

# **MySQL-HA Installation Guide**

Installation guide for beta-0.6.9

Author: Fernando Ipar

© 2006-2007 Fernando Ipar

Revision: 76

Last update: 01/30/07

## Status

This document is still a draft and might be incomplete in it's coverage. The goal is to have complete and revised document by the time the cluster gets to 1.0

## Introduction

The goal of this document is to help System Administrators setup a highly available MySQL system using the scripts that make up the mysql-ha project.

The audience is expected to have some basic skills in shell usage and MySQL administration.

We'll start off by covering the requisites checklist (what's verified by the pre-install.sh script on the automatic installation) and then move onto the prerequisite installation part.

Finally, we'll describe the mysql-ha.conf file and we'll explain how to start and stop the cluster.

## Prerequisites

The following is a checklist of what you'll need in order to run mysql-ha. If you can make the cluster run without any of these components, please let us know.

- MySQL, with internal replication configured between Master and Slave host. Development is currently tested on version 5 of the engine, but the code includes compatibility for versions 3.X and 4.X.
- The mysql and mysqladmin client binaries
- bash 2 or greater
- syslog
- perl
- gcc and make
- ssh
- smbclient, **only if you want netbios alerts**

Unfortunately, I don't have the minimal versions required for perl, gcc and make. If you find out that your installed version doesn't work, please let us know.

The system is known to work on **Fedora Core 5** and **6** and in **CentOS 4**, with **MySQL 4** and **5** (support for 3.23 is included in the code, but we haven't tested that in years and we won't be testing it anytime soon either).

## Package installation

To begin the installation process, extract the **mysql-ha-<release>.tar.bz2** into a directory of your choice. The **/usr/mysql-ha** directory is the one I've used in my test systems, and the default directory, but it's by no means mandatory. You'll need to set the environment variable **MYSQLHA\_HOME** to point to the place where you untar the files.

## External packages

Next, you'll need to install fping, fake and mysql.monitor, according to these instructions:

### *fping*

Change directory to the place where you untarred the mysql-ha.tar.bz2 file. All the external packages are found in the extern directory. Change directory to fping-2.2b2 and type:

```
#> ./configure && make && make check && make install && make clean
```

This should run with no errors.

## ***fake***

Change directory to extern/ again, and type:

```
#> tar xzvf fake.gz && rm -f fake.gz; make patch && make && make install
```

Once again, this should run with no errors.

## ***mysql.monitor***

Just copy this script from the extern directory to any directory on your **\$PATH**.

# Cluster installation

At this point, you need to populate the /etc/mysql-ha.conf file.

The cluster configuration file is sourced by our shell scripts, so you should leave no blank space between a configuration key (i.e. 'CLUSTER\_IP') and it's value.

We'll review each parameter providing a description and a default value.

<i>Parameter</i>	<i>Description</i>	<i>Default value</i>
CLUSTER_IP	It's the public IP that is shared between master and slave nodes. This is the unique service IP provided to the outside world.	None
CLUSTER_NETMASK	The network mask associated with the previous IP address	None
CLUSTER_BROADCAST	The broadcast address associated with the previous IP address	None
CLUSTER_DEVICE	The device to which to attach CLUSTER_IP	eth0
MYSQL_USER	The MySQL user used for service verification. The mysql.monitor connects to the other node using this user.	replicator
MYSQL_PASSWORD	The password associated with the previous user	replicatorpwd
MYSQL_DATABASE	The database we verify upon testing the	test

<i>Parameter</i>	<i>Description</i>	<i>Default value</i>
	connection	
MASTER_SLEEP_TIME	The amount of seconds the master routine sleeps between rounds	60
SLAVE_SLEEP_TIME	The same, for the slave routine	60
SSH_PATIENCE	Timeout, in seconds, for remote commands executed through ssh	40
MONITOR_PATIENCE	Timeout in seconds for mysql.monitor	20
MONITOR_CHK_THRES HOLD	If mysql.monitor fails, we wait this amount of time and perform a second test before marking the service as down. If no units are specified, the time is interpreted as seconds. Otherwise, you can specify ms (milliseconds) and us (microseconds).	20
MYSQL_KILL_WAIT	Amount of time we wait before checking after running mysql_kill. mysql_kill kills every MySQL process except for the replication thread, in case the server is not responding because it's choked. The units are handled in the same way as MONITOR_CHK_THRESHOLD,	60
MYSQL_RESTART_WAI T	The same, for mysql_restart. mysql_restart attempts to restart mysqld.	60
ATTEMPT_KILL	This boolean flag controls the behavior of the slave routine. If set to 1, it'll attempt to run mysql_kill on the remote host before doing a mysql_restart and finally a takeover. If set to the default, it'll go straight to the mysql_restart.	0
FPING_ATTEMPTS	When we decide the master is gone, we try to fping it to see if the machine is running. We try ATTEMPTS times.	3
SLAVE	IP or host name for the slave node	slave-node
SIG_KILL_WAIT	This is used on the takeover, and it's the amount of time we wait after issuing a SIGTERM on any remaining mysql processes and before issuing a SIGKILL. Keep in mind that the slave is waiting for	5

<i><b>Parameter</b></i>	<i><b>Description</b></i>	<i><b>Default value</b></i>
	us to finish the failover and won't start the service until we do. The units are handled in the same way as MONITOR_CHK_THRESHOLD.	
DB_USER	This is the root user for the MySQL database server	root
DB_PASSWORD	The associated password	rootpwd
NOTIFY_EMAIL	The email address where to send notifications	root@localhost

## ***Fake configuration***

If you installed fake properly, you'll find the **/etc/fake** directory on your system, and inside, the **instance\_config** subdirectory.

There, you must create a file named **CLUSTER\_IP.cfg**, with the following contents:

```
----- BEGIN CLUSTER_IP.cfg ----  
  
SPOOF_IP=CLUSTER_IP  
  
SPOOF_NETMASK=CLUSTER_NETMASK  
  
SPOOF_BROADCAST=CLUSTER_BROADCAST  
  
TARGET_INTERFACE=CLUSTER_DEVICE  
  
----- END CLUSTER_IP.cfg ----
```

You can also create this file by issuing the following commands:

```
#> cd /usr/mysql-ha  
#> export MYSQLHA_HOME=/usr/mysql-ha  
#> ./setup_fake.sh
```

## **Configure passwordless ssh**

In order to work, the scripts need passwordless ssh working for either the root user or some unprivileged user between both nodes. The unprivileged user account is user selectable, but we use mysqlha in our systems and it's a nice name for coherence. You must create a file named 'ssh\_user' under the MYSQLHA\_HOME directory with a single line containing the user name you'll use for passwordless ssh.

In order to setup passwordless ssh between your nodes, create a private/public key pair on each node with the following command<sup>1</sup>:

```
#> ssh-keygen -t dsa
```

---

1 If you're using the mysqlha or another unprivileged account, su – to that account before issuing this commands

### SECURITY CONSIDERATION:

It's up to you to decide if you want to protect the private key with a passphrase or not. The cluster uses the ssh-agent at startup, so if you decide to use a passphrase, you'll need to enter it every time you start the cluster.

Next, copy the generated `/root/.ssh/id_dsa.pub` file to the other node, and add it to the `$HOME/.ssh/authorized_keys` file, where `$HOME` is the cluster user (root, mysqlha or the user you chose) home directory.

Repeat this procedure for the other node.

You can improve security by including `'from=<host>'` before the ssh-dsa string in the `authorized_keys` file.

## Configure sudo

If you decide to setup passwordless ssh using an underprivileged account, **then your system must have sudo installed**, and you must configure this properly.

Create a file in `$MYSQLHA_HOME` called `sudo_prefix` with the text 'sudo ' (**that's WITH A SPACE after the last letter**).

Finally, modify `/etc/sudoers` by adding the following line:

```
mysqlha  ALL=NOPASSWD:/usr/local/sbin/fping, /sbin/fuser, /bin/ps, /bin/kill,  
/etc/init.d/mysqld, /sbin/shutdown, /usr/bin/fake, /sbin/ifconfig
```

Of course, **the full path names for those commands are provided as examples, you should check the full path on your system using 'type'**.

## Configure replication

Set up replication according to the instructions found on MySQL's manual:

<http://www.mysql.com/doc/R/e/Replication.html>

## Configure MySQL access

In order to check the service health, mysql-ha uses the `mysql.monitor` script from the `mon` package (available from <ftp.kernel.org> and its mirrors, please use the mirrors). For this script to work, you'll



need to issue the following sentences in MySQL:

```
GRANT SELECT ON test.* TO '<USER>'@'<HOST>' IDENTIFIED BY  
'<PASSWORD>'
```

where:

'<USER>' is **MYSQL\_USER**.

'<HOST>' is the **CLUSTER\_IP** when you issue this on the slave node, and the slave node's IP when you issue this on the master.

'<PASSWORD>' is **MYSQL\_PASSWORD**.

## Install the rc-script

Copy the rc-script from \$MYSQLHA\_HOME to /etc/init.d and chmod it 755

You can now start the service using this script. We're still working on issues when the script is started with the rc-script and the private key has been protected with a passphrase, so you might want to start the cluster using ./configurator.sh directly from \$MYSQLHA\_HOME while we work it out.

## Run the cluster

- Configure the CLUSTER\_IP on the master node
- Start mysqld on the master node
- Start mysql-ha on the master node, by either:
  - service mysql-ha start (in redhat/fedora)
  - /usr/mysql-ha/configurator.sh (better for testing, since you'll be hooked up to the output)
- Start mysql-ha on the slave node.

Test failover scenarios and report bugs to [mysql-ha-devel@lists.sourceforge.net](mailto:mysql-ha-devel@lists.sourceforge.net)