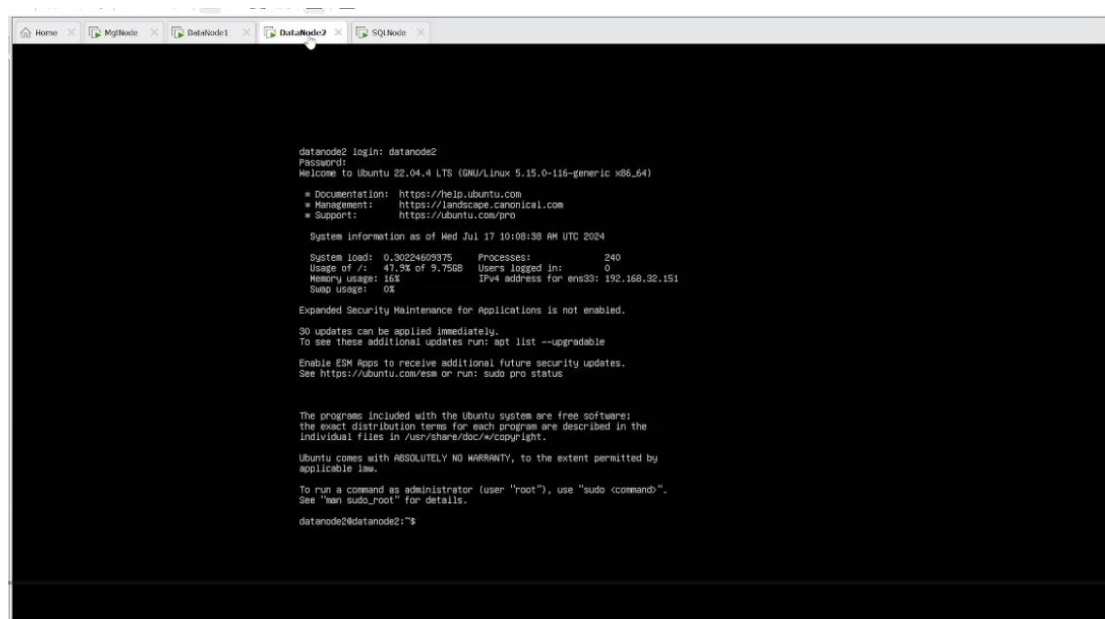


## MySQL NDB Cluster Configuration in Ubuntu 22.04 LTS—Step by Step process

### Prerequisites :

Server Usage Type	Public/Internet IP address	Node ID
Management node	192.168.32.149	1
Data node 1	192.168.32.150	2
Data node 2	192.168.32.151	3
SQL node	192.168.32.152	4

Configuring 4 nodes :



```
datanode2 login: datanode2
Password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 5.15.0-116-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

System information as of Wed Jul 17 10:08:38 AM UTC 2024
System load:  0.30224609375   Processes:            240
Usage of /:   47.9% of 9.75GB   Users logged in:      0
Memory usage: 15%            IPv4 address for ens3: 192.168.32.151
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.
30 updates can be applied immediately.
To see these additional updates run: apt list --upgradable
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

datanode2@datanode2:~$
```

### Step 1: Defining MySQL NDB Cluster Management Node (ndb\_mgmd)

First, we will set up and configure the MySQL NDB Cluster management node. This daemon will be responsible for reading the cluster configuration file and distributing the information to all processes (nodes participating in the clustered network)..

The `ndb_mgmd` is also responsible for maintaining a log file that details the activities of all nodes in the system.

So, connect to the management node using an SSH client.

### Download NDB Management Server

Then, we need grab the link of the latest NDB Management Server for our Ubuntu 22.04 server from [MySQL Cluster download page](http://dev.mysql.com/downloads/cluster/). (<http://dev.mysql.com/downloads/cluster/>)

Once you enter the [download link](#) on a web browser, scroll down to Generally Available (GA) Releases. Then, select **Ubuntu Linux** as the Operating System and **Ubuntu Linux 16.04(x86, 64 Bit)** version as shown below:

# MySQL NDB Cluster 9.0.0 Innovation

Select Version:

9.0.0 Innovation

Select Operating System:

Ubuntu Linux

Select OS Version:

Ubuntu Linux 22.04 (x86, 64-bit)

## DEB Package, NDB Management Server

9.0.0

2.2M

Download

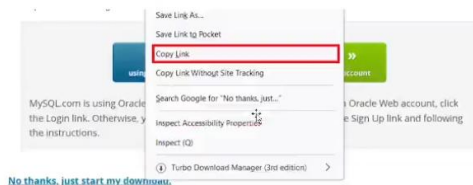
(mysql-cluster-community-management-server\_9.0.0-1ubuntu22.04\_amd64.deb)

MD5: 4dfe2c7bda2c0b8817e3956a16e13750

| Signature

On the next screen, navigate to the bottom and right-click **No thanks, just start my download** to copy the link address e.g. [https://dev.mysql.com/get/Downloads/MySQL-Cluster-9.0/mysql-cluster-community-management-server\\_9.0.0-1ubuntu22.04\\_amd64.deb](https://dev.mysql.com/get/Downloads/MySQL-Cluster-9.0/mysql-cluster-community-management-server_9.0.0-1ubuntu22.04_amd64.deb)

Refer to the image below:



Next, ensure you are connected on the terminal window of the management server. Then cd to the **tmp** directory:

```
$ cd /tmp
```

Download the MySQL cluster management server DEB file using **wget** command and the link you have copied above:

```
$ wget https://dev.mysql.com/get/Downloads/MySQL-Cluster-9.0/mysql-cluster-community-management-server_9.0.0-1ubuntu22.04_amd64.deb
```

## Installing the NDB Management Server

You can now use the **Debian** based **dpkg** package manager to install the **ndb\_mgmd** as shown below:

```
$ sudo dpkg -i mysql-cluster-community-management-server_9.0.0-1ubuntu22.04_amd64.deb
```

## Configuring the MySQL NDB Management Node

We will first create a directory for log files **/var/lib/mysql-cluster/**:

```
$ sudo mkdir /var/lib/mysql-cluster/
```

Then, we will create and open a config.ini file in the same directory. This is the global configuration file read by the management server, which in turn redistributes the information to other nodes in the cluster.

The configuration file lists all the details of the management nodes, data nodes and [SQL\(API\)](#) nodes. So, using a [nano](#) editor, open the file:

```
$ sudo nano /var/lib/mysql-cluster/config.ini
```

Then, paste the below minimum configuration information and remember to change the IP addresses to match your **IP addresses** that you wish to include in the database cluster.

In below we have change ip according to server

```
GNU nano 6.2 /var/lib/mysql-cluster/config.ini *
# Default settings
[ndbd default]
# Options affecting ndbd processes on all data nodes:
NoOfReplicas=2 # Number of replicas

# Management node
[ndb_mgmd]
# Management process options:
hostname=192.168.32.149 # Hostname of the manager
datadir=/var/lib/mysql-cluster # Directory for the log files

# 1st data node
[ndbd]
hostname=192.168.32.150 # Hostname/IP of the first data node
NodeId=2 # Node ID for this data node
datadir=/usr/local/mysql/data # Remote directory for the data files

# 2nd data node
[ndbd]
hostname=192.168.32.151 # Hostname/IP of the second data node
NodeId=3 # Node ID for this data node
datadir=/usr/local/mysql/data # Remote directory for the data files

# SQL node
[mysqld]
hostname=192.168.32.152 # In our case the MySQL server/client is different server cluster manager
```

Press **CTRL +X**, **Y** and **Enter** to save the file.

In the above file, the [NoOfReplicas=2](#) is a default parameter that can only be defined in the [\[ndbd default\]](#) section. The parameter defines the number of replicas each table will have on the cluster. The default, maximum and recommended value is 2 a value greater than this won't work in a production server.

This parameter is also used to specify the size of node groups in the network. So, if you have 6 data nodes and the [NoOfReplicas](#) is set to 2, then the first group is going to be formed by server 1,2 and 3 while the second one will come from server 4, 5 and 6.

In our case, we are using only two data servers and since we are setting the [NoOfReplicas](#) to 2. Each data node(server), will be taken as a group.

To avoid a single point of failure, nodes in the same group should not be setup in the same server because a hardware problem may cause the entire cluster to stop working.

On the configuration file, a management node is defined under the [\[ndb\\_mgmd\]](#) section while data nodes are listed under [\[ndbd\]](#). The API/SQL or MySQL server nodes are defined under the [\[mysqld\]](#) section.

Nodes participating in the cluster are assigned a unique id with the `NodeId` parameter. The data nodes must define a data directory using the `datadir` parameter. The `datadir` on the management node points to the directory for the log files.

Once we have created the master MySQL cluster configuration file, we will start the **ndb\_mgmd** Management daemon.

The configuration file we created must be specified when the `ndb_mgmd` is started for the first time using the `-f` option as shown below:

```
$ sudo ndb_mgmd -f /var/lib/mysql-cluster/config.ini
```

You should get the below output:

```
MySQL Cluster Management Server mysql-9.0.0 ndb-9.0.0
2024-07-17 04:06:35 [MgmtSrvr] INFO      -- The default config directory
'/usr/mysql-cluster' does not exist. Trying to create it...
2024-07-17 04:06:35 [MgmtSrvr] INFO      -- Successfully created config
directory
```

This shows that MySQL NDB Management server is setup and running on my VMWare Workstation.

In order for the process to start at boot, we need to run a few commands:

Before we create the service, we need to kill the running server:

```
$ sudo pkill -f ndb_mgmd
```

Then, we need to create and open a `systemd` unit file for the **ndb\_mgmd** service:

```
$ sudo nano /etc/systemd/system/ndb_mgmd.service
```

Then, paste the code below to instruct the `systemd` daemon how to start, stop and restart the service:

```
[Unit]
Description=MySQL NDB Cluster Management Server
After=network.target auditd.service

[Service]
Type=forking
ExecStart=/usr/sbin/ndb_mgmd -f /var/lib/mysql-cluster/config.ini
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

Press **CTRL +X**, **Y** and **Enter** to save the file.

```
GNU nano 6.2 /etc/systemd/system/ndb_mgmd.service *
[Unit]
Description=MySQL NDB Cluster Management Server
After=network.target auditd.service

[Service]
Type=forking
ExecStart=/usr/sbin/ndb_mgmd -f /var/lib/mysql-cluster/config.ini
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure

[Install]
WantedBy=multi-user.target
```



Reload `systemd` manager for the changes to take effect:

```
$ sudo systemctl daemon-reload
```

Next, enable the `ndb_mgmd` service:

```
$ sudo systemctl enable ndb mgmd
```

Then, start the service:

```
$ sudo systemctl start ndb mgmd
```

You can check the status of the `ndb_mgmd` process by running the command below:

```
$ sudo systemctl status ndb mgmd
```

The final step for setting up the Cluster Manager is to allow incoming connections from other MySQL Cluster nodes on our private network.

If you did not configure the `ufw` firewall when setting up this into server, you can skip ahead to the next section.

We'll add rules to allow local incoming connections from both data nodes:

```
$ sudo ufw allow from 192.168.32.150
$ sudo ufw allow from 192.168.32.151
```

After entering these commands, you should see the following output:

```
Output
Rule added
```

The Cluster Manager should now be up and running, and able to communicate with other Cluster nodes over the private network.

## Step 2: Configuring MySQL Cluster on Data Nodes

With the management node configured, we can now go ahead and configure our 2 data nodes. Please note, you must follow and repeat the below procedure for every data node on the cluster for replication to work.

The data node is responsible for storing MySQL cluster data and thus, they offer redundancy and high availability, 2 or more servers must be used.

Remember, your data nodes must be homogeneous in nature. This means you should setup the data node ECS instances with the same VCPU's, RAM, disk space and bandwidth.

A RAM of at least 1GB is recommended for the data nodes because the cluster engine uses a lot of memory.

So, let start by installing and configuring the first data node. In our case, we will SSH to the server with the IP address 198.18.0.2.

Then, cd to the `tmp` directory:

```
$ cd /tmp
```

Just like we have done for the management node, we are going to pull the latest NDB cluster data node deb package from [MySQL download page](#). Select **Ubuntu Linux** and **Ubuntu Linux (22.04(x86, 64-bit))** as the version.

Connect datanode1 and datanode2

General Availability (GA) Releases Archives

### MySQL NDB Cluster 9.0.0 Innovation

Select Version:  
9.0.0 Innovation

Select Operating System:  
Ubuntu Linux

Select OS Version:  
Ubuntu Linux 22.04 (x86, 64-bit)

Then, scroll down on the list and select **DEB Package, NDB Data Node Binaries**. Then, click **Download** as shown below:

Download the package

<b>DEB Package, NDB Data Node Binaries</b>	9.0.0	6.9M	<a href="#">Download</a>
(mysql-cluster-community-data-node_9.0.0-1ubuntu22.04_amd64.deb)		MD5: 3ec0e19423747e4944f7c49c688287c8   <a href="#">Signature</a>	

[No thanks, just start my download.](#)

ORACLE © 2022

Privacy / Do Not Sell My Info | Terms of Service | Privacy Policy | Cookie Preferences

- Open link in new tab
- Open link in new window
- Open link in incognito window
- Save link as...
- Copy link address
- Inspect

Paste the link and download using **wget**

```
root@datanode1:/home/datanode1# cd /tmp
root@datanode1:/tmp# wget https://dev.mysql.com/get/Downloads/MySQL-Cluster-9.0/mysql-cluster-community-data-node_9.0.0-1ubuntu22.04_amd64.deb
root@datanode2:~$ sudo su
[sudo] password for datanode2:
root@datanode2:/home/datanode2# cd /tmp
root@datanode2:/tmp#
```

After download and install update using **apt update** on both **datanode1** and **datanode2**

Then, download the deb file from the link you have copied using the **wget** command:

```
$ wget https://dev.mysql.com/get/Downloads/MySQL-Cluster-9.0/mysql-cluster-community-data-node_9.0.0-1ubuntu22.04_amd64.deb
```

The MySQL cluster community data node has known dependency requirements, so we are going to install **libclass-methodmaker-perl** package before we install the daemon.

I

So first, update the package list information index:

```
$ sudo apt-get update
```

Before we install the data node binary, we need to install a dependency, **libclass-methodmaker-perl**:

```
$ sudo apt-get install libclass-methodmaker-perl
```

Now next step to install the Mysql cluster on both **datanode1** and **datanode2** :

We can now go ahead and install MySQL cluster data node:

```
$ sudo dpkg -i mysql-cluster-community-data-node_9.0.0-1ubuntu22.04_amd64.deb
```

Next, we need to instruct the data node to connect to the management cluster to retrieve the configuration file that we created. When there are changes in the data node, the events will be transferred to the management node to be written to the cluster log file.

So, open the `/etc/my.cnf` file to enter the MySQL cluster management node information. That is the IP address where the cluster management node resides, in our case **192.168.32.149**

```
$ sudo nano /etc/my.cnf
```

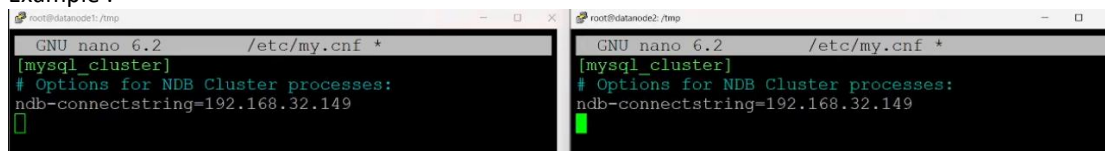
Paste the content below on the file and remember to replace **192.168.32.149** with the private IP address of your MySQL cluster management node:

```
[mysql_cluster]
# Options for NDB Cluster processes:
ndb-connectstring=192.168.32.149
```

Save the file by pressing **CTRL + X, Y** and **Enter**.

Next, create the data directory `/usr/local/mysql/data` because this is what we specified in the management node configuration file:

Example :



Next, create the data directory `/usr/local/mysql/data` because this is what we specified in the management node configuration file:

```
$ sudo mkdir -p /usr/local/mysql/data
```

Now we can start the data node using the following command:

```
$ sudo ndbd
```

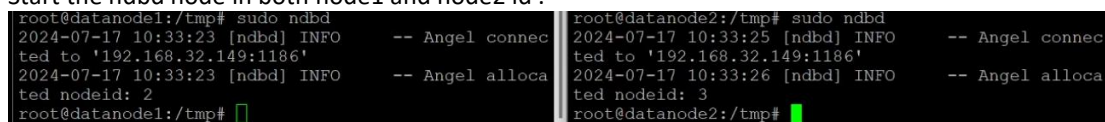
You should get the below output:

```
2023-07-22 17:59:22 [ndbd] INFO      -- Angel connected to
'192.168.32.149:1186'
2023-07-22 17:59:22 [ndbd] INFO      -- Angel allocated nodeid: 2
```

The NDB data node daemon has been successfully installed and is now running on your server.

We also need to allow incoming connections from other MySQL Cluster nodes over the private network.

Start the ndbd node in both node1 and node2 id .



If you did not configure the `ufw` firewall when setting up this servers, you can skip ahead to setting up the `systemd` service for `ndbd`.

We'll add rules to allow incoming connections from the Cluster Manager and other data nodes:

```
sudo ufw allow from 192.168.32.150
sudo ufw allow from 192.168.32.151
```

After entering these commands, you should see the following output:

```
Output
Rule added
```

Your MySQL data node Droplet can now communicate with both the Cluster Manager and other data node over the private network.

Finally, we'd also like the data node daemon to start up automatically when the server boots. We'll follow the same procedure used for the Cluster Manager, and create a `systemd` service.

To enable the `ndbd` service to start at boot, we will add the process in the `systemd` daemon. First, let's kill the process that we initialized:

```
$ sudo pkill -f ndbd
```

Then, open the `/etc/systemd/system/ndbd.service` file to instruct `systemd` on how to start, stop and restart the `ndbd` service:

```
$ sudo nano /etc/systemd/system/ndbd.service
```

```
root@datanode1:/tmp# sudo pkill -f ndbd
```

```
ted nodeid: 3
```

```
root@datanode2:/tmp# sudo pkill -f ndbd
```

Now next open in both node file `sudo nano /etc/systemd/system/ndbd.service`

```
root@datanode2: /tmp
/etc/systemd/system/ndbd.service *
[Unit]
Description=MySQL NDB Data Node Daemon
After=network.target auditd.service

[Service]
Type=forking
ExecStart=/usr/sbin/ndbd
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

Run below on both node

Press **CTRL +X**, **Y** and **Enter** to save the file.

Restart the `systemd` process for the changes to take effect:

```
$ sudo systemctl daemon-reload
```

Then, enable the `ndbd` process using the `systemctl` command:

```
$ sudo systemctl enable ndbd
```

Then, start the `ndbd` process:

```
$ sudo systemctl start ndbd
```

You can check whether the `ndbd` process is running by typing the command below:

```
$ sudo systemctl status ndbd
```



You should see the following output:  
Output

```
● ndbd.service - MySQL NDB Data Node Daemon
   Loaded: loaded (/etc/systemd/system/ndbd.service; enabled; vendor preset:
   enabled)
   Active: active (running) since Thu 2024-07-06 10:55:49 UTC; 8s ago
     Process: 11972 ExecStart=/usr/sbin/ndbd (code=exited, status=0/SUCCESS)
    Main PID: 11984 (ndbd)
      Tasks: 46 (limit: 4915)
```

```
CGroup: /system.slice/ndbd.service
└─11984 /usr/sbin/ndbd
└─11987 /usr/sbin/ndbd
```

### Step 3: Configuring SQL Cluster Node

In this step, we will install a custom MySQL community server that is bundled with the NDB storage engine. This is the SQL node and will reside in our 4th server

The SQL node will be used specifically to access the clustered data through the NDBCLUSTER storage engine. It is simply a mysqld process that works as the API(Application Programming Interface) node to manipulate data on the cluster.

Your web application or website should connect to this SQL node. So, if you are designing a software or a website, use the public IP address of this node as your host.

To setup the SQL node, we will SSH to the server with the public IP address **192.168.32.152**. The MySQL cluster server applications require some dependencies, so will first update the package list index and install them.

```
$ sudo apt-get update
```

Then install libaio1 and libmecab2 packages:

```
$ sudo apt install libaio1 libmecab2
```

Press Y and hit enter when prompted to confirm the installation

Then , cd to the tmp directory on the server .

```
$ cd /tmp
```

Download the Mysql Cluster DEB bundle link from Mysql cluster download page select ubuntu linux as the operating system as the version .

<b>DEB Bundle</b>	9.0.0	454.9M	<a href="#">Download</a>
(mysql-cluster_9.0.0-1ubuntu22.04_amd64.deb-bundle.tar)	MD5: bfe02de0eca0db65f7101439f90c20ba   <a href="#">Signature</a>		

Select appropriate bundle and click on download and right click and copy the url link and now follow the below steps .

```
$ wget https://dev.mysql.com/get/Downloads/MySQL-Cluster-9.0/mysql-cluster_9.0.0-1ubuntu22.04_amd64.deb-bundle.tar
```

The above command will download a tar archive file e.g. **mysql-cluster\_9.0.0-1ubuntu22.04\_amd64.deb-bundle.tar**.

The zipped file contains several deb packages that we require for the installation. First create a working installation directory e.g. **install**:

```
$ sudo mkdir install
```

Then, unzip the **mysql-cluster\_9.0.0-1ubuntu22.04\_amd64.deb-bundle.tar** deb files to the installation directory:

```
$ sudo tar -xvf mysql-cluster_9.0.0-1ubuntu22.04_amd64.deb-bundle.tar -C install/
```

Once the deb files are extracted and copied in the **installation** directory, cd to the directory.

```
$ cd install
```

Then, run the commands below one by one to install all required MySQL Cluster packages. Enter a strong password for the root user of the database server when prompted to do so:

```
$ sudo dpkg -i mysql-common_9.0.0-1ubuntu22.04_amd64.deb
$ sudo dpkg -i mysql-cluster-community-client-plugins_9.0.0-1ubuntu22.04_amd64.deb
$ sudo dpkg -i mysql-cluster-community-client-core_9.0.0-1ubuntu22.04_amd64.deb
$ sudo dpkg -i mysql-cluster-community-client_9.0.0-1ubuntu22.04_amd64.deb
$ sudo dpkg -i mysql-client_9.0.0-1ubuntu22.04_amd64.deb
$ sudo dpkg -i mysql-cluster-community-server-core_9.0.0-1ubuntu22.04_amd64.deb
$ sudo dpkg -i mysql-cluster-community-server_9.0.0-1ubuntu22.04_amd64.deb
```

Next, we need to configure MySQL server to connect to the cluster. We can do this by editing the `/etc/mysql/my.cnf` file:

```
$ sudo nano /etc/mysql/my.cnf
```

Enter the values below and replace the IP address `192.168.32.149` with the private IP address of your Cluster Management Server :

```
[mysqld]
# Options for mysqld process:
ndbcluster                                # run NDB storage engine

[mysql_cluster]
# Options for NDB Cluster processes:
ndb-connectstring=192.168.32.149         # location of management server
```

Add in `sqlnode` my.cnf ip of ndb server

```
!includedir /etc/mysql/conf.d/
!includedir /etc/mysql/mysql.conf.d/
[mysqld]
# Options for mysqld process:
ndbcluster                                # run NDB storage engine

[mysql_cluster]
# Options for NDB Cluster processes:
ndb-connectstring=192.168.32.149         # location of management server
```

The `ndbcluster` directive enables the `ndbcluster` storage engine because it is not enabled by default to save resources. Then, the `ndb-connectstring` points to the private IP address of the `mysql` management server.

Restart MySQL server for the changes to take effect:

```
$ sudo systemctl restart mysql
```

## Step 4: Verifying MySQL Cluster Installation

The following MySQL cluster nodes should now be up and running.

- MySQL cluster Management node
- MySQL cluster data node 1

- MySQL cluster data node 2
- SQL node

To test if the MySQL cluster is working, enter the command below on the **SQL node**:

```
$ mysql -u root -p
```

Enter your MySQL cluster password that you created when prompted and hit **Enter**. You should see the output below:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 11  
Server version: 9.0.0-cluster MySQL Cluster Community Server - GPL
```

Create a sample database:

```
mysql > create database test database;
```

Now create table in this database and insert some records .

Once inside the mysql client run the following command

```
Mysql> SHOW ENGINE NDB STATUS \G;
```

And connect ndbd and perform activity.