

MySQL Cluster on Centos 7

Content:

- Prerequisites
- Installation of CentOS(64x) Minimal in VMWare
- Step 1 - Setup Management Node
 - A. Download the MySQL Cluster software
 - B. Install and Remove Packages
 - C. Install MySQL Cluster
 - D. Configure MySQL Cluster
 - E. Start the Management Node
- Step 2 - Setup the MySQL Cluster Data Nodes
 - A. Login as root user and download the MySQL Cluster software
 - B. Install and Remove Packages
 - C. Install MySQL Cluster
 - D. Configure Data Node
 - E. Redo step 2.A - 2.D on db3 server
- Step 3 - Setup SQL Node
 - A. Log in and Download MySQL Cluster
 - B. Install and Remove Packages
 - C. Install MySQL Cluster
 - D. Configure the SQL Node
 - E. Redo step 3.A - 3.D on db5 server
- Step 4 - Monitor the Cluster
- Step 5 - Testing the Cluster
- Conclusion

Percona XtraDB Cluster on CentOS 7

Content:

Installation of CentOS 7 (64x) Minimal in VMWare

How to Install Percona XtraDB Cluster on CentOS 7

Step 1 – Setup the hosts file

Step 2 – Configure Firewallld

Step 3 – Install the Epel Repository and Socat

Step 4 – Install Percona XtraDB Cluster

Step 5 – Configure Percona XtraDB Cluster

Step 6 – Start the Percona XtraDB Cluster Server

Step 7 – Testing

How to Install and Configure MySQL Cluster on CentOS 7

MySQL Cluster is designed to provide a MySQL compatible database with high availability and low latency. The MySQL Cluster technology is implemented through the NDB (Network Database) and NDBCLUSTER storage engines and provides shared-nothing clustering and auto-sharing for MySQL database systems. In the shared-nothing architecture, each of nodes has its own memory and disk, the use of shared storage such as NFS, SANs are not recommended and supported.

To implement a MySQL Cluster, we have to install three types of nodes. Each node type will be installed on its own server. The components are:

1. Management Node - NDB_MGMD/MGM

The Cluster management server is used to manage the other node of the cluster. We can create and configure new nodes, restart, delete, or backup nodes on the cluster from the management node.

2. Data Nodes - NDBD/NDB

This is the layer where the process of synchronizing and data replication between nodes happens.

3. SQL Nodes - MySQLD/API

The interface servers that are used by the applications to connect to the database cluster. In this tutorial, I will guide you through the installation and configuration of a MySQL Cluster with CentOS 7. We will configure the management node, two data nodes, and two SQL nodes.

Prerequisites

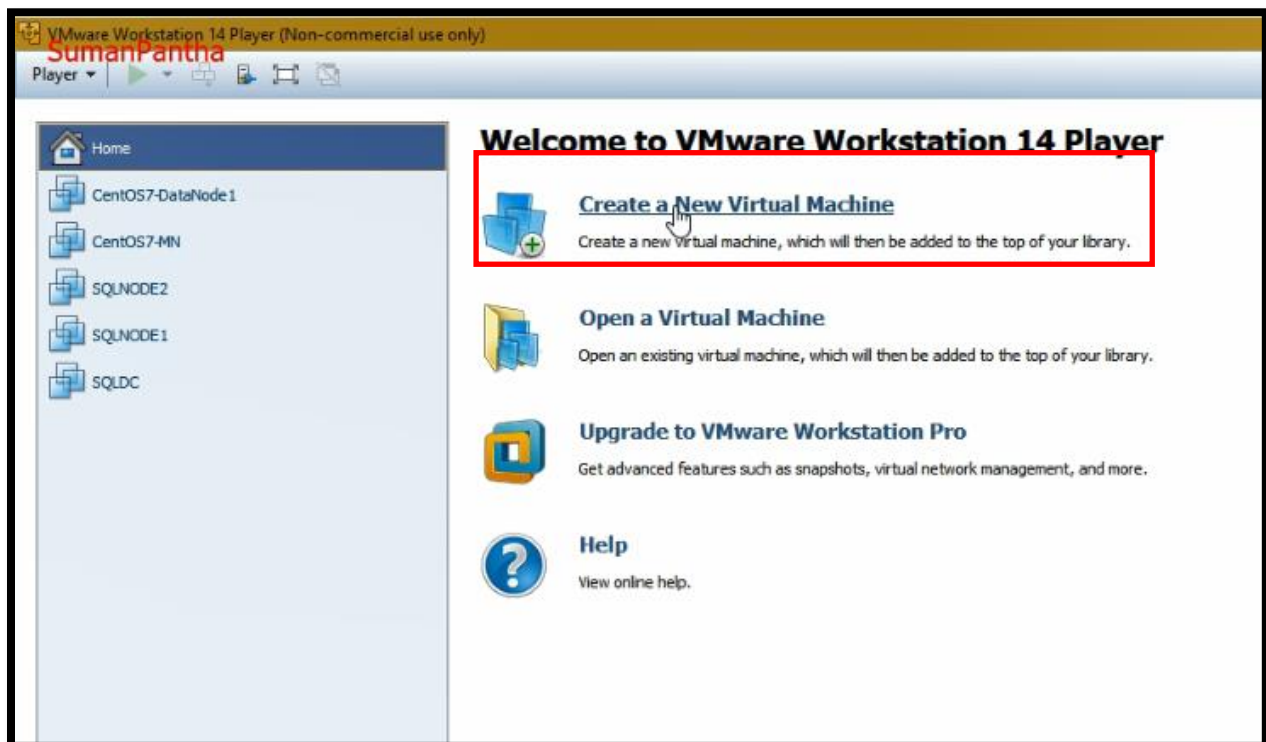
5 CentOS servers or virtual machines. I will use the hostnames and IP addresses as shown below:

- **Management Node**
`centosmn` = 192.168.19.130
- **Data Nodes**
`datanode1` = 192.168.19.129
`datanode2` = 192.168.19.131
- **SQL Nodes**
`sqlnode1` = 192.168.19.132
`sqlnode2` = 192.168.19.133
- The OS is CentOS 7 - 64bit.

Installation of CentOS 7 (64X) Minimal in VMWare

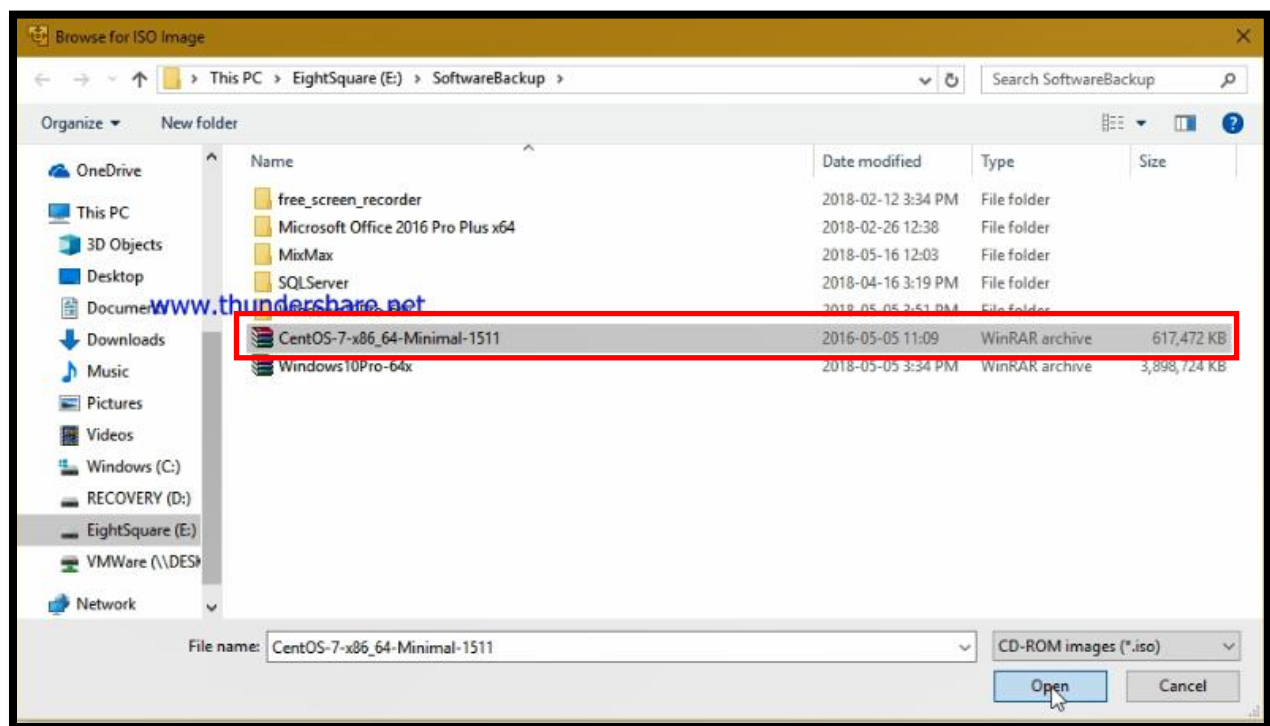
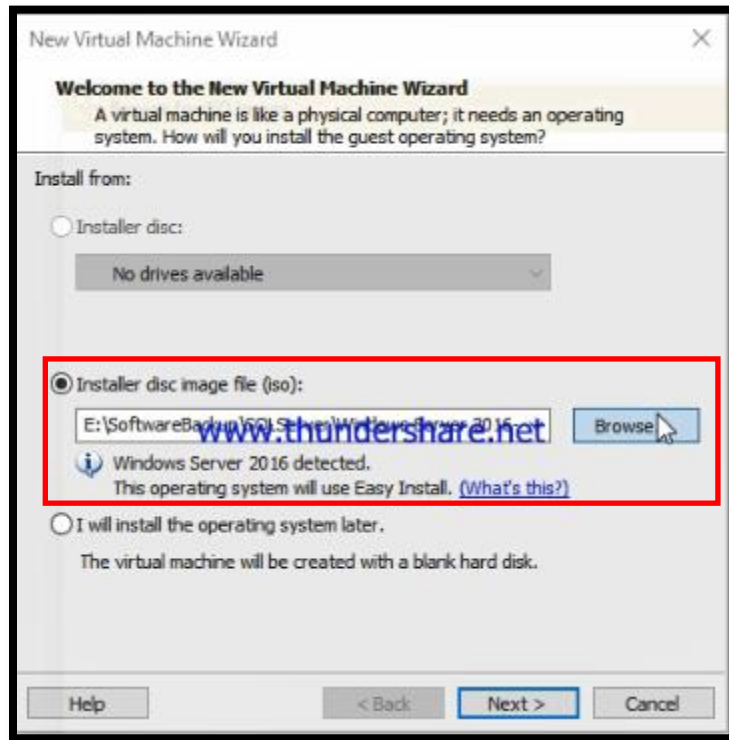
As we need 5 Virtual Machine to implement the MySQL Clustering so we need to first install Operation System (CentOS 7). The steps to install CentOS in VMWare is represented by the screenshot below. We need to install CentOS in five different VM. So, repeat the steps to install the OS on five different VM.

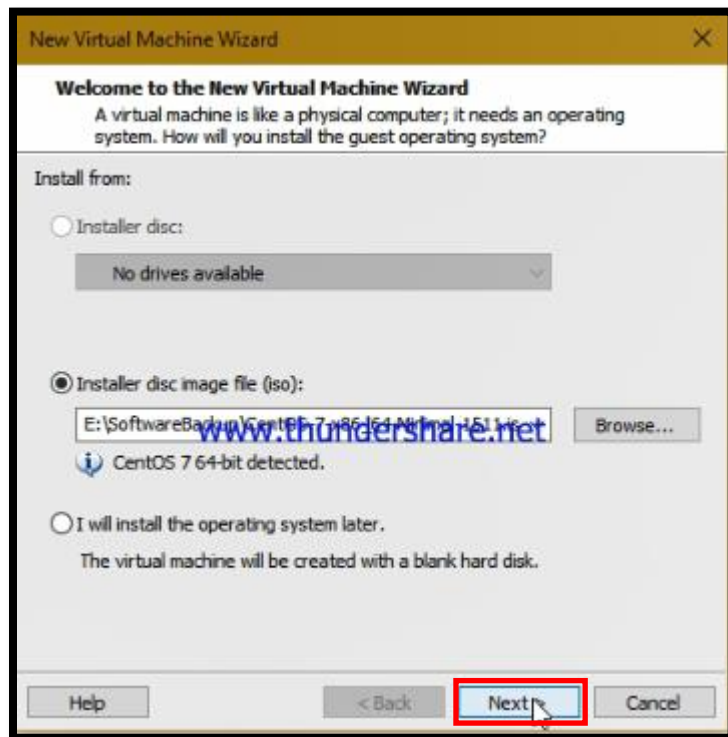
For the very first time we need to create the Virtual Machine for the OS.



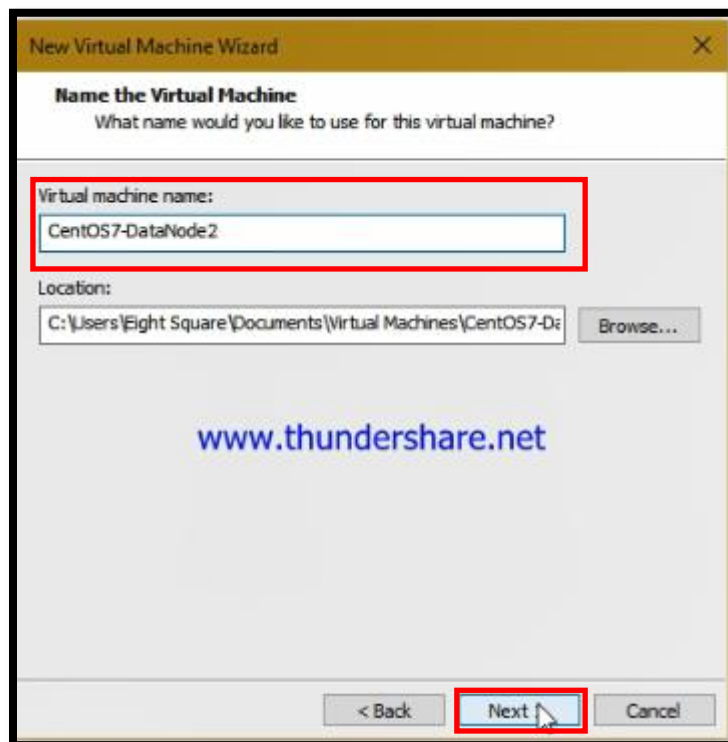
5 | MySQL Cluster on CentOS-7

Browse the OS file and click Next to continue the installation.



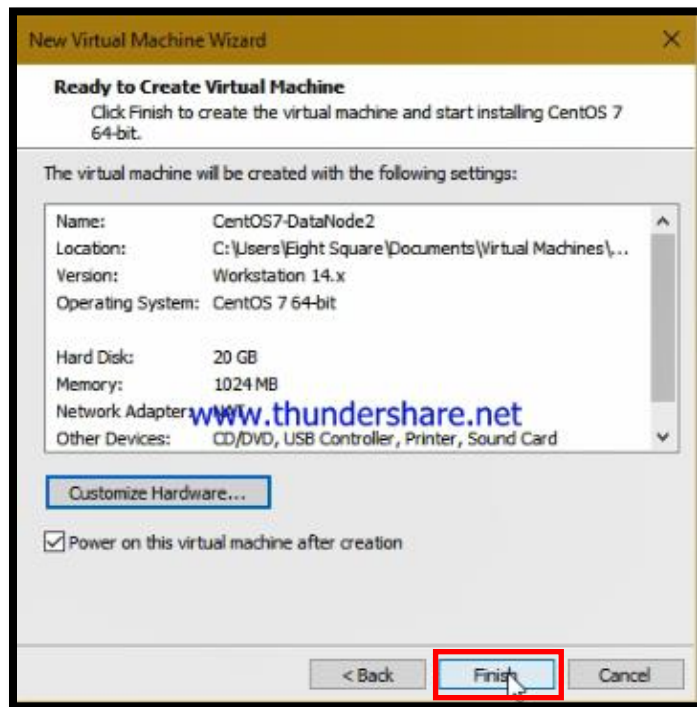
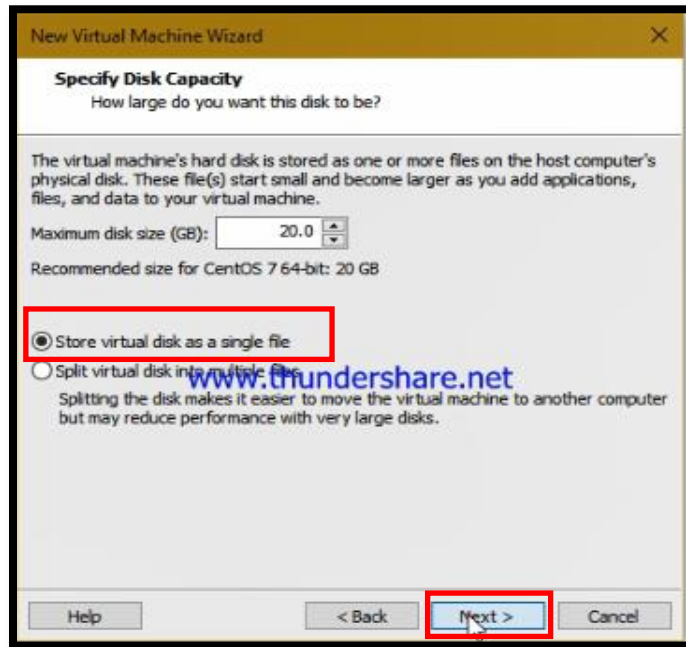


Give the Virtual Machine a Identical name and click Next



7 | MySQL Cluster on CentOS-7

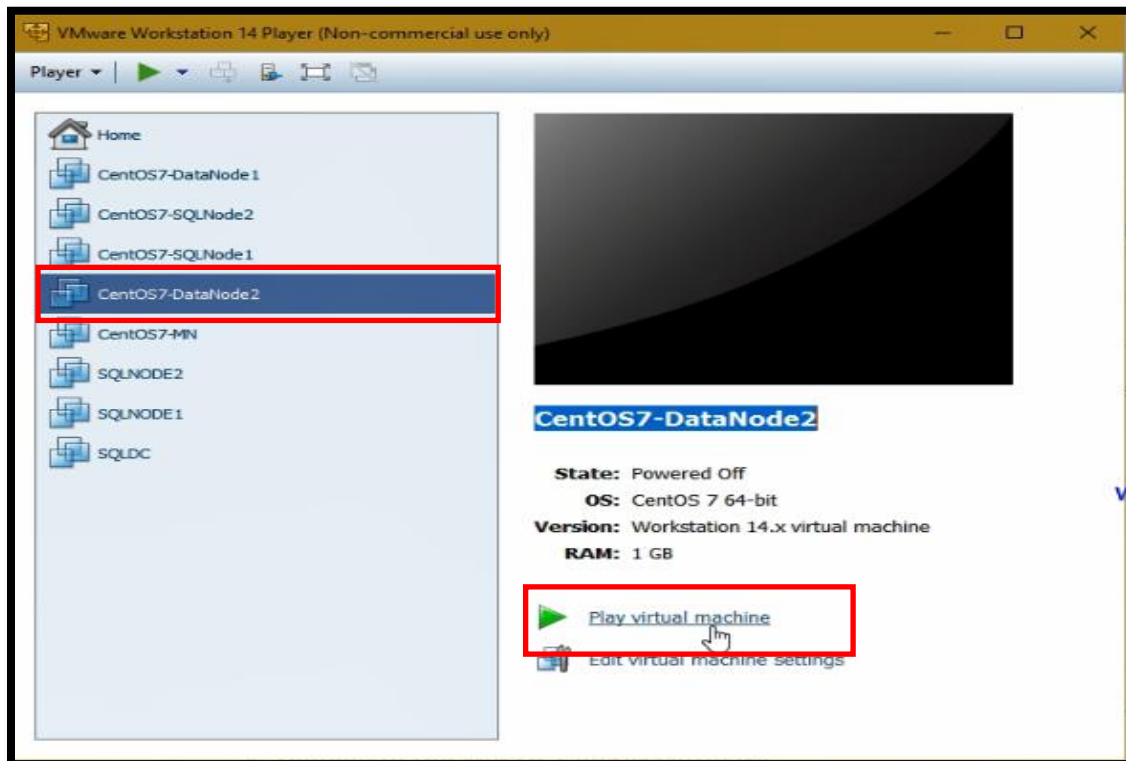
Instead of splitting the virtual disk files select to store in a single file and click Next to continue.



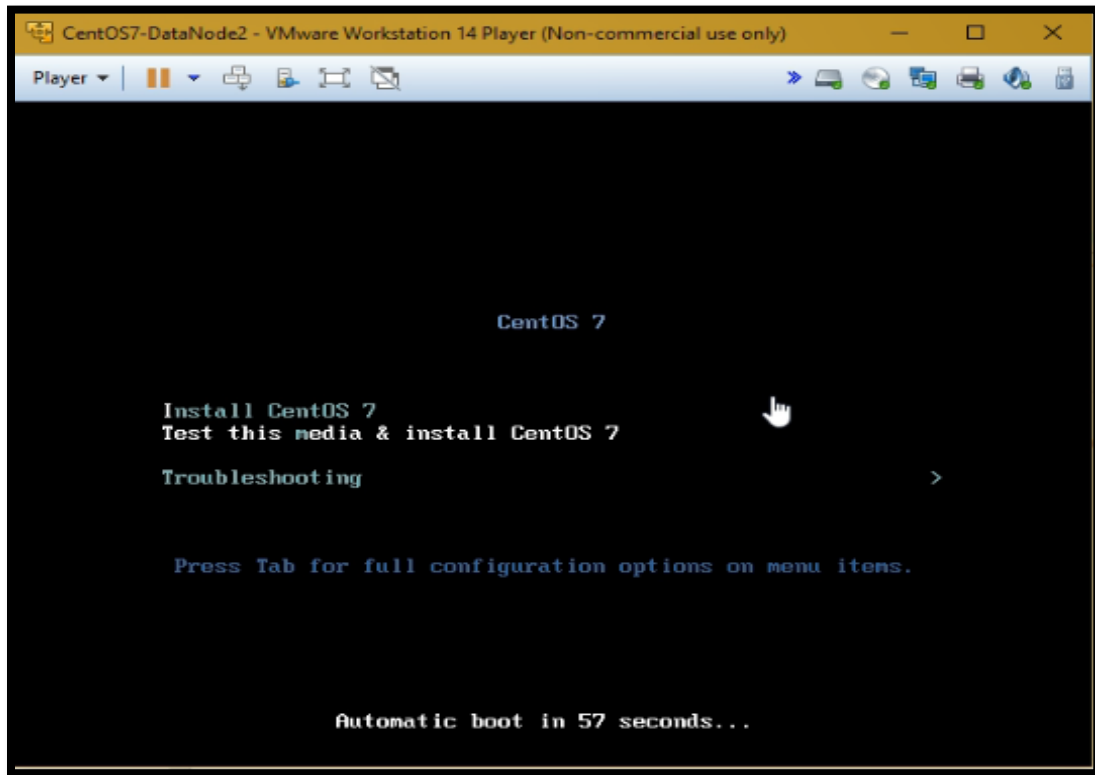
Click Finish to complete the creation of the Virtual Machine.



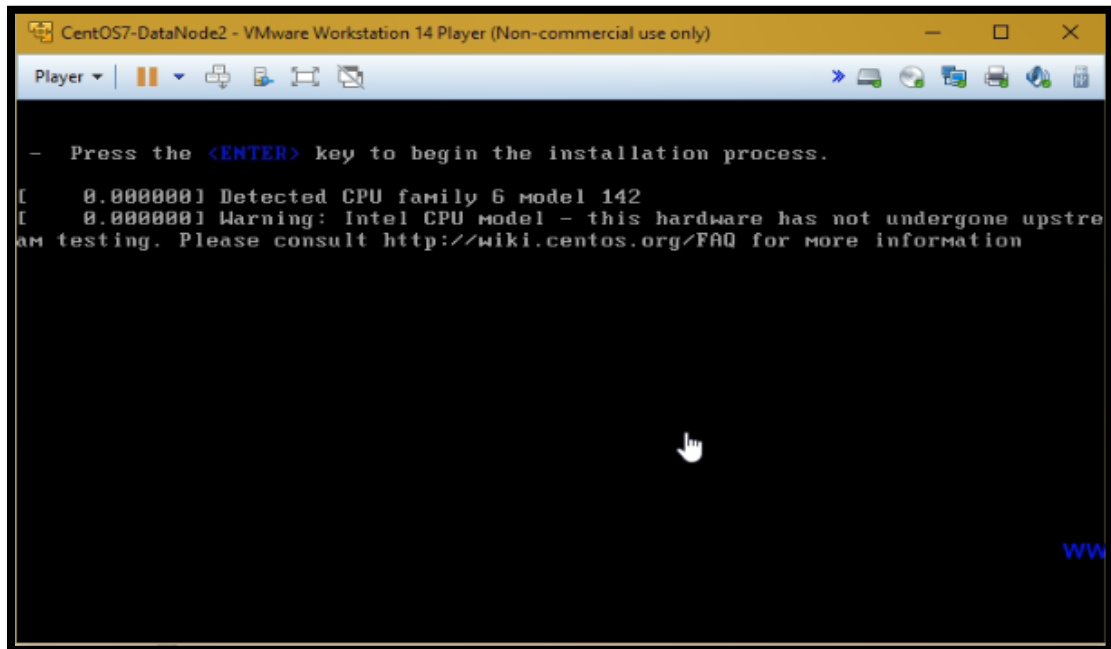
Select the Virtual Machine and Click on **Play virtual machine**

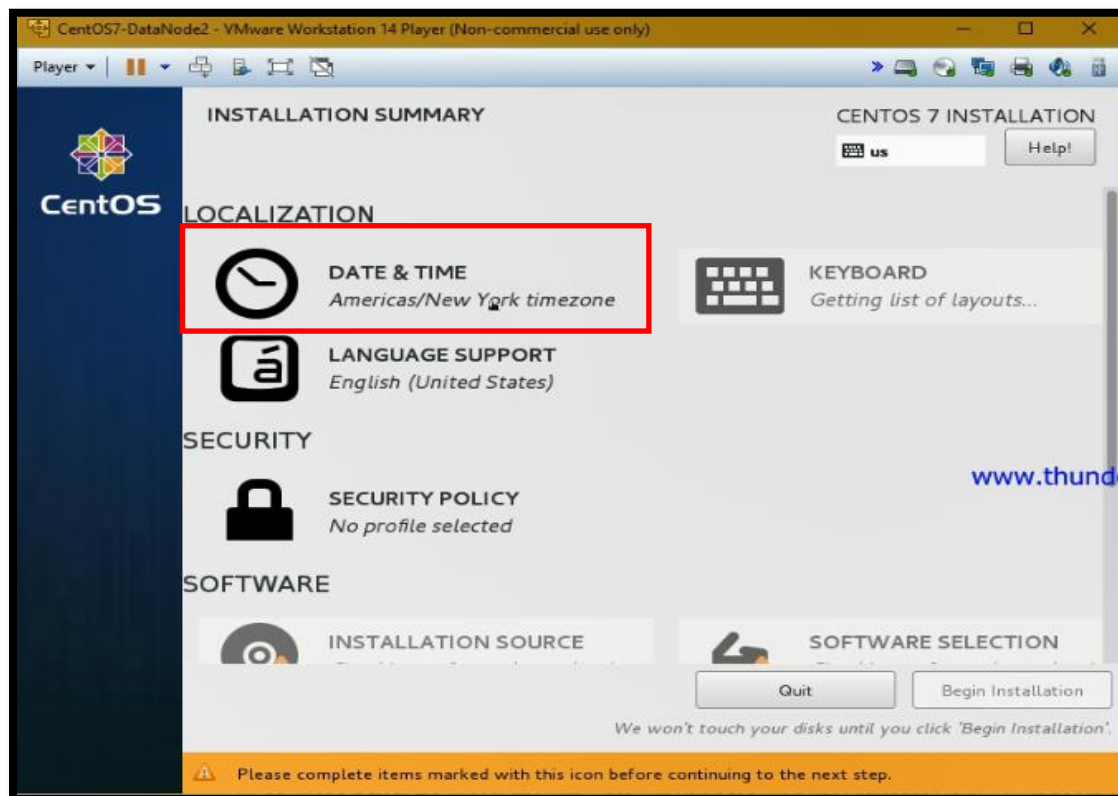
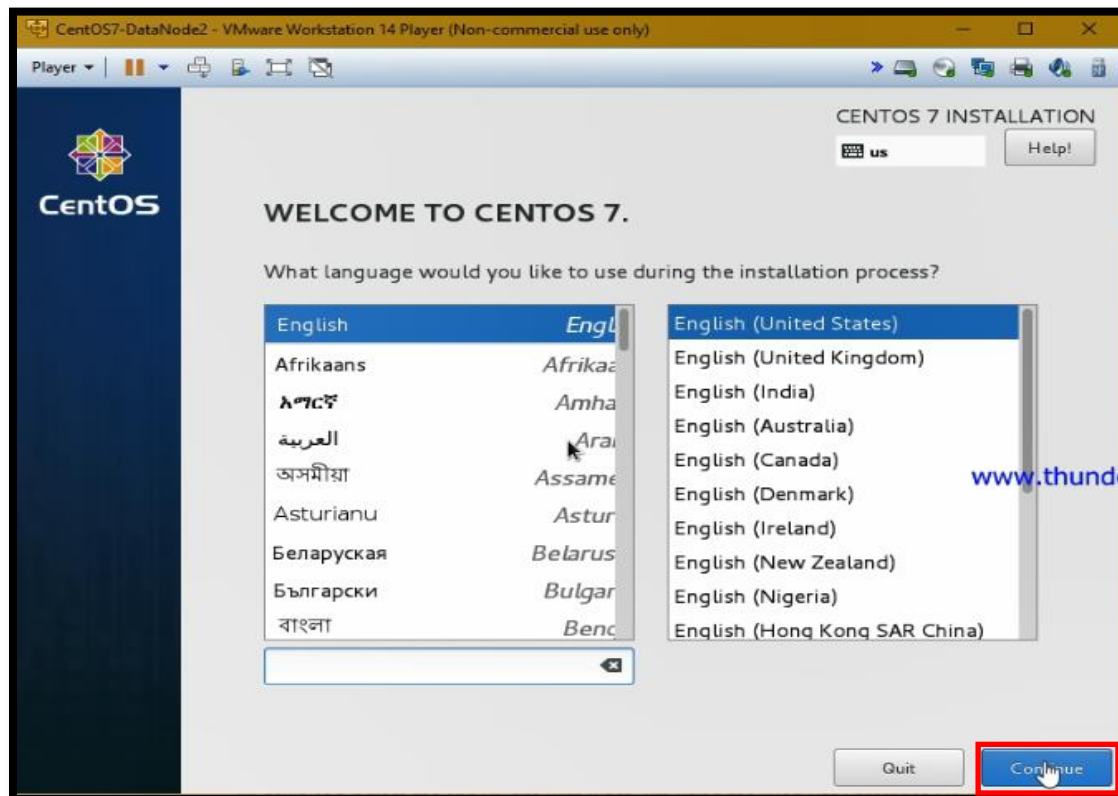


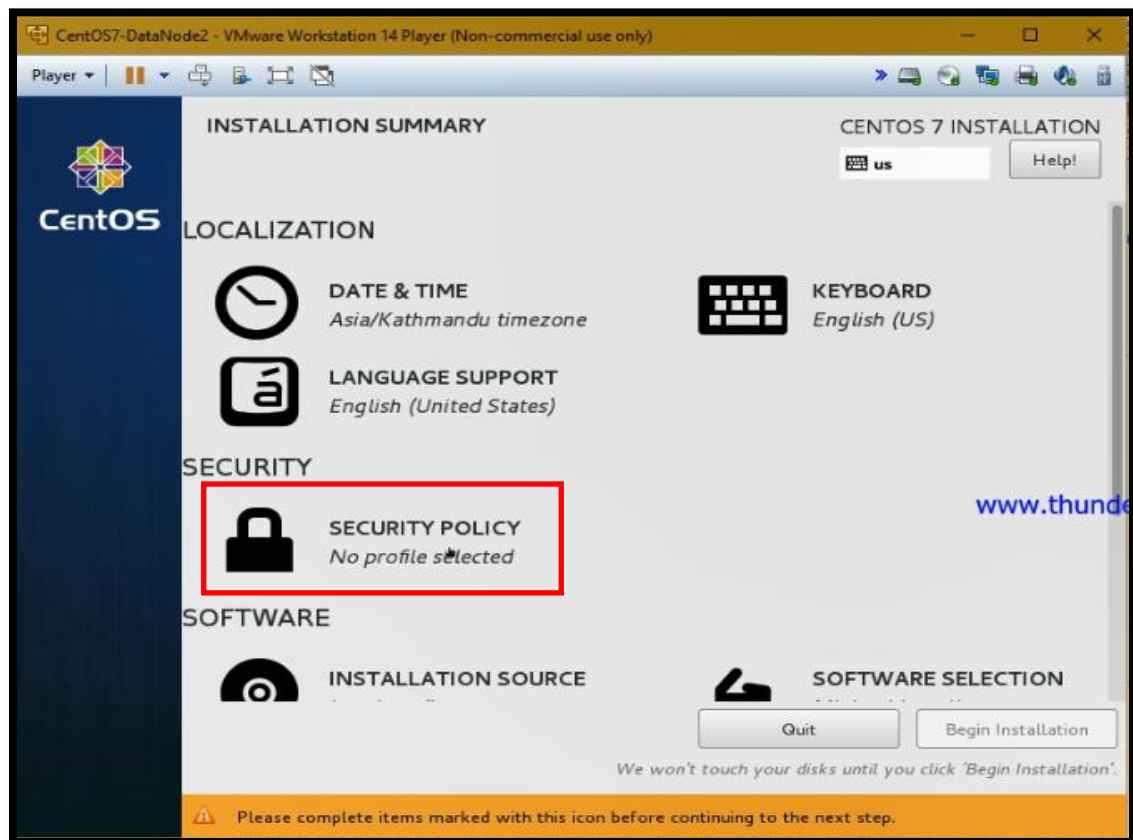
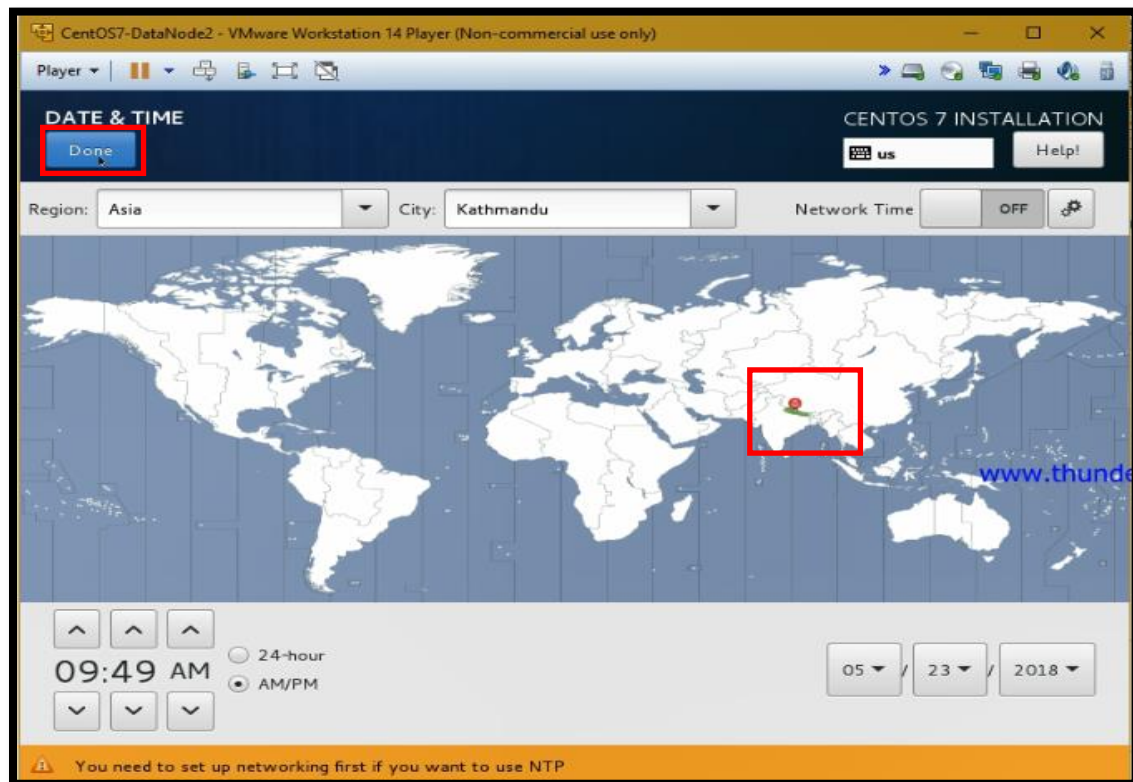
9 | MySQL Cluster on CentOS-7

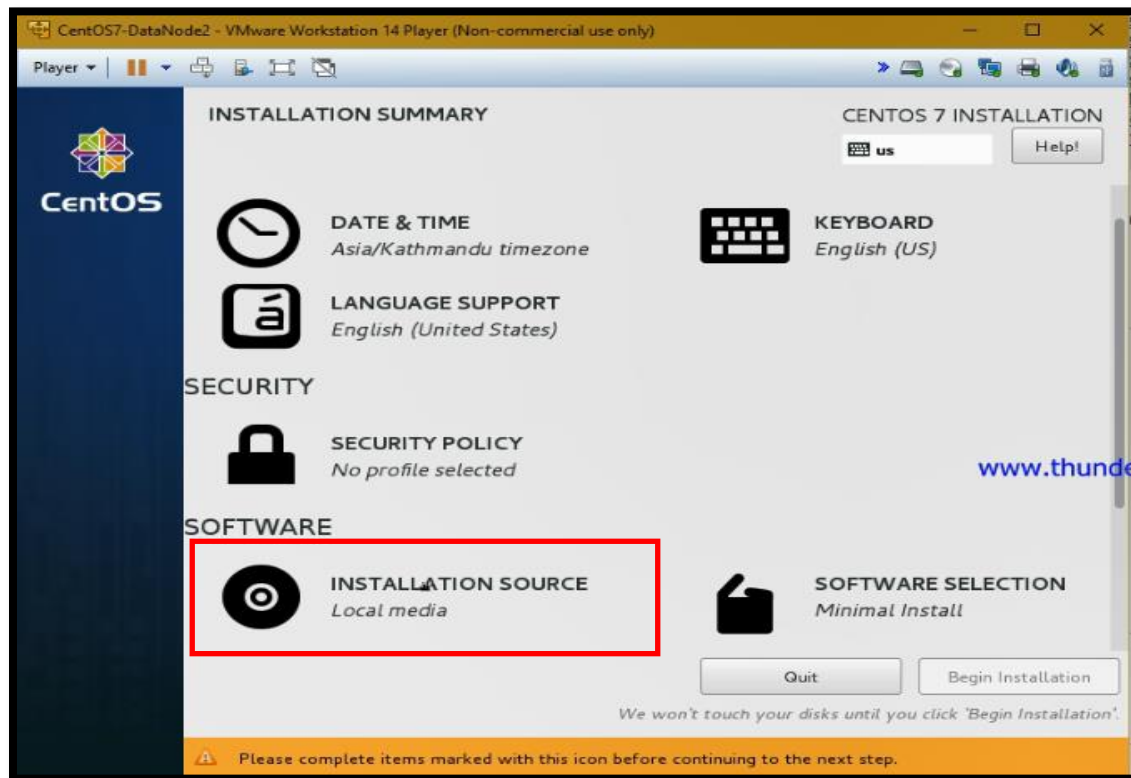
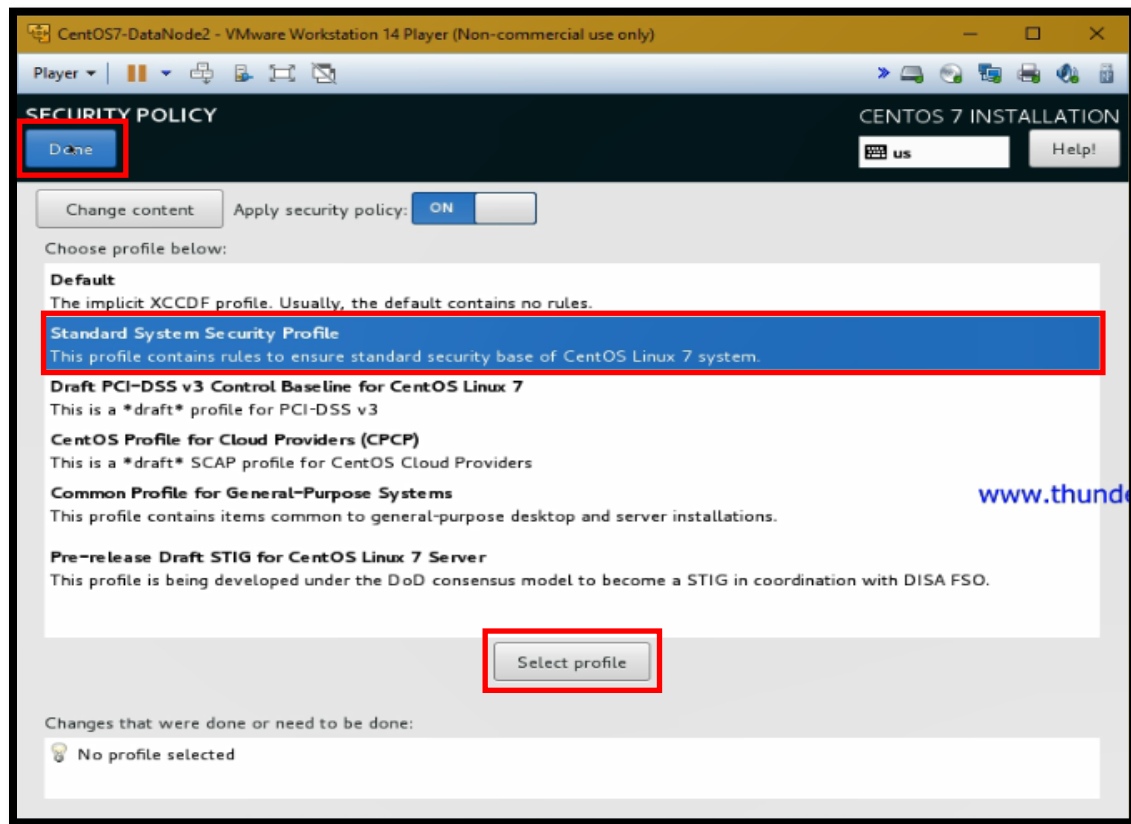


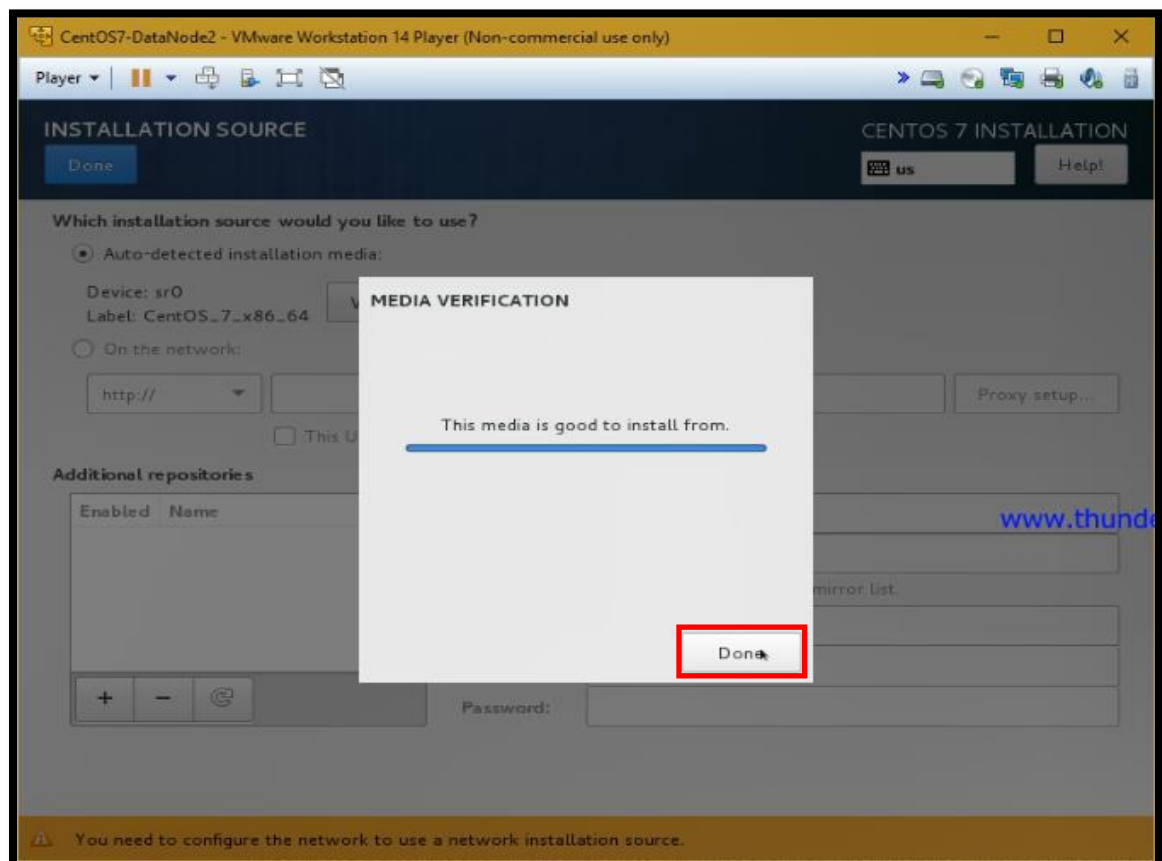
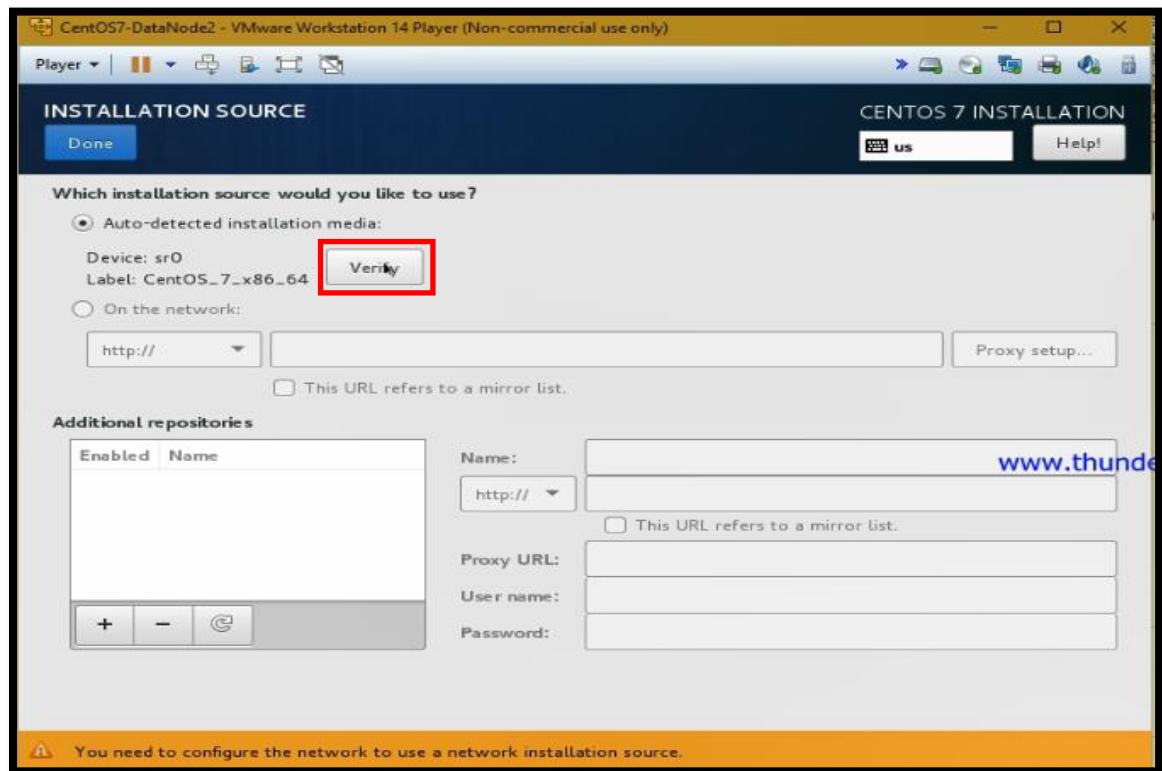
Press **Enter** key to continue.

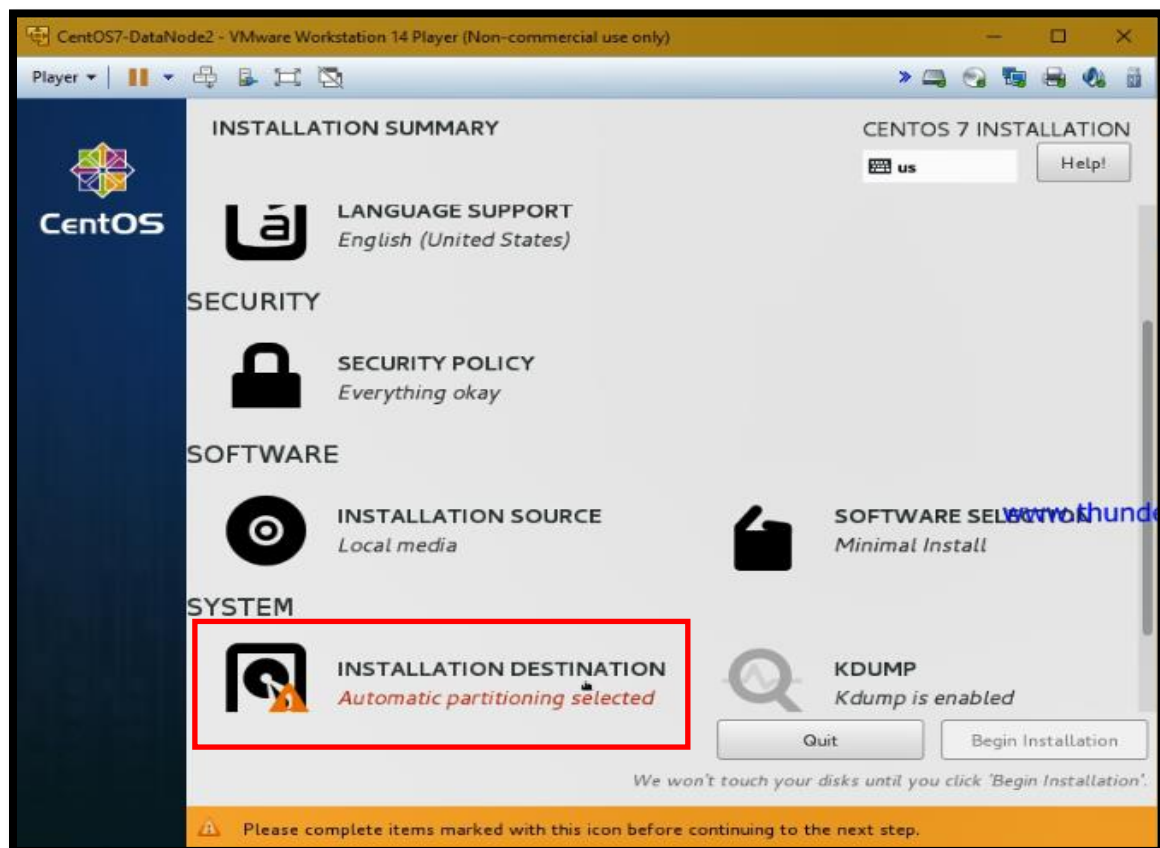
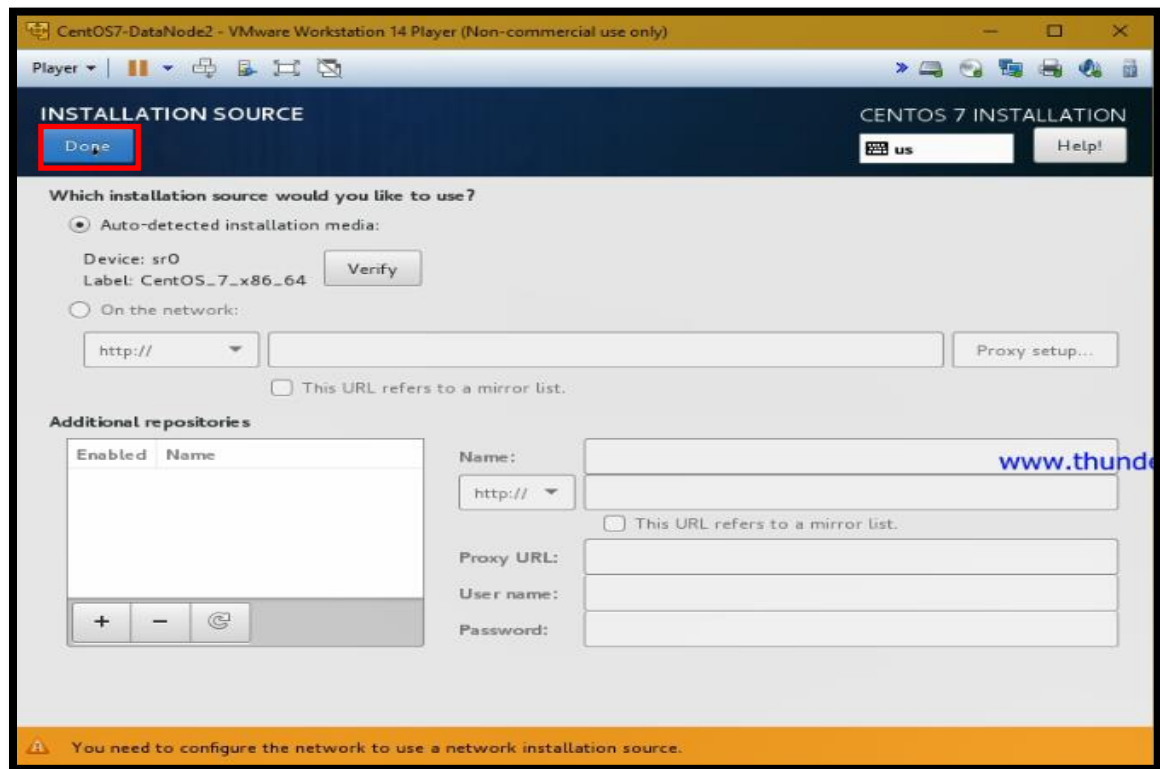


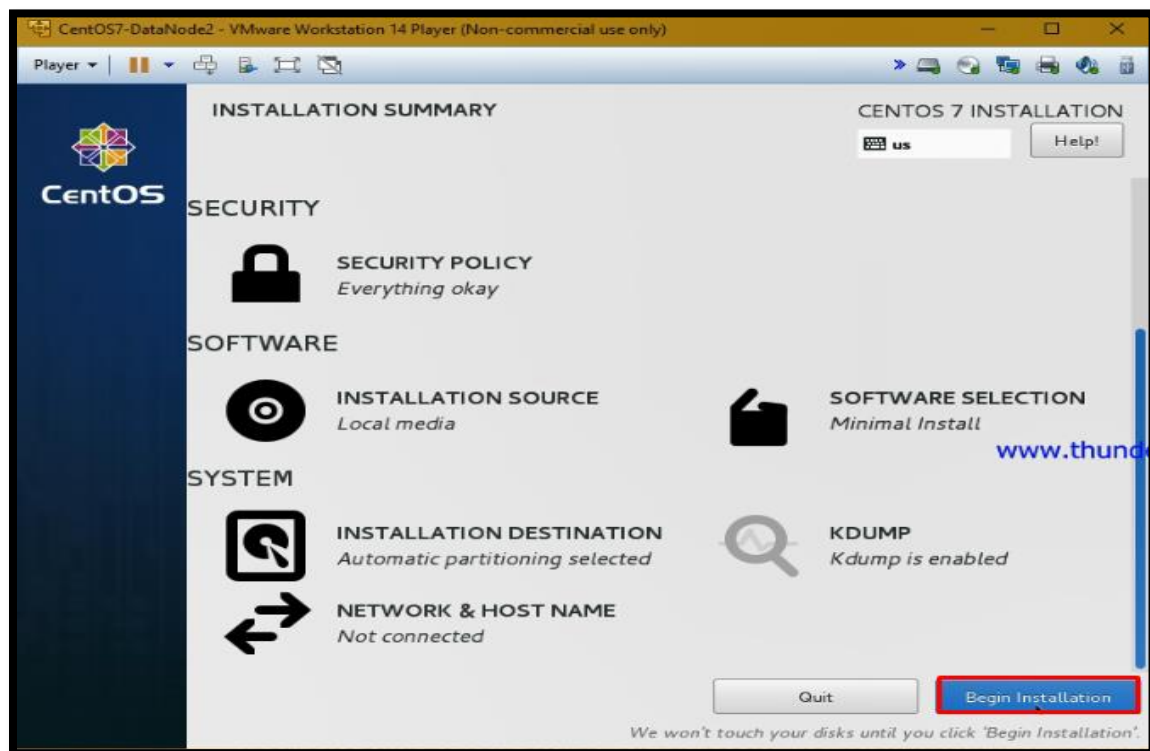
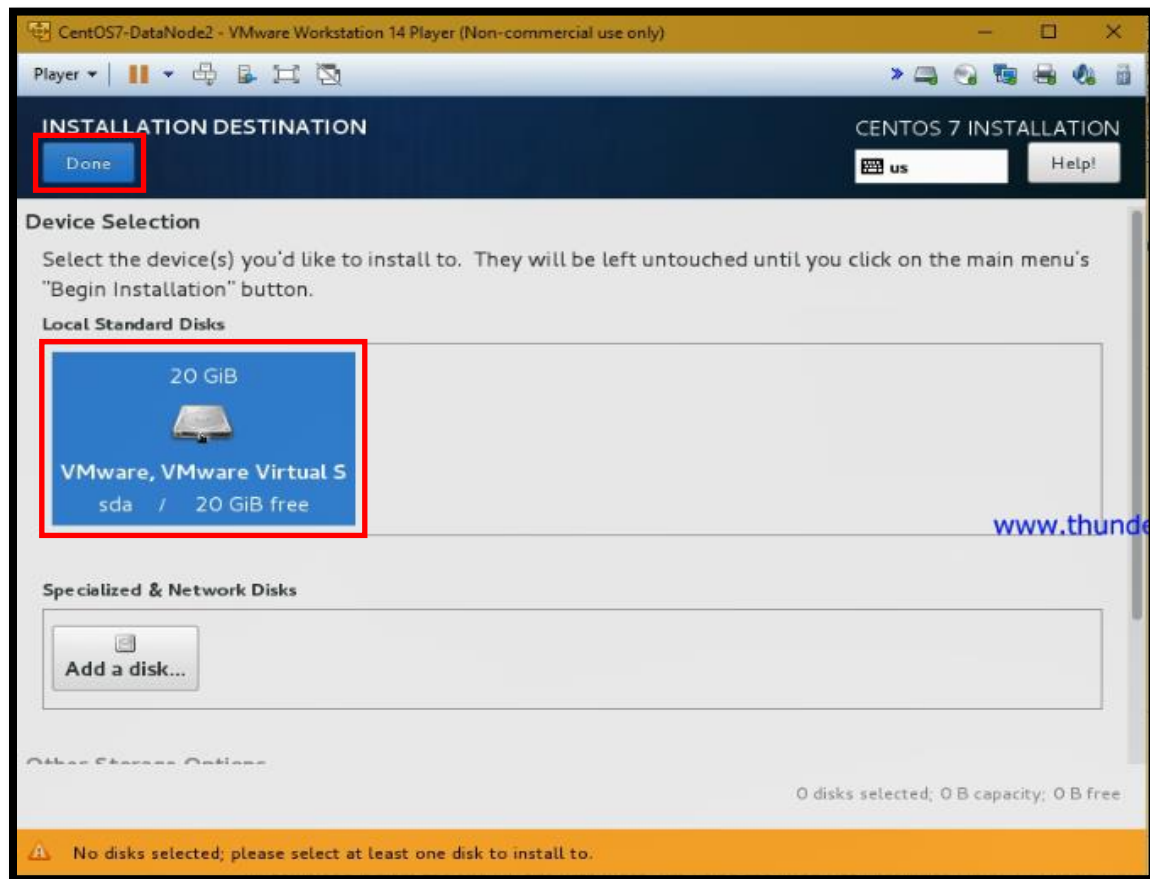


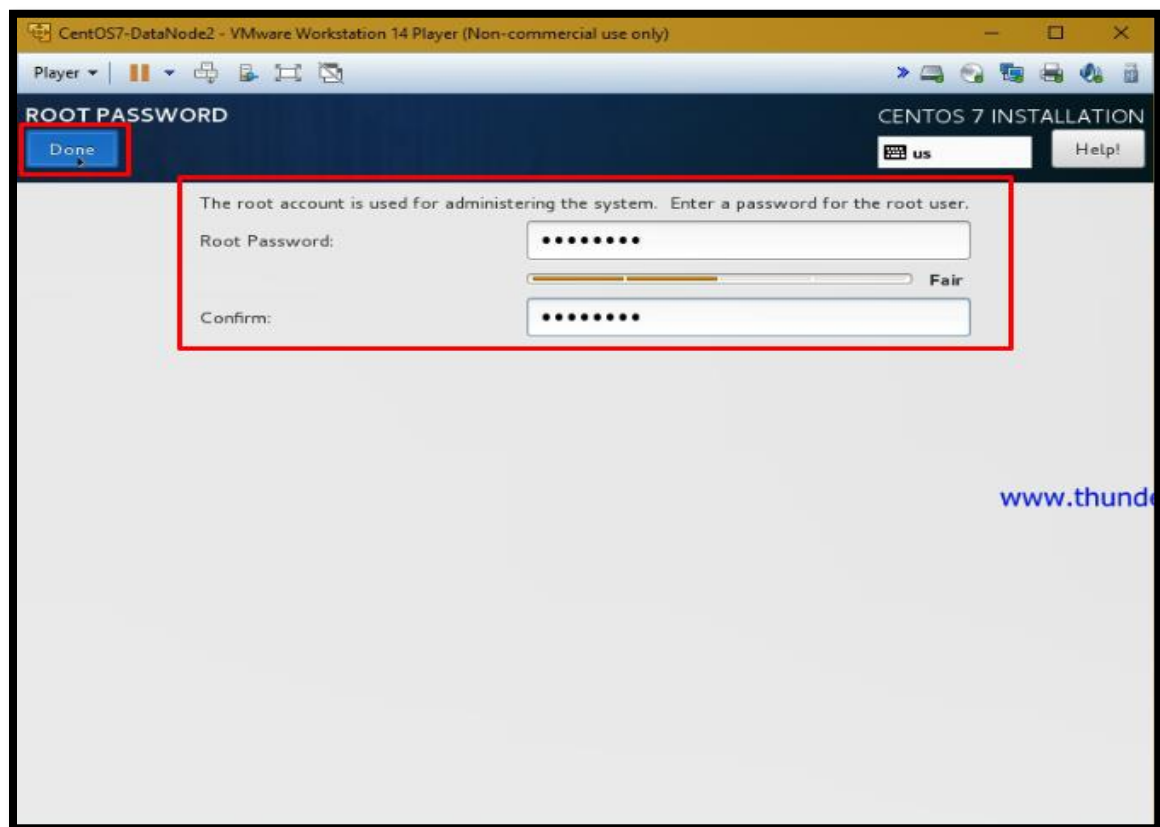
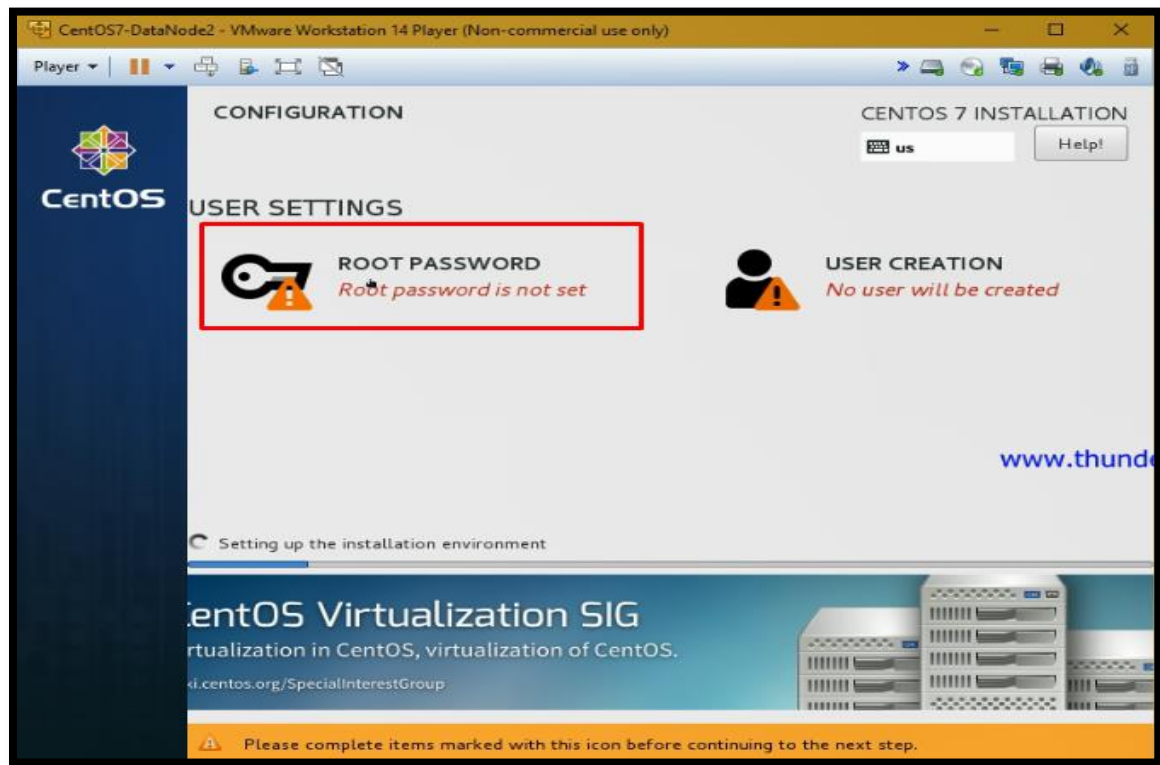


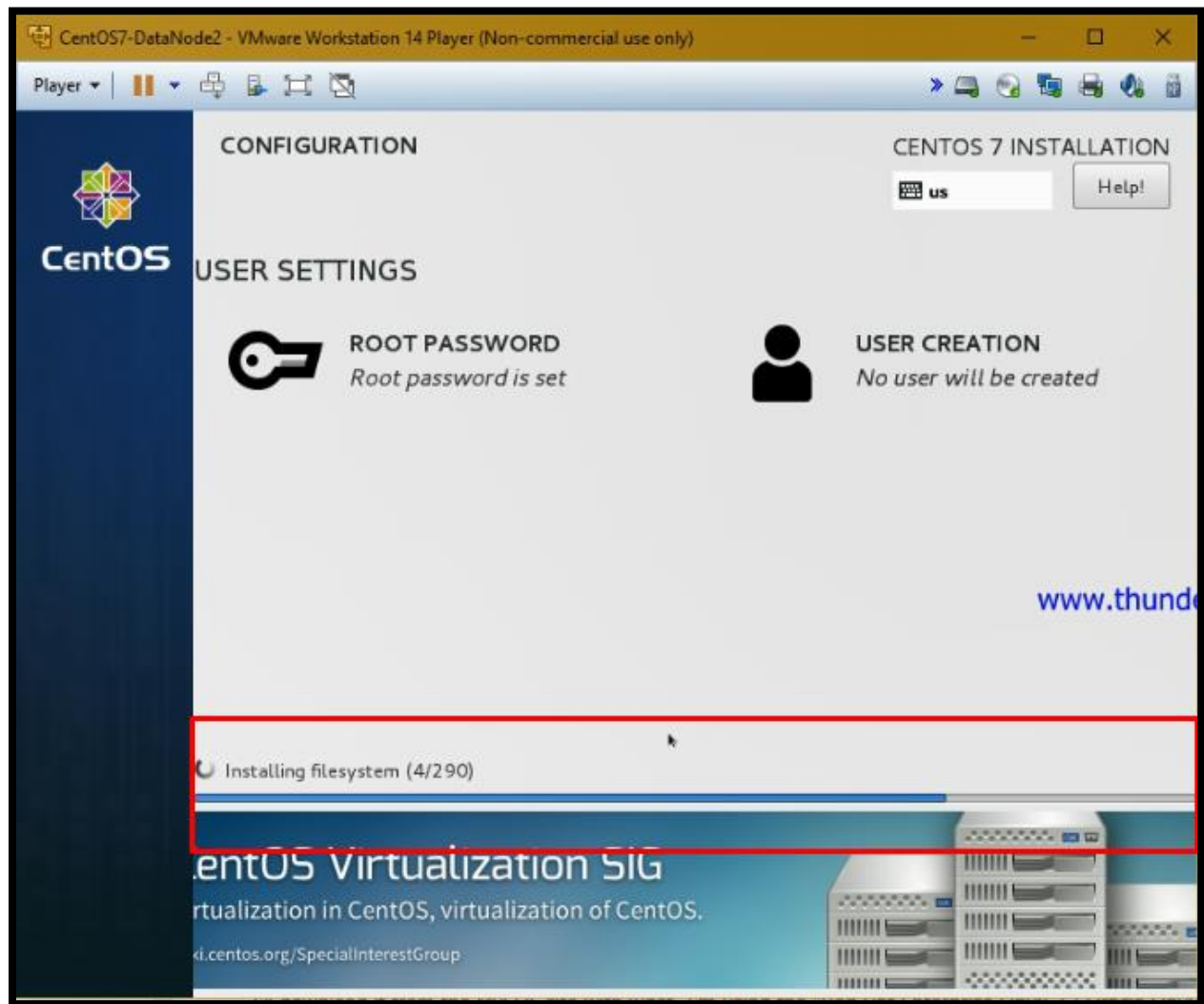












Change the host name before you start the process as readable hostname helps to identify the server easily. To change the hostname as per the desired name follow the link below for more info:
<https://github.com/sumanpantha/Linux-Research/blob/master/Change%20HostName>
Also update all the OS with command: `yum update`.

Syntax To change hostname:
(root@localhost ~] # hostnamectl set-hostname your-new-hostname)

For Management Nodes:

```
[root@localhost ~] # hostnamectl set-hostname centosmn
```

For Data Nodes:

```
[root@localhost ~] # hostnamectl set-hostname datanode1
```

```
[root@localhost ~] # hostnamectl set-hostname datanode2
```

For SQL Nodes:

```
[root@localhost ~] # hostnamectl set-hostname sqlnode1
```

```
[root@localhost ~] # hostnamectl set-hostname sqlnode2
```

- Need to Disable SELinux in all the nodes:
To Disable SELinux

```
[root@centos-mn ~] # vi /etc/sysconfig/selinux
```

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#   enforcing - SELinux security policy is enforced.
#   permissive - SELinux prints warnings instead of enforcing.
#   disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of these two values:
#   targeted - Targeted processes are protected,
#   minimum - Modification of targeted policy. Only selected processes are protected.
#   mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Then change the directive **SELinux=enforcing** to **SELinux=disabled** as shown in the below

```
SELINUX=disabled
```

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#   enforcing - SELinux security policy is enforced.
#   permissive - SELinux prints warnings instead of enforcing.
#   disabled - No SELinux policy is loaded.
SELINUX=disabled
# SELINUXTYPE= can take one of these two values:
#   targeted - Targeted processes are protected,
#   minimum - Modification of targeted policy. Only selected processes are protected.
#   mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Then, save and exit the file, for the changes to take effect, you need to reboot your system and then check the status of SELinux using **sestatus** command as shown:

```
[root@centos-mn ~] # sestatus
```

```
[root@centosmn ~]# sestatus
SELinux status:                disabled
[root@centosmn ~]#
```

Step 1 - Setup Management Node

The first step is to create the **Management Node** with CentOS 7 **centosms** and IP *192.168.19.130*. Make sure you are logged into the **datanode1** server as root user.

A. Download the MySQL Cluster software

I'll download it from the MySQL site with wget before download we need to install wget. I'm using the "CentOS Linux 7 (x86, 64-bit), RPM Bundle " here. Then extract the tar file.

```
[root@centos-mn ~] # yum install wget
[root@centos-mn ~] # wget http://dev.mysql.com/get/Downloads/MySQL-Cluster-7.4/MySQL-Cluster-gpl-7.4.10-1.el7.x86\_64.rpm-bundle.tar
[root@centos-mn ~] # tar -xvf MySQL-Cluster-gpl-7.4.10-1.el7.x86_64.rpm-bundle.tar
```

B. Install and Remove Packages

Before you install the rpm package for MySQL Cluster, you need to install *perl-Data-Dumper* that is required by the MySQL-Cluster server. And you need to remove *MariaDB-libs* before we can install MySQL Cluster.

```
[root@centos-mn ~] # yum install perl-Data-Dumper
[root@centos-mn ~] # yum remove mariadb-libs
[root@centos-mn ~] # yum install net-tools
```

C. Install MySQL Cluster

Install MySQL Cluster package with these rpm commands:

```
[root@centos-mn ~] # rpm -Uvh MySQL-Cluster-client-gpl-7.4.10-1.el7.x86_64.rpm
[root@centos-mn ~] # rpm -Uvh MySQL-Cluster-server-gpl-7.4.10-1.el7.x86_64.rpm
[root@centos-mn ~] # rpm -Uvh MySQL-Cluster-shared-gpl-7.4.10-1.el7.x86_64.rpm
```

Make sure there is no error.

D. Configure MySQL Cluster

Create a new directory for the configuration files. I will use the `/var/lib/mysql-cluster` directory.

```
[root@centos-mn ~] # mkdir -p /var/lib/mysql-cluster
```

Then create new configuration file for the cluster management named `config.ini` in the `mysql-cluster` directory.

```
[root@centos-mn ~] # cd /var/lib/mysql-cluster  
[root@centos-mn ~] # vi config.ini
```

Paste the configuration below:

```
[ndb_mgmd default]  
# Directory for MGM node log files  
DataDir=/var/lib/mysql-cluster  
  
[ndb_mgmd]  
#Management Node centosmn  
HostName=192.168.19.130  
  
[ndbd default]  
NoOfReplicas=2      # Number of replicas  
DataMemory=256M     # Memory allocate for data storage  
IndexMemory=128M    # Memory allocate for index storage  
#Directory for Data Node  
DataDir=/var/lib/mysql-cluster  
  
[ndbd]  
#Data Node datanode1  
HostName=192.168.19.129  
  
[ndbd]  
#Data Node datanode2  
HostName=192.168.19.131  
  
[mysqld]  
#SQL Node sqlnode1  
HostName=192.168.19.132  
  
[mysqld]  
#SQL Node sqlnode2  
HostName=192.168.19.133
```

Save the file and exit.

E. Start the Management Node

Next start the management node with the command below:

```
[root@centos-mn ~] # ndb_mgmd --config-file=/var/lib/mysql-cluster/config.ini
```

The result should be similar to this:

```
MySQL Cluster Management Server mysql-5.6.28 ndb-7.4.10
2018-05-24 19:26:08 [MgmtSrvr] INFO      -- The default config directory
'/usr/mysql-cluster' does not exist. Trying to create it...
2018-05-24 19:26:08 [MgmtSrvr] INFO      -- Successfully created config
directory
```

The management node is started, now you can use command **ndb_mgm** to monitor the node:

```
[root@centos-mn ~] # ndb_mgm
ndb_mgm> show
```

```
[root@centosmn mysql-cluster]# ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm> show
Connected to Management Server at: localhost:1186
Cluster Configuration
-----
[ndbd(NDB)]      2 node(s)
id=2 (not connected, accepting connect from 192.168.19.129)
id=3 (not connected, accepting connect from 192.168.19.131)

[ndb_mgmd(MGM)]  1 node(s)
id=1 @192.168.19.130 (mysql-5.6.28 ndb-7.4.10)

[mysqld(API)]    2 node(s)
id=4 (not connected, accepting connect from 192.168.19.132)
id=5 (not connected, accepting connect from 192.168.19.133)
```

You can see the management node has been started with: mysql-5.6 and ndb-7.4.

Step 2 - Setup the MySQL Cluster Data Nodes

We will use 2 CentOS servers for the Data Nodes.

1. **datanode1** = 192.168.19.129
2. **datanode2** = 192.168.19.131

A. Login as root user and download the MySQL Cluster software

Login to the **datanode1** server with ssh:

```
[root@centos-mn ~] # ssh root@192.168.19.129
```

Then download the MySQL Cluster package and extract it:

```
[root@datanode1 ~] # yum install wget
```

```
[root@datanode1 ~] # wget http://dev.mysql.com/get/Downloads/MySQL-Cluster-7.4/MySQL-Cluster-gpl-7.4.10-1.el7.x86\_64.rpm-bundle.tar
```

```
[root@datanode1 ~] # tar -xvf MySQL-Cluster-gpl-7.4.10-1.el7.x86_64.rpm-bundle.tar
```

B. Install and Remove Packages

Install perl-Data-Dumper and remove the mariadb-libs also need to install net-tools for the connection between the nodes:

```
[root@datanode1 ~] # yum install perl-Data-Dumper
```

```
[root@datanode1 ~] # yum remove mariadb-libs
```

```
[root@centos-mn ~] # yum install net-tools
```

C. Install MySQL Cluster

Now we can install the MySQL Cluster packages for the Data Nodes with these rpm commands:

```
[root@datanode1 ~] # rpm -Uvh MySQL-Cluster-client-gpl-7.4.10-1.el7.x86_64.rpm

[root@datanode1 ~] # rpm -Uvh MySQL-Cluster-server-gpl-7.4.10-1.el7.x86_64.rpm

[root@datanode1 ~] # rpm -Uvh MySQL-Cluster-shared-gpl-7.4.10-1.el7.x86_64.rpm
```

Make sure there is no error.

D. Configure Data Node

Create a new configuration file in the /etc directory with the vi editor:

```
[root@datanode1 ~] # vi /etc/my.cnf
```

Paste configuration below:

```
[mysqld]
ndbcluster
ndb-connectstring=192.168.19.130      # IP address of Management Node

[mysql_cluster]
ndb-connectstring=192.168.19.130      # IP address of Management Node
```

Save the file and exit.

Then create the new directory for the database data that we defined in the management node config file **config.ini**.

```
[root@datanode1 ~] # mkdir -p /var/lib/mysql-cluster
```

Now start the data node/ndbd:

```
[root@datanode1 ~] # ndbd
```


Results:

datanode1

```
[root@datanode1 ~]# ndbd
2018-05-25 10:17:08 [ndbd] INFO      -- Angel connected to '192.168.19.130:1186'
2018-05-25 10:17:08 [ndbd] INFO      -- Angel allocated nodeid: 2
[root@datanode1 ~]# _
```

datanode2

```
[root@datanode2 ~]# ndbd
2018-05-25 10:16:52 [ndbd] INFO      -- Angel connected to '192.168.19.130:1186'
2018-05-25 10:16:52 [ndbd] INFO      -- Angel allocated nodeid: 3
[root@datanode2 ~]# _
```

Data Node **datanode1** connected to the management node IP **192.168.19.130**.

E. Redo step 2.A - 2.D on db3 server.

As we have 2 data nodes, please redo the steps 2.A - 2.D on our second data node.

Step 3 - Setup SQL Node

This step contains the setup for the SQL Node that provides the application access to the database. We use 2 CentOS servers for the SQL Nodes:

1. **sqlnode1** = 192.168.19.132
2. **sqlnode2** = 192.168.19.133

A. Log in and Download MySQL Cluster

Login to the **sqlnode1** server as **root** user:

```
[root@centosmn ~] # ssh root@192.168.19.132
```

And download MySQL Cluster package:

```
[root@sqlnode1 ~] # yum install wget
[root@sqlnode1 ~] # wget http://dev.mysql.com/get/Downloads/MySQL-Cluster-7.4/MySQL-Cluster-gpl-7.4.10-1.el7.x86\_64.rpm-bundle.tar
[root@sqlnode1 ~] # tar -xvf MySQL-Cluster-gpl-7.4.10-1.el7.x86_64.rpm-bundle.tar
```

B. Install and Remove Packages

Install perl-Data-Dumper and remove the mariadb-libs that conflict with MySQL Cluster.

```
[root@sqlnode1 ~] # yum install perl-Data-Dumper
[root@sqlnode1 ~] # yum remove mariadb-libs
[root@sqlnode1 ~] # yum install net-tools
```

C. Install MySQL Cluster

Install the MySQL Cluster server, client and shared package with the rpm commands below:

```
[root@sqlnode1 ~] # rpm -Uvh MySQL-Cluster-client-gpl-7.4.10-1.el7.x86_64.rpm
[root@sqlnode1 ~] # rpm -Uvh MySQL-Cluster-server-gpl-7.4.10-1.el7.x86_64.rpm
[root@sqlnode1 ~] # rpm -Uvh MySQL-Cluster-shared-gpl-7.4.10-1.el7.x86_64.rpm
```

D. Configure the SQL Node

Create a new **my.cnf** file in the **/etc** directory:

```
[root@sqlnode1 ~] # vi /etc/my.cnf
```

And paste configuration below:

```
[mysqld]
ndbcluster
ndb-connectstring=192.168.19.130      # IP address for server management node
default_storage_engine=ndbcluster    # Define default Storage Engine used by MySQL

[mysql_cluster]
ndb-connectstring=192.168.19.130      # IP address for server management node
```

Save the file and exit the editor.

Start the SQL Node by starting the MySQL server:

```
[root@sqlnode1 ~] # service mysql start
```

Start **sqlnode1**

```
[root@sqlnode1 ~]# mysqld_safe &
[1] 1711
[root@sqlnode1 ~]# 180525 22:32:09 mysqld_safe Logging to '/var/lib/mysql/sqlnode1.err'.
180525 22:32:09 mysqld_safe Starting mysqld daemon with databases from /var/lib/mysql
```

Start **sqlnode2**

```
[root@sqlnode2 ~]# mysqld_safe &
[1] 10649
[root@sqlnode2 ~]# 180525 10:36:57 mysqld_safe Logging to '/var/lib/mysql/sqlnode2.err'.
180525 10:36:57 mysqld_safe Starting mysqld daemon with databases from /var/lib/mysql
```

Disable Firewall or allow the port used for MySQL database (3306) and cluster port
(<https://bytetfreaks.net/gnulinux/how-to-startstop-or-enabledisable-firewalld-on-centos-7>)
Allow mysql port on both the SQL Nodes **sqlnode1** and **sqlnode2**.

```
[root@sqlnode1 ~] # firewall-cmd --permanent --add-port=3306/tcp

[root@sqlnode1 ~] # firewall-cmd --reload
```

E. Redo step 3.A - 3.D on db5 server.

Please redo the steps 3.A - 3.D on the second SQL server (**sqlnode2**).

Step 4 - Monitor the Cluster

To see the cluster status, we have to log into the management node `centosmn`.

```
ssh root@192.168.19.130
```

We can use the `ndb_mgm` command to see the cluster status:

```
[root@centos-mn ~] # ndb_mgm
ndb_mgm> show
```

```
[root@centosmn ~]# ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm> show
Connected to Management Server at: localhost:1186
Cluster Configuration
-----
[ndbd(NDB)] 2 node(s)
id=2 @192.168.19.129 (mysql-5.6.28 ndb-7.4.10, Nodegroup: 0, *)
id=3 @192.168.19.131 (mysql-5.6.28 ndb-7.4.10, Nodegroup: 0)

[ndb_mgmd(MGM)] 1 node(s)
id=1 @192.168.19.130 (mysql-5.6.28 ndb-7.4.10)

[mysqld(API)] 2 node(s)
id=4 @192.168.19.132 (mysql-5.6.28 ndb-7.4.10)
id=5 @192.168.19.133 (mysql-5.6.28 ndb-7.4.10)
```

Another useful command is:

```
ndb_mgm -e "all status"
ndb_mgm -e "all report memory"
```

Step 5 - Testing the Cluster

To perform a test on our new MySQL Cluster, we have to login to the SQL Nodes **sqlnode1** or **sqlnode2** servers.

Login to the **sqlnode1** server:

```
ssh root@192.168.19.132
```

Change the default MySQL password that stored in **.mysql_secret** file in root directory:

```
[root@sqlnode1 ~] # cat .mysql_secret
```

This is my sample:

```
# The random password set for the root user at Tue Mar 22 19:44:07 2016  
(local time): qna3AwbJMuOnw23T
```

Now change the password with command below:

```
[root@sqlnode1 ~] # mysql_secure_installation
```

Type your old MySQL password and then type the new one, press enter to confirm all.
If all is done, you can login to the MySQL shell with your password:

```
[root@sqlnode1 ~] # mysql -u root -p
```

After you logged in, create a new root user with host "**@**", so we will be able to access the MySQL from outside.

```
mysql> CREATE USER 'root'@'%' IDENTIFIED BY 'admin123';
```

Replace *admin123* with your own secure password! Now you can see the new root user with host "**@**" on the MySQL user list:

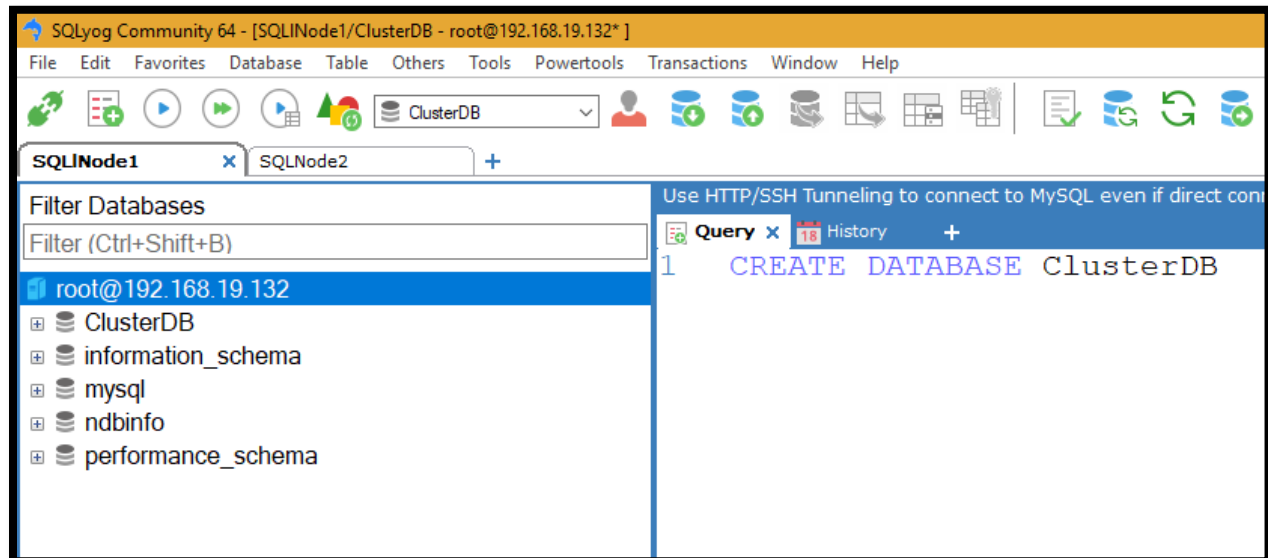
```
mysql> select user, host, password from mysql.user;
```

And grant the new root user read and write access from the remote node:

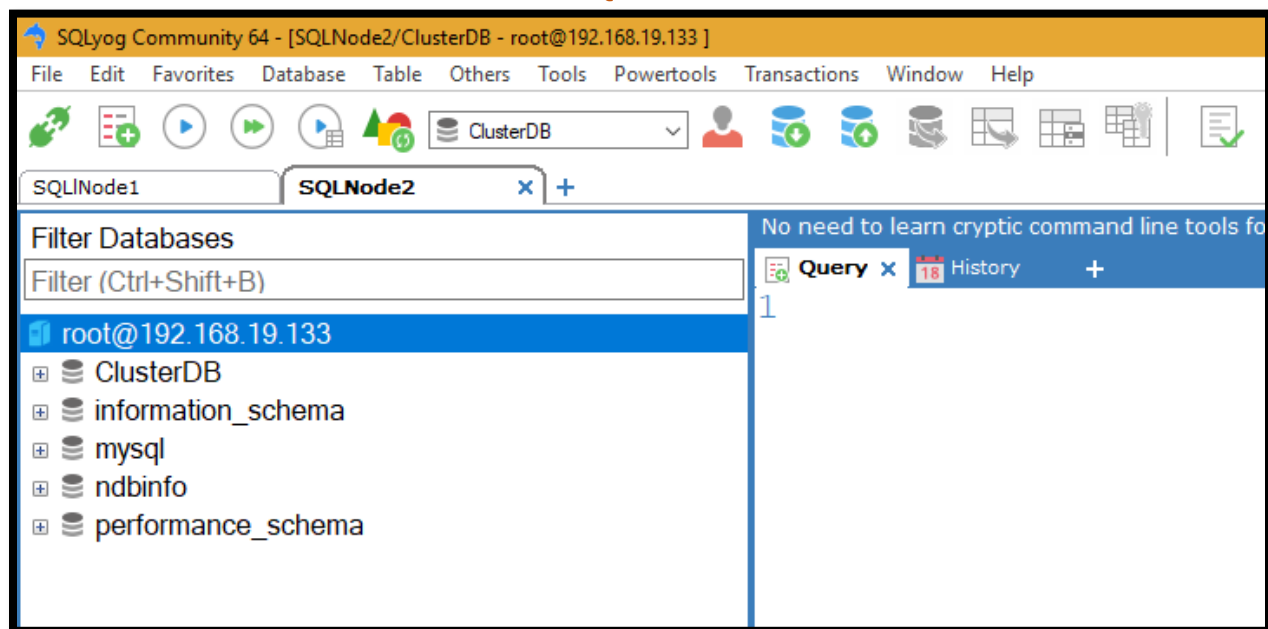
```
GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY PASSWORD  
'*94CC7BF027327993D738E11...(Encrypted PASSWORD)' WITH GRANT OPTION;
```

Now try to create a new database(ClusterDB) from **sqlnode1** server and you will see the database on **sqlnode2** too. This is just a sample result for testing the cluster data replication.

SQLNODE1



SQLNODE2



The MySQL Cluster has been setup successfully on CentOS 7 with 5 server nodes.

Conclusion

MySQL Cluster is a technology that provides High Availability and Redundancy for MySQL databases. It uses NDB or NDBCLUSTER as the storage engine and provides shared-nothing clustering and auto-sharing for MySQL databases. To implement the cluster, we need 3 components: Management Node(MGM), Data Nodes (NDB) and SQL Nodes (API). Each of node must have its own memory and disk. It is not recommended to use network storage such as NFS. To install MySQL Cluster on a CentOS 7 minimal system, we have to remove the mariadb-libs package, mariadb-libs conflict with MySQL-Cluster-server and you have to install the perl-Data-Dumper package, it's needed by MySQL-Cluster-server. A MySQL Cluster is easy to install and configure on multiple CentOS servers.

Percona XtraDB Cluster on CentOS 7

Installation of CentOS 7 (64X) Minimal in VMWare

As we plan to cluster 3 node Cluster need 3 Virtual Machine to implement the MySQL Clustering so we need to first install Operation System (CentOS 7). The steps to install CentOS in VMWare is represented by the screenshot on [MySQL Cluster on CentOS 7](#). We need to install CentOS in five different VM. So, repeat the steps to install the OS on three different VM.

For the very first time we need to create the Virtual Machine for the OS. And install OS on all three machine and continue the process then after.

How to Install Percona XtraDB Cluster on CentOS 7

This tutorial is all about installing and configuring Percona XtraDB Cluster on a CentOS 7 server. We will use Percona XtraDB Cluster 5.6 that is fully compatible to MySQL and the Percona Server. Percona is a company of MySQL and MongoDB database experts founded in 2006. Percona builds and maintains open source software for MySQL and MongoDB: the Percona Server (database server for MySQL with high availability performance enhancements), Percona XtraDB Cluster (high availability solution for MySQL cluster), Percona Server for MongoDB and other tools for managing the databases such as the Percona toolkit, Percona monitoring tools and Percona XtraBackup.

Prerequisite

- 3 CentOS 7 server nodes.
Percona1: 192.168.19.129
Percona2: 192.168.19.135
Percona3: 192.168.19.138
- Root privileges.
- Basic CentOS 7 knowledge.
- Change the host name before you start the process as readable hostname helps to identify the server easily. To change the hostname as per the desired name, follow the link below for more info: <https://github.com/sumanpantha/Linux-Research/blob/master/Change%20HostName>
Also update all the OS with command: `yum update`.

Syntax To change hostname:

```
(root@localhost ~] # hostnamectl set-hostname your-new-hostname)
```



```
[root@localhost ~] # hostnamectl set-hostname parcona1
[root@localhost ~] # hostnamectl set-hostname parcona2
[root@localhost ~] # hostnamectl set-hostname parcona3
```

- Need to Disable SELinux in all the nodes:
To Disable SELinux

```
[root@parcona1 ~] # vi /etc/sysconfig/selinux
```

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#   enforcing - SELinux security policy is enforced.
#   permissive - SELinux prints warnings instead of enforcing.
#   disabled - No SELinux policy is loaded.
SELINUX=enforcing
# SELINUXTYPE= can take one of three two values:
#   targeted - Targeted processes are protected,
#   minimum - Modification of targeted policy. Only selected processes are protected.
#   mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Then change the directive `SELINUX=enforcing` to `SELINUX=disabled` as shown in the below

```
SELINUX=disabled
```

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#   enforcing - SELinux security policy is enforced.
#   permissive - SELinux prints warnings instead of enforcing.
#   disabled - No SELinux policy is loaded.
#SELINUX=enforcing
SELINUX=disabled
# SELINUXTYPE= can take one of three two values:
#   targeted - Targeted processes are protected,
#   minimum - Modification of targeted policy. Only selected processes are protected.
#   mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

Then, save and exit the file, for the changes to take effect, you need to **reboot** your system and then check the status of SELinux using **sestatus** command as shown:

```
[root@percona1 ~] # sestatus
```

```
[root@percona1 ~]# sestatus
SELinux status: disabled
[root@percona1 ~]# _
```

Step 1 - Setup the hosts file

The first thing to do is to configure the hostnames of all servers. I've 3 servers with CentOS 7 as listed below:

Node		Server IP	Hostname
Node1	-	192.168.19.129	percona1
Node2	-	192.168.19.135	percona2
Node3	-	192.168.19.138	percona3

Connect to all servers by using your terminal:

```
ssh root@serverip
```

If you have logged into all server, edit the `'/etc/hosts'` file on each server with vim:

```
vim /etc/hosts
```

Paste the hosts configuration below:

```
192.168.19.129 percona1
192.168.19.135 percona2
192.168.19.138 percona3
```

Replace the IP addresses with the ones that match your local network configuration. Save and exit.

Step 2 - Configure Firewalld

Firewalld is the new default firewall interface on CentOS 7. The `firewall-cmd` command is used to configure the firewall. We can define and configure specific groups or zones, or we can configure a firewall for services like ssh, MySQL databases etc.

In this step, we will use firewalld for the firewall configuration. We will use the '`firewall-cmd`' command to open the percona server port and other ports that are needed for the cluster. Start firewalld with this systemctl command:

```
systemctl start firewalld
```

Then run the command below to open the port used by the MySQL/percona server:

```
firewall-cmd --zone=public --add-service=mysql --permanent
```

Next, add the other ports for the cluster with command below:

```
firewall-cmd --zone=public --add-port=3306/tcp --permanent
firewall-cmd --zone=public --add-port=4567/tcp --permanent
firewall-cmd --zone=public --add-port=4568/tcp --permanent
firewall-cmd --zone=public --add-port=4444/tcp --permanent
firewall-cmd --zone=public --add-port=4567/udp --permanent
```

Reload the firewall rules:

```
firewall-cmd --reload
```

To see the list of all the firewall rules, use the option '`--list-all`':

```
firewall-cm --list-all
```

```
[root@parconal ~]# firewall-cmd --list-all
public
  target: default
  icmp-block-inversion: no
  interfaces:
  sources:
  services: ssh dhcpv6-client
  ports: 3306/tcp 4567/tcp 4568/tcp 4444/tcp 4567/udp
  protocols:
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

Step 3 - Install the Epel Repository and Socat

To get Percona XtraDB Cluster running on the server, we need to install socat, and it's available in the epel-repository. So, we need to install the Epel repository first and then install socat. Further we have to remove the mariadb-libs from the server because they conflict with the Percona XtraDB Cluster.

Install epel-repository and socat:

```
yum -y install epel-release
yum -y install socat
```

Remove the mariadb-libs to avoid the package conflict between mariadb-libs and Percona XtraDB Cluster:

```
yum -y remove mariadb-libs
```

The Epel repository and socat are installed.

Step 4 - Install Percona XtraDB Cluster

In this step, we will install the Percona xtradb cluster with all package dependencies. We need to add the Percona repository for the installation and then we start the Percona server and configure the root user and password for the database server.

Install the Percona repository with yum:

```
yum -y install http://www.percona.com/downloads/percona-release/redhat/0.1-3/percona-release-0.1-3.noarch.rpm
```

Now install Percona XtraDB cluster and the other packages required for this tutorial:

```
yum install Percona-XtraDB-Cluster-server-56 Percona-XtraDB-Cluster-client-56 Percona-XtraDB-Cluster-shared-56 percona-toolkit percona-xtrabackup Percona-XtraDB-Cluster-galera-3 rsync nc
```

Percona XtraDB Cluster is installed, start the Percona server with this systemctl command:

```
Systemctl start mysql.service  
OR  
systemctl start mysql
```

Next, configure the root password for all percona/mysql servers:

```
mysql_secure_installation
```

Set the percona/mysql password:

```
Enter current password for root (enter for none): PRESS ENTER  
Set root password? [Y/n] Y  
New password: TYPE YOUR PASSWORD  
Re-enter new password: REPEAT PASSWORD  
Remove anonymous users? [Y/n] Y  
Disallow root login remotely? [Y/n] Y  
Reload privilege tables now? [Y/n] Y
```

Note:

Run step 1 - 4 on all 3 CentOS servers.

Step 5 - Configure Percona XtraDB Cluster

In step 4 we've already installed Percona XtraDB Cluster and configured the root password for all Percona/MySQL server nodes. In this step, we will create a new user for SST authentication and edit the MySQL configuration `my.cnf` on each of server.

SST (State Snapshot Transfer) is a full data copy from one server as a donor to another server as the joiner. For the SST authentication, we need to create a new user called `'8Square'` with password `'8Square@'`. And for the SST method, we will use `xtrabackup-v2` instead of `rsync`. Please use a different and secure password for your cluster!

Login to the percona/mysql shell on each server:

```
mysql -u root -p
TYPE YOUR PASSWORD
```

And create the new `'8Square'` with password `'8Square@'`:

```
create user '8square'@'%' identified by '8Square@';
grant all on *.* to 8square@'%';
flush privileges;
```

```
mysql> create user 8Square@'%' identified by '8Square@';
Query OK, 0 rows affected (0.01 sec)

mysql> grant all on *.* to 8Square@'%';
Query OK, 0 rows affected (0.01 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```

Then stop the MySQL service on each server before editing the config file:

```
systemctl stop mysql
```

Next, edit the mysql configuration file `my.cnf` on each server with the vi editor.

On **Percona1** server:

```
vi /etc/my.cnf
```

For the very first-time take the backup of default my.cnf file and replace my.cnf file with below my.cnf file.

```
# Template my.cnf for PXC
# Edit to your requirements.

[mysqld]

datadir=/var/lib/mysql

server_id =1

# Path to Galera library
wsrep_provider=/usr/lib64/libgalera_smm.so

# For the first node of the cluster we don't use wsrep cluster address.
wsrep_cluster_address=gcomm://

# In order for Galera to work correctly binlog format should be ROW
binlog_format=ROW

# MyISAM storage engine has only experimental support
default_storage_engine=InnoDB

# This InnoDB autoincrement locking mode is a requirement for Galera
innodb_autoinc_lock_mode=2

# Node 1 address
wsrep_node_address=192.168.19.129

# SST method
wsrep_sst_method=xtrabackup

# Cluster name
wsrep_cluster_name=my_centos_cluster

# Authentication for SST method
wsrep_sst_auth=8square:8Square@
```

Save and exit.

On **Percona2** server:

```
vim /etc/my.cnf
```

For the very first-time take the backup of default my.cnf file and replace my.cnf file with below my.cnf file.

Template my.cnf for PXC
Edit to your requirements.
[mysqld]
server_id =2
datadir=/var/lib/mysql
Path to Galera library
wsrep_provider=/usr/lib64/galera3/libgalera_smm.so
Cluster connection URL contains IPs of node#1, node#2 and node#3
wsrep_cluster_address=gcomm://192.168.19.129:4567
In order for Galera to work correctly binlog format should be ROW
binlog_format=ROW
MyISAM storage engine has only experimental support
default_storage_engine=InnoDB
This InnoDB autoincrement locking mode is a requirement for Galera
innodb_autoinc_lock_mode=2
Node 2 address
wsrep_node_address=percona2
Cluster name
wsrep_cluster_name=my_centos_cluster
SST method
wsrep_sst_method=xtrabackup
#Authentication for SST method
wsrep_sst_auth=8square:8Square@

Save and exit.

On **Percona3** server:

```
vim /etc/my.cnf
```

For the very first-time take the backup of default my.cnf file and replace my.cnf file with below my.cnf file.

Template my.cnf for PXC
Edit to your requirements.
[mysqld]
server_id = 3
datadir=/var/lib/mysql
Path to Galera library
wsrep_provider=/usr/lib64/libgalera_smm.so
Cluster connection URL contains IPs of node#1, node#2 and node#3
wsrep_cluster_address=gcomm://192.168.19.129:4567
In order for Galera to work correctly binlog format should be ROW
binlog_format=ROW
MyISAM storage engine has only experimental support
default_storage_engine=InnoDB
This InnoDB autoincrement locking mode is a requirement for Galera
innodb_autoinc_lock_mode=2
Node 3 address
wsrep_node_address=192.168.19.138
Cluster name
wsrep_cluster_name=my_centos_cluster
SST method
wsrep_sst_method=xtrabackup
#Authentication for SST method
wsrep_sst_auth=8square:8Square@

Save and exit.

Step 6 - Start the Percona XtraDB Cluster Server

We've configured Percona XtraDB cluster on all servers, and now we can start the server. On Percona1 server, bootstrapping or getting the cluster up and running:

```
systemctl start mysql@bootstrap
```

Next, start Percona/MySQL server on percona2 and percona3 server with the command below:

```
systemctl start mysql
```

make sure there are no error messages. If there is an error after typing to start MySQL, check the log file `/var/log/messages`.

Step 7 - Testing

We must log in to the percona/mysql shell on each server for testing the Percona XtraDB cluster. Login to the Percona/MySQL shell on all servers/nodes:

```
mysql -u root -p  
TYPE YOUR PASSWORD
```

Testing the High Availability with mysql command:

```
SHOW STATUS LIKE 'wsrep_local_state_comment';
```

Try that command on every node, if you see **"Synced"** as the result, then that node is ready to handle traffic.

Testing available nodes of the cluster:

```
show global status like 'wsrep_cluster_size';
```

You will get the current number of nodes in the Percona cluster.

```
mysql> show status like 'wsrep_local_state_comment';
+-----+
| Variable_name      | Value |
+-----+
| wsrep_local_state_comment | Synced |
+-----+
1 row in set (0.00 sec)

mysql> show global status like 'wsrep_cluster_size';
+-----+
| Variable_name      | Value |
+-----+
| wsrep_cluster_size | 3      |
+-----+
1 row in set (0.00 sec)
```

For the full results, you can use the command below:

```
show global status like 'wsrep%';
```