Configure MySQL Master-Master replication

Configure MySQL Master-Master replication

Category: Databases &nbsp

Configuring MySQL in a master-slave replicate model adds complexity to the application utilizing the database. The application has to be programmed to send the write queries only to the master and distribute the read operations between the master and the slave. In the master-master model both the servers in the cluster can handle reads as well as writes. Changes to the database in any one of the servers is replicated to the other.

The following are the steps to configure Master-master MySQL replication. Please replace the IP addresses and other variables as per your environment.

## Step 1: Configure the first server

Edit the `/etc/mysql/my.cnf` file, find the following line

```
bind-address = 127.0.0.1
```

and change it to

```
bind-address = 0.0.0.0
```

This makes MySQL listen on all the ports. Find the following lines and uncomment them (remove the “#” at the beginning).

```
server-id = 1
log_bin = /var/log/mysql/mysql-bin.log
```

Set the `binlog_ignore_db` option to ignore the database named `mysql` during replication. All other databases will be replicated.

```
binlog_ignore_db = mysql
```

Restart the MySQL server with the following comment.

```
service mysqld restart
```

Create a user with replication privileges.

```
root@master01# mysql -u root -p
master01> CREATE USER ‘repl_user’@'<IP of master02>’ IDENTIFIED BY ‘password';
master01> GRANT REPLICATION SLAVE ON *.* TO ‘repl_user’@'<IP of master02>';
```

Replace `password` with a strong one. Make note of master log file and its position.

```
master01> SHOW MASTER STATUS;
+—————————+————-+———————+—————————+
| File | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+—————————+————-+———————+—————————+
| mysql-bin.000003 | 441 | | mysql |
+—————————+————-+———————+—————————+
1 row in set (0.00 sec)
```

## Step 2: Configure the second server

The same steps have to be repeated on this server too. Edit the `/etc/mysql/my.cnf` file, find the following line

```
bind-address = 127.0.0.1
```

and change it to

```
bind-address = 0.0.0.0
```

Uncomment the `server-id` line, change its value to `2` and also set the value of the `binlog_ignore_db` directive.

```
server-id = 2
log_bin = /var/log/mysql/mysql-bin.log
binlog_ignore_db = mysql
```

Restart the MySQL service.

```
service mysqld restart
```

Create a user with replication privileges.

```
root@master02# mysql -u root -p
master02> CREATE USER ‘repl_user’@'<IP of master01>’ IDENTIFIED BY ‘password';
master02> GRANT REPLICATION SLAVE ON *.* TO ‘repl_user’@'<IP of master01>';
```

Make sure to place the IP address of the first server in the above queries.

Note down the values of the master log file and its position.

```
master02> SHOW MASTER STATUS;
+——————+———-+—————+——————+
| File | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+——————+———-+—————+——————+
| mysql-bin.000002 | 441 | | mysql |
+——————+———-+—————+——————+
1 row in set (0.00 sec)
```

## Step 3: Start replication on the first server

Set the master varibles on the first server. Use the `MASTER_LOG_FILE` and `MASTER_LOG_POS` values taken from the **second server**.

```
root@master01# mysql -u root -p
master01> SLAVE STOP;
master01> CHANGE MASTER TO MASTER_HOST = ‘<IP of master02>’, MASTER_USER = ‘repl_user’, MASTER_PASSWORD = ‘password’,
MASTER_LOG_FILE = ‘mysql-bin.000002, MASTER_LOG_POS = 441;
master01> SLAVE START;
```

Check the status of replication.

```
mysql> SHOW SLAVE STATUS\G;
```

## Step 4: Start replication on the second server

Execute the query used in the previous step with the values changed accordingly. Use the `MASTER_LOG_FILE` and `MASTER_LOG_POS` values taken from the **first server**.

```
root@master02# mysql -u root -p
master02> SLAVE STOP;
master02> CHANGE MASTER TO MASTER_HOST = ‘<IP of master01>’, MASTER_USER = ‘repl_user’, MASTER_PASSWORD = ‘password’,
MASTER_LOG_FILE = ‘mysql-bin.000003, MASTER_LOG_POS = 441;
master02> SLAVE START;
```

Check the status of replication.

```
master02> SHOW SLAVE STATUS\G;
```

## Step 5: Secure the MySQL ports

Since MySQL is listening on all interfaces it becomes a potential target to attacker. In this step we will configure the firewall to allow MySQL connections only between these two servers.

The following IPTables firewall rule only permits MySQL traffic from the trusted server.

```
root@mysql01# iptables -I INPUT -m state –state NEW -p tcp –dport 3306 -s <IP of master02> -j ACCEPT
```

On the second server replace the IP with that of the first.

```
root@mysql02# iptables -I INPUT -m state –state NEW -p tcp –dport 3306 -s <IP of master02> -j ACCEPT
```

Related Articles

- MySQL Replication
- How to increase /tmp partition size
- Getting Started with Iptables

Save this article