MySQL Replication for Failover Protection

Table of contents

1 Replication Models

This document describes how to configure MySQL for failover protection based on two replication models:

- Master-Slave (MS)
- Master-Master in Active-Passive mode (MMAP)

1.1 Master-Slave Model

The Master-Slave model is the most basic replication model, where one box acts as a master, generating transaction records which are then replicated over to the slave. This creates a cold standby that can then be used as a backup or as a replacement for the master when it goes down. Manual configuration can also result in swapping around the order of the master and the slave.

- Pros: Robust, well-understood. Allows for backups, simple to configure and manage.
- Cons: Requires explicit manual intervention to switch it over when failure of the master happens.

1.1.1 M-S Runtime Configurations

Configurations Set on Master	
`server_id`	Change from default 1 to a unique id.
`log_bin`	On, set to a filename where all transactions are recorded for transfer to the slave.
`binlog_format`	ROW. The default isolation level for the Hive metastore is repeatable-read, and bin logs are not supported in STATEMENT or MIXED modes with that setting.
`innodb_flush_logs_at_trx_commit`	1 — recommended setting, default with newer MySQL versions.
`innodb_support_xa`	1 — recommended setting, default with newer MySQL versions.

Configurations Set on Slave	
`server_id`	Change from default 1 to a unique id separate from the master.
`read_only`	Turned on — prevents any application from accidentally attempting to write as a result of a misconfiguration.

Configurations Set on Slave	
`relay_log`	On, set to a filename to use.
`log_slave_updates`	Turned on — this is not strictly necessary and is mostly used for slave chains, but makes it much easier to recover and rebuild the original master after a failure; therefore using `log_slave_updates` is recommended.
`log_bin`	On — again not strictly necessary on the slave, but useful to have when rebuilding.
`binlog_format`	ROW. The default isolation level for the Hive metastore is repeatable-read, and bin logs are not supported in STATEMENT or MIXED modes with that setting.
Master setting	Set to the ip/name of the master (done directly in MySQL, not in `my.cnf`).

1.2 M-M-A-P Model

First of all, it should be noted that MySQL does not support what is often considered to be a multi-master model, in that no slave can receive replication records from more than one master. However, it is possible for us to set up two machines which each refer to each other as master, and as long as they each have only one master (each other), we can have multiple "masters". It is also possible to have a Master-chain setup, but that brings up more problems than it solves, so we'll restrict ourselves primarily to the 2-master model in this discussion, and compare it to a typical M-S setup.

While a Master-Master model does support, in theory, updates to both masters, it is recommended to not use it in that fashion, as it can lead to inconsistencies between the two masters in cases of updates involving auto-increment fields or updates that update the same field. Also, there isn't any major benefit to writing across two masters in terms of write scaling, since the update will have to be processed on both computers anyway. For the most part, the main advantage that an M-M model gives over an M-S model is simply the ability to switch over to a secondary master upon failure of the primary master and to continue using the application that uses the database, and eventually, when the original master comes back up, it can recover much more easily than in an M-S situation. Thus, for the purposes of using a MySQL database as a backend for the Hive metastore, it is recommended that the M-M model not be used in what's referred to as Active-Active mode, but rather, that it be used in an Active-Passive mode, where writes happen to only one master at any given time.

1.2.1 M-M-A-P Runtime Configurations

Configurations Set on Master-A	
`server_id`	Change from default 1 to a unique id.
`read_only`	Turned off — this will be our "active" master.
`relay_log`	On, set to a filename to use.
`log_slave_updates`	Turned on.
`log_bin`	On, set to a filename to use.
`binlog_format`	ROW. The default isolation level for the Hive metastore is repeatable-read, and bin logs are not supported in STATEMENT or MIXED modes with that setting.
`innodb_flush_logs_at_trx_commit`	1 — recommended setting, default with newer MySQL versions.
`innodb_support_xa`	1 — recommended setting, default with newer MySQL versions.
Master setting	Set to master-B (done directly in MySQL, not in `my.cnf`).

Configurations Set on Master-B	
`server_id`	Change from default 1 to a unique id separate from the master-A.
`read_only`	Turned on — prevents any application from accidentally attempting to write as a result of a misconfiguration.
`relay_log`	On, set to a filename to use.
`log_slave_updates`	Turned on.
`log_bin`	On, set to a filename to use.
`binlog_format`	ROW. The default isolation level for the Hive metastore is repeatable-read, and bin logs are not supported in STATEMENT or MIXED modes with that setting.
`innodb_flush_logs_at_trx_commit`	1 — recommended setting, default with newer MySQL versions.

Configurations Set on Master-B	
`innodb_support_xa`	1 — recommended setting, default with newer MySQL versions.
Master setting	Set to master-A (done directly in MySQL, not in `my.cnf`).

2 Configuring and Using Replication

This section explains how to configure MySQL replication with either the Master-Slave model (M-S) or the Master-Master model in Active-Passive mode (M-M-A-P), and describes how to recover from failures in each case.

2.1 Disclaimer and Assumptions

- A lot of the following instructions assume that you are using InnoDB for all your MySQL tables. If you are using some other storage engine, other changes might be required and it would be best to verify with a DBA. InnoDB is the recommended storage engine for storing Hive metadata.
- Assume we want to use `server-A.my.example.com` as the primary master and `server-B.my.example.com` as the slave (or secondary passive master in the M-M-A-P case). Let's say we have installed Hive so as to use server-A as its metastore database, as one would when installing with the HDP installer, and we've simply installed MySQL (using yum or otherwise) on server-B, but done nothing else.

2.2 M-S Replication

2.2.1 Setup of Master-Slave Replication

First, on `server-A.my.example.com`:

1. Shut down the mysql server process if it is running.

```
> mysqladmin shutdown
```

2. Edit the `my.cnf` files with the following values:

```
log_bin=mysql-bin
binlog_format=ROW
server_id=10
innodb_flush_logs_at_trx_commit=1
innodb_support_xa=1
```

3. Bring up the mysql server process again — this step may vary based on your installation; in a typical RHEL setup, I can use the system service startup for mysql as follows:

```
> service mysql start
```

4. To verify that the server is now logging bin-logs, you can use the SQL command: "`SHOW MASTER STATUS;`". It should show you a binlog filename and a position.
5. Make sure that your current user is able to do a dump of the MySQL database, by running the following as a mysql-root-capable user, for example, on a default installation, "root".

```
CREATE USER 'root'@'server-A.my.example.com' identified by 'p4ssw0rd';
GRANT ALL ON *.* to 'root'@'server-A.my.example.com';
```

   Also, create a replication user that will be used to conduct future replications:

```
GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'repl'@'server-B.my.example.com'\
  IDENTIFIED BY 'r3plpwd';
```

6. Run `mysqldump` to dump all tables from the master and load them onto the slave as follows:

```
> mysqldump --single-transaction --all-databases --master-data=1 --host=server-
A.my.example.com > dump.out
```

   (You may need to specify `-pp4ssw0rd` to specify the password.)
7. Copy this dump.out file over to the server-B.

Then, on `server-B.my.example.com`:

1. Shut down the mysql server process if it is running.
2. Edit the my.cnf files with the following values:

```
log_bin = mysql-bin
binlog_format = ROW
server_id = 11
relay_log = mysql-relay-bin
log_slave_updates = 1
read_only = 1
```

3. Bring up the mysql server process.
4. Make sure that your current user is able to load the prepared dump of the mysql database, by running the following as a mysql-root-capable user, for eg., on a default installation, "root".

```
create user 'root'@'server-B.my.example.com' identified by 'p4ssw0rd';
grant all on *.* to 'root'@'server-B.my.example.com';
```

5. Load the dump that was dumped out by `mysqldump` by running the following:

```
> mysql --host=server-B.my.example.com -pp4ssw0rd < dump.out
```

6. Verify that the metastore database was transferred over by running a 'SHOW DATABASES' call on MySQL.
7. Look through the MySQL dump, and locate values for MASTER_LOG_FILE and MASTER_LOG_POS. We will need to specify values for these to start replication on the

slave. Assuming these values were 'mysql-bin.000001' and the position was 0, then to copy new entries from the master, run the following:

```
CHANGE MASTER TO MASTER_HOST='server-A.my.example.com', MASTER_USER='repl',\
  MASTER_PASSWORD='r3plpwd', MASTER_LOG_FILE='mysql-bin.000001', MASTER_LOG_POS=0;
```

Note that these values can also be obtained from the master by running 'SHOW MASTER STATUS' on it.

8. Restart the mysql server.
9. Check that the replication is correctly configured by running

```
SHOW SLAVE STATUS;
```

or, for increased readability:

```
SHOW SLAVE STATUS\G
```

It should show correct values as set previously for Master_User and Master_Host. If the slave is caught up to the master, then this field will show a value for Seconds_Behind_Master as being 0.

And with that, you now have M-S replication set up.

2.2.2 What To Do in the Case of a Master Failure

1. Stop all writes on the existing master — for example, shut down the Hive metastore server and stop any Hive processes that connect directly to the database.
2. If the master is still accessible, we can flush the tables to make sure that everything has been written out to logs with the command "FLUSH TABLES WITH READ LOCK" and set the master to read_only.
3. Make sure that replication in the slave is caught up to the transactions recorded by the master by checking 'SHOW SLAVE STATUS'.
4. Run 'STOP SLAVE' on the slave.
5. Remove existing master settings for replication by running:

```
CHANGE MASTER TO MASTER_HOST=''
```

6. Note the new master's binary log coordinates with 'SHOW MASTER STATUS'
7. Configure a new slave to replicate from this new master in a manner similar to how we configured the M-S setup before.
8. Change hive-site.xml to point to the new master.
9. If we intend to bring back the old master in this rotation, it can easily be configured to be a slave of the new master.
10. Turn off the read-only flag on the new master (old slave).

2.3 M-M-A-P Replication

2.3.1 Configuration for Master-Master-Active-Passive

1. Follow the same M-S setup instructions as above, including log_bin, binlog_format and log_slave_updates as recommended. While these settings were not essential for a typical M-S setup, they are needed for an M-M-A-P setup.
2. Verify that both machines are in sync, that one-way replication is working.
3. Configure the server-A's master as server-B, in a manner similar to how we configured server-B's master as server-A. To wit, as follows:

   On server-B, after verifying sync:

   ```
   GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'repl'@'server-A.my.example.com'
    IDENTIFIED BY 'r3plpwd';
   SHOW MASTER STATUS;
   ```

   Let's say, for example, that we see that the replication file is mysql-bin.000004 and the position is 296. Then, we go to server-A, and run the following:

   ```
   CHANGE MASTER TO MASTER_HOST='server-B.my.example.com', MASTER_USER='repl',
    MASTER_PASSWORD='r3plpwd',\
   MASTER_LOG_FILE='mysql-bin.000004',MASTER_LOG_POS=296;
   START SLAVE;
   ```

That's all there is to it — you're set. If you still don't see a slave process running, you might need to restart the slave database as a means of ensuring all settings being flushed and loaded.

2.3.2 Switching Active and Passive Roles in the M-M-A-P Setup

This is a simple matter of doing the following:

1. Shutting down the applications (metastore server) so as to not perform any new writes to the database.
2. Switching the "active" master to read-only.
3. Waiting for the "passive" master to be caught up to replicating all updates from the old "active" master.
4. Switching the passive master's read-only flag to off.
5. Switching config in the applications (metastore server) to point to the old passive/new active master and restarting the application.

That's it — this switches which master is active and which is passive.

2.3.3 Recovering from Failure When the Active Master in an M-M-A-P Fails

- If the old master still has not had any data corruption, it should recover gracefully when its replication from the new master is caught up.
- If there was data loss/data corruption, proceed to set up a database as if it were a fresh slave in the M-S case, and/or the secondary master in the M-M-A-P case, and continue.