Setup Basic MySQL Master-Slave Replication

by: *Biondi Septian S*

Requirements

- Two MySQL Server running on 2 different servers.
  First Server is called **Source**.
  Second Server is called **Replica**.
  In this guide we will use IP addresses below:
  - Source: **10.225.10.10**
  - Replica: **10.225.10.20**
- Sudoer users on both servers already setup. In this guide we will use **vmuser.**
- Already setup public/private key authentication, so the source can copy data to the replica.
- Firewall between both servers is already setup properly so two servers can communicate in both ways. Protocol to be allowed: SSH (default: **22**) and MySQL (default: **3306**).
- Example database to be replicated is already running on the source. In this guide we will use **example_db**.

Legend

In this guide, to make things clear on where to execute the script, here are the agreements below based on background color:

Execute on Source
Execute on Replica

Configure Database in the Source

Edit MySQL Server configuration file:

```
sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
```

Add below lines:

```
port            = 3306
bind-address    = 10.225.10.10
server-id       = 1
binlog_do_db    = example_db
```

Save file and exit.

Restart MySQL:

```
sudo systemctl restart mysql
```

Create a Replication User

Enter the MySQL Prompt as root:
```
mysql -u root -p
```

Create user **repuser** with password **reppwd123** for replication user:
```
CREATE USER 'repuser'@'10.225.10.20' IDENTIFIED WITH mysql_native_password BY 'reppwd123';

GRANT REPLICATION SLAVE ON *.* TO 'repuser'@'10.225.10.20';

FLUSH PRIVILEGES;
```

Retrieve Binary Log Position from Source DB

Lock the database:
```
FLUSH TABLES WITH READ LOCK;
```

Show the status of Source:
```
SHOW MASTER STATUS;
```
Make note of the **File** and **Position** value.
In this guide, we assume below:

- File: **binlog.000006**
- Position: **855**

Next step is to make the logical backup or dump of the current database.
To ensure the data integrity of the backup, we need to keep this mysql prompt terminal open, therefore the tables are still locked while doing the dump.

Make Logical Backup of the Source DB

Open a new terminal window for source DB.

From home directory, execute the logical backup of **example_db** using mysqldump utility and name the backup file **example_db_for_slave.sql**:
```
sudo mysqldump -u root -p example_db > example_db_for_slave.sql
```

Close the terminal window after the dump finished.

Copy The Backup File to the Replica DB

Back to the previous opened mysql prompt terminal.

We can unlock the tables now, therefore the transaction can roll again:

```
UNLOCK TABLES;
exit;
```

Copy the backup file to the Replica vmuser home directory using scp:

```
scp example_db_for_slave.sql vmuser@10.225.10.20:/home/vmuser/
```

Restore the Dump in the Replica DB

ssh to the Replica:

Enter MySQL prompt terminal as root:

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

Create the database **example_db** for dump restore target:

```
CREATE DATABASE ‘example_db’;
```

```
exit
```

Restore the database:

```
sudo mysql -u root -p example_db < /home/vmuser/example_db_for_slave.sql
```

Configure Database in the Replica

Edit MySQL Server configuration file:

```
sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
```

Add below lines:

```
server-id        = 2
binlog_do_db     = example_db
```

Save file and exit.

Restart MySQL:

```
sudo systemctl restart mysql
```

Start Replication

Enter MySQL prompt terminal as root:

```
mysql -u root -p

CHANGE REPLICATION SOURCE TO
SOURCE_HOST='10.225.10.10',
SOURCE_USER='repuser',
SOURCE_PASSWORD='reppwd123',
SOURCE_LOG_FILE='binlog.000006',
SOURCE_LOG_POS=855;

START REPLICA;

SHOW REPLICA STATUS\G;
```

We should see result similar to below if we succeed:

```
*************************** 1. row ***************************
             Replica_IO_State: Waiting for source to send event
                  Source_Host: 10.225.10.10
                  Source_User: repuser
                  Source_Port: 3306
                Connect_Retry: 60
              Source_Log_File: binlog.000006
          Read_Source_Log_Pos: 855
               Relay_Log_File: mysql-node-b-relay-bin.000002
                Relay_Log_Pos: 323
        Relay_Source_Log_File: binlog.000006
……
……
             Source_Info_File: mysql.slave_master_info
                    SQL_Delay: 0
          SQL_Remaining_Delay: NULL
    Replica_SQL_Running_State: Replica has read all relay log; waiting for more updates
           Source_Retry_Count: 86400
                  Source_Bind:
```

Test Replication

The replication is now running properly. We might want to make sure that any changes we make on the source should be reflected also on the replica.
We can create a table on Source and then insert data into it, and then check the data on the Replica:

```
mysql -u root -p

use example_db;

CREATE TABLE example_tab( example_col VARCHAR(50) );

INSERT INTO example_tab(example_col) VALUES('example_row_1');
INSERT INTO example_tab(example_col) VALUES('example_row_2');
INSERT INTO example_tab(example_col) VALUES('example_row_3');
INSERT INTO example_tab(example_col) VALUES('example_row_4');
INSERT INTO example_tab(example_col) VALUES('example_row_5');

SELECT * FROM example_tab;
+---------------+
| example_col   |
+---------------+
| example_row_1 |
| example_row_2 |
| example_row_3 |
| example_row_4 |
| example_row_5 |
+---------------+
5 rows in set (0.00 sec)

mysql -u root -p

use example_db;

SELECT * FROM example_tab;
+---------------+
| example_col   |
+---------------+
| example_row_1 |
| example_row_2 |
| example_row_3 |
| example_row_4 |
| example_row_5 |
+---------------+
5 rows in set (0.00 sec)
```

The result are both the same.
Therefore we can conclude the replication is running well.