****

Replication in MySQL

1) Master - Slave

2) Master - Master

****

****

1) Master Slave Replication in MySQL

****

About MySQL replication

Here i configure master-slave for two different scenario
1) For a particular database replication
2) For a all database 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 helpful for many reasons including facilating 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 and one of the slave.

This tutorial will use the following IP addresses:

192.168.1.48 :: Master Database
192.168.1.234 :: Slave
Database database user :: root
Database password :: ubuntu

Setup

This article assumes that you have 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
commend this line

 #bind-address   = 127.0.0.1

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.
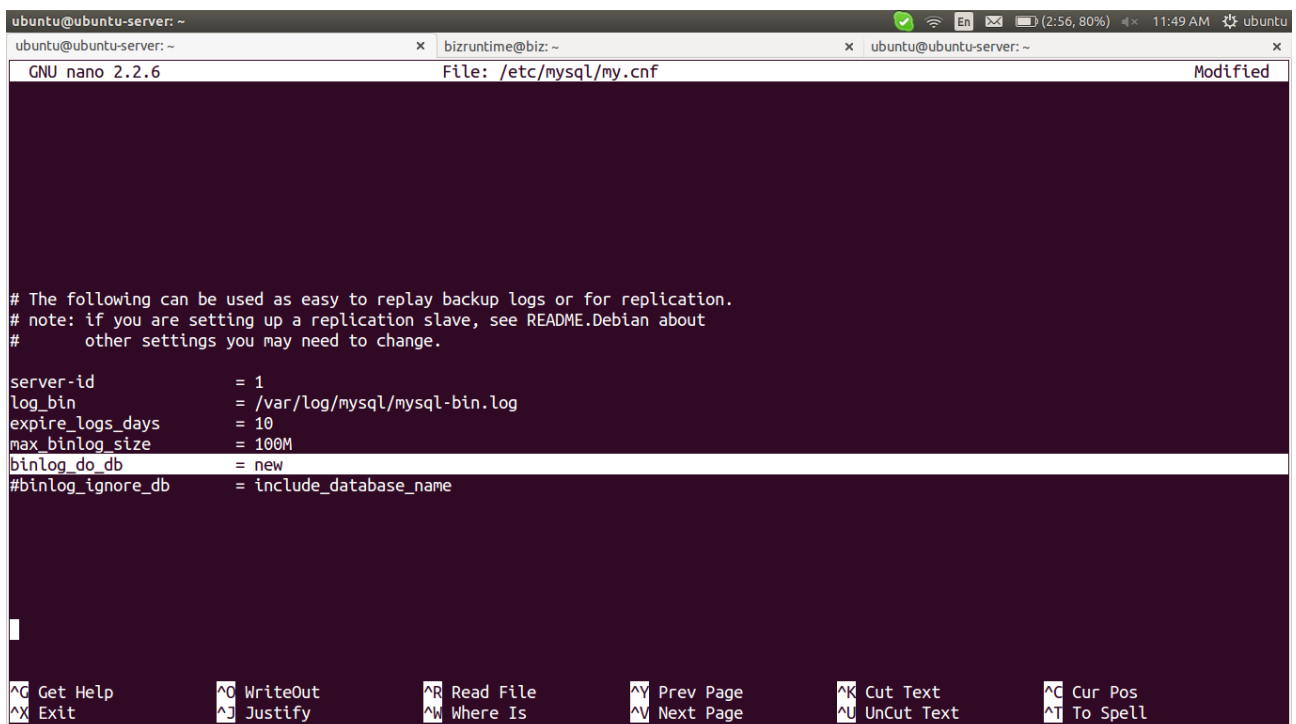
server-id   = 1

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.

For a particular database ::
binlog_do_db   = newdatabase

For all database ::
#binlog_do_db   = newdatabase

```
ubuntu@ubuntu-server: ~                          ×   bizruntime@biz: ~                          ×   ubuntu@ubuntu-server: ~                          ×
 GNU nano 2.2.6                      File: /etc/mysql/my.cnf                                       Modified




# The following can be used as easy to replay backup logs or for replication.
# note: if you are setting up a replication slave, see README.Debian about
#       other settings you may need to change.

server-id               = 1
log_bin                 = /var/log/mysql/mysql-bin.log
expire_logs_days        = 10
max_binlog_size         = 100M
#binlog_do_db            = new
#binlog_ignore_db        = include_database_name




                                      [ Unknown Command ]
^G Get Help      ^O WriteOut      ^R Read File     ^Y Prev Page    ^K Cut Text     ^C Cur Pos
^X Exit          ^J Justify       ^W Where Is      ^V Next Page    ^U UnCut Text   ^T To Spell
```

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;

For particular database ::
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".

Create database 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:

```
+------------------+----------+-------------+------------------+
| 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.

For all databases ::
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.

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

FLUSH TABLES WITH READ LOCK;
Then type in:

SHOW MASTER STATUS;
You'll see something like this:

```
+------------------+----------+-------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+-------------+------------------+
| mysql-bin.000037 |    14462 |             |                  |
+------------------+----------+-------------+------------------+
1 row in set (0.00 sec)
```

My server has been logging for some time, hence the .000037 at the end of the log fie. Make a note of these coordinates. If all goes well you won't need them – but if you run into trouble, these will come in handy.

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.

mysqldump --all-databases --user=root --password --master-data > everything.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):

For particular database ::
CREATE DATABASE newdatabase;

For all databases ::
no need to create databases

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

For particular database ::
mysql -u root -p newdatabase < /path/to/newdatabase.sql

For all databases ::
mysql -u root -p < everything.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 your 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

For particular database ::
binlog_do_db   = newdatabase

For all database ::
#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:

For particular database ::
CHANGE MASTER TO MASTER_HOST='192.168.1.48', MASTER_USER='slave_user', MASTER_PASSWORD='password', MASTER_LOG_FILE='mysql-bin.000001', MASTER_LOG_POS= 107;

For all database ::
change master to MASTER_HOST='192.168.1.48', MASTER_USER='slave_user', MASTER_PASSWORD='password';

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;

****

Testing ::

For particular database ::

create some tables inside that particular database in master and check whether it is replicate on slave side or not

CREATE TABLE testing(ID   INT   NOT NULL);

For all database ::

create some database in master and check whether it is replicate on slave side or not

create database new;

****

links ::

For particular database ::
https://www.digitalocean.com/community/tutorials/how-to-set-up-master-slave-replication-in-mysql

For all databases ::
http://wpguru.co.uk/2013/05/how-to-setup-mysql-masterslave-replication-with-existing-data/

****

****

2) Master-Master Replication in MySQL

****

MySQL Master-Master replication adds speed and redundancy for active websites. With replication, two separate MySQL servers act as a cluster. Database clustering is particularly useful for high availability website configurations. Use two separate Linodes to configure database replication, each with private IPv4 addresses.

Note :: This guide is written for a non-root user. Commands that require elevated privileges are prefixed with sudo. If you're not familiar with the sudo command, you can check our Users and Groups guide.
This guide is written for Debian 7 or Ubuntu 14.04.

Install MySQL

Use the following commands to install MySQL on each of the Linodes:

sudo apt-get update
sudo apt-get upgrade -y
sudo apt-get install mysql-server mysql-client

Edit MySQL's Configuration
Edit the /etc/mysql/my.cnf file on each of the Linodes. Add or modify the following values:

Server 1:

sudo nano /etc/mysql/my.cnf

 # bind-address   = 127.0.0.1

server_id   = 1
log_bin   = /var/log/mysql/mysql-bin.log
log_bin_index   = /var/log/mysql/mysql-bin.log.index
relay_log   = /var/log/mysql/mysql-relay-bin
relay_log_index   = /var/log/mysql/mysql-relay-bin.index
expire_logs_days   = 10
max_binlog_size   = 100M
log_slave_updates   = 1
auto-increment-increment = 2
auto-increment-offset = 1

Server 2:

sudo nano /etc/mysql/my.cnf

 # bind-address   = 127.0.0.1

server_id   = 2
log_bin   = /var/log/mysql/mysql-bin.log
log_bin_index   = /var/log/mysql/mysql-bin.log.index
relay_log   = /var/log/mysql/mysql-relay-bin
relay_log_index   = /var/log/mysql/mysql-relay-bin.index
expire_logs_days   = 10
max_binlog_size   = 100M
log_slave_updates   = 1
auto-increment-increment = 2
auto-increment-offset = 2

Once completed, restart the MySQL application:

sudo service mysql restart

Create Replication Users
Log in to MySQL on each of the Linodes:

Server 1 & Server 2

mysql -u root -p

Configure the replication users on each Linode. Replace x.x.x.x with the private IP address of the opposing Linode, and password with a strong password:

*GRANT REPLICATION SLAVE ON *.* TO 'replication'@'%' IDENTIFIED BY 'password';*

Run the following command to test the configuration. Use the private IP address of the opposing Linode:

mysql -ureplication -p -h x.x.x.x -P 3306

This command should connect you to the remote server's MySQL instance.

Configure Database Replication
While logged into MySQL on *Server 1*, query the master status:

SHOW MASTER STATUS;

Note the file and position values that are displayed:

```
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000001 |      277 |              |                  |
+------------------+----------+--------------+------------------+
1 row in set (0.00 sec)
```

On *Server 2* at the MySQL prompt, set up the slave functionality for that database. Replacex.x.x.x with the private IP from the first server. Also replace the value for master_log_file with the file value from the previous step, and the value for master_log_pos with the position value.

SLAVE STOP;
CHANGE MASTER TO master_host='x.x.x.x', master_port=3306, master_user='replication', master_password='password', master_log_file='mysql-bin.000001', master_log_pos=106;
SLAVE START;

On *Server 2*, query the master status. Again note the file and position values.

SHOW MASTER STATUS;

```
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000001 |      277 |              |                  |
+------------------+----------+--------------+------------------+
1 row in set (0.00 sec)
```

Set the slave database status on *Server 1*, replacing the same values swapped in step 2 with those from the *Server 2*.

SLAVE STOP;
CHANGE MASTER TO master_host='x.x.x.x', master_port=3306, master_user='replication', master_password='password', master_log_file='mysql-bin.000001', master_log_pos=277;
SLAVE START;

****
Test by creating a database and inserting a row:

Server 1:

create database test;
create table test.flowers (`id` varchar(10));

Server 2:

show tables in test;

When queried, you should see the tables from *Server 1* replicated on *Server 2*. Congratulations, you now have a MySQL Master-Master cluster!
****

****

Link ::

https://www.linode.com/docs/databases/mysql/mysql-master-master-replication

error while run command CHANGE MASTER TO master_host .....

http://stackoverflow.com/questions/5029485/error-could-not-initailize-master-info-structure-while-doing-master-slave-rep

****