By petero
Published: 2008-06-15 19:21

How To Set Up A Load-Balanced MySQL Cluster With MySQL 5.1

Version 1.0
Author: Peter Okupski <okupski [at] widzew [dot] net>
Last edited 06/02/2008

This tutorial is based on Falko Timme's tutorial for MySQL Cluster 5.0. It shows how to configure a MySQL 5.1 cluster with five nodes: 1 x management, 2 x storage nodes and 2 x balancer nodes. This cluster is load-balanced by an *Ultra Monkey* package which provides *heartbeat* (for checking if the other node is still alive) and *ldirectord* (to split up the requests to the nodes of the MySQL cluster).

In this document I use Debian Etch 4.0 for all nodes. Therefore the setup might differ a bit for other distributions. The two data nodes were x64 to use all of the 8GB RAM. Servers were compiled from source so you should be able to make it running on any platform. The MySQL version I use in this setup is 5.1.24-rc. It's a release candidate, but I wanted to use 5.1 to take advantage of Memory-Disk Based tables.

Beginning with MySQL 5.1.6, it is possible to store the non-indexed columns of NDB tables on disk, rather than in RAM as with previous versions of MySQL Cluster.[**More here**]

This howto is just a scratch to make it running, for many of you I am suggesting to read some off docs from MySQL page to be prepared to how manage the whole cluster and always know whats going on.

This document comes without warranty of any kind! Bare in mind you need to make tests and prepare your databases before using it in production mode.

1 My Servers

I will use the following Debian servers that are all in the same network (10.0.1.x in this example):

- *mysql-mngt.example.com:***10.0.1.30** [MySQL cluster management server]
- *lb1.example.com:* **10.0.1.31** [Load Balancer 1]

- *lb2.example.com:* **10.0.1.32** [Load Balancer 2]
- *mysql-data1.example.com:* **10.0.1.33** [MySQL cluster node 1]
- *mysql-data2.example.com:* **10.0.1.34** [MySQL cluster node 2]

In addition to that we need a virtual IP address : 10.0.1.10. It will be assigned to the MySQL cluster by the load balancer so that applications have a single IP address to access the cluster.

Although we want to have two MySQL cluster nodes in our MySQL cluster, we still need a third node, the MySQL cluster management server, for mainly one reason: if one of the two MySQL cluster nodes fails, and the management server is not running, then the data on the two cluster nodes will become inconsistent ("*split brain*"). We also need it for configuring the MySQL cluster.

2 MySQL data nodes + 1 cluster management server + 2 Load Balancers = 5

Here is my hardware configuration:

MySQL Data : DELL R300 Intel(R) Quad Core Xeon(R) CPU X3353 @ 2.66GHz, 2x SAS 146GB Drives (Raid 1), 8GB RAM

MySQL LoadBalancer : DELL R200 Intel(R) Xeon(R) CPU 3065 @ 2.33GHz, 2x SATA 250 GB Drives (Raid 1), 1GB RAM

MySQL Management : DELL R200 Intel(R) Celeron(R) CPU 430 @ 1.80GHz, 1x SATA 160 GB Drives (Raid 1), 1GB RAM

As the MySQL cluster management server does not use many resources, you can put additional load balancer on this machine or you can use it for monitoring the whole cluster by **Nagios** or **Cacti**.

2 Set Up The MySQL Cluster Management Server

First we have to download MySQL 5.1.24 (the sources version) and install the cluster management server (*ndb_mgmd*) and the cluster management client (*ndb_mgm* - it can be used to monitor what's going on in the cluster). The following steps are carried out on *mysql-mngt.example.com* (*10.0.1.30*):

mysql-mngt.example.com:

```
cd /usr/src
```



```
wget http://mysql.mirrors.pair.com/Downloads/MySQL-5.1/mysql-5.1.24-rc.tar.gz

tar xvzf mysql-5.1.24-rc.tar.gz

#Lets add proper user and group

groupadd mysql

useradd -g mysql mysql

./configure --prefix=/usr/local/mysql --enable-community-features \

--with-mysqld-user=mysql --with-mysqlmanager --with-plugins=ndbcluster

make

make install
```

This way we are wasting about 124MB space since we do not need all the actual MySQL files, but believe me it's way easier to make any cleanup/upgrade in just one dir `/usr/local/mysql` instead of searching for all files in `/usr/bin` and so on. After compiling we have two directories we are interested in `/usr/local/mysql/bin` and `/usr/local/mysql/libexec` [last one contains the ndb management exec].

Just to make life easier let's add this below to your `PATH` environment, to do so we have to edit file `/root/.bash_profile`:

mysql-mngt.example.com:

```
echo "PATH=$PATH:/usr/local/mysql/bin:/usr/local/mysql/libexec" >>/root/.bash_profile

echo "export PATH" >>/root/.bash_profile
```

Next, we must create the cluster configuration file, `/usr/local/mysql/var/mysql-cluster/config.ini:` [**Hints here**]

mysql-mngt.example.com:

```
mkdir /usr/local/mysql/var/mysql-cluster


cd /usr/local/mysql/var/mysql-cluster

vi config.ini
```

```
[NDBD DEFAULT]
NoOfReplicas=2
DataMemory=80M    # How much memory to allocate for data storage
IndexMemory=18M   # How much memory to allocate for index storage
# For DataMemory and IndexMemory, we have used the
# default values. Since the "world" database takes up
# only about 500KB, this should be more than enough for
# this example Cluster setup.
[MYSQLD DEFAULT]
[NDB_MGMD DEFAULT]
[TCP DEFAULT]
# Section for the cluster management node
[NDB_MGMD]
# IP address of the management node (this system)
HostName=10.0.1.30
# Section for the storage nodes
[NDBD]
# IP address of the first storage node
HostName=10.0.1.33
DataDir=/usr/local/mysql/var/mysql-cluster
BackupDataDir=/usr/local/mysql/var/mysql-cluster/backup
DataMemory=2048M
```

```
[NDBD]
# IP address of the second storage node
HostName=10.0.1.34
DataDir=/usr/local/mysql/var/mysql-cluster
BackupDataDir=/usr/local/mysql/var/mysql-cluster/backup
DataMemory=2048M

# one [MYSQLD] per storage node
[MYSQLD]
[MYSQLD]
```

Please replace the IP addresses in the file appropriately to your setup.

Then we proceed to DataSQL nodes to make necessary dirs and files setup.

mysql-mngt.example.com:

```
ndb_mgmd -f /usr/local/mysql/var/mysql-cluster/config.ini
```

It makes sense to automatically start the management server at system boot time, so we create a very simple init script and the appropriate startup links:

mysql-mngt.example.com:

```
echo '/usr/local/mysql/libexec/ndb_mgmd -f /usr/local/mysql/var/mysql-cluster/config.ini' > /etc/init.d/ndb_mgmd

chmod 755 /etc/init.d/ndb_mgmd

update-rc.d ndb_mgmd defaults
```

3 Set Up The MySQL Cluster Nodes (Storage Nodes)

Now we install *mysql-5.1.14-rc* on both mysql-*data1.example.com* and mysql-*data2.example.com*:

mysql-data1.example.com / mysql-data2.example.com:

```
cd /usr/src

wget http://mysql.mirrors.pair.com/Downloads/MySQL-5.1/mysql-5.1.24-rc.tar.gz

tar xvzf mysql-5.1.24-rc.tar.gz

groupadd mysql

useradd -g mysql mysql

cd /usr/src/mysql-5.1.24-rc

./configure --prefix=/usr/local/mysql --enable-community-features --with-mysqld-user=mysql --with-plugins=ndbcluster

make

make install

/usr/src/mysql-5.1.24-rc/scripts/mysql_install_db --user=mysql

cd /usr/local/mysql

chown -R root:mysql .

chown -R mysql.mysql /usr/local/mysql/var
```

```
cd /usr/src/mysql-5.1.24-rc
```

```
cp support-files/mysql.server /etc/init.d/

chmod 755 /etc/init.d/mysql.server

cd /etc/init.d

update-rc.d mysql.server defaults
```

Then we create the MySQL configuration file */etc/my.cnf* on both nodes:

mysql-data1.example.com / mysql-data2.example.com:

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

```
[client]
port=3306
socket=/tmp/mysql.sock
[mysqld]
ndbcluster
# IP address of the cluster management node
ndb-connectstring=10.0.1.30
default-storage-engine=NDBCLUSTER

#Those are for future tuning
#max_connections=341
#query_cache_size=16M
#thread_concurrency = 4
[mysql_cluster]
# IP address of the cluster management node
ndb-connectstring=10.0.1.30
```

Make sure to fill in the correct IP address of the MySQL cluster management server.

Lets add PATH env. to data nodes also:

mysql-data1.example.com / mysql-data2.example.com:

```
echo "PATH=$PATH:/usr/local/mysql/bin:/usr/local/mysql/libexec" >>/root/.bash_profile

echo "export PATH" >>/root/.bash_profile
```

Next we create the data and backup directories and start the MySQL server on both cluster nodes:

mysql-data1.example.com / mysql-data2.example.com:

```
mkdir /usr/local/mysql/var/mysql-cluster

mkdir /usr/local/mysql/var/mysql-cluster/backup
cd /var/lib/mysql-cluster

ndbd --initial

/etc/init.d/mysql.server start
```

(Please note: we have to run `ndbd --initial` only when we start MySQL for the first time, and if `/usr/local/mysql/mysql-cluster/config.ini` on `mysql-mngt.example.com` changes.)

Now is a good time to set a password for the MySQL root user:

mysql-data1.example.com / mysql-data2.example.com:

```
mysqladmin -u root password yourrootsqlpassword
```

We want to start the cluster nodes at boot time, so we create an *ndbd* init script and the appropriate system startup links:

mysql-data1.example.com / mysql-data2.example.com:

```
echo '/usr/local/mysql/libexec/ndbd' > /etc/init.d/ndbd

chmod 755 /etc/init.d/ndbd

update-rc.d ndbd defaults
```

4 Test The MySQL Cluster

Our MySQL cluster configuration is already finished, now it's time to test it. On the cluster management server (*loadb1.example.com*), run the cluster management client *ndb_mgm* to check if the cluster nodes are connected:

mysql-mngt.example.com:

```
ndb_mgm
```

You should see this:

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

Now type *show;* at the command prompt:

```
show;
```

The output should be like this:

```
ndb_mgm> show;
Connected to Management Server at: localhost:1186
Cluster Configuration
---------------------
[ndbd(NDB)] 2 node(s)
id=2 @10.0.1.33 (Version: 5.1.24, Nodegroup: 0, Master)
id=3 @10.0.1.34 (Version: 5.1.24, Nodegroup: 0)

[ndb_mgmd(MGM)] 1 node(s)
id=1 @10.0.1.30 (Version: 5.1.24)

[mysqld(API)] 2 node(s)
id=4 @10.0.1.34 (Version: 5.1.24)
id=5 @10.0.1.33 (Version: 5.1.24)

ndb_mgm>
```

If you see that your nodes are connected, then everything's ok!

Type

```
quit;
```

to leave the *ndb_mgm* client console.

Now we have to create a test database with a test table and some data on *mysql-data1.example.com*:

mysql-data1.example.com:

```
mysql -u root -p

CREATE DATABASE mysqlclustertest;

USE mysqlclustertest;

CREATE TABLE testtable (I INT) ENGINE=NDBCLUSTER;

INSERT INTO testtable () VALUES (1);

SELECT * FROM testtable;

quit;
```

(Have a look at the *CREATE* statement: We must use *ENGINE=NDBCLUSTER* for all database tables that we want to get clustered! If you use another engine, then clustering will not work!)

The result of the *SELECT* statement should be:

```
mysql> SELECT * FROM testtable;
+------+
| I    |
+------+
| 1    |
+------+
1 row in set (0.03 sec)
```

Now we have to create the same database on *sql2.example.com* (yes, we still have to create it, but afterwards *testtable* and its data should be replicated to *mysql-data2.example.com* because *testtable* uses *ENGINE=NDBCLUSTER*):

mysql-data2.example.com:

```
mysql -u root -p

CREATE DATABASE mysqlclustertest;

USE mysqlclustertest;

SELECT * FROM testtable;
```

The *SELECT* statement should deliver you the same result as before on *mysql-data1.example.com*: [The CREATE statement should fail due to NDBCLUSTER Engine]

```
mysql> SELECT * FROM testtable;
+------+
| I    |
+------+
| 1    |
+------+
1 row in set (0.04 sec)
```

So the data was replicated from *mysql-data1.example.com* to *mysql-data2.example.com*. Now we insert another row into *testtable*:

mysql-data2.example.com:

```
INSERT INTO testtable () VALUES (2);
```

```
quit;
```

Now let's go back to *mysql-data1.example.com* and check if we see the new row there:

mysql-data1.example.com:

```
mysql -u root -p

USE mysqlclustertest;

SELECT * FROM testtable;

quit;
```

You should see something like this:

```
mysql> SELECT * FROM testtable;
+------+
| I    |
+------+
| 1    |
| 2    |
+------+
2 rows in set (0.05 sec)
```

So both MySQL cluster nodes always have the same data!

Now let's see what happens if we stop node 1 (*mysql-data1.example.com*): Run

mysql-data1.example.com:

```
killall ndbd
```

and check with

```
ps aux | grep ndbd | grep -iv grep
```

that all *ndbd* processes have terminated. If you still see *ndbd* processes, run another

```
killall ndbd
```

until all *ndbd* processes are gone.

Now let's check the cluster status on our management server (*mysql-mngt.example.com*):

mysql-mngt.example.com:

```
ndb_mgm
```

On the *ndb_mgm* console, issue

```
show;
```

and you should see this:


```
ndb_mgm> show;
Connected to Management Server at: localhost:1186
Cluster Configuration
---------------------
[ndbd(NDB)] 2 node(s)
id=2 (not connected, accepting connect from 10.0.1.33)
id=3 @10.0.1.34 (Version: 5.1.24, Nodegroup: 0, Master)

[ndb_mgmd(MGM)] 1 node(s)
id=1 @10.0.1.30 (Version: 5.1.24)

[mysqld(API)] 2 node(s)
id=4 @10.0.1.34 (Version: 5.1.24)
id=5 @10.0.1.33 (Version: 5.1.24)

ndb_mgm>
```

You see, *mysql-data1.example.com* is not connected anymore.

Type

```
quit;
```

to leave the *ndb_mgm* console.

Let's check *mysql-data2.example.com*:

mysql-data2.example.com:

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

```
USE mysqlclustertest;

SELECT * FROM testtable;

quit;
```

The result of the *SELECT* query should still be

```
mysql> SELECT * FROM testtable;
+------+
| I    |
+------+
| 1    |
| 2    |
+------+
2 rows in set (0.17 sec)
```

Ok, all tests went fine, so let's start our *mysql-data1.example.com* node again:

mysql-data1.example.com:

```
ndbd
```

5 How To Restart The Cluster

Now let's asume you want to restart the MySQL cluster, for example because you have changed */usr/local/mysql/var/mysql-cluster/config.ini* on *mysql-mngt.example.com* or for some other reason. To do this, you use the *ndb_mgm* cluster management client on *mysql-mngt.example.com*:

mysql-mngt.example.com:

```
ndb_mgm
```

On the *ndb_mgm* console, you type

```
shutdown;
```

You will then see something like this:

```
ndb_mgm> shutdown;
Node 3: Cluster shutdown initiated
Node 2: Node shutdown completed.
2 NDB Cluster node(s) have shutdown.
NDB Cluster management server shutdown.
ndb_mgm>
```

This means that the cluster data nodes *mysql-data1.example.com* and mysql-data*2.example.com* and also the cluster management server have shut down.

Run

```
quit;
```

to leave the *ndb_mgm* console.

To start the cluster management server, do this on *mysql-mngt.example.com*:

mysql-mngt.example.com:

```
ndb_mgmd -f /usr/local/mysql/mysql-cluster/config.ini
```

and on *mysql-data1.example.com* and *mysql-data2.example.com* you can run

mysql-data1.example.com / mysql-data2.example.com:

```
ndbd
```

or, if you have changed */usr/local/mysql/var/mysql-cluster/config.ini* on *mysql-mngt.example.com*:

```
ndbd --initial
```

Afterwards, you can check on *mysql-mngt.example.com* if the cluster has restarted:

mysql-mngt.example.com:

```
ndb_mgm
```

On the *ndb_mgm* console, type

```
show;
```

to see the current status of the cluster. It might take a few seconds after a restart until all nodes are reported as connected.

Type

```
quit;
```

to leave the *ndb_mgm* console.

6 Configure The Load Balancers

Our MySQL cluster is finished now, and you could start using it now. However, we don't have a single IP address that we can use to access the cluster, which means you must configure your applications in a way that a part of it uses the MySQL cluster node 1 (*mysql-data1.example.com*), and the rest uses the other node (*mysql-data2.example.com*). Of course, all your applications could just use one node, but what's the point then in having a cluster if you do not split up the load between the cluster nodes? Another problem is, what happens if one of the cluster nodes fails? Then the applications that use this cluster node cannot work anymore.

The solution is to have a load balancer in front of the MySQL cluster which (as its name suggests) balances the load between the MySQL cluster nodes. The load blanacer configures a virtual IP address that is shared between the cluster nodes, and all your applications use this virtual IP address to access the cluster. If one of the nodes fails, then your applications will still work, because the load balancer redirects the requests to the working node.

Now in this scenario the load balancer becomes the bottleneck. What happens if the load balancer fails? Therefore we will configure two load balancers (*mysql-lb1.example.com* and *mysql-b2.example.com*) in an active/passive setup, which means we have one active load balancer, and the other one is a *hot-standby* and becomes active if the active one fails. Both load balancers use *heartbeat* to check if the other load balancer is still alive, and both load balancers also use *ldirectord*, the actual load balancer the splits up the load onto the cluster nodes. *heartbeat* and *ldirectord* are provided by the *Ultra Monkey* package that we will install.

It is important that *mysql-lb1.example.com* and *mysql-lb2.example.com* have support for *IPVS* (IP Virtual Server) in their kernels. *IPVS* implements transport-layer load balancing inside the Linux kernel.

6.1 Install Ultra Monkey

Ok, let's start: first we enable *IPVS* on *mysql-lb1.example.com* and *mysql-lb2.example.com*:

mysql-lb1.example.com / mysql-lb2.example.com:

```
modprobe ip_vs_dh
```

```
modprobe ip_vs_ftp

modprobe ip_vs

modprobe ip_vs_lblc

modprobe ip_vs_lblcr

modprobe ip_vs_lc

modprobe ip_vs_nq

modprobe ip_vs_rr

modprobe ip_vs_sed

modprobe ip_vs_sh

modprobe ip_vs_wlc

modprobe ip_vs_wrr
```

In order to load the *IPVS* kernel modules at boot time, we list the modules in */etc/modules*:

mysql-lb1.example.com / mysql-lb2.example.com:

```
vi /etc/modules
```

```
ip_vs_dh
```

```
ip_vs_ftp
ip_vs
ip_vs_lblc
ip_vs_lblcr
ip_vs_lc
ip_vs_nq
ip_vs_rr
ip_vs_sed
ip_vs_sh
ip_vs_wlc
ip_vs_wrr
```

Now we edit */etc/apt/sources.list* and add the *Ultra Monkey* repositories (don't remove the other repositories), and then we install *Ultra Monkey*:

mysql-lb1.example.com / mysql-lb2.example.com:

```
vi /etc/apt/sources.list
```

```
deb http://www.ultramonkey.org/download/3/ sarge main
deb-src http://www.ultramonkey.org/download/3 sarge main
```

```
apt-get update

apt-get install ultramonkey libdbi-perl libdbd-mysql-perl libmysqlclient14-dev
```

Now *Ultra Monkey* is being installed. If you see this warning:

```
Â¦ libsensors3 not functional                    Â¦
Â¦                                               Â¦
Â¦ It appears that your kernel is not compiled with sensors support. As a  Â¦
Â¦ result, libsensors3 will not be functional on your system.              Â¦
Â¦                                               Â¦
Â¦ If you want to enable it, have a look at "I2C Hardware Sensors Chip     Â¦
Â¦ support" in your kernel configuration.                                  Â¦
```

you can ignore it.

Answer the following questions:

```
Do you want to automatically load IPVS rules on boot?
<-- No

Select a daemon method.
<-- none
```

The *libdbd-mysql-perl* package we've just installed does not work with MySQL 5 (we use MySQL 5 on our MySQL cluster...), so we install the newest *DBD::mysql* Perl package:

mysql-lb1.example.com / mysql-lb2.example.com:

```
cd /tmp

wget http://search.cpan.org/CPAN/authors/id/C/CA/CAPTTOFU/DBD-mysql-3.0002.tar.gz

tar xvfz DBD-mysql-3.0002.tar.gz
```

```
cd DBD-mysql-3.0002

perl Makefile.PL

make

make install
```

We must enable packet forwarding:

mysql-lb1.example.com / mysql-lb2.example.com:

```
vi /etc/sysctl.conf
```

```
# Enables packet forwarding
net.ipv4.ip_forward = 1
```

```
sysctl -p
```

6.2 Configure heartbeat

Next we configure *heartbeat* by creating three files (all three files must be identical on *mysql-lb1.example.com* and *mysql-lb2.example.com*):

mysql-lb1.example.com / mysql-lb2.example.com:

```
vi /etc/ha.d/ha.cf
```

```
logfacility     local0
bcast     eth0
mcast eth0 225.0.0.1 694 1 0
auto_failback off
node     mysql-lb1
node     mysql-lb2
respawn hacluster /usr/lib/heartbeat/ipfail
apiauth ipfail gid=haclient uid=hacluster
```

Please note: you must list the node names (in this case *mysql-lb1* and *lmysql-lb2*) as shown by

```
uname -n
```

IP addresses does not work here, it is also good idea to add proper entries on both load balancers(*mysql-lb1* and *lmysql-lb2*) in

```
vi /etc/hosts
```

```
127.0.0.1     localhost
10.0.1.31     mysql-lb1.example.com    mysql-lb1
10.0.1.32     mysql-lb2.example.com    mysql-lb2
```

Now let's change here your Virtual MySQL IP:

```
vi /etc/ha.d/haresources
```



```
loadb1       \
      ldirectord::ldirectord.cf \
      LVSSyncDaemonSwap::master \
      IPaddr2::
10.0.1.10
```

/24/eth0/**10.0.1.255**

You must list one of the load balancer node names (here: *mysql-lb1*) and list the virtual IP address (*10.0.1.10*) together with the correct netmask (*24*) and broadcast address (*10.0.1.255*). If you are unsure about the correct settings, **http://www.subnetmask.info/** might help you.

```
vi /etc/ha.d/authkeys
```

```
auth 3
3 md5 somerandomstring
```

somerandomstring is a password which the two *heartbeat* daemons on *loadb1* and *loadb2* use to authenticate against each other. Use your own string here. You have the choice between three authentication mechanisms. I use *md5* as it is the most secure one.

/etc/ha.d/authkeys should be readable by root only, therefore we do this:

mysql-lb1.example.com / mysql-lb2.example.com:

```
chmod 600 /etc/ha.d/authkeys
```

6.3 Configure ldirectord

Now we create the configuration file for *ldirectord*, the load balancer:

mysql-lb1.example.com / mysql-lb2.example.com:

```
vi /etc/ha.d/ldirectord.cf
```

```
# Global Directives
checktimeout=10
checkinterval=2
autoreload=no
logfile="local0"
quiescent=yes
virtual = 10.0.1.10:3306
        service = mysql
        real = 10.0.1.33:3306 gate
        real = 10.0.1.34:3306 gate
        checktype = negotiate
        login = "ldirector"
        passwd = "ldirectorpassword"
        database = "ldirectordb"
        request = "SELECT * FROM connectioncheck"
        scheduler = wrr
```

Please fill in the correct virtual IP address (*10.0.1.10*) and the correct IP addresses of your MySQL cluster nodes (*10.0.1.33* and *10.0.1.34*). *3306* is the port that MySQL runs on by default. We also specify a MySQL user (*ldirector*) and password (*ldirectorpassword*), a database (*ldirectordb*) and an SQL query. *ldirectord* uses this information to make test requests to the MySQL cluster nodes to check if they are still available. We are going to create the *ldirector* database with the *ldirector* user in the next step.

Now we create the necessary system startup links for *heartbeat* and remove those of *ldirectord* (bacause *ldirectord* will be started by *heartbeat*):

mysql-lb1.example.com / mysql-lb2.example.com:

```
update-rc.d -f heartbeat remove

update-rc.d heartbeat start 75 2 3 4 5 . stop 05 0 1 6 .

update-rc.d -f ldirectord remove
```

6.4 Create A Database Called ldirector

Next we create the *ldirector* database on our MySQL cluster nodes *mysql-data1.example.com* and *mysql-data2.example.com*. This database will be used by our load balancers to check the availability of the MySQL cluster nodes.

mysql-data1.example.com:

```
mysql -u root -p

GRANT ALL ON ldirectordb.* TO 'ldirector'@'%' IDENTIFIED BY 'ldirectorpassword';

FLUSH PRIVILEGES;

CREATE DATABASE ldirectordb;

USE ldirectordb;

CREATE TABLE connectioncheck (I INT) ENGINE=NDBCLUSTER;

INSERT INTO connectioncheck () VALUES (1);

quit;
```

mysql-data2.example.com:

```
mysql -u root -p

GRANT ALL ON ldirectordb.* TO 'ldirector'@'%' IDENTIFIED BY 'ldirectorpassword';

FLUSH PRIVILEGES;

CREATE DATABASE ldirectordb;

quit;
```

6.5 Prepare The MySQL Cluster Nodes For Load Balancing

Finally we must configure our MySQL cluster nodes *mysql-data1.example.com* and *mysql-data2.example.com* to accept requests on the virtual IP address *192.168.0.105*.

mysql-data1.example.com / mysql-data2.example.com:

```
apt-get install iproute
```

Add the following to */etc/sysctl.conf*:

mysql-data1.example.com / mysql-data2.example.com:

```
vi /etc/sysctl.conf
```

```
# Enable configuration of arp_ignore option
net.ipv4.conf.all.arp_ignore = 1
```

```
# When an arp request is received on eth0, only respond if that address is
# configured on eth0. In particular, do not respond if the address is
# configured on lo
net.ipv4.conf.eth0.arp_ignore = 1

# Ditto for eth1, add for all ARPing interfaces
#net.ipv4.conf.eth1.arp_ignore = 1



# Enable configuration of arp_announce option
net.ipv4.conf.all.arp_announce = 2

# When making an ARP request sent through eth0 Always use an address that
# is configured on eth0 as the source address of the ARP request.  If this
# is not set, and packets are being sent out eth0 for an address that is on
# lo, and an arp request is required, then the address on lo will be used.
# As the source IP address of arp requests is entered into the ARP cache on
# the destination, it has the effect of announcing this address.  This is
# not desirable in this case as adresses on lo on the real-servers should
# be announced only by the linux-director.
net.ipv4.conf.eth0.arp_announce = 2

# Ditto for eth1, add for all ARPing interfaces
#net.ipv4.conf.eth1.arp_announce = 2
```

```
sysctl -p
```

Add this section for the virtual IP address to */etc/network/interfaces*:

mysql-data1.example.com / mysql-data2.example.com:

```
vi /etc/network/interfaces
```

```
auto lo:0
iface lo:0 inet static
  address 10.0.1.10
  netmask 255.255.255.255
  pre-up sysctl -p > /dev/null
```

```
ifup lo:0
```

7 Start The Load Balancer And Do Some Testing

Now we can start our two load balancers for the first time:

mysql-lb1.example.com / mysql-lb2.example.com:

```
/etc/init.d/ldirectord stop

/etc/init.d/heartbeat start
```

If you don't see errors, you should now reboot both load balancers.If you do see errors go to the end of this tutorial I might know whats the problem :) [Chapter 8]

mysql-lb1.example.com / mysql-lb2.example.com:

```
shutdown -r now
```

After the reboot we can check if both load balancers work as expected :

mysql-lb1.example.com / mysql-lb2.example.com:

```
ip addr sh eth0
```

The active load balancer should list the virtual IP address (`10.0.1.10`):

```
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:16:3e:45:fc:f8 brd ff:ff:ff:ff:ff:ff
    inet 10.0.1.31/24 brd 192.168.0.255 scope global eth0
    inet 10.0.1.10/24 brd 192.168.0.255 scope global secondary eth0
```

The hot-standby should show this:

```
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:16:3e:16:c1:4e brd ff:ff:ff:ff:ff:ff
    inet 10.0.1.32/24 brd 192.168.0.255 scope global eth0
```

mysql-lb1.example.com / mysql-lb2.example.com:

```
ldirectord ldirectord.cf status
```

Output on the active load balancer:

```
ldirectord for /etc/ha.d/ldirectord.cf is running with pid: 1603
```

Output on the hot-standby:

```
ldirectord is stopped for /etc/ha.d/ldirectord.cf
```

mysql-lb1.example.com / mysql-lb2.example.com:

```
ipvsadm -L -n
```

Output on the active load balancer:

```
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
  -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
TCP  10.0.1.10:3306 wrr
  -> 10.0.1.33:3306               Route   1      0          0
  -> 10.0.1.34:3306               Route   1      0          0
```

Output on the hot-standby:

```
IP Virtual Server version 1.2.1 (size=4096)
Prot LocalAddress:Port Scheduler Flags
```

```
 -> RemoteAddress:Port           Forward Weight ActiveConn InActConn
```

mysql-lb1.example.com / mysql-lb2.example.com:

```
/etc/ha.d/resource.d/LVSSyncDaemonSwap master status
```

Output on the active load balancer:

```
master running
(ipvs_syncmaster pid: 1766)
```

Output on the hot-standby:

```
master stopped
(ipvs_syncbackup pid: 1440)
```

If your tests went fine, you can now try to access the MySQL database from a totally different server in the same network (*10.0.1.x*) using the virtual IP address *10.0.1.10*:

```
mysql -h 10.0.1.10 -u ldirector -p
```

(Please note: your MySQL client must at least be of version 4.1; older versions do not work with MySQL 5.)

You can now switch off one of the MySQL cluster nodes for test purposes; you should then still be able to connect to the MySQL database.

8 Annotations and hints

Errors with **ldirectord:**

```
/etc/init.d/ldirectord stop

/etc/init.d/heartbeat start

Stopping ldirectord Error [] reading file /etc/ha.d/ldirectord.cf at line 8: >Unknown command service=mysql
```

This error is due to formatting, it took me some time to figure it out.
The `serivce= mysql` is probably connected to `/etc/services` which is correct.
If you have libmysqlclient15 or 14 installed, fixing right formatting will do the trick like here:

```
vi /etc/heartbeat/ldirectord.cf
```

```
# Global Directives
checktimeout=10
checkinterval=2
autoreload=no
logfile="local0"
quiescent=yes
#after virtual=... make TABS (or double spaces) in every line
virtual = 10.0.1.10:3306
  service = mysql
  real = 10.0.1.33:3306 gate
  real = 10.0.1.34:3306 gate
  checktype = negotiate
  login = "ldirector"
  passwd = "ldirector"
  database = "ldirectordb"
  request = "SELECT * FROM connectioncheck"
  scheduler = wrr
```

And that should be it.

There are some important things to keep in mind when running a MySQL cluster:

1.If you have your databases and you want to use them in MySQL cluster please read **this** as it will guide you through how to change ENGINE=MyISAM to NDBCLUSTER.

2. Adding user to mysql database and changing GRANTS **must be** done on all data nodes since mysql database is MyISAM/InoDB. You can of course convert it's engine.

3.**All data is stored in RAM!** as a default, but Therefore you need lots of RAM on your cluster nodes. The formula how much RAM you need on each node goes like this:

```
(SizeofDatabase * NumberOfReplicas * 1.1 ) / NumberOfDataNodes
```

So if you have a database that is 1 GB of size, you would need 1.1 GB RAM **on each node**!

4.The cluster management node listens on port 1186, and anyone can connect. So that's definitely not secure, and therefore you should run your cluster in an isolated private network! Since it would be harder to manage it may be a good idea to make changes to /etc/hosts.deny or prepare iptables based firewall.

5.It's a good idea to have a look at the MySQL Cluster FAQ: **http://dev.mysql.com/doc/refman/5.1/en/mysql-cluster-faq.html** and also at the MySQL Cluster documentation: **http://dev.mysql.com/doc/refman/5.1/en/ndbcluster.html**

Links

- MySQL: **http://www.mysql.com/**
- MySQL Cluster documentation: **http://dev.mysql.com/doc/refman/5.1/en/ndbcluster.html**
- MySQL Cluster FAQ: **http://dev.mysql.com/doc/refman/5.1/en/mysql-cluster-faq.html**
- Ultra Monkey: **http://www.ultramonkey.org/**
- The High-Availability Linux Project: **http://www.linux-ha.org/**
- MySQL 5.1 Cluster Replication **http://blog.dbadojo.com/2007/08/mysql-51-ndb-cluster-replication-on-ec2.html**

- How to stress test my MySQL Cluster 5.1 **http://blogs.techrepublic.com.com/howdoi/?p=133**