By Falko Timme
Published: 2006-01-15 00:53

This is a *"copy & paste"* HowTo! The easiest way to follow this tutorial is to use a command line client/SSH client (like ***PuTTY*** for Windows) and simply copy and paste the commands (except where you have to provide own information like IP addresses, hostnames, passwords,...). This helps to avoid typos.

How To Set Up Database Replication In MySQL

Version 1.1
Author: Falko Timme <ft [at] falkotimme [dot] com>
Last edited: 01/14/2006

This tutorial describes how to set up database replication in MySQL. MySQL replication allows you to have an exact copy of a database from a master server on another server (slave), and all updates to the database on the master server are immediately replicated to the database on the slave server so that both databases are in sync. This is not a backup policy because an accidentally issued DELETE command will also be carried out on the slave; but replication can help protect against hardware failures though.

In this tutorial I will show how to replicate the database ***`exampledb`*** from the master with the IP address ***`192.168.0.100`*** to a slave. Both systems (master and slave) are running ***Debian Sarge***; however, the configuration should apply to almost all distributions with little or no modification.

Both systems have MySQL installed, and the database ***`exampledb`*** with tables and data is already existing on the master, but not on the slave.

I want to say first that this is not the only way of setting up such a system. There are many ways of achieving this goal but this is the way I take. I do not issue any guarantee that this will work for you!

1 Configure The Master

First we have to edit ***/etc/mysql/my.cnf***. We have to enable networking for MySQL, and MySQL should listen on all IP addresses, therefore we comment out these lines (if existant):

```
#skip-networking
#bind-address           = 127.0.0.1
```

Furthermore we have to tell MySQL for which database it should write logs (these logs are used by the slave to see what has changed on the master), which log file it should use, and we have to specify that this MySQL server is the master. We want to replicate the database ***exampledb***, so we put the following lines into ***/etc/mysql/my.cnf***:

```
log-bin = /var/log/mysql/mysql-bin.log
binlog-do-db=exampledb
server-id=1
```

Then we restart MySQL:

```
/etc/init.d/mysql restart
```

Then we log into the MySQL database as ***root*** and create a user with replication privileges:

```
mysql -u root -p
Enter password:
```

Now we are on the MySQL shell.

GRANT REPLICATION SLAVE ON *.* TO 'slave_user'@'%' IDENTIFIED BY '<some_password>'; (Replace ***<some_password>*** with a real password!)
FLUSH PRIVILEGES;

Next (still on the MySQL shell) do this:

```
USE exampledb;
  FLUSH TABLES WITH READ LOCK;
SHOW MASTER STATUS;
```

The last command will show something like this:

```
+---------------+----------+--------------+------------------+
| File          | Position | Binlog_do_db | Binlog_ignore_db |
+---------------+----------+--------------+------------------+
| mysql-bin.006 | 183      | exampledb    |                  |
+---------------+----------+--------------+------------------+
1 row in set (0.00 sec)
```

Write down this information, we will need it later on the slave!

Then leave the MySQL shell:

```
quit;
```

There are two possibilities to get the existing tables and data from ***exampledb*** from the master to the slave. The first one is to make a database dump, the second one is to use the ***LOAD DATA FROM MASTER;*** command on the slave. The latter has the disadvantage the the database on the master will be ***locked*** during this operation, so if you have a large database on a high-traffic production system, this is not what you want, and I recommend to follow the first method in this case. However, the latter method is very fast, so I will describe both here.

If you want to follow the first method, then do this:

mysqldump -u root -p<password> --opt exampledb > exampledb.sql (Replace ***<password>*** with the real password for the MySQL user ***root***! ***Important:*** There is ***no*** space between ***-p*** and ***<password>***!)

This will create an SQL dump of ***exampledb*** in the file ***exampledb.sql***. Transfer this file to your slave server!

If you want to go the ***LOAD DATA FROM MASTER;*** way then there is nothing you must do right now.

Finally we have to unlock the tables in ***exampledb***:

```
mysql -u root -p
  Enter password:
  UNLOCK TABLES;
quit;
```

Now the configuration on the master is finished. On to the slave...

2 Configure The Slave

On the slave we first have to create the database ***exampledb***:

```
mysql -u root -p
  Enter password:
  CREATE DATABASE exampledb;
quit;
```

If you have made an SQL dump of ***exampledb*** on the master and have transferred it to the slave, then it is time now to import the SQL dump into our newly created ***exampledb*** on the slave:

mysql -u root -p<password> exampledb < /path/to/exampledb.sql (Replace ***<password>*** with the real password for the MySQL user `root`! ***Important:*** There is ***no*** space between `-p` and ***<password>***!)

If you want to go the ***LOAD DATA FROM MASTER;*** way then there is nothing you must do right now.

Now we have to tell MySQL on the slave that it is the slave, that the master is ***192.168.0.100***, and that the master database to watch is ***exampledb***.

Therefore we add the following lines to ***/etc/mysql/my.cnf***:

```
server-id=2
master-host=192.168.0.100
master-user=slave_user
master-password=secret
master-connect-retry=60
replicate-do-db=exampledb
```

Then we restart MySQL:

```
/etc/init.d/mysql restart
```

If you have not imported the master ***exampledb*** with the help of an SQL dump, but want to go the ***LOAD DATA FROM MASTER;*** way, then it is time for you now to get the data from the master ***exampledb***:

```
mysql -u root -p
  Enter password:
  LOAD DATA FROM MASTER;
quit;
```

If you have ***phpMyAdmin*** installed on the slave you can now check if all tables/data from the master ***exampledb*** is also available on the slave ***exampledb***.

Finally, we must do this:

```
mysql -u root -p
  Enter password:
SLAVE STOP;
```

In the next command (still on the MySQL shell) you have to replace the values appropriately:

CHANGE MASTER TO MASTER_HOST='192.168.0.100', MASTER_USER='slave_user', MASTER_PASSWORD='<some_password>
', MASTER_LOG_FILE='mysql-bin.006', MASTER_LOG_POS=183;

- MASTER_HOST is the IP address or hostname of the master (in this example it is ***192.168.0.100***).
- MASTER_USER is the user we granted replication privileges on the master.
- MASTER_PASSWORD is the password of ***MASTER_USER*** on the master.
- MASTER_LOG_FILE is the file MySQL gave back when you ran ***SHOW MASTER STATUS;*** on the master.
- MASTER_LOG_POS is the position MySQL gave back when you ran ***SHOW MASTER STATUS;*** on the master.

Now all that is left to do is start the slave. Still on the MySQL shell we run

```
START SLAVE;
quit;
```

That's it! Now whenever ***exampledb*** is updated on the master, all changes will be replicated to ***exampledb*** on the slave. Test it!

Links

- MySQL: ***http://www.mysql.com***