BRYAN KENNEDY

JANUARY 24, 2013

Setting up MySQL replication without the downtime

I clearly don't need to expound on the benefits of master-slave replication for your MySQL database. It's simply a good idea; one nicety I looked forward to was the ability to run backups from the slave without impacting the performance of our production database. But the benefits abound.

Most tutorials on master-slave replication use a read lock to accomplish a consistent copy during initial setup. Barbaric! With our users sending thousands of cards and gifts at all hours of the night, I wanted to find a way to accomplish the migration without any downtime.

@pQd via ServerFault suggests enabling bin-logging and taking a non-locking dump with the binlog position included. In effect, you're creating a copy of the db marked with a timestamp, which allows the slave to catch up once you've migrated the data over. This seems like the best way to set up a MySQL slave with no downtime, so I figured I'd document the step-by-step here, in case it proves helpful for others.

First, you'll need to configure the master's */etc/mysql/my.cnf* by adding these lines in the [mysqld] section:

```
server-id=1
binlog-format    = mixed
log-bin=mysql-bin
datadir=/var/lib/mysql
innodb_flush_log_at_trx_commit=1
sync_binlog=1
```

Restart the master mysql server and create a replication user that your slave server will use to connect to the master:

```
CREATE USER replicant@<<slave-server-ip>>;
GRANT REPLICATION SLAVE ON *.* TO replicant@<<slave-server-ip>>
IDENTIFIED BY '<<choose-a-good-password>>';
```

Note: Mysql allows for passwords up to 32 characters for replication users.

Next, create the backup file with the binlog position. It will affect the performance of your database server, but won't lock your tables:

```
mysqldump --skip-lock-tables --single-transaction --flush-logs --hex-
blob --master-data=2 -A  > ~/dump.sql
```

Now, examine the head of the file and jot down the values for *MASTER_LOG_FILE* and *MASTER_LOG_POS*. You will need them later:

```
head dump.sql -n80 | grep "MASTER_LOG_POS"
```

Because this file for me was huge, I gzip'ed it before transferring it to the slave, but that's optional:

```
gzip ~/dump.sql
```

Now we need to transfer the dump file to our slave server (if you didn't gzip first, remove the *.gz* bit):

```
scp ~/dump.sql.gz mysql-user@<<slave-server-ip>>:~/
```

While that's running, you should log into your slave server, and edit your */etc/mysql/my.cnf* file to add the following lines:

```
server-id                = 101
binlog-format          = mixed
log_bin                  = mysql-bin
relay-log                = mysql-relay-bin
```

```
log-slave-updates = 1
read-only             = 1
```

Restart the mysql slave, and then import your dump file:

```
gunzip ~/dump.sql.gz
mysql -u root -p < ~/dump.sql
```

Log into your mysql console on your slave server and run the following commands to set up and start replication:

```
CHANGE MASTER TO MASTER_HOST='<<master-server-
ip>>',MASTER_USER='replicant',MASTER_PASSWORD='<<slave-server-
password>>', MASTER_LOG_FILE='<<value from above>>', MASTER_LOG_POS=
<<value from above>>;
START SLAVE;
```

To check the progress of your slave:

```
SHOW SLAVE STATUS \G
```

If all is well, *Last_Error* will be blank, and *Slave_IO_State* will report “Waiting for master to send event”. Look for *Seconds_Behind_Master* which indicates how far behind it is. It took me a few hours to accomplish all of the above, but the slave caught up in a matter of minutes. YMMV.

And now you have a newly minted mysql slave server without experiencing any downtime!

A parting tip: Sometimes errors occur in replication. For example, if you accidentally change a row of data on your slave. If this happens, fix the data, then run:

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

Update: In following my own post when setting up another slave, I ran into an issue with authentication. The slave status showed an error of 1045 (credential error) even though I was able to directly connect using the replicant credentials. It turns out that MySQL allows passwords up to 32 characters in length for master-slave replication.

Update #2: An astute reader noted that he ran into a “MySQL server has gone away” error while running the initial dump. The solution he found was to add the following during the import on slave:

```
[mysqld]
max_allowed_packet=16M
```

4,747 KUDOS

NOW READ THIS

Stealing is the future of retail

I stole something from the Apple Store today. Or rather, it felt a lot like stealing. I walked in, found what I wanted, opened the “Apple Store” app, scanned the barcode, and walked out. It seemed so much like stealing, I felt a little... Continue →

BRYAN KENNEDY @plusbryan aboutbryan.com