MySQL Master Slave Replication: 7 Easy Steps:

MySQL replication is a process that allows data to be copied/replicated from one server to the other at the same time. It is mainly done to increase the availability of data. One of the main reasons that people go for MySQL master-slave replication is for data recovery. In the case of any catastrophe or a hardware failure, MySQL replication ensures that an accurate backup exists all the time.

In this article, you will look in detail at the MySQL master-slave replication process along with a step-by-step guide about how to achieve replication.

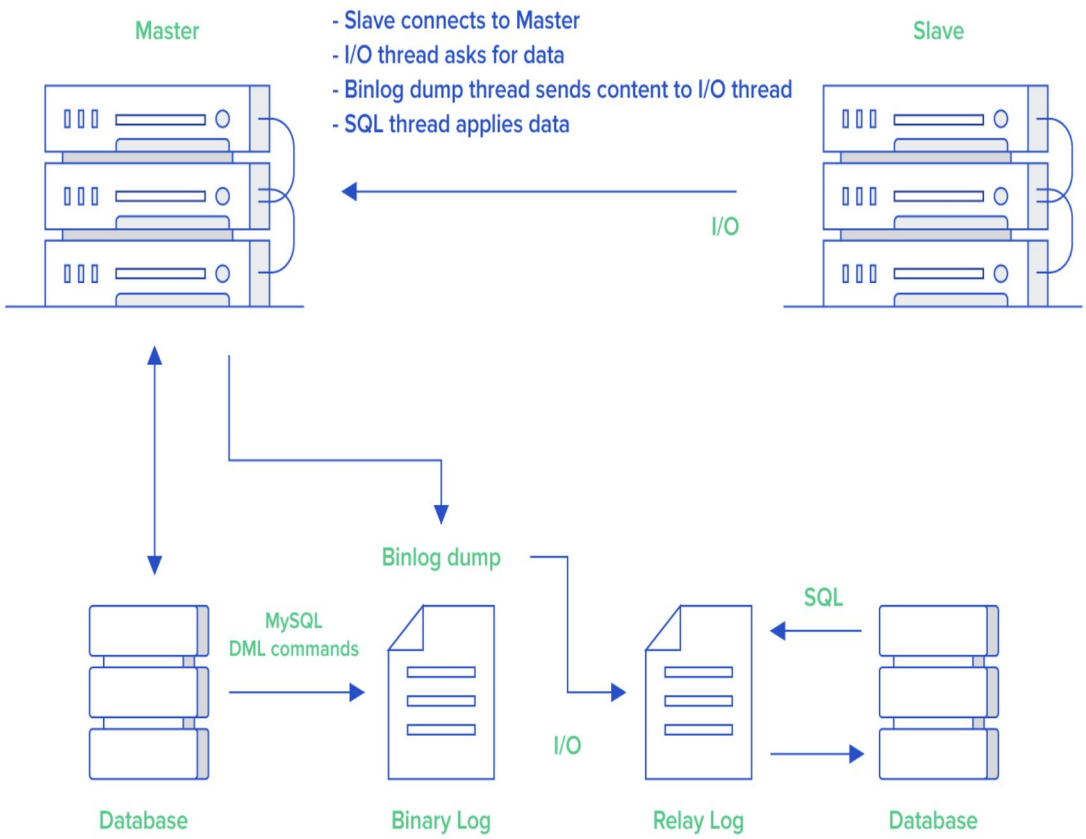
What Is Master-Slave Replication?

The master-slave replication process enables database administrators to replicate or copy data stored in more than one server simultaneously. This helps the database administrator to create a live backup of the database all the time. During some situations, when the master-slave is down to any issues, they can instantly switch over the slave database and keep the application up and running. The replication process ensures that your application doesn't face any kind of downtime at all.

In this replication, there are various types of replication processes. You can have a single master and multiple slaves or multiple masters and multiple slaves, etc.

In this process, it is always a single or one-way transmission of data. The data is stored in the master first and then copied on to the slaves. Hence, the write operation

is performed only on the master database. The read operation is done in both the master and the slave. The slaves can also be used for data accessibility to reduce the load on the master database.



Purpose Of Master-Slave Replication

One of the main purposes of going for a master-slave replication system is to have a standby system with a live backup that can be promoted as the master when the original master server crashes. Apart from this, there are several benefits as outlined below:

•**Scalability**: All query requests to the database can be routed to multiple database servers to reduce the load on the server and allow faster accessibility. Most of the web applications and sites that you encounter nowadays come loaded with more read operations than write operations to the database. Hence, website administrators need to provide the perfect setup for the quick loading of details on the website.

•Performance: All database write operations are done on the master database. Once these changes are made to the master database, they get updated from the master to the slave. But the read requests from websites can be shared across multiple slaves to increase the performance of the website.

•Backup: You can just replicate the latest snapshot of the database to another database and create a backup in just a couple of minutes. Data corruption is largely reduced as the master server runs without any issues and provides 99.9% Uptime. This allows applications to process large amounts of reading or write operations without any hassle.

•**Analytics And Bench-Marking**: This process allows database analysts to run all kinds of data analysis tests and experiments on the slaves without disturbing the master.

Prerequisites

To set up a MySQL master-slave replication, you need to have the following:

1. Two Servers with root access and MySQL should be installed on both.

2. Working Internet for connectivity and enough disk space on both the servers to replicate data.

To start setting up the MySQL master-slave replication, please follow the step-by-step guide provided below:

1. **Setting Up The Master**
2. **Create A New User For Slave**
3. **Move Data From Master To Slave**
4. **Configure Slave Server**
5. **Import Data Dump**
6. **Start Slave Server**
7. **Test MySQL Master-Slave Replication**

STEP:1 Setting Up The Master

login to master server and make changes in configuration(my.cnf) file.

ssh user@master_server_ip

root@repl-master:~# sudo vi /etc/mysql/mysql.conf.d/mysqld.cnf

bind-address = 127.0.0.1 or <Master IP>

Uncomment below lines:-

server-id = 1

log_bin = /var/log/mysql/mysql-bin.log

Restart MySQL service.

STEP:2 Create A New User For Slave

The next step is to create a new user for your slave server. Use the following command to create it:

root@repl-master:~# mysql -uroot -p;

mysql> CREATE USER 'slave'@'slave_IP' IDENTIFIED BY 'SLAVE_PASSWORD';

mysql> GRANT REPLICATION SLAVE ON *.* TO 'slave'@'slave_IP';

mysql> FLUSH PRIVILEGES;

STEP:3 Copy Data From Master To Slave

You can start moving the data from the master to the slave.

You need to create a MySQL dump file to move the data.

Use the following command to create the dump file:

root@repl-master:~# mysqldump -u root -p –all-databases –master-data > data.sql

or

root@repl-master:~# mysqldump -u root -p --master-data=2 --databases db1 db2 db3 > data.sql

To copy the dump file to the slave, use the following command:

scp data.sql root@slave_IP:/tmp

STEP:4 Configure Slave Server

Now, all you need to do is configure the slave server and test if replication is working. Ensure MySQL is installed.

Login and Open the **configuration file** in your slave server and update these lines:

ssh sammy@source_server_ip

root@repl-slave:~# sudo vi /etc/mysql/mysql.conf.d/mysqld.cnf

In the same way that you did for the master server,

you need to bind the IP address and uncomment those two lines for the slave server.

bind-address = 0.0.0.0 or <Slave_IP>

Uncomment below lines:-

server-id = 2

log_bin = /var/log/mysql/mysql-bin.log

relay-log = /var/log/mysql/mysql-relay-bin.log

Now, restart the MySQL server using the following command:

root@repl-slave:~# sudo service mysql restart

STEP:5 Import Data Dump

Use the following command to import the dump file to the slave server:

root@repl-slave:~# mysql -uroot -p < /tmp/data.sql

Once the data is imported, you need to stop MySQL in the slave server using the following command:

root@repl-slave:~# mysql -uroot -p;

mysql> STOP SLAVE;

Get the binary log file and position to start replication from the point from master and output will be:

root@repl-slave:~# head -n23 /tmp/data.sql | grep -i change

--- CHANGE MASTER TO MASTER_LOG_FILE='1.000082', MASTER_LOG_POS=1061787221;

Give those coordinates in below command.

mysql> CHANGE REPLICATION SOURCE TO

SOURCE_HOST='source_server_ip',

SOURCE_USER='replica_user',

SOURCE_PASSWORD='password',

SOURCE_LOG_FILE='mysql-bin.000001',

SOURCE_LOG_POS=899;

You have finally imported the dump files and updated the master IP address, password, log file name, and position, to enable the master to communicate with the slave without any issues.

STEP:6 Start Slave Server

Next, use the “Start Slave” command to start operating the slave server.

root@repl-slave:~#START SLAVE;

STEP:7 Test MySQL Master Slave Replication

To test if your MySQL master slave replication works, just create a database in your master server and see if it is replicated in the slave server. If you can see the database in the slave, then it is working fine.

Create a test database in a master server called 'sampledb'.

mysql>CREATE DATABASE sampledb;

Now login to your slave server and list the databases, and if you see the “sampledb” there, then the master slave replication process is working fine.

Login to your slave server and use the following command to list all databases:

mysql>show databases;

If any issue comes and slave gets broken then run below command:

stop slave; set global sql_slave_skip_counter = 1; start slave;

references:

https://hevodata.com/learn/mysql-master-slave-replication/

https://www.digitalocean.com/community/tutorials/how-to-set-up-replication-in-mysql