Muhammad Abyan Dzaka
05111640007003

INSTALASI MySQL CLUSTER DENGAN MENGGUNAKAN PROXY SQL SEBAGAI LOAD BALACER

1. Yang Perlu Disiapkan

Dalam instalasi MySQL Cluster ini, saya membutuhkan beberapa *tool* dan bahan dalam mengimplementasikannya, di antaranya:

- Oracle Virtual Box release tahun 2019
- Vagrant
- Ubuntu Bento 18.04
- Data sampel MySQL
- SQLYog

2. Arsitektur

Sebelum melakukan instalasi, saya merancang terlebih dahulu arsitektur *node* yang akan dibangun sebagai berikut:

No	IP Address	Hostname	Deskripsi
1	192.168.33.11	clusterdb1	Manager Node
2	192.168.33.12	clusterdb2	Data Node
3	192.168.33.13	clusterdb3	Data Node
4	192.168.33.14	clusterdb4	Server (API) / Service
5	192.168.33.15	clusterdb5	Server (API) / Service
6	192.168.33.16	clusterdb6	Proxy SQL

3. Instalasi dan Konfigurasi Manager Node

Jadi saya akan membangun 6 *node* tersebut berbasiskan virtual machine yang berjalan pada Oracle Virtual Box. Langkah awalnya, kita download terlebih dahulu file NDB Management Server di sini. Lalu log in ke Manager Node dan jalankan skrip di bawah ini:

```
$ wget https://dev.mysql.com/get/Downloads/MySQL-Cluster-7.6/mysql-cluster-community-management-server_7.6.9-1ubuntu18.04_amd64.deb
```

Instal ndb_mgmd menggunakan dpkg:

```
$ sudo dpkg -i  mysql-cluster-community-management-server_7.6.9-1ubuntu18.04_amd64.deb
```

Konfigurasi ndb_mgmd dengan cara membuat file baru yang diletakkan di /var/lib/mysql-cluster/config.ini.

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

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

Lalu masukkan skrip berikut ini:

config.ini

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

[ndb_mgmd]
# Management process options:
hostname=192.168.33.11 # Hostname of the manager
datadir=/var/lib/mysql-cluster  # Directory for the log files

[ndbd]
hostname=192.168.33.12 # 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=192.168.33.13 # 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=192.168.33.14 # In our case the MySQL server/client is on the same Droplet as the cluster manager

[mysqld]
hostname=192.168.33.15
```

[ndb_mgmd] adalah *node* untuk Manager. [ndbd] adalah *node* untuk Data. [mysqld] adalah *node* untuk service-nya. Jika sudah selesai, simpan file dan tutup *editor*-nya. Sekarang kita akan mencoba menjalankan managernya sebagai berikut:

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

Jika berhasil, maka akan ada notifikasi sukses. Nah, sekarang kita akan membuat agar Manager ini akan aktif otomatis saat pertama kali menyalakannya dengan cara:

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

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

Masukkan skrip berikut:

/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
```

Silahkan disimpan dan ditutup *file*-nya. Sekarang kita akan me-*reload* sistem Manager-nya menggunakan daemon-reload:

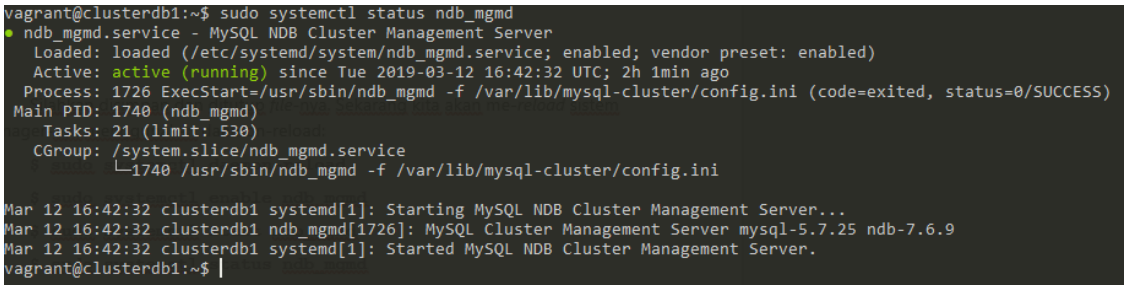
```
$ sudo systemctl daemon-reload
```

```
$ sudo systemctl enable ndb_mgmd
```

```
$ sudo systemctl start ndb_mgmd
```

```
$ sudo systemctl status ndb_mgmd
```

Jika berhasil, seharusnya akan muncul status berikut:



Langkah terakhir, kita harus mengizinkan koneksi yang masuk dari *node* MySQL Cluster yang lain ke *private network* kita.

```
$ sudo ufw allow from 192.168.33.12
```

```
$ sudo ufw allow from 192.168.33.13
```

```
$ sudo ufw allow from 192.168.33.14
```

```
$ sudo ufw allow from 192.168.33.15
```

4. Instalasi dan Konfigurasi Data Node

Langkah awalnya, kita download terlebih dahulu file NDB Data Node Binaries di sini. Lalu log in ke Data Node dan jalankan skrip di bawah ini:

```
$ wget https://dev.mysql.com/get/Downloads/MySQL-Cluster-7.6/mysql-cluster-community-data-node_7.6.9-1ubuntu18.04_amd64.deb
```

```
$ sudo apt update
```

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

```
$ sudo dpkg -i mysql-cluster-community-data-node_7.6.9-1ubuntu18.04_amd64.deb
```

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

Masukkan skrip berikut:

```
[mysql_cluster]
# Options for NDB Cluster processes:
ndb-connectstring=192.168.33.11  # location of cluster manager
```

Simpan lalu tutup file. Sebelum memulai *daemon*-nya, kita membuat direktori dalam *node*-nya menjalankan *data node*-nya.

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

```
$ sudo ndbd
```

Maka akan menghasilkan output demikian:

```
vagrant@clusterdb2:~$ sudo ndbd
2019-03-12 18:58:22 [ndbd] INFO     -- Angel connected to '192.168.33.11:1186'
2019-03-12 18:58:22 [ndbd] INFO     -- Angel allocated nodeid: 2
vagrant@clusterdb2:~$
```

Nah, *daemon data node* NDB sudah berhasil diinstal dan berjalan di server. Kita juga harus mengizinkan koneksi yang masuk dari *node* MySQL Cluster yang lain ke *private network* kita.

```
$ sudo ufw allow from 192.168.33.11
```

```
$ sudo ufw allow from 192.168.33.12/13
```

```
$ sudo ufw allow from 192.168.33.14
```

```
$ sudo ufw allow from 192.168.33.15
```

Langkah terakhir, mengatur *data node daemon* agar aktif otomatis saat pertama kali dijalankan.

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

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

Masukkan skrip berikut:

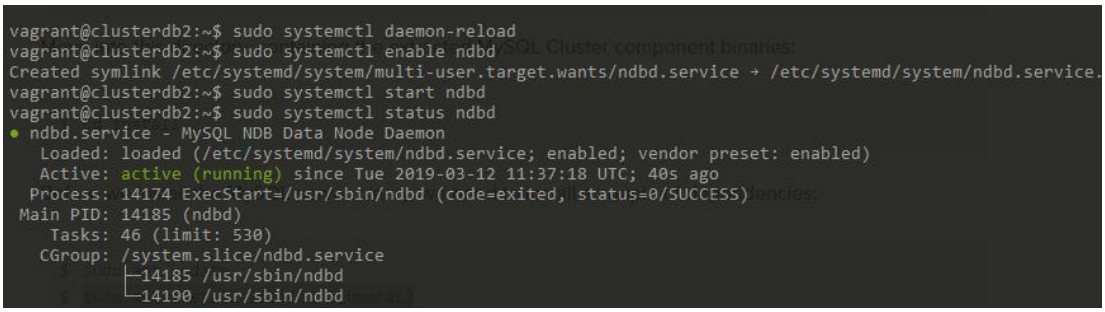
/etc/systemd/system/ndbd.service

```
[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
```

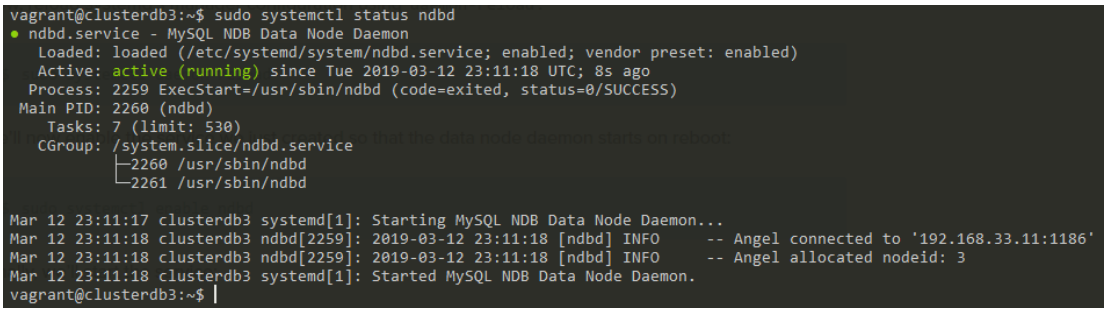
Simpan dan tutup file tersebut. Sekarang kita tinggal melakukan *daemon-reload*:

```
$ sudo systemctl daemon-reload

$ sudo systemctl enable ndbd

$ sudo systemctl start ndbd

$ sudo systemctl status ndbd
```

Maka akan muncul *output* seperti ini:



Status "active" pada clusterdb2 atau Data Node 1.



Status "active" pada clusterdb3 atau Data Node 2.

5. Instalasi dan Konfigurasi Service (API)

Langkah awalnya, kita download terlebih dahulu file DEB Bundle di sini. Lalu log in ke Service Node dan jalankan skrip di bawah ini:

```
$ wget mysql-cluster_7.6.9-1ubuntu18.04_amd64.deb-bundle.tar

$ mkdir install

$ tar -xvf mysql-cluster_7.6.9-1ubuntu18.04_amd64.deb-bundle.tar -C install/

$ cd install

$ sudo apt-get update

$ sudo apt install libaio1 libmecab2
```

Menginstal dari file hasil ekstraksi:

```
$ sudo dpkg -i mysql-common-mysql-common_7.6.9-1ubuntu18.04_amd64.deb
```

```
$ sudo dpkg -i mysql-cluster-community-client_7.6.9-1ubuntu18.04_amd64.deb
```

```
$ sudo dpkg -i mysql-client_7.6.9-1ubuntu18.04_amd64.deb
```

```
$ sudo dpkg -i mysql-cluster-community-server_7.6.9-1ubuntu18.04_amd64.deb
```

Masukkan password MySQL dan *re-enter* lagi.

```
$ sudo dpkgi -i mysql-server_7.6.9-1ubuntu18.04_amd64.deb
```

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

Tambahkan konfigurasi ini ke file my.cnf:

```
[mysqld]
# Options for mysqld process:
ndbcluster                      # run NDB storage engine
bind-address = 192.168.33.14    # Listen address
#bind-address = 192.168.33.12    # Listen address

[mysql_cluster]
# Options for NDB Cluster processes:
ndb-connectstring=192.168.33.11  # location of management server
```

Simpan dan tutup file. Sekarang kita akan me-*restart Service Node* nya agar dapat langsung berjalan saat pertama kali dinyalakan.

```
$ sudo systemctl restart mysql
```

```
$ sudo systemctl enable mysql
```

6. Mengecek Hasil Instalasi MySQL Cluster

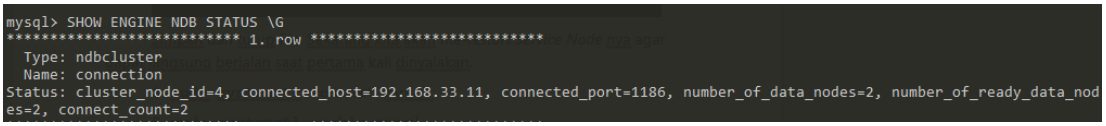
Untuk mengecek hasil instalasi MySQL Cluster, silahkan *log in* ke Cluster Manager atau SQL Server Node. Di sini kita *log in* ke SQL Server Node atau Service Node-nya. Lalu ketikkan skrip sebagai berikut untuk masuk ke *root*:

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

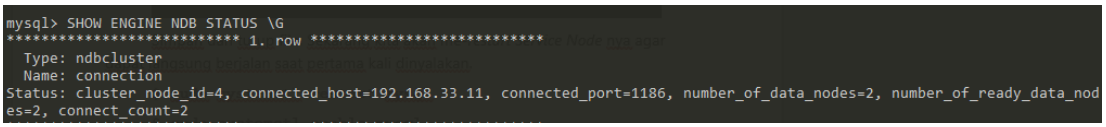
Saat sudah masuk ke *root* mysql, jalankan *command* berikut:

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

Maka, akan muncul informasi tentang NDB Cluster Engine-nya.



Status NDB Cluster Engine pada clusterdb4 atau Service 1.

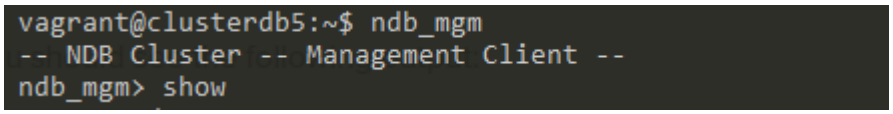


Status NDB Cluster Engine pada clusterdb5 atau Service 2.

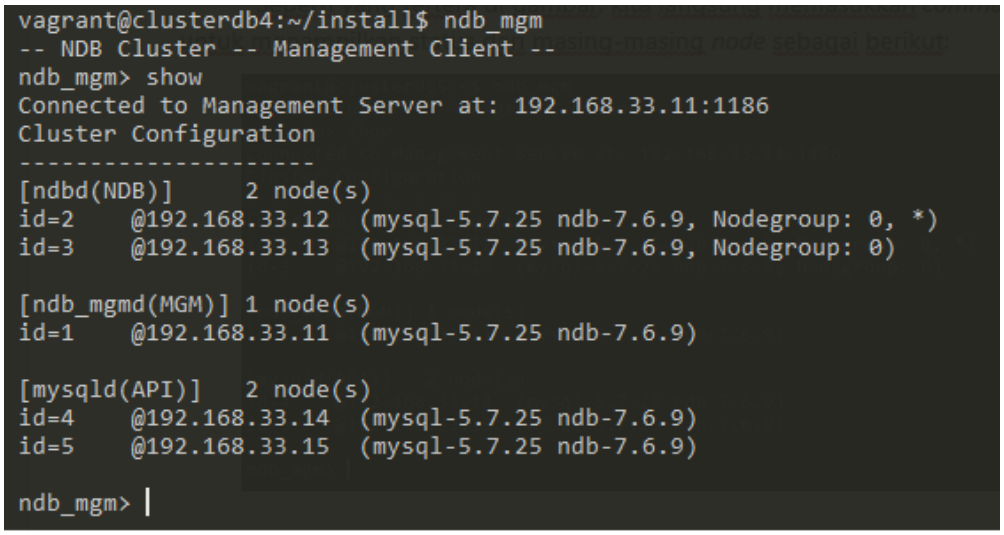
Ini menunjukkan kita telah berhasil terkoneksi dengan MySQL Cluster kita. Selain itu, kita akan mencoba menguji kembali untuk mengkonfirmasi sekali lagi kalau *cluster*-nya berfungsi dengan baik. Sebelumnya, kita keluar dulu dari *root* mysql dengan menggunakan *command* "exit". Setelah kembali ke vagrant, kita masukkan *command* sebagai berikut:

```
$ ndb_mgm
```

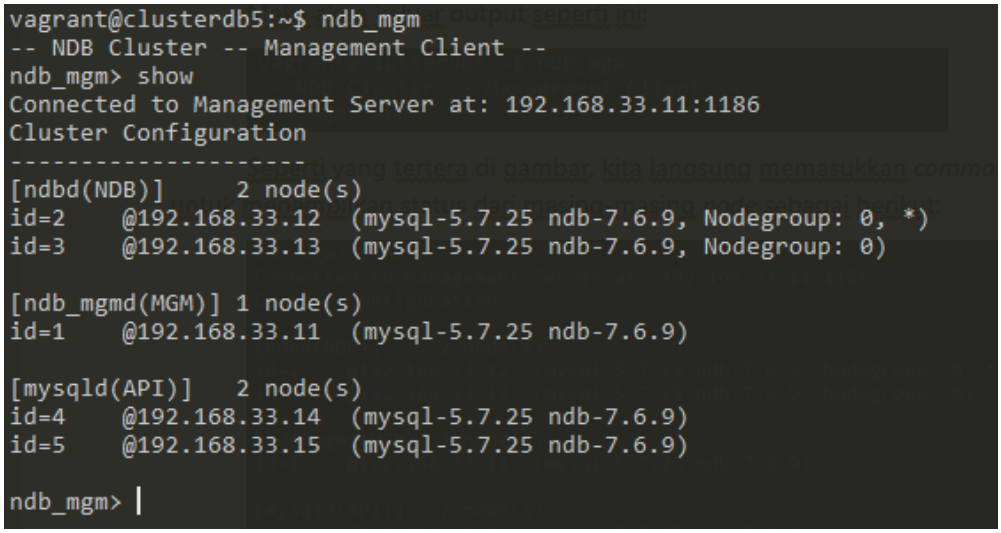
Maka akan keluar output seperti ini:



Seperti yang tertera di gambar, kita langsung memasukkan *command* "show" untuk menampilkan status dari masing-masing *node* sebagai berikut:



Informasi *cluster* pada clusterdb4 atau Service 1.



Informasi *cluster* pada clusterdb5 atau Service 2.

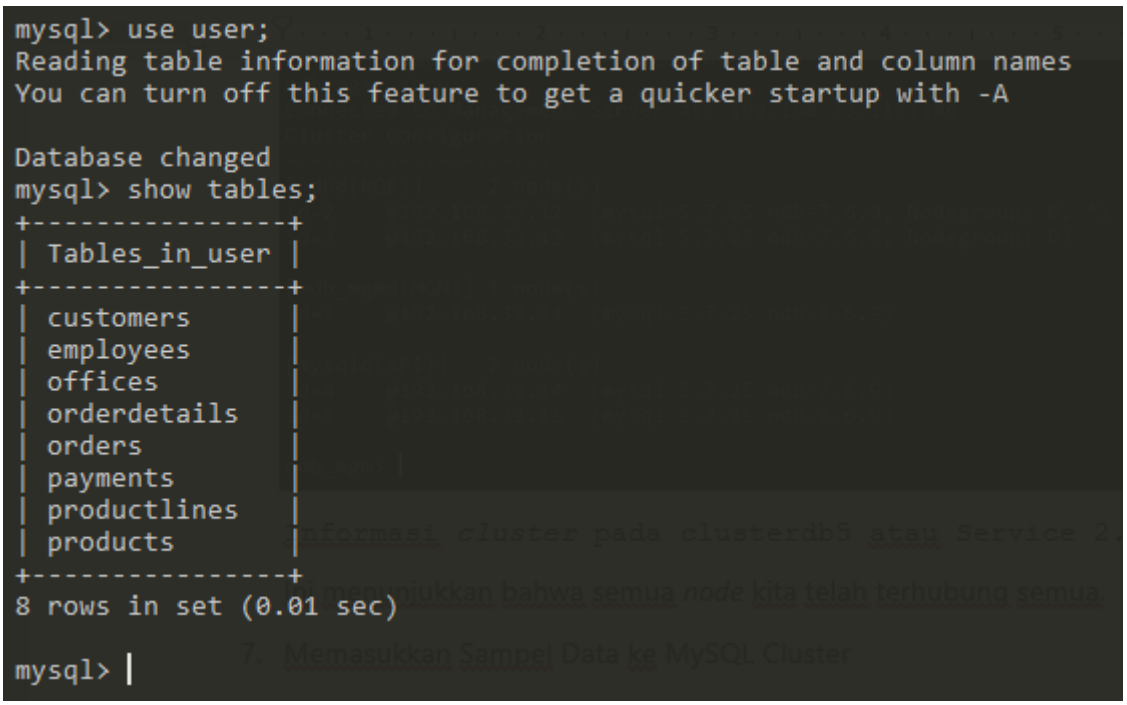
Ini menunjukkan bahwa semua *node* kita telah terhubung semua.

7. Memasukkan Sampel Data ke MySQL Cluster

Sekarang kita akan memasukkan *dummy data* dari internet yang telah disediakan ke salah satu Service Node. Kita gunakan Service Node 2 atau clusterdb5 untuk langsung mengimpor file ".sql"-nya dengan cara menggunakan *command* sebagai berikut:

```
$ mysql -u root -p < /vagrant/config/mysqlsample.sql
```

Berikut daftar table dari mysqlsample.sql di Service 2:



```
mysql> use user;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+----------------+
| Tables_in_user |
+----------------+
| customers      |
| employees      |
| offices        |
| orderdetails   |
| orders         |
| payments       |
| productlines   |
| products       |
+----------------+
8 rows in set (0.01 sec)

mysql>
```

Pada Service 1, juga bisa menampilkan tabel yang diimpor dari Service 2

8. Proses Query di Node MySQL API yang Berbeda

Kita akan mencoba menambahkan data ke dalam database yang baru diimpor tadi, juga memastikan bahwa apabila salah satu dari MySQL API mati, layanan tetap berjalan. Di sini saya akan hanya menampilkan “last name”, “first name”, dan “jobtitle” yang “officecode”-nya angka “6”.

```
mysql> select lastName, firstName, jobTitle from employees where officeCode =6;
+-----------+-----------+----------------------+
| lastName  | firstName | jobTitle             |
+-----------+-----------+----------------------+
| Marsh     | Peter     | Sales Rep            |
| Fixter    | Andy      | Sales Rep            |
| Patterson | William   | Sales Manager (APAC) |
| King      | Tom       | Sales Rep            |
+-----------+-----------+----------------------+
4 rows in set (0.04 sec)

mysql>
```

Saya akan mencoba memasukkan data saya ke dalam table tersebut, lalu kita lihat hasilnya di Service 2, sebab kita akan meng-*input*-kan di Service 1.

```
Database changed
mysql> select lastName, firstName, jobTitle from employees where officeCode=6;
+-----------+-----------+---------------------+
| lastName  | firstName | jobTitle            |
+-----------+-----------+---------------------+
| King      | Tom       | Sales Rep           |
| Marsh     | Peter     | Sales Rep           |
| Fixter    | Andy      | Sales Rep           |
| Patterson | William   | Sales Manager (APAC) |
+-----------+-----------+---------------------+
4 rows in set (0.00 sec)

mysql> insert into `user`.`employees`(`employeeNumber`,`lastName`,`firstName`,`extension`,`email`,`officeCode`,`reportsTo`,`jo
bTitle`) values ( '8888','Dzaka','Abyan','x4566','abyan@gmail.com','6','2222','Owner')
    -> ;
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint fails (`user`.`employees`, CONSTRAINT `employee
s_ibfk_1` FOREIGN KEY (`reportsTo`) REFERENCES `employees` (`employeeNumber`) ON DELETE NO ACTION ON UPDATE NO ACTION)
mysql> insert into `user`.`employees`(`employeeNumber`,`lastName`,`firstName`,`extension`,`email`,`officeCode`,`reportsTo`,`jo
bTitle`) values ( '8888','Dzaka','Abyan','x4566','abyan@gmail.com','6','1002','Owner');
Query OK, 1 row affected (0.00 sec)

mysql>
```

Data saya berhasil ditambahkan ke dalam tabelnya.

Sekarang, kita coba lihat hasil datanya di Service 2.

```
mysql> use user;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> select lastName, firstName, jobTitle from employees where officeCode=6;
+-----------+-----------+----------------------+
| lastName  | firstName | jobTitle             |
+-----------+-----------+----------------------+
| King      | Tom       | Sales Rep            |
| Marsh     | Peter     | Sales Rep            |
| Fixter    | Andy      | Sales Rep            |
| Dzaka     | Abyan     | Owner                |
| Patterson | William   | Sales Manager (APAC) |
+-----------+-----------+----------------------+
5 rows in set (0.01 sec)

mysql>
```

Data yang saya masukkan di Service 1, bisa dilihat juga melalu Service 2. Sekarang kita coba jika Service 1 kita matikan, kita akan hapus salah satu data di atas, lalu kita nyalakan lagi Service 1, terus kita lihat apakah datanya ikut ter-update atau tidak.

```
vagrant@clusterdb5:~$ ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm> show
Connected to Management Server at: 192.168.33.11:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=2    @192.168.33.12  (mysql-5.7.25 ndb-7.6.9, Nodegroup: 0, *)
id=3    @192.168.33.13  (mysql-5.7.25 ndb-7.6.9, Nodegroup: 0)

[ndb_mgmd(MGM)] 1 node(s)
id=1    @192.168.33.11  (mysql-5.7.25 ndb-7.6.9)

[mysqld(API)]   2 node(s)
id=4 (not connected, accepting connect from 192.168.33.14)
id=5    @192.168.33.15  (mysql-5.7.25 ndb-7.6.9)

ndb_mgm>
```

Memastikan bahwa Service 1 atau clusterdb4 telah dimatikan. Lalu kita coba hapus data “Tom King” di Service 2.

```
mysql> delete from employees where officeCode = 6 AND lastName = King;
ERROR 1054 (42S22): Unknown column 'King' in 'where clause'
mysql> select lastName, firstName, jobTitle from employees where officeCode=6;
+-----------+-----------+----------------------+
| lastName  | firstName | jobTitle             |
+-----------+-----------+----------------------+
| Marsh     | Peter     | Sales Rep            |
| Fixter    | Andy      | Sales Rep            |
| Dzaka     | Abyan     | Owner                |
| Patterson | William   | Sales Manager (APAC) |
| King      | Tom       | Sales Rep            |
+-----------+-----------+----------------------+
5 rows in set (0.00 sec)

mysql> delete from employees where officeCode = 6 AND lastName = 'King';
Query OK, 1 row affected (0.04 sec)

mysql>
```

Data telah dihapus, tapi belum dilihat di Service 2, karena kita akan melihat di Service 1. Berhubung status Service 1 masih mati, maka kita nyalakan lagi agar bisa kita cek apakah datanya sama dengan Service 2 atau tidak.

```
vagrant@clusterdb5:~$ ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm> show
Connected to Management Server at: 192.168.33.11:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=2    @192.168.33.12  (mysql-5.7.25 ndb-7.6.9, Nodegroup: 0, *)
id=3    @192.168.33.13  (mysql-5.7.25 ndb-7.6.9, Nodegroup: 0)

[ndb_mgmd(MGM)] 1 node(s)
id=1    @192.168.33.11  (mysql-5.7.25 ndb-7.6.9)

[mysqld(API)]   2 node(s)
id=4    @192.168.33.14  (mysql-5.7.25 ndb-7.6.9)
id=5    @192.168.33.15  (mysql-5.7.25 ndb-7.6.9)

ndb_mgm>
```

Service 1 sudah menyala kembali

```
mysql> select lastName, firstName, jobTitle from employees where officeCode=6;
+-----------+-----------+----------------------+
| lastName  | firstName | jobTitle             |
+-----------+-----------+----------------------+
| Marsh     | Peter     | Sales Rep            |
| Fixter    | Andy      | Sales Rep            |
| Dzaka     | Abyan     | Owner                |
| Patterson | William   | Sales Manager (APAC) |
+-----------+-----------+----------------------+
4 rows in set (0.00 sec)

mysql>
```

Data di Service 1 juga ikut ter-*update*. Tidak ada “King” di sana.

9. Instalasi ProxySQL

Arahkan direktori ke "tmp" lalu *download* paket terbaru ProxySQL di sini. Kemudian instal menggunakan "dpkg". Jika sudah, file ".deb" bisa dihapus.

```
$ cd /tmp
```

```
$ curl -OL https://github.com/sysown/proxysql/releases/download/v1.4.4/proxysql_1.4.4-ubuntu16_amd64.deb
```

```
$ sudo dpkg -i proxysql_*
```

```
$ rm proxysql_*
```

Kemudian, *update* repository untuk menginstal mysql-client.

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

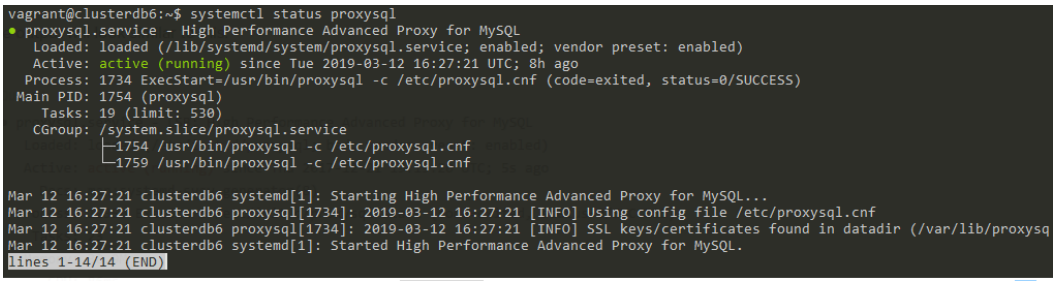
```
$ sudo apt-get install mysql-client
```

Untuk menjalankan ProxySQL-nya, harus dijalankan secara manual.

```
$ sudo systemctl start proxysql
```

```
$ systemctl status proxysql
```

Kalau sudah aktif, maka statusnya akan menjadi demikian:



10. Mengatur Password Administrator ProxySQL

Password *default* dari admin ProxySQL adalah "admin". Jadi, saat mengeksekusi *command* untuk akses *root* ProxySQL, passwordnya adalah "admin".

```
$ mysql -u admin -p -h 127.0.0.1 -P 6032 --prompt='ProxySQLAdmin> '
```

Kalau sudah berhasil masuk ke *root,* sekarang mengganti password *default* tersebut ke password yang diinginkan.

```
ProxySQLAdmin> UPDATE global_variables SET variable_value='admin:password' WHERE variable_name='admin-admin_credentials';
```

Kata-kata "password" di dalam *syntax* tersebut adalah sandi kita. Jadi saya mengubah dari sandi dasarnya (admin) menjadi "password". Kemudian eksekusi juga *command* berikut agar sistemnya termodifikasi.

```
ProxySQLAdmin> LOAD ADMIN VARIABLES TO RUNTIME;

ProxySQLAdmin> SAVE ADMIN VARIABLES TO DISK;
```

11. Konfigurasi Monitor dan User di MySQL (API)

Masuk ke *root* MySQL di salah satu MySQL (API) node-nya sekaligus mengimpor data "addition_to_sys.sql"

```
mysql> mysql -u admin -p <
/vagrant/config/addition_to_sys.sql
```

Setelah itu, kita akan membuat *dedicated user* yang dinamakan "monitor" dan *user* biasa yang dinamakan "abyan" dengan cara mengimpor data "proxy_connection.sql".

```
mysql> mysql -u admin -p <
/vagrant/config/proxy_connection.sql
```

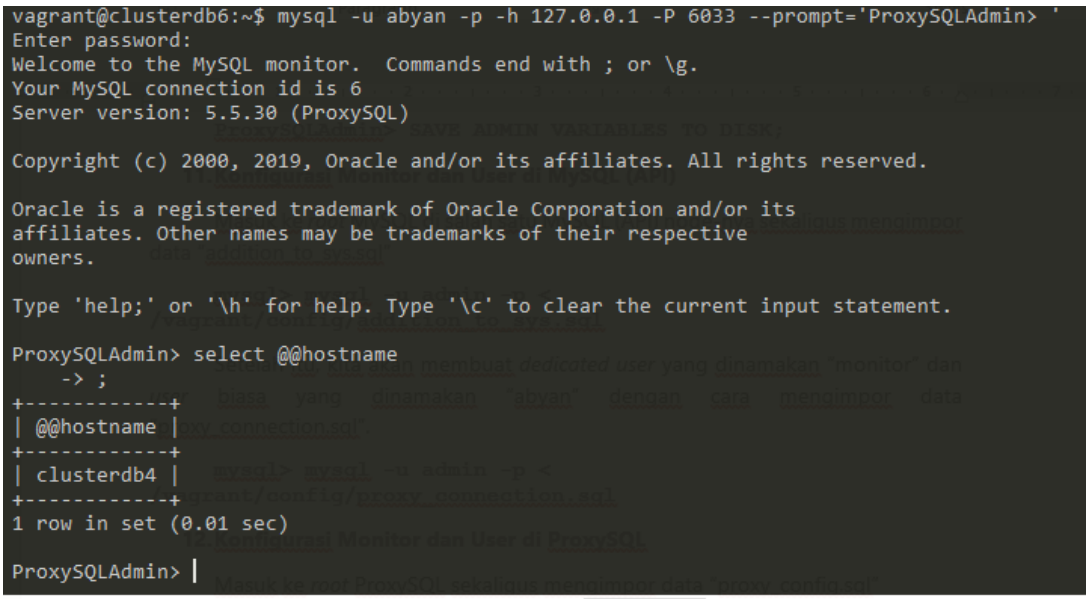
12. Konfigurasi Monitor dan User di ProxySQL

Masuk ke *root* ProxySQL sekaligus mengimpor data "proxy_config.sql"

```
ProxySQL> sudo mysql -u admin -p -h 127.0.0.1 -P 6032 –
prompt='ProxySQLAdmin>' < /vagrant/config/proxy_config.sql
```

Sekarang kita cek user baru yang telah dibuat melalui impor data ".sql" di atas untuk mengetahui Service Node mana yang sedang kita akses dengan cara masuk ke ProxySQL sebagai user yang baru dibuat tadi, lalu eksekusi *command* berikut:

```
ProxySQLAdmin> select @@hostname
```



13. Verifikasi Konfigurasi Pada ProxySQL

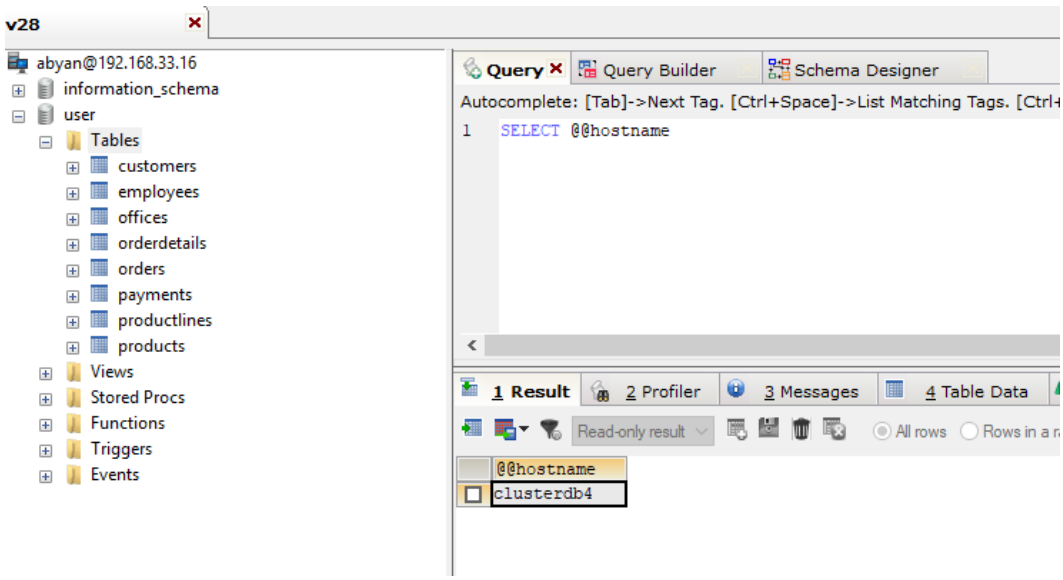
Sekarang kita bisa mengecek status Service Node yang terhubung dengan ProxySQL. Apakah Service tersebut "online" atau "shunned" bisa dipantau melalui ProxySQL. Sebelumnya, masuk ke ProxySQL-nya sebagai "admin".

```
ProxySQLAdmin> SELECT hostgroup_id, hostname, status FROM runtime_mysql_servers;
```

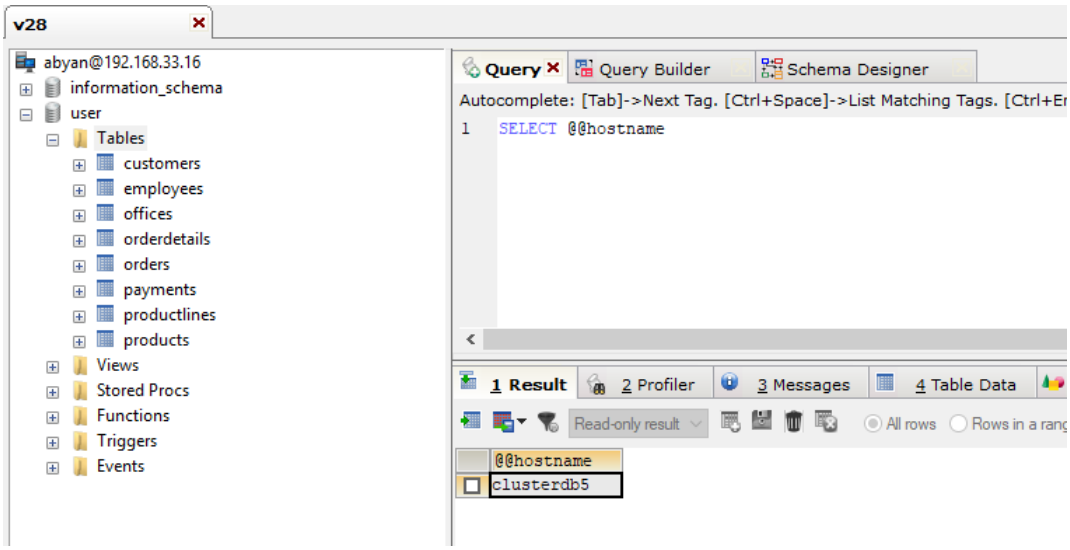
```
ProxySQLAdmin> SELECT hostgroup_id, hostname, status FROM runtime_mysql_servers;
+--------------+---------------+--------+
| hostgroup_id | hostname      | status |
+--------------+---------------+--------+
| 2            | 192.168.33.14 | ONLINE |
| 2            | 192.168.33.15 | ONLINE |
+--------------+---------------+--------+
2 rows in set (0.00 sec)

ProxySQLAdmin>
```

14. Tes Koneksi ke Luar Menggunakan SQLYog

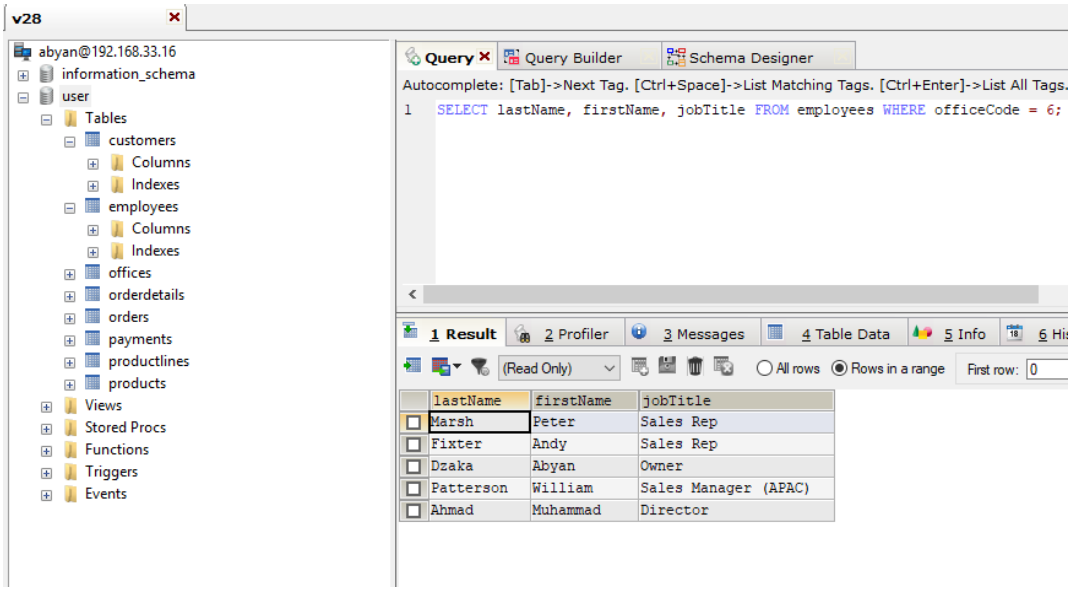
Kita akan mencoba mengakses database melalui SQLYog yang notabene berada di luar environment *virtual machine*-nya.



Coba clusterdb4 yang merupakan Service 1 kita matikan, maka nanti yang diakses kembali adalah clusterdb5 atau Service 2. Seperti gambar berikut:



Lalu coba kita masukkan sebuah data di "employees" melalui SQL Yog, apakah nanti akan ter-*update* juga di MySQL Node-nya.



Sudah diubah melalui SQLYog, sekarang kita cek di MySQL Node-nya.

```
Database changed
mysql> SELECT lastName, firstName, jobTitle FROM employees WHERE officeCode = 6;
+-----------+-----------+----------------------+
| lastName  | firstName | jobTitle             |
+-----------+-----------+----------------------+
| Ahmad     | Muhammad  | Director             |
| Marsh     | Peter     | Sales Rep            |
| Fixter    | Andy      | Sales Rep            |
| Dzaka     | Abyan     | Owner                |
| Patterson | William   | Sales Manager (APAC) |
+-----------+-----------+----------------------+
5 rows in set (0.00 sec)

mysql>
```

Ternyata data di MySQL Node juga ikut ter-*update*.