
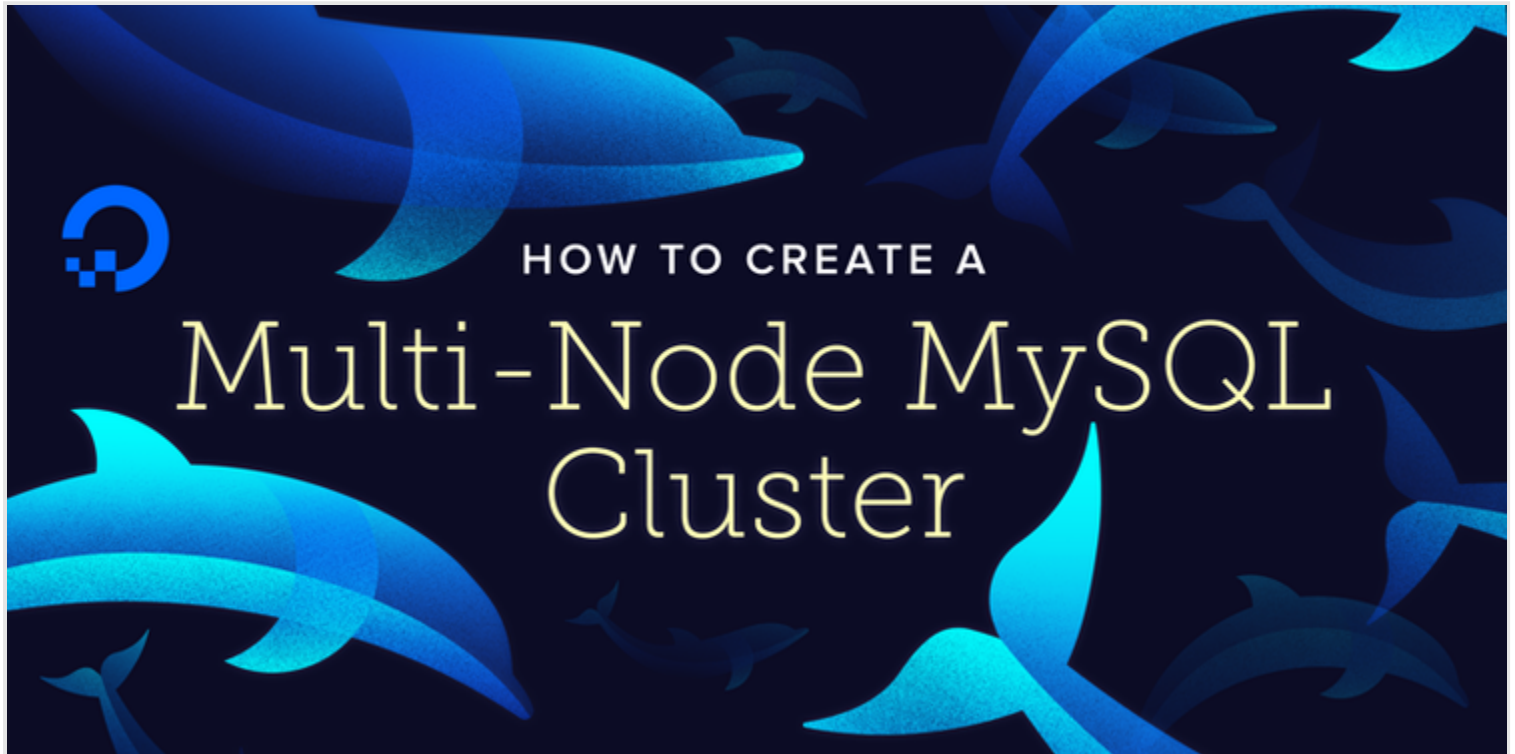


 Subscribe

 Share

 Contents ▾


How To Create a Multi-Node MySQL Cluster on Ubuntu 18.04


7

Posted July 26, 2018

 20.9k

MYSQL

DATABASES

CLUSTERING

UBUNTU

UBUNTU 18.04

By: Anatoliy Dimitrov By: Hanif Jetha

 Not using **Ubuntu 18.04**? Choose a different version:

Ubuntu 16.04



Automated: Docker

[request](#)

Automated: Bash

[request](#)

Introduction

The MySQL Cluster distributed database provides high availability and throughput for your MySQL database management system. A MySQL Cluster consists of one or more management nodes (`ndb_mgmd`) that store the cluster's configuration and control the data nodes (`ndbd`), where cluster data is stored. After communicating with the management node, clients (MySQL clients, servers, or native APIs) connect directly to these data nodes.

With MySQL Cluster there is typically no replication of data, but instead data node synchronization. For this purpose a special data engine must be used — NDBCluster (NDB). It's helpful to think of the cluster as a single logical MySQL environment with redundant components. Thus, a MySQL Cluster can participate in replication with other MySQL Clusters.

MySQL Cluster works best in a shared-nothing environment. Ideally, no two components should share the same hardware. For simplicity and demonstration purposes, we'll limit ourselves to using only three servers. We will set up two servers as data nodes which sync data between themselves. The third server will be used for the Cluster Manager and also for the MySQL server/client. If you spin up additional servers, you can add more data nodes to the cluster, decouple the cluster manager from the MySQL server/client, and configure more servers as Cluster Managers and MySQL servers/clients.

Prerequisites

To complete this tutorial, you will need a total of three servers: two servers for the redundant MySQL data nodes (`ndbd`), and one server for the Cluster Manager (`ndb_mgmd`) and MySQL server/client (`mysqld` and `mysql`).

In the **same DigitalOcean data center**, create the following Droplets with **private networking enabled**:

- Three Ubuntu 18.04 Droplets with private networking enabled
- A non-root user with sudo privileges configured for each Droplet. You can learn how to do this in Initial Server Setup with Ubuntu 18.04.

Be sure to note down the **private** IP addresses of your three Droplets. In this tutorial our cluster nodes have the following private IP addresses:

- **198.51.100.0** will be the first MySQL Cluster data node
- **198.51.100.1** will be the second data node
- **198.51.100.2** will be the Cluster Manager & MySQL server node

Once you've spun up your Droplets, configured a non-root user, and noted down the IP addresses for the 3 nodes, you're ready to begin with this tutorial.

Step 1 — Installing and Configuring the Cluster Manager

We'll first begin by downloading and installing the MySQL Cluster Manager, `ndb_mgmd`.

To install the Cluster Manager, we first need to fetch the appropriate `.deb` installer file from the the official MySQL Cluster [download page](#).

From this page, under **Select Operating System**, choose **Ubuntu Linux**. Then, under **Select OS Version**, choose **Ubuntu Linux 18.04 (x86, 64-bit)**.

Scroll down until you see **DEB Package, NDB Management Server**, and click on the **Download** link for the one that does *not* contain `dbgsym` (unless you require debug symbols). You will be brought to a **Begin Your Download** page. Here, right click on **No thanks, just start my download.** and copy the link to the `.deb` file.

Now, log in to your Cluster Manager Droplet (in this tutorial, `198.51.100.2`), and download this `.deb` file:

```
$ cd ~
$ wget https://dev.mysql.com/get/Downloads/MySQL-Cluster-7.6/mysql-cluster-community-management-server_7.6.6-1ubuntu18.04_amd64.deb
```

Install `ndb_mgmd` using `dpkg`:

```
$ sudo dpkg -i mysql-cluster-community-management-server_7.6.6-1ubuntu18.04_amd64.deb
```

We now need to configure `ndb_mgmd` before first running it; proper configuration will ensure correct synchronization and load distribution among the data nodes.

The Cluster Manager should be the first component launched in any MySQL cluster. It requires a configuration file, passed in as an argument to its executable. We'll create and use the following configuration file: `/var/lib/mysql-cluster/config.ini`.

On the Cluster Manager Droplet, create the `/var/lib/mysql-cluster` directory where this file will reside:

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

Then create and edit the configuration file using your preferred text editor:

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

Paste the following text into your editor:

```
/var/lib/mysql-cluster/config.ini
```

```
[ndbd default]
# Options affecting ndbd processes on all data nodes:
NoOfReplicas=2 # Number of replicas
```

```
[ndb_mgmd]
# Management process options:
```

```
hostname=198.51.100.2 # Hostname of the manager
datadir=/var/lib/mysql-cluster # Directory for the log files

[ndbd]
hostname=198.51.100.0 # 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

[ndbd]
hostname=198.51.100.1 # 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

[mysqld]
# SQL node options:
hostname=198.51.100.2 # In our case the MySQL server/client is on the same Droplet as the cluster manager
```

After pasting in this text, being sure to replace the `hostname` values above with the correct IP addresses of the Droplets you've configured. Setting this `hostname` parameter is an important security measure that prevents other servers from connecting to the Cluster Manager.

Save the file and close your text editor.

This is a pared-down, minimal configuration file for a MySQL Cluster. You should customize the parameters in this file depending on your production needs. For a sample, fully configured `ndb_mgmd` configuration file, consult the [MySQL Cluster documentation](#).

In the above file you can add additional components like data nodes (`ndbd`) or MySQL server nodes (`mysqld`) by appending instances to the appropriate section.

We can now start the manager by executing the `ndb_mgmd` binary and specifying its config file using the `-f` flag:

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

You should see the following output:

Output

```
MySQL Cluster Management Server mysql-5.7.22 ndb-7.6.6
2018-07-25 21:48:39 [MgmtSrvr] INFO      -- The default config directory '/usr/mysql-cluster' does not exist
2018-07-25 21:48:39 [MgmtSrvr] INFO      -- Successfully created config directory
```

This indicates that the MySQL Cluster Management server has successfully been installed and is now running on your Droplet.

Ideally, we'd like to start the Cluster Management server automatically on boot. To do this, we're going to create and enable a systemd service.

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

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

Now, open and edit the following systemd Unit file using your favorite editor:

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

Paste in the following code:

```
/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
```

Here, we've added a minimal set of options instructing systemd on how to start, stop and restart the `ndb_mgmd` process. To learn more about the options used in this unit configuration, consult the [systemd manual](#).

Save and close the file.

Now, reload systemd's manager configuration using `daemon-reload`:

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

We'll enable the service we just created so that the MySQL Cluster Manager starts on reboot:

```
$ sudo systemctl enable ndb_mgmd
```

Finally, we'll start the service:

```
$ sudo systemctl start ndb_mgmd
```

You can verify that the NDB Cluster Management service is running:

```
$ sudo systemctl status ndb_mgmd
```

You should see the following output:

```
● ndb_mgmd.service - MySQL NDB Cluster Management Server
   Loaded: loaded (/etc/systemd/system/ndb_mgmd.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2018-07-26 21:23:37 UTC; 3s ago
     Process: 11184 ExecStart=/usr/sbin/ndb_mgmd -f /var/lib/mysql-cluster/config.ini (code=exited, status=0/SUCCESS)
    Main PID: 11193 (ndb_mgmd)
      Tasks: 11 (limit: 4915)
     CGroup: /system.slice/ndb_mgmd.service
            └─11193 /usr/sbin/ndb_mgmd -f /var/lib/mysql-cluster/config.ini
```

Which indicates that the `ndb_mgmd` MySQL Cluster Management server is now running as a systemd service.

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 Droplet, 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 198.51.100.0
$ sudo ufw allow from 198.51.100.1
```

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 — Installing and Configuring the Data Nodes

Note: All the commands in this section should be executed on both data nodes.

In this step, we'll install the `ndbd` MySQL Cluster data node daemon, and configure the nodes so they can communicate with the Cluster Manager.

To install the data node binaries we first need to fetch the appropriate `.deb` installer file from the official [MySQL download page](#).

From this page, under **Select Operating System**, choose **Ubuntu Linux**. Then, under **Select OS Version**, choose **Ubuntu Linux 18.04 (x86, 64-bit)**.

Scroll down until you see **DEB Package, NDB Data Node Binaries**, and click on the **Download** link for the one that does **not** contain `dbgsym` (unless you require debug symbols). You will be brought to a **Begin Your Download** page. Here, right click on **No thanks, just start my download.** and copy the link to the `.deb` file.

Now, log in to your first data node Droplet (in this tutorial, `198.51.100.0`), and download this `.deb` file:

```
$ cd ~
$ wget https://dev.mysql.com/get/Downloads/MySQL-Cluster-7.6/mysql-cluster-community-data-node_7.6.6
```

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

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

We can now install the data node binary using `dpkg`:

```
$ sudo dpkg -i mysql-cluster-community-data-node_7.6.6-1ubuntu18.04_amd64.deb
```

The data nodes pull their configuration from MySQL's standard location, `/etc/my.cnf`. Create this file using your favorite text editor and begin editing it:

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

Add the following configuration parameter to the file:

```
/etc/my.cnf

[mysql_cluster]
# Options for NDB Cluster processes:
ndb-connectstring=198.51.100.2 # location of cluster manager
```

Specifying the location of the Cluster Manager node is the only configuration needed for `ndbd` to start. The rest of the configuration will be pulled from the manager directly.

Save and exit the file.

In our example, the data node will find out that its data directory is `/usr/local/mysql/data`, per the manager's configuration. Before starting the daemon, we'll create this directory on the node:

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

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

```
$ sudo ndbd
```

You should see the following output:

Output

```
2018-07-18 19:48:21 [ndbd] INFO      -- Angel connected to '198.51.100.2:1186'
2018-07-18 19:48:21 [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.

If you did not configure the `ufw` firewall when setting up this Droplet, 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 198.51.100.0
$ sudo ufw allow from 198.51.100.2
```

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.

Before we create the service, we'll kill the running `ndbd` process:


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

Now, open and edit the following systemd Unit file using your favorite editor:

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

Paste in the following code:

```
                                /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
```

Here, we've added a minimal set of options instructing systemd on how to start, stop and restart the `ndbd` process. To learn more about the options used in this unit configuration, consult the [systemd manual](#).

Save and close the file.

Now, reload systemd's manager configuration using `daemon-reload`:

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

We'll now enable the service we just created so that the data node daemon starts on reboot:

```
$ sudo systemctl enable ndbd
```

Finally, we'll start the service:

```
$ sudo systemctl start ndbd
```

You can verify that the NDB Cluster Management service is running:

```
$ 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 2018-07-26 20:56:29 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
```

Which indicates that the `ndbd` MySQL Cluster data node daemon is now running as a `systemd` service. Your data node should now be fully functional and able to connect to the MySQL Cluster Manager.

Once you've finished setting up the first data node, repeat the steps in this section on the other data node (`198.51.100.1` in this tutorial).

Step 3 — Configuring and Starting the MySQL Server and Client

A standard MySQL server, such as the one available in Ubuntu's APT repository, does not support the MySQL Cluster engine NDB. This means we need to install the custom SQL server packaged with the other MySQL Cluster software we've installed in this tutorial.

We'll once again grab the MySQL Cluster Server binary from the official MySQL Cluster [download page](#).

From this page, under **Select Operating System**, choose **Ubuntu Linux**. Then, under **Select OS Version**, choose **Ubuntu Linux 18.04 (x86, 64-bit)**.

Scroll down until you see **DEB Bundle**, and click on the **Download** link (it should be the first one in the list). You will be brought to a **Begin Your Download** page. Here, right click on **No thanks, just start my download**, and copy the link to the `.tar` archive.

Now, log in to the Cluster Manager Droplet (in this tutorial, `198.51.100.2`), and download this `.tar` archive (recall that we are installing MySQL Server on the same node as our Cluster Manager – in a production setting you should run these daemons on different nodes):

```
$ cd ~
$ wget https://dev.mysql.com/get/Downloads/MySQL-Cluster-7.6/mysql-cluster_7.6.6-1ubuntu18.04_amd64.
```

We'll now extract this archive into a directory called `install`. First, create the directory:

```
$ mkdir install
```

Now extract the archive into this directory:

```
$ tar -xvf mysql-cluster_7.6.6-1ubuntu18.04_amd64.deb-bundle.tar -C install/
```

Move into this directory, containing the extracted MySQL Cluster component binaries:

```
$ cd install
```

Before we install the MySQL server binary, we need to install a couple of dependencies:

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

Now, we need to install the MySQL Cluster dependencies, bundled in the `tar` archive we just extracted :

```
$ sudo dpkg -i mysql-common_7.6.6-1ubuntu18.04_amd64.deb
$ sudo dpkg -i mysql-cluster-community-client_7.6.6-1ubuntu18.04_amd64.deb
$ sudo dpkg -i mysql-client_7.6.6-1ubuntu18.04_amd64.deb
$ sudo dpkg -i mysql-cluster-community-server_7.6.6-1ubuntu18.04_amd64.deb
```

When installing `mysql-cluster-community-server` , a configuration prompt should appear, asking you to set a password for the **root** account of your MySQL database. Choose a strong, secure password, and hit **<Ok>**. Re-enter this **root** password when prompted, and hit **<Ok>** once again to complete installation.

We can now install the MySQL server binary using `dpkg` :

```
$ sudo dpkg -i mysql-server_7.6.6-1ubuntu18.04_amd64.deb
```

We now need to configure this MySQL server installation.

The configuration for MySQL Server is stored in the default `/etc/mysql/my.cnf` file.

Open this configuration file using your favorite editor:

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

You should see the following text:

```
/etc/mysql/my.cnf
```

```
# Copyright (c) 2015, 2016, Oracle and/or its affiliates. All rights reserved.
```

```

#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; version 2 of the License.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
#
# The MySQL Cluster Community Server configuration file.
#
# For explanations see
# http://dev.mysql.com/doc/mysql/en/server-system-variables.html
#
# * IMPORTANT: Additional settings that can override those from this file!
# The files must end with '.cnf', otherwise they'll be ignored.
#
!includedir /etc/mysql/conf.d/
!includedir /etc/mysql/mysql.conf.d/

```

Append the following configuration to it:

```

                                /etc/mysql/my.cnf

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

[mysql_cluster]
# Options for NDB Cluster processes:
ndb-connectstring=198.51.100.2 # location of management server

```

Save and exit the file.

Restart the MySQL server for these changes to take effect:

```
$ sudo systemctl restart mysql
```

MySQL by default should start automatically when your server reboots. If it doesn't, the following command should fix this:

```
$ sudo systemctl enablemysql
```

A SQL server should now be running on your Cluster Manager / MySQL Server Droplet.

In the next step, we'll run a few commands to verify that our MySQL Cluster installation is functioning as expected.

Step 4 — Verifying MySQL Cluster Installation

To verify your MySQL Cluster installation, log in to your Cluster Manager / SQL Server node.

We'll open the MySQL client from the command line and connect to the **root** account we just configured by entering the following command:

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

Enter your password when prompted, and hit **ENTER**.

You should see an output similar to:

Output

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.7.22-ndb-7.6.6 MySQL Cluster Community Server (GPL)

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

```
mysql>
```

Once inside the MySQL client, run the following command:

```
mysql> SHOW ENGINE NDB STATUS \G
```

You should now see information about the NDB cluster engine, beginning with connection parameters:

Output

```
***** 1. row *****
```

```
Type: ndbcluster
Name: connection
Status: cluster_node_id=4, connected_host=198.51.100.2, connected_port=1186, number_of_data_nodes=2,
. . .
```

This indicates that you've successfully connected to your MySQL Cluster.

Notice here the number of `ready_data_nodes` : 2. This redundancy allows your MySQL cluster to continue operating even if one of the data nodes fails. It also means that your SQL queries will be load balanced across the two data nodes.

You can try shutting down one of the data nodes to test cluster stability. The simplest test would be to restart the data node Droplet in order to fully test the recovery process. You should see the value of `number_of_ready_data_nodes` change to 1 and back up to 2 again as the node reboots and reconnects to the Cluster Manager.

To exit the MySQL prompt, simply type `quit` or press `CTRL-D`.

This is the first test that indicates that the MySQL cluster, server, and client are working. We'll now go through an additional test to confirm that the cluster is functioning properly.

Open the Cluster management console, `ndb_mgm` using the command:

```
$ ndb_mgm
```

You should see the following output:

Output

```
-- NDB Cluster -- Management Client --
ndb_mgm>
```

Once inside the console enter the command `SHOW` and hit `ENTER`:

```
ndb_mgm> SHOW
```

You should see the following output:

Output

```
Connected to Management Server at: 198.51.100.2:1186
Cluster Configuration
-----
[ndbd(NDB)] 2 node(s)
id=2      @198.51.100.0 (mysql-5.7.22 ndb-7.6.6, Nodegroup: 0, *)
```

```
id=3      @198.51.100.1 (mysql-5.7.22 ndb-7.6.6, Nodegroup: 0)
```

```
[ndb_mgmd(MGM)] 1 node(s)
```

```
id=1      @198.51.100.2 (mysql-5.7.22 ndb-7.6.6)
```

```
[mysqld(API)] 1 node(s)
```

```
id=4      @198.51.100.2 (mysql-5.7.22 ndb-7.6.6)
```

The above shows that there are two data nodes connected with node-id s 2 and 3. There is also one management node with node-id 1 and one MySQL server with node-id 4. You can display more information about each id by typing its number with the command `STATUS` as follows:

```
ndb_mgm> 2 STATUS
```

The above command shows you the status, MySQL version, and NDB version of node 2:

Output

```
Node 2: started (mysql-5.7.22 ndb-7.6.6)
```

To exit the management console type `quit` , and then hit `ENTER` .

The management console is very powerful and gives you many other options for administering the cluster and its data, including creating an online backup. For more information consult the [official MySQL documentation](#).

At this point, you've fully tested your MySQL Cluster installation. The concluding step of this guide shows you how to create and insert test data into this MySQL Cluster.

Step 5 — Inserting Data into MySQL Cluster

To demonstrate the cluster's functionality, let's create a new table using the NDB engine and insert some sample data into it. Note that in order to use cluster functionality, the engine must be specified explicitly as **NDB**. If you use InnoDB (default) or any other engine, you will not make use of the cluster.

First, let's create a database called `clustertest` with the command:

```
mysql> CREATE DATABASE clustertest;
```

Next, switch to the new database:

```
mysql> USE clustertest;
```

Now, create a simple table called `test_table` like this:

```
mysql> CREATE TABLE test_table (name VARCHAR(20), value VARCHAR(20)) ENGINE=ndbcluster;
```

We have explicitly specified the engine `ndbcluster` in order to make use of the cluster.

Now, we can start inserting data using this SQL query:

```
mysql> INSERT INTO test_table (name,value) VALUES('some_name','some_value');
```

To verify that the data has been inserted, run the following select query:

```
mysql> SELECT * FROM test_table;
```

When you insert data into and select data from an `ndbcluster` table, the cluster load balances queries between all the available data nodes. This improves the stability and performance of your MySQL database installation.

You can also set the default storage engine to `ndbcluster` in the `my.cnf` file that we edited previously. If you do this, you won't need to specify the `ENGINE` option when creating tables. To learn more, consult the [MySQL Reference Manual](#).

Conclusion

In this tutorial, we've demonstrated how to set up and configure a MySQL Cluster on Ubuntu 18.04 servers. It's important to note that this is a minimal, pared-down architecture used to demonstrate the installation procedure, and there are many advanced options and features worth learning about before deploying MySQL Cluster in production (for example, performing backups). To learn more, consult the official [MySQL Cluster documentation](#).

By: Anatoliy Dimitrov By: Hanif Jetha

♡ Upvote (7)

📄 Subscribe

🔗 Share



We just made it easier for you to deploy faster.

[TRY FREE](#)

Related Tutorials

[How To Ensure Code Quality with SonarQube on Ubuntu 18.04](#)

[How To Sync and Share Your Files with Seafile on Ubuntu 18.04](#)

[How To Troubleshoot Issues in MySQL](#)



[How To Set Up a Remote Database to Optimize Site Performance with MySQL on Ubuntu 18.04](#)

[How To Set Up Laravel, Nginx, and MySQL with Docker Compose](#)

4 Comments

Leave a comment...

[Log In to Comment](#)

-  [matriawan](#) *September 10, 2018*
-  just fyi, in step 3 about "\$mysql-server7.6.6-1ubuntu18.04amd64.deb" shoud be "\$sudo dpkg -i mysql-server7.6.6-1ubuntu18.04amd64.deb"

^ [hjet](#) MOD December 19, 2018

o Great catch!

Thank you.

^ [supportis](#) September 16, 2018

o Hello,

I have followed every thing and finally got this installed but this is fine when we have 2 cluster node's and 1 sql node . For a production set up we need to deploy via loadbalancer. Can you please help us with steps on how to install 2 Sql nodes points to same 2 cluster node . So in other words

There are 2 mysql node and 2 cluster node's . In case of 1 mysql node fails load balancer switches to other mysql node.

Your help on this is highly appreciated as for production use we cannot rely on single instance of sql .

^ [ran541959](#) October 26, 2018

o root@mySQLCluster-Manager:~# sudo dpkg -i mysql-cluster-community-management-server7.6.8-1ubuntu18.04amd64.deb

Selecting previously unselected package mysql-cluster-community-management-server.

(Reading database ... 58794 files and directories currently installed.)

Preparing to unpack mysql-cluster-community-management-server7.6.8-1ubuntu18.04amd64.deb ...

Unpacking mysql-cluster-community-management-server (7.6.8-1ubuntu18.04) ...

dpkg: dependency problems prevent configuration of mysql-cluster-community-management-server:
mysql-cluster-community-management-server depends on libtinfo5 (>= 6); however:

Package libtinfo5 is not installed.

dpkg: error processing package mysql-cluster-community-management-server (--install):

dependency problems - leaving unconfigured

Processing triggers for man-db (2.8.4-2) ...

Errors were encountered while processing:

mysql-cluster-community-management-server



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.



Copyright © 2019 DigitalOcean™ Inc.

[Community](#) [Tutorials](#) [Questions](#) [Projects](#) [Tags](#) [Newsletter](#) [RSS](#) 

[Distros & One-Click Apps](#) [Terms, Privacy, & Copyright](#) [Security](#) [Report a Bug](#) [Write for DOnations](#) [Shop](#)