How To Set Up MySQL Master-Master Replication

Authored by: ***ASPHostServer Administrator*** *[asphostserver@gmail.com]*
Saved From: http://faq.asphosthelpdesk.com/article.php?id=183

Intro

This second installment of "Scaling Web Applications" will list out the steps necessary for scaling a mysql deployment over two server. The first article in this series laid out the steps needed to load-balance nginx over two server, and it is recommended that you read that article first.

MySQL replication is the process by which a single data set, stored in a MySQL database, will be live-copied to a second server. This configuration, called "master-slave" replication, is a typical setup. Our setup will be better than that, because master-master replication allows data to be copied from either server to the other one. This subtle but important difference allows us to perform mysql read or writes from either server. This configuration adds redundancy and increases efficiency when dealing with accessing the data.
The examples in this article will be based on two server, named Server C and Server D.
Server C: 3.3.3.3
Server D: 4.4.4.4

Step 1 - Install and Configure MySQL on Server C

The first thing we need to do is to install the mysql-server and mysql-client packages on our server. We can do that by typing the following:

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

By default, the mysql process will only accept connections on localhost (127.0.0.1). To change this default behavior and change a few other settings necessary for replication to work properly, we need to edit /etc/mysql/my.cnf on Server C.
There are four lines that we need to change, which are currently set to the following:

```
#server-id              = 1
#log_bin                = /var/log/mysql/mysql-bin.log
#binlog_do_db           = include_database_name
bind-address            = 127.0.0.1
```

The first of those lines is to uniquely identify our particular server, in our replication configuration. We need to uncomment that line, by removing the "#" before it. The second line indicates the file in which changes to any mysql database or table will be logged. The third line indicates which databases we want to replicate between our servers. You can add as many databases to this line as you'd like. The article will use a single database named "example" for the purposes of simplicity. And the last line tells our server to accept connections from the internet (by not listening on 127.0.0.1).

```
server-id               = 1
log_bin                 = /var/log/mysql/mysql-bin.log
binlog_do_db            = example
# bind-address            = 127.0.0.1
```

Now we need to restart mysql:

```
sudo service mysql restart
```

We next need to change some command-line settings within our mysql instance. Back at our shell, we can get to our root mysql user by typing the following:

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

Please note that the password this command will prompt you for is that of the root mysql user, not the root user on our control panel.

To confirm that you are logged in to the mysql shell, the prompt should look like the following.

```
mysql>
```

Once we are logged in, we need to run a few commands.

We need to create a pseudo-user that will be used for replicating data between our two server. The examples in this article will assume that you name this user "replicator". Replace "password" with the password you wish to use for replication.

```
create user 'replicator'@'%' identified by 'password';
```

Next, we need to give this user permissions to replicate our mysql data:

```
grant replication slave on *.* to 'replicator'@'%';
```

Permissions for replication cannot, unfortunately, be given on a per-database basis. Our user will only replicate the database(s) that we instruct it to in our config file.
For the final step of the initial Server C configuration, we need to get some information about the current MySQL instance which we will later provide to Server D.
The following command will output a few pieces of important information, which we will need to make note of:

```
show master status;
```

The output will looking similiar to the following, and will have two pieces of critical information:

```
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000001 |      107 | example      |                  |
+------------------+----------+--------------+------------------+
1 row in set (0.00 sec)
```

We need to make a note of the file and position which will be used in the next step.

Step 2 - Install and Configure MySQL on Server D

We need to repeat the same steps that we followed on Server C.

First we need to install it, which we can do with the following command:

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

Once the two packages are properly installed, we need to configure it in much the same way as we configured Server C. We will start by editing the /etc/mysql/my.cnf file.

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

We need to change the same four lines in the configuration file as we changed earlier.
The defaults are listed below, followed by the changes we need to make.

```
#server-id              = 1
#log_bin                = /var/log/mysql/mysql-bin.log
#binlog_do_db           = include_database_name
bind-address            = 127.0.0.1
```

We need to change these four lines to match the lines below. Please note, that unlike Server C, the server-id for Server D cannot be set to 1.

```
server-id              = 2
log_bin                = /var/log/mysql/mysql-bin.log
binlog_do_db           = example
# bind-address            = 127.0.0.1
```

After you save and quit that file, you need to restart mysql:

```
sudo service mysql restart
```

It is time to go into the mysql shell and set some more configuration options.

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

First, just as on Server C, we are going to create the pseudo-user which will be responsible for the replication. Replace "password" with the password you wish to use.

```
create user 'replicator'@'%' identified by 'password';
```

Next, we need to create the database that we are going to replicate across our server.

```
create database example;
```

And we need to give our newly created 'replication' user permissions to replicate it.

```
grant replication slave on *.* to 'replicator'@'%';
```

The next step involves taking the information that we took a note of earlier and applying it to our mysql instance. This will allow replication to begin. The following should be typed at the mysql shell:

```
slave stop;
CHANGE MASTER TO MASTER_HOST = '3.3.3.3', MASTER_USER = 'replicator',
MASTER_PASSWORD = 'password', MASTER_LOG_FILE = 'mysql-bin.000001',
MASTER_LOG_POS = 107;
slave start;
```

You need to replace 'password' with the password that you have chosen for replication. Your values for MASTER_LOG_FILE and MASTER_LOG_POS may differ than those above. You should copy the values that "SHOW MASTER STATUS" returns on Server C.

The last thing we have to do before we complete the mysql master-master replication is to make note of the master log file and position to use to replicate in the other direction (from Server D to Server C).
We can do that by typing the following:

```
SHOW MASTER STATUS;
```

The output will look similiar to the following:

```
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000004 |      107 | example      |                  |
+------------------+----------+--------------+------------------+
1 row in set (0.00 sec)
```

Take note of the file and position, as we will have to enter those on server C, to complete the two-way replication. The next step will explain how to do that.

Step 3 - Completing Replication on Server C

Back on Server C, we need to finish configuring replication on the command line. Running this command will replicate all data from Server D.

```
slave stop;
CHANGE MASTER TO MASTER_HOST = '4.4.4.4', MASTER_USER = 'replicator',
MASTER_PASSWORD = 'password', MASTER_LOG_FILE = 'mysql-bin.000004',
MASTER_LOG_POS = 107;
slave start;
```

Keep in mind that your values may differ from those above. Please also replace the value of MASTER_PASSWORD with the password you created when setting up the replication user.
The output will look similiar to the following:

```
Query OK, 0 rows affected (0.01 sec)
```

The last thing to do is to test that replication is working on both server. The last step will explain an easy way to test this configuration.

Step 4 - Testing Master-Master Replication

Now that have all the configuration set up, we are going to test it now. To do this, we are going to create a table in our example database on Server C and check on Server D to see if it shows up. Then, we are going to delete it from Server D and make sure it's no longer showing up on Server C.

We now need to create the database that will be replicated between the servers. We can do that by typing the following at the mysql shell:

```
create database example;
```

Once that's done, let's create a dummy table on Server C:

```
create table example.dummy (`id` varchar(10));
```

We now are going to check Server D to see if our table exists.

```
show tables in example;
```

We should see output similiar to the following:

```
+------------------+
| Tables_in_example |
+------------------+
| dummy             |
+------------------+
1 row in set (0.00 sec)
```

The last test to do is to delete our dummy table from Server D. It should also be deleted from Server C. We can do this by entering the following on Server D:

```
DROP TABLE dummy;
```

To confirm this, running the "show tables" command on Server C will show no tables:

```
Empty set (0.00 sec)
```

And there you have it! Working mysql master-master replication.