



By: Etel Sverdlov  39  40

How To Set Up Master Slave Replication in MySQL

Jul 25, 2012  MySQL

About MySQL replication

MySQL replication is a process that allows you to easily maintain multiple copies of a MySQL data by having them copied automatically from a master to a slave database. This can be helpful for many reasons including facilitating a backup for the data, a way to analyze it without using the main database, or simply as a means to scale out.

This tutorial will cover a very simple example of MySQL replication—one master will send information to a single slave. For the process to work you will need two IP addresses: one of the master server and one of the slave.

This tutorial will use the following IP addresses:

12.34.56.789- Master Database

12.23.34.456- Slave Database

Setup

This article assumes that you have a user with sudo privileges and have MySQL installed. If you do not have MySQL, you can install it with this command:

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

Step One—Configure the Master Database

Open up the mysql configuration file on the master server.

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

Once inside that file, we need to make a few changes.

The first step is to find the section that looks like this, binding the server to the local host:

```
bind-address          = 127.0.0.1
```

Replace the standard IP address with the IP address of server.

```
bind-address          = 12.34.56.789
```

The next configuration change refers to the server-id, located in the [mysqld] section. You can choose any number for this spot (it may just be easier to start with 1), but the number must be unique and cannot match any other server-id in your replication group. I'm going to go ahead and call this one 1.

Make sure this line is uncommented.

```
server-id             = 1
```

Move on to the log_bin line. This is where the real details of the replication are kept. The slave is going to copy all of the changes that are registered in the log. For this step we simply need to uncomment the line that refers to log_bin:

```
log_bin               = /var/log/mysql/mysql-bin.log
```

Finally, we need to designate the database that will be replicated on the slave server. You

can include more than one database by repeating this line for all of the databases you will need.

```
binlog_do_db          = newdatabase
```

After you make all of the changes, go ahead and save and exit out of the configuration file.

Refresh MySQL.

```
sudo service mysql restart
```

The next steps will take place in the MySQL shell, itself.

Open up the MySQL shell.

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

We need to grant privileges to the slave. You can use this line to name your slave and set up their password. The command should be in this format:

```
GRANT REPLICATION SLAVE ON *.* TO 'slave_user'@'%' IDENTIFIED BY 'password'
```

Follow up with:

```
FLUSH PRIVILEGES;
```

The next part is a bit finicky. To accomplish the task you will need to open a *new window or tab* in addition to the one that you are already using a few steps down the line.

In *your current tab* switch to “newdatabase”.

```
USE newdatabase;
```

Following that, lock the database to prevent any new changes:

```
FLUSH TABLES WITH READ LOCK;
```

Then type in:

```
SHOW MASTER STATUS;
```

You will see a table that should look something like this:

```
mysql> SHOW MASTER STATUS;
+-----+-----+-----+-----+
| File           | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+-----+-----+-----+-----+
| mysql-bin.000001 |      107 | newdatabase  |                   |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

This is the position from which the slave database will start replicating. Record these numbers, they will come in useful later.

If you make any new changes in the same window, the database will automatically unlock. For this reason, you should open the new tab or window and continue with the next steps there.

Proceeding the with the database still locked, export your database using mysqldump in the new window (make sure you are typing this command in the bash shell, not in MySQL).

```
mysqldump -u root -p --opt newdatabase > newdatabase.sql
```

Now, returning to your your original window, unlock the databases (making them writeable again). Finish up by exiting the shell.

```
UNLOCK TABLES;
```

```
QUIT;
```

Now you are all done with the configuration of the the master database.

Step Two—Configure the Slave Database

Once you have configured the master database. You can put it aside for a while, and we will now begin to configure the slave database.

Log into your slave server, open up the MySQL shell and create the new database that you will be replicating from the master (then exit):

```
CREATE DATABASE newdatabase;
```

```
EXIT;
```

Import the database that you previously exported from the master database.

```
mysql -u root -p newdatabase < /path/to/newdatabase.sql
```

Now we need to configure the slave configuration in the same way as we did the master:

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

We have to make sure that we have a few things set up in this configuration. The first is the server-id. This number, as mentioned before needs to be unique. Since it is set on the default (still 1), be sure to change it's something different.

```
server-id          = 2
```

Following that, make sure that you have the following three criteria appropriately filled out:

```
relay-log          = /var/log/mysql/mysql-relay-bin.log
```

```
log_bin            = /var/log/mysql/mysql-bin.log
```

```
binlog_do_db      = newdatabase
```

You will need to add in the relay-log line: it is not there by default. Once you have made all of the necessary changes, save and exit out of the slave configuration file.

Restart MySQL once again:

```
sudo service mysql restart
```

The next step is to enable the replication from within the MySQL shell.

Open up the the MySQL shell once again and type in the following details, replacing the values to match your information:

```
CHANGE MASTER TO MASTER_HOST='12.34.56.789',MASTER_USER='slave_user', MASTER
```

This command accomplishes several things at the same time:

1. It designates the current server as the slave of our master server.
2. It provides the server the correct login credentials
3. Last of all, it lets the slave server know where to start replicating from; the master log file and log position come from the numbers we wrote down previously.

With that—you have configured a master and slave server.

Activate the slave server:

```
START SLAVE;
```

You be able to see the details of the slave replication by typing in this command. The \G rearranges the text to make it more readable.

```
SHOW SLAVE STATUS\G
```

If there is an issue in connecting, you can try starting slave with a command to skip over it:

```
SET GLOBAL SQL_SLAVE_SKIP_COUNTER = 1; SLAVE START;
```

All done.

See More

MySQL replication has a lot different options, and this was just a brief overview.

If you have any further questions about the specific capabilities of MySQL, feel free to post your questions in our [Q&A Forum](#) and we'll be happy to answer them.

By Etel Sverdlov

 39

Subscribe

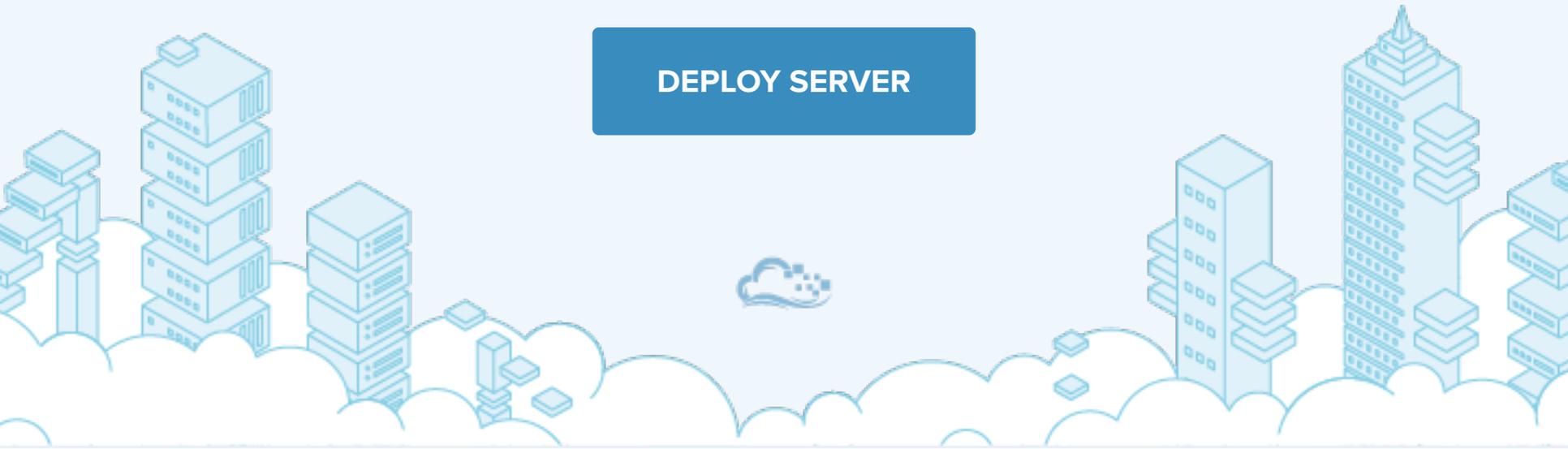


Author:
Etel Sverdlov

Spin up an SSD cloud server in under a minute.

Simple setup. Full root access.
Straightforward pricing.

[DEPLOY SERVER](#)



Related Tutorials

[Understanding SQL And NoSQL Databases And Different Database Models](#)

[How To Migrate a MySQL Database Between Two Servers](#)

[How To Import and Export Databases and Reset a Root Password in MySQL](#)

[How To Create a New User and Grant Permissions in MySQL](#)

[How to Install MemSQL on Ubuntu 14.04](#)

40 Comments

Leave a comment...

Log In to Comment

jason *March 5, 2013*

A few suggestions I would like to make in regards to this article:

- 1) I think it should be mentioned that MySQL has the capabilities to bind to multiple IP addresses and that to take advantage of this feature, an additional "bind-address" line is required.
- 2) In the following section (regarding server-ids) you mentioned that this line has to be "uncommented" but I'm not sure all users will know that means that they have to remove the "#" preceding it if there is one (which there will be in the installed version of my.cnf).
- 3) In the section regarding the databases that they want to replicate, I believe it should be mentioned that leaving this line commented will result in all databases being replicated. In some, though admittedly rather rare situations, this is the desired behavior. This, too, will allow new databases to be replicated automatically without any additionally editing of the configuration files.
- 4) Some of the wording in the explanations is a bit awkward, specifically: "This is the position from which the slave database will start replicating. Record these numbers, they will come in useful later." This might be more grammatically correct if stated as "This is the file name and

position that the master server is using for logging purposes. Make a note of these, as you will need to enter them later".

5) After the mysqldump step, there should be an additional step explaining how to send the newly created .sql file to the remote host. SCP would be the easiest way, in my opinion, to accomplish this.

 4

himanshuchanda *March 16, 2013*

Well, i just followed the steps as mentioned and everything went fine. But the changes in Master DB did not reflect in slave DB

In my case, Master DB & Slave DB resides on two different droplets. Where could i have gone wrong ?



nithin *April 11, 2013*

Nice post, works great.

It would be nice if you could extend this article to include how one would add additional slave servers to the setup, since most people following this would want to scale someday.



manan728 *May 6, 2013*

This is a good link to add yet another slave and from my experience it works great.

<http://www.redips.net/mysql/add-new-slave/>



rutgergeelen *June 18, 2013*

As a backup I would not recommended this approach. If data is corrupted, lost or deleted in the master that will be synced to the slave. Restoring the slave to the master does not get you your data back.

What I miss is the explanation how to use the slave for failover. If somebody could add that that would be helpful (at least for me)

Ruter

 1

kobano *February 4, 2015*

MySQL 5.6 supports delayed replication such that a slave server deliberately lags behind the

master by at least a specified amount of time.



mikeg *July 26, 2013*

@Ruter

Good point. What do you suggest?

Thanks, Mike



hlima *July 30, 2013*

Great article !!

Just one comment from my experience : Use short short passwords to the replica user :) password like bKpGpJIQEm1KHhbEuf6zueTBvfW84ml6XYCcxas2 WON'T work !! half of it works well, took me some hours checking everything before find it.



hbokhoven *August 5, 2013*

Good starter, but I wonder: why are you running bin-logs on the slave?



rajmunees *August 22, 2013*

How often slave updated from master? Is it configurable?



kamaln7 *August 22, 2013*

@Sahaya: I believe it's instant by default. It is configurable:

<http://alexalexander.blogspot.com/2013/03/mysql-slave-delay-how-to.html>



fof *November 14, 2013*

This works really well but I have had a problem where the server-id config variable set in the [mysqld] section of the my.cnf was not being picked up by mysql on restart. No matter what I set it to on the slave it would always be the default which is 0.

When I ran "start-slave" I would get an error message..

"The server is not configured as slave; fix in config file or with CHANGE MASTER TO"

I ended up running the following command to give it a server-id of 2 before I could run start slave.

```
set global server_id=2
```



traio *November 19, 2013*

thanks for the excellent article. however, i could not get it to work, until i changed this on the slave:

```
binlog_do_db = dbnae
```

change to

```
replicate_do_db = dbname
```



matan *December 4, 2013*

Thanks for the great article. What happens when a slave db is modified? Does it update the master which then in-turn updates any other slave dbs?



1

kamaln7 *December 4, 2013*

@matan: The write will fail as it is not supposed to accept writes. You might want to check out master-master replication: <https://www.digitalocean.com/community/articles/how-to-set-up-mysql-master-master-replication>



1

chris51324 *December 26, 2013*

Works GREAT, thanks for the tutorial!



kevin_thulin *February 5, 2014*

Great tutorial, I am going to add a 1GB slave server where I will do my backups.

Will the slave server slow down my prod/master server?



1

kamaln7 *February 8, 2014*

@Kiwot: It shouldn't noticeably affect the master's performance.



m.kamran237 *March 5, 2014*

where do you set the tables from which it will copy from the master and also where do you set where you want to copy that data TO on the slave?



m.kamran237 *March 5, 2014*

so i have database called web1.0 which has a table called contacts. Then i have a different database on another server called web2.0.

i want to replicate everything contacts on web1.0 to web2.0

how to do this pls?



kamaln7 *March 5, 2014*

@m.kamran237: As far as I know you can't do that. It has to be exactly the same, e.g. web1.0 gets replicated to web2.0. I'm not sure if you can choose which tables you want to be replicated or if it just replicates all of the database.



Load More Comments



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.



Copyright © 2015 DigitalOcean™ Inc.

[Community](#) [Tutorials](#) [Questions](#) [Projects](#) [Tags](#) [RSS](#) 

[Terms, Privacy, & Copyright](#) [Security](#) [Report a Bug](#) [Get Paid to Write](#)