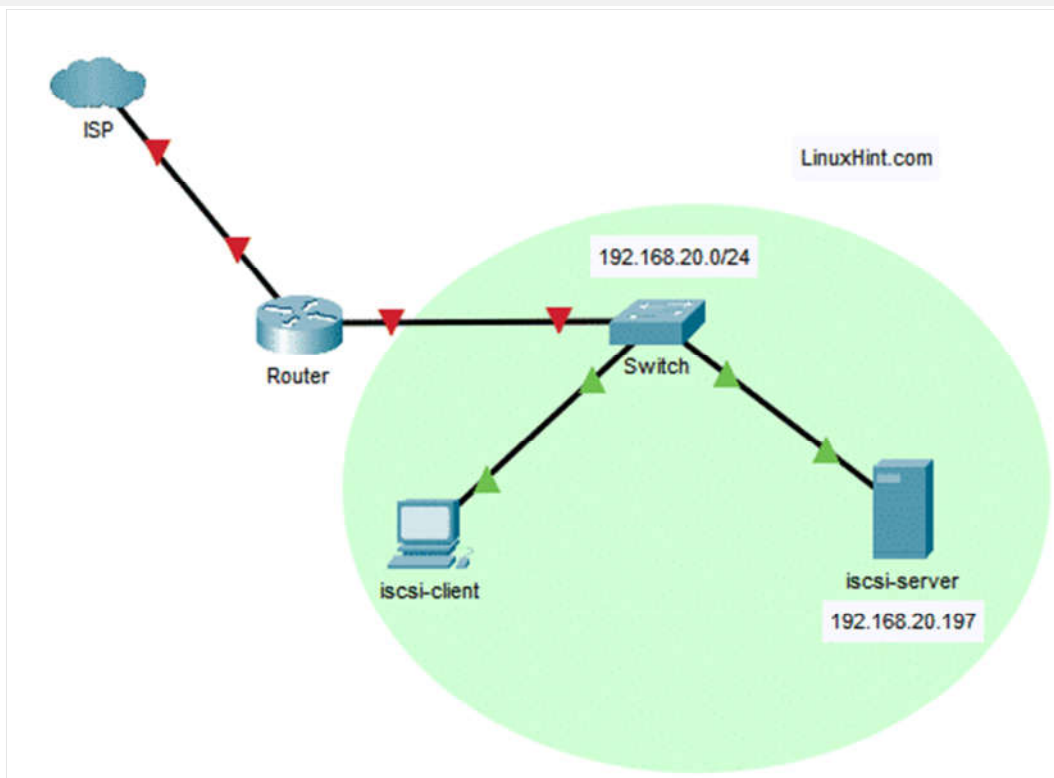
The iSCSI target name and initiator name must be unique.

The target naming format is: ***iqn.YYYY-MM.reverse-domain-name:target-name***

iqn.2020-03.com.linuxhint:www, iqn.2020-03.com.linuxhint:logs,
iqn.2020-03.com.linuxhint:user-bob etc.

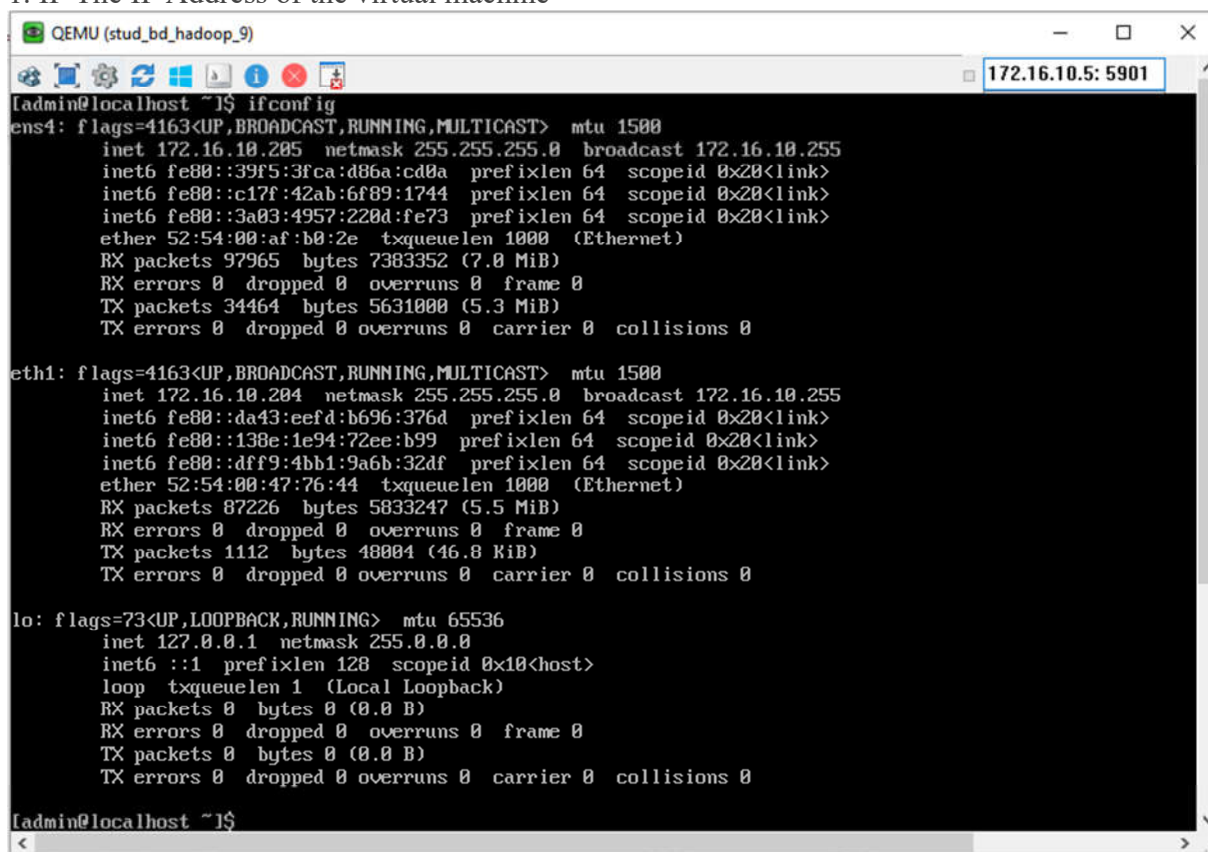
The initiator naming format is: ***iqn.YYYY-MM.reverse-domain-name:initiator-name***

iqn.2020-03.com.linuxhint:initiator01, iqn.2020-03.com.linuxhint:initiator02,
iqn.2020-03.com.linuxhint:initiator03 etc.



Here, we configure a CentOS 8 machine as an iSCSI server. The iSCSI server has a static IP address 192.168.20.197. The iSCSI client is also on the network 192.168.20.0/24. So, it can access the iSCSI server.

1. IP The IP Address of the virtual machine



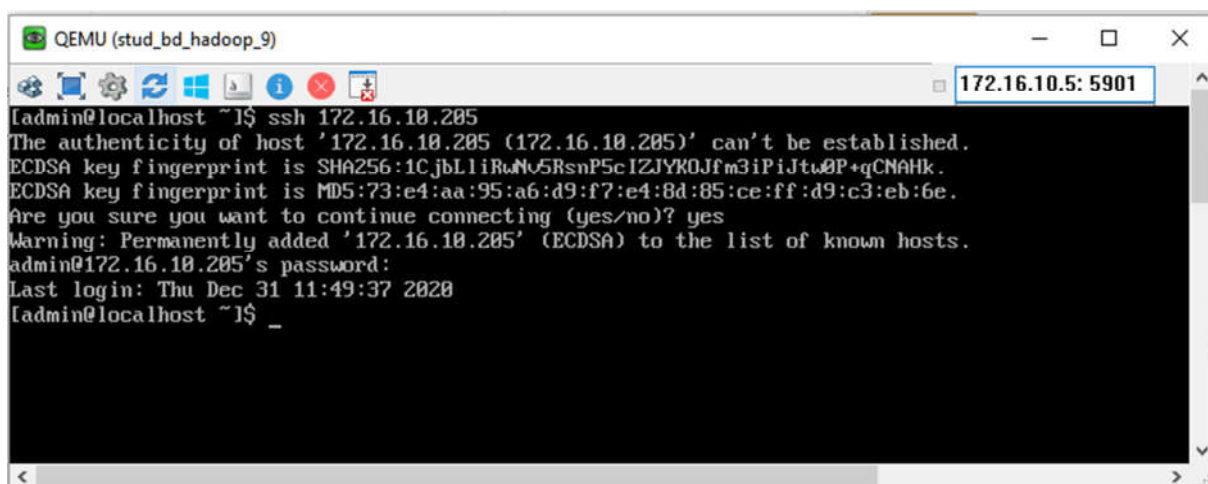
```
[admin@localhost ~]$ ifconfig
ens4: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.16.10.205 netmask 255.255.255.0 broadcast 172.16.10.255
    inet6 fe80::39f5:3fca:d86a:cd0a prefixlen 64 scopeid 0x20<link>
    inet6 fe80::c17f:42ab:6f89:1744 prefixlen 64 scopeid 0x20<link>
    inet6 fe80::3a03:4957:220d:fe73 prefixlen 64 scopeid 0x20<link>
    ether 52:54:00:af:b0:2e txqueuelen 1000 (Ethernet)
    RX packets 97965 bytes 7383352 (7.0 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 34464 bytes 5631000 (5.3 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.16.10.204 netmask 255.255.255.0 broadcast 172.16.10.255
    inet6 fe80::da43:ee4d:b696:376d prefixlen 64 scopeid 0x20<link>
    inet6 fe80::138e:1e94:72ee:b99 prefixlen 64 scopeid 0x20<link>
    inet6 fe80::dff9:4bb1:9a6b:32df prefixlen 64 scopeid 0x20<link>
    ether 52:54:00:47:76:44 txqueuelen 1000 (Ethernet)
    RX packets 87226 bytes 5833247 (5.5 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1112 bytes 40004 (46.0 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

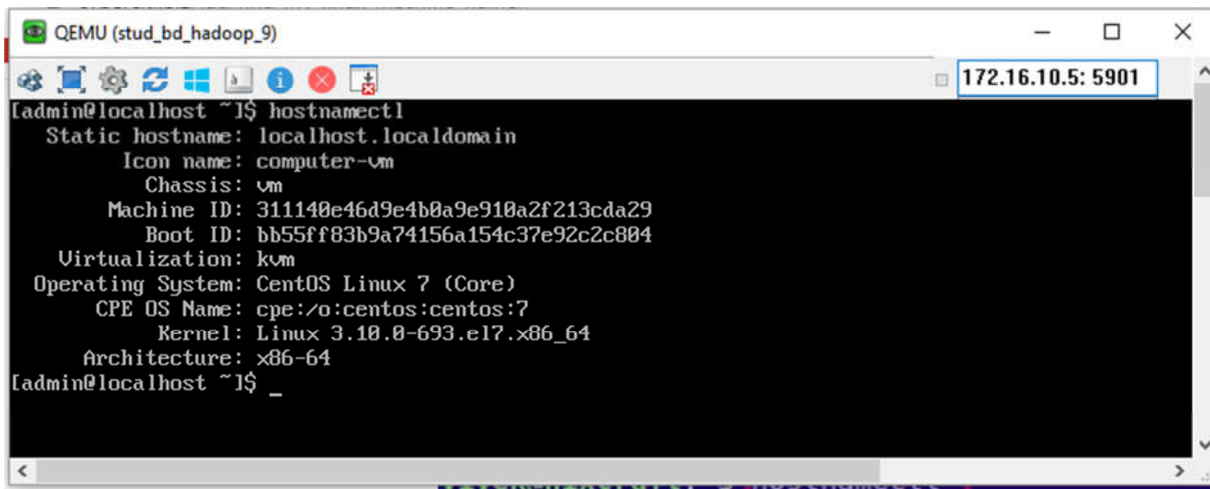
[admin@localhost ~]$
```

2. SSH



```
[admin@localhost ~]$ ssh 172.16.10.205
The authenticity of host '172.16.10.205 (172.16.10.205)' can't be established.
ECDSA key fingerprint is SHA256:1CjbLlRwNv5RsnP5cIZJYK0Jfm3iPiJtw0P+qCNAHk.
ECDSA key fingerprint is MD5:73:e4:aa:95:a6:d9:f7:e4:8d:85:ce:ff:d9:c3:eb:6e.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.16.10.205' (ECDSA) to the list of known hosts.
admin@172.16.10.205's password:
Last login: Thu Dec 31 11:49:37 2020
[admin@localhost ~]$ _
```


3. Find out the name of the virtual machine

A screenshot of a QEMU terminal window titled 'QEMU (stud_bd_hadoop_9)'. The terminal shows the output of the 'hostnamectl' command. The output includes: Static hostname: localhost.localdomain, Icon name: computer-vm, Chassis: vm, Machine ID: 311140e46d9e4b0a9e918a2f213cda29, Boot ID: bb55ff83b9a74156a154c37e92c2c804, Virtualization: kvm, Operating System: CentOS Linux 7 (Core), CPE OS Name: cpe:/o:centos:centos:7, Kernel: Linux 3.10.0-693.el7.x86_64, and Architecture: x86-64. The prompt is [admin@localhost ~]\$.

```
QEMU (stud_bd_hadoop_9)
[admin@localhost ~]$ hostnamectl
  Static hostname: localhost.localdomain
        Icon name: computer-vm
        Chassis: vm
  Machine ID: 311140e46d9e4b0a9e918a2f213cda29
  Boot ID: bb55ff83b9a74156a154c37e92c2c804
  Virtualization: kvm
  Operating System: CentOS Linux 7 (Core)
  CPE OS Name: cpe:/o:centos:centos:7
    Kernel: Linux 3.10.0-693.el7.x86_64
  Architecture: x86-64
[admin@localhost ~]$ _
```

4. Connect to storage

Installing iSCSI Client Tools and connecting to storage:

On the iSCSI client, you must have **iscsi-initiator-utils** package installed in order to access the shared iSCSI storage devices.

First, update the DNF package repository cache as follows:

Now, install **iscsi-initiator-utils** package on the client machine as follows:

```
$ sudo dnf install iscsi-initiator-utils
```

Now, open the **/etc/iscsi/initiatorname.iscsi** configuration file as follows:

```
$ sudo vi /etc/iscsi/initiatorname.iscsi
```

Now, set your initiator name to **InitiatorName** and save the file.

```
InitiatorName=iqn.2020-03.com.linuxhint:init1
```

Now, start the iscsi and iscsid services as follows:

```
$ sudo systemctl start iscsi
```

```
$ sudo systemctl start iscsid
```

```
$ sudo systemctl status iscsi iscsid
```

```
QEMU (stud_bd_hadoop_9) 172.16.10.5:5901
[root@localhost ~]# sudo systemctl status iscsi iscsid
iscsi.service - Login and scanning of iSCSI devices
Loaded: loaded (/usr/lib/systemd/system/iscsi.service; enabled; vendor preset: disabled)
Active: active (exited) since 4r 2020-12-31 13:27:04 MSK; 1 day 2h ago
Docs: man:iscsiadm(8)
      man:iscsid(8)
Process: 22025 ExecStart=/sbin/iscsiadm -m node --loginall=automatic (code=exited, status=0/SUCCESS)
Main PID: 22025 (code=exited, status=0/SUCCESS)
CGroup: /system.slice/iscsi.service

дек 31 13:27:04 localhost.localdomain systemd[1]: Starting Login and scanning of iSCSI devices...
дек 31 13:27:04 localhost.localdomain systemd[1]: Started Login and scanning of iSCSI devices.

iscsid.service - Open-iSCSI
Loaded: loaded (/usr/lib/systemd/system/iscsid.service; enabled; vendor preset: disabled)
Active: active (running) since 4r 2020-12-31 13:40:02 MSK; 1 day 2h ago
Docs: man:iscsid(8)
      man:iscsiuio(8)
      man:iscsiadm(8)
Main PID: 27066 (iscsid)
Status: "Ready to process requests"
CGroup: /system.slice/iscsid.service
       └─27066 /sbin/iscsid -f

январь 01 15:24:34 localhost.localdomain iscsid[27066]: iscsid: Connection2:0 to [target: iqn.2006-08.com.huawei:oceanst...l now
январь 01 15:26:28 localhost.localdomain iscsid[27066]: iscsid: Connection2:0 to [target: iqn.2006-08.com.huawei:oceanst...down.
январь 01 15:26:40 localhost.localdomain iscsid[27066]: iscsid: Could not set session3 priority. READ/WRITE throughout a...cted.
январь 01 15:26:40 localhost.localdomain iscsid[27066]: iscsid: Connection3:0 to [target: iqn.2006-08.com.huawei:oceanst...l now
январь 01 15:26:40 localhost.localdomain iscsid[27066]: iscsid: Connection3:0 to [target: iqn.2006-08.com.huawei:oceanst...down.
январь 01 15:30:17 localhost.localdomain iscsid[27066]: iscsid: Could not set session4 priority. READ/WRITE throughout a...cted.
январь 01 15:30:17 localhost.localdomain iscsid[27066]: iscsid: Connection4:0 to [target: iqn.2006-08.com.huawei:oceanst...l now
январь 01 15:32:36 localhost.localdomain iscsid[27066]: iscsid: Connection4:0 to [target: iqn.2006-08.com.huawei:oceanst...down.
январь 01 15:32:44 localhost.localdomain iscsid[27066]: iscsid: Could not set session5 priority. READ/WRITE throughout a...cted.
январь 01 15:32:44 localhost.localdomain iscsid[27066]: iscsid: Connection5:0 to [target: iqn.2006-08.com.huawei:oceanst...l now
Hint: Some lines were ellipsized, use -l to show in full.
[root@localhost ~]# _
```

```
QEMU (stud_bd_hadoop_9) 172.16.10.5: 5901
[root@localhost admin]# sudo systemctl start tgt
[root@localhost admin]# sudo systemctl enable tgt
Created symlink from /etc/systemd/system/multi-user.target.wants/tgt.service to /usr/lib/systemd/system/tgt.service.
[root@localhost admin]# _
```

Now, scan for the targets as follows:

```
$ sudo iscsiadm -m discovery -t sendtargets -p 172.16.10.20
```

```
QEMU (stud_bd_hadoop_9) 172.16.10.5: 5901
[root@localhost admin]# iscsiadm -m discovery -t sendtargets -p 172.16.10.20
172.16.10.20:3260,8196 iqn.2006-08.com.huawei:oceanstor:210050605f00d9e7:70:22003:172.16.10.20
[root@localhost admin]#
```



```
QEMU (stud_bd_hadoop_9)
[172.16.10.5: 5901]
[root@localhost admin]# iscsiadm -m discovery -P 1
SENDTARGETS:
DiscoveryAddress: 172.16.10.20,3260
Target: iqn.2006-08.com.huawei:oceastor:210058605f00d9e7:70:22003:172.16.10.20
Portal: 172.16.10.20:3260,8196
Iface Name: default
iSNS:
No targets found.
STATIC:
No targets found.
FIRMWARE:
No targets found.
[root@localhost admin]# iscsiadm -m node -P 1
Target: iqn.2006-08.com.huawei:oceastor:210058605f00d9e7:70:22003:172.16.10.20
Portal: 172.16.10.20:3260,8196
Iface Name: default
[root@localhost admin]# _
```

To log in into specific system target, enter the following command:
iscsiadm --mode node --target <IQN> --portal x.x.x.x --login

```
QEMU (stud_bd_hadoop_9)
[172.16.10.5:5901]
[root@localhost ~]# iscsiadm --mode node -l
Logging in to [iface: default, target: iqn.2006-08.com.huawei:oceastor:210058605f00d9e7:70:22003:172.16.10.20, portal: 172.16.10.20,3260] (multiple)
Login to [iface: default, target: iqn.2006-08.com.huawei:oceastor:210058605f00d9e7:70:22003:172.16.10.20, portal: 172.16.10.20,3260] successful.
[root@localhost ~]# _
```

View the current session

```
admin@localhost:~
[admin@localhost ~]$ sudo iscsiadm -m session -P 1
Target: iqn.2006-08.com.huawei:oceastor:210058605f00d9e7:70:22003:172.16.10.20 (non-flash)
Current Portal: 172.16.10.20:3260,8196
Persistent Portal: 172.16.10.20:3260,8196
*****
Interface:
*****
Iface Name: default
Iface Transport: tcp
Iface Initiatorname: iqn.1996-04.de.suse:01:myserver0

1
Iface IPaddress: 172.16.10.205
Iface HWaddress: <empty>
Iface Netdev: <empty>
SID: 8
iSCSI Connection State: LOGGED IN
iSCSI Session State: LOGGED_IN
Internal iscsid Session State: NO CHANGE
[admin@localhost ~]$
```

```
admin@localhost:~  
[admin@localhost ~]$ sudo iscsiadm -m session -P 1 | grep 'Target\\|disk'  
Target: iqn.2006-08.com.huawei:oceanstor:210058605f00d9e7:70:22003:172.16.10.20 (non-flash)  
[admin@localhost ~]$
```

To log out of all your established sessions, enter the following command:
iscsiadm --mode node --logoutall=all

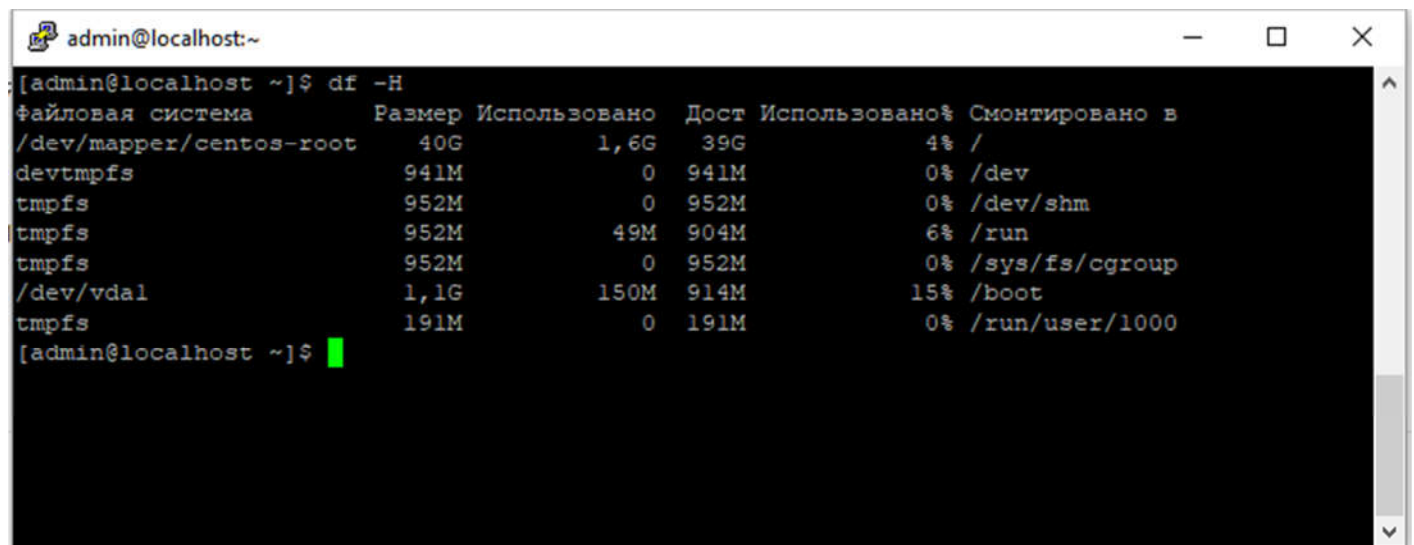
```
QEMU (stud_bd_hadoop_9) 172.16.10.5:5901  
[root@localhost ~]# iscsiadm --mode node --portal 172.16.10.20 --logout  
Logging out of session [sid: 3, target: iqn.2006-08.com.huawei:oceanstor:210058605f00d9e7:70:22003:172.16.10.20, portal: 172.16.10.20,3260]  
Logout of [sid: 3, target: iqn.2006-08.com.huawei:oceanstor:210058605f00d9e7:70:22003:172.16.10.20, portal: 172.16.10.20,3260] successful.  
[root@localhost ~]#
```

5 View storage structure

Results of fdisk -l

```
admin@localhost:~  
[admin@localhost ~]$ sudo fdisk -l  
  
Disk /dev/vda: 42.9 GB, 42949672960 bytes, 83886080 sectors  
Units = sectors of 1 * 512 = 512 bytes  
Sector size (logical/physical): 512 bytes / 512 bytes  
I/O size (minimum/optimal): 512 bytes / 512 bytes  
Disk label type: dos  
Disk identifier: 0x000aa4ca  
  
Устройство Начало Конец Блоки Id Система  
/dev/vda1 * 2048 2099199 1048576 83 Linux  
/dev/vda2 2099200 83886079 40893440 8e Linux LVM  
  
Disk /dev/mapper/centos-root: 39.7 GB, 39720058880 bytes, 77578240 sectors  
Units = sectors of 1 * 512 = 512 bytes  
Sector size (logical/physical): 512 bytes / 512 bytes  
I/O size (minimum/optimal): 512 bytes / 512 bytes  
  
Disk /dev/mapper/centos-swap: 2147 MB, 2147483648 bytes, 4194304 sectors  
Units = sectors of 1 * 512 = 512 bytes  
Sector size (logical/physical): 512 bytes / 512 bytes  
I/O size (minimum/optimal): 512 bytes / 512 bytes  
  
Disk /dev/sda: 0 MB, 16384 bytes, 32 sectors  
Units = sectors of 1 * 512 = 512 bytes  
Sector size (logical/physical): 512 bytes / 512 bytes  
I/O size (minimum/optimal): 512 bytes / 512 bytes  
  
[admin@localhost ~]$
```

Results of df -H



```
admin@localhost:~$ df -H
файловая система      Размер  Использовано  Дост  Использовано%  Смонтировано в
/dev/mapper/centos-root 40G      1,6G      39G          4% /
devtmpfs                941M      0      941M          0% /dev
tmpfs                   952M      0      952M          0% /dev/shm
tmpfs                   952M     49M      904M          6% /run
tmpfs                   952M      0      952M          0% /sys/fs/cgroup
/dev/vda1               1,1G     150M      914M         15% /boot
tmpfs                   191M      0      191M          0% /run/user/1000
admin@localhost:~$
```

Mounting iSCSI Disk:

You can mount the iSCSI disks permanently on the iSCSI client using the `/etc/fstab` file. First, format the iSCSI disk if it's not already formatted.

```
sudo mkfs.ext4 -L data /dev/sda
```

To create and setup LUNs using LVM on RHEL/CentOS/Fedora

LUN is a Logical Unit Number that shared from the iSCSI Storage Server. The iSCSI target server physical drive shares its drive to initiate over TCP/IP network. It form a large storage as SAN (Storage Area Network) with a collection of drives. LUNs are defined in LVM as per space availability in real environment. Creating LUNs using LVM is explained in this article.

Importance of LUN

- It is used for storage purpose.
- It is used to install Operating systems and also in Clusters, Virtual servers, SAN etc.

Install targetcli and Creating LUNS

```
admin@localhost:~  
[admin@localhost ~]$ sudo systemctl enable target  
[admin@localhost ~]$ sudo systemctl start target  
[admin@localhost ~]$ targetcli  
opening lockfile failed: [Errno 13] Permission denied: '/var/run/targetcli.lock'  
[admin@localhost ~]$ sudo targetcli  
targetcli shell version 2.1.53  
Copyright 2011-2013 by Datera, Inc and others.  
For help on commands, type 'help'.  
/>
```

```
admin@localhost:~  
/> cd iscsi/  
/iscsi> create ign.2020-01.example.com:target01  
Created target ign.2020-01.example.com:target01.  
Created TPG 1.  
Default portal not created, TPGs within a target cannot share ip:port.  
/iscsi> cd ign.2020-01.example.com:target01/tpg1/acls  
/iscsi/ign.20...t01/tpg1/acls> create ign.2020-01.example.com:server01  
Created Node ACL for ign.2020-01.example.com:server01  
/iscsi/ign.20...t01/tpg1/acls> cd ign.2020-01.example.com:server01  
/iscsi/ign.20....com:server01> set auth userid=amos  
Parameter userid is now 'amos'.  
/iscsi/ign.20....com:server01> set auth password=qwerty  
Parameter password is now 'qwerty'.  
/iscsi/ign.20....com:server01> cd ../ ../  
set 2 positional parameters, expected at most 1
```

```
admin@localhost:~  
№1) Уважайте частную жизнь других.  
№2) Думайте, прежде что-то вводить.  
№3) С большой властью приходит большая ответственность.  
  
[sudo] пароль для admin:  
targetcli shell version 2.1.53  
Copyright 2011-2013 by Datera, Inc and others.  
For help on commands, type 'help'.  
  
/backstores/block> cd /  
/> ls  
o- / ..... [...]  
  o- backstores ..... [...]  
    | o- block ..... [Storage Objects: 0]  
    | o- fileio ..... [Storage Objects: 0]  
    | o- pscsi ..... [Storage Objects: 0]  
    | o- ramdisk ..... [Storage Objects: 0]  
  o- iscsi ..... [Targets: 2]  
    | o- iqn.2003-01.org.linux-iscsi.localhost.x8664:sn.dfbe270cac68 ..... [TPGs: 1]  
    |   | o- tpg1 ..... [no-gen-acls, no-auth]  
    |   |   | o- acls ..... [ACLs: 0]  
    |   |   | o- luns ..... [LUNs: 0]  
    |   |   | o- portals ..... [Portals: 0]  
    | o- iqn.2020-01.example.com:target01 ..... [TPGs: 1]  
    |   | o- tpg1 ..... [no-gen-acls, no-auth]  
    |   |   | o- acls ..... [ACLs: 1]  
    |   |   |   | o- iqn.2020-01.example.com:server01 ..... [Mapped LUNs: 0]  
    |   |   |   | o- luns ..... [LUNs: 0]  
    |   |   |   | o- portals ..... [Portals: 0]  
  o- loopback ..... [Targets: 0]  
/>
```

To Create LUNs using LVM in iSCSI Target Server

Using 'fdisk -l' command, find out the list of drives in every partitions on the system.

Creation of Logical Volume for LUNs

Create Physical volume using 'pvcreate' command.

lets use /dev/sdb like iscsi target
make file system Ext4 on this device

```
[root@storage-unixmen ~]# mkfs.ext4 /dev/sdb
```

References

- <https://www.unixmen.com/how-to-create-an-iscsi-target/>
- https://linuxhint.com/iscsi_storage_server_centos/
- <https://www.linuxhelp.com/how-to-create-and-setup-luns-using-lvm>
- <https://www.tecmint.com/create-luns-using-lvm-in-iscsi-target/>
- https://linux-admins.ru/article.php?id_article=66&article_title=%D0%A3%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BA%D0%B0%20%D0%B8%20%D0%BD%D0%B0%D1%81%D1%82%D1%80%D0%BE%D0%B9%D0%BA%D0%B0%20Iscsi%20%D0%BD%D0%B0%20CentOS7
- <https://unix.ru/%D0%BD%D0%B0%D1%81%D1%82%D1%80%D0%BE%D0%B9%D0%BA%D0%B0-iscsi-target-%D0%B8-initiator-%D0%BD%D0%B0-linux/>
- <https://www.namecheap.com/support/knowledgebase/article.aspx/10266/2194/how-to-connect-to-remote-storage-on-centos-server/>
- <https://cloud.ibm.com/docs/BlockStorage?topic=BlockStorage-mountingLinux>