Step to configure MYSQL Cluster

Link followed to configure the MySQL Cluster are as follows

<https://www.alibabacloud.com/blog/how-to-setup-mysql-ndb-cluster-on-ubuntu-16-04_594458>

Pre-requisites:

1.Ubuntu 16.04 OS for Manager Node with Private IP 192.168.1.1

2.Ubuntu 16.04 OS for Data Node-1 with Private IP 192.168.1.2

3.Ubuntu 16.04 OS for Data Node-2 with Private IP 192.168.1.3

4.Ubuntu 16.04 OS for SQL Node with Private IP 192.168.1.4

Step 1: Configure Management Node on AWS Cloud

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-7.6/mysql-cluster-community-management-server\_7.6.7-1ubuntu16.04\_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\_7.6.7-1ubuntu16.04\_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 on the same directory. This is the global configuration file read by the management server that in turn redistributes the information to other node 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 the **private IP addresses** identified by your Alibaba Cloud ECS instances that you wish to include in the database cluster.

# Default settings

[ndbd default]

NoOfReplicas=2

# Management node

[ndb\_mgmd]

hostname=192.168.1.1

datadir=/var/lib/mysql-cluster

# 1st data node

[ndbd]

hostname=192.168.1.2

NodeId=2

datadir=/usr/local/mysql/data

# 2nd data node

[ndbd]

hostname=192.168.1.3

NodeId=3

datadir=/usr/local/mysql/data

# SQL node

[mysqld]

hostname=192.168.1.4

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-5.7.23 ndb-7.6.7

2018-08-31 06:11:22 [MgmtSrvr] INFO -- The default config directory '/usr/mysql-cluster' does not exist. Trying to create it...

2018-08-31 06:11:22 [MgmtSrvr] INFO -- Sucessfully created config directory

This shows that MySQL NDB Management server is setup and running on your Alibaba Cloud.

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

First, let’s kill the process that we have started above:

$ 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.

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

## Step 2: Configuring MySQL Cluster Data Nodes on AWS Cloud

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 are 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

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

$ wget https://dev.mysql.com/get/Downloads/MySQL-Cluster-7.6/mysql-cluster-community-data-node\_7.6.7-1ubuntu16.04\_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.

So first, update the package list information index:

$ sudo apt-get update

Then install **libclass-methodmaker-perl** using the command below:

$ sudo apt-get install libclass-methodmaker-perl

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

$ sudo dpkg -i mysql-cluster-community-data-node\_7.6.7-1ubuntu16.04\_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 private IP address where the cluster management node resides, in our case **192.168.1.1**

$ sudo nano /etc/my.cnf

Paste the content below on the file and remember to replace **192.168.1.1** with the private IP address of your MySQL cluster management node:

[mysql\_cluster]

ndb-connectstring=192.168.1.1

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:

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

Start the data node process by running the command below:

$ ndbd

You should get the below output:

2018-08-31 18:39:48 [ndbd] INFO -- Angel connected to '192.168.1.1:1186'

2018-08-31 18:39:48 [ndbd] INFO -- Angel allocated nodeid: 2

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

Paste the content below:

[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

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

## Step 3: Configuring SQL Cluster Node on AWS Cloud

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 198.18.0.4. 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

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

$ wget https://dev.mysql.com/get/Downloads/MySQL-Cluster-7.6/mysql-cluster\_7.6.7-1ubuntu16.04\_amd64.deb-bundle.tar

The above command will download a tar archive file e.g. **mysql-cluster\_7.6.7-1ubuntu16.04\_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. **installation**:

$ sudo mkdir installation

Then, unzip the **mysql-cluster\_7.6.7-1ubuntu16.04\_amd64.deb-bundle.tar** deb files to the installation directory:

$ sudo tar -xvf mysql-cluster\_7.6.7-1ubuntu16.04\_amd64.deb-bundle.tar -C installation/

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

$ cd installation

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\_7.6.7-1ubuntu16.04\_amd64.deb

$ sudo dpkg -i mysql-cluster-community-client\_7.6.7-1ubuntu16.04\_amd64.deb

$ sudo dpkg -i mysql-client\_7.6.7-1ubuntu16.04\_amd64.deb

$ sudo dpkg -i mysql-cluster-community-server\_7.6.7-1ubuntu16.04\_amd64.deb

$ sudo dpkg -i mysql-server\_7.6.7-1ubuntu16.04\_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 172.16.0.1 with the private IP address of your Cluster Management Server :

[mysqld]

ndbcluster

[mysql\_cluster]

ndb-connectstring=192.168.1.1 # private ip address of the cluster manager

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: Testing MySQL NDB Cluster on AWS Cloud

The following MySQL cluster nodes should now be up and running on your Alibaba Cloud console.

* 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 5

Server version: 5.7.23-ndb-7.6.7 MySQL Cluster Community Server (GPL)

Create a sample database:

mysql > create database test\_database;

Switch to the database:

mysql > use test\_database;

Create a sample table:

mysql> create table students(student\_Id BIGINT NOT NULL, student\_name VARCHAR(40) NOT NULL) ENGINE=ndbcluster;

**Insert data**:

mysql> Insert into students(student\_Id, student\_name) values('1', 'JOHN DOE');

mysql> Insert into students(student\_Id, student\_name) values('2', 'MARY DOE');

Confirm the presence of data:

mysql> select \* from students;

+------------+--------------+

| student\_Id | student\_name |

+------------+--------------+

| 1 | JOHN DOE |

| 2 | MARY DOE |

+------------+--------------+

2 rows in set (0.00 sec)

You can also test the if the MySQL cluster is working on Alibaba cloud by issuing the command below on the **SQL** node to open the **NDB Cluster Management Client(ndb\_mgm)**. Don’t confuse this process with **NDB Cluster Management Server (ndb\_mgmd**) that we created in step 1:

$ ndb\_mgm

**Output:**

-- NDB Cluster -- Management Client --

ndb\_mgm>

Once you get a prompt from **ndb\_mgm**, enter the **show** command and hit **Enter** to check the cluster’s status as shown below:

ndb\_mgm> show;

Connected to Management Server at: 192.168.1.1:1186

Cluster Configuration

---------------------

[ndbd(NDB)] 2 node(s)

id=2 @192.168.1.2 (mysql-5.7.23 ndb-7.6.7, Nodegroup: 0, \*)

id=3 @192.168.1.3 (mysql-5.7.23 ndb-7.6.7, Nodegroup: 0)

[ndb\_mgmd(MGM)] 1 node(s)

id=1 @192.168.1.1 (mysql-5.7.23 ndb-7.6.7)

[mysqld(API)] 1 node(s)

id=4 @192.168.1.4 (mysql-5.7.23 ndb-7.6.7)