#### **MASTER – SLAVE REPLICATION**

Setting Up Master-Slave Replication in MySQL

Master-Slave replication allows one Master (writes) to replicate data to one or more Slaves (readonly).

**★ Steps to Set Up Master-Slave Replication in MySQL** 

Let's assume:

• Master Server (Primary): 192.168.1.100

• Slave Server (Replica): 192.168.1.200

Database to replicate: do\_db

Step 1: Configure MySQL on the Master

Edit MySQL Config (my.cnf or my.ini)

On Master (192.168.1.100), update:

ini

[mysqld]

server-id=1

log-bin=mysql-bin

binlog-do-db=do\_db # Replicate only this DB

**Z**Restart MySQL

Restart MySQL to apply changes:

bash

systemctl restart mysql # Linux

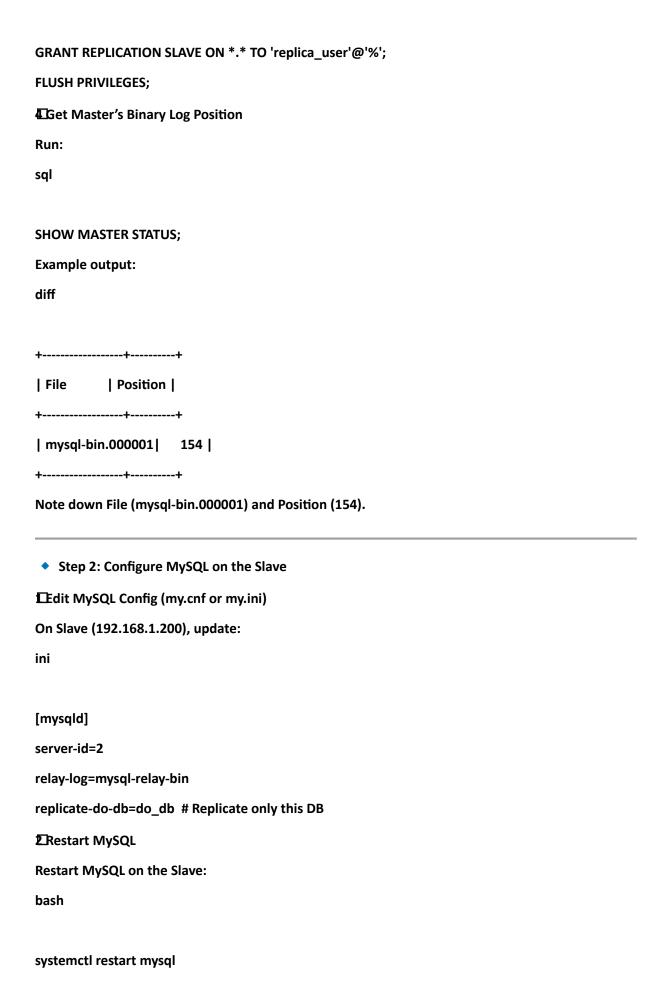
net stop mysql && net start mysql # Windows

**E**Create a Replication User

Run on the Master:

sql

CREATE USER 'replica\_user'@'%' IDENTIFIED WITH mysql\_native\_password BY 'password';



```
EConnect Slave to Master
Run on Slave (192.168.1.200):
sql
CHANGE MASTER TO
MASTER_HOST='192.168.1.100',
MASTER_USER='replica_user',
MASTER_PASSWORD='password',
MASTER_LOG_FILE='mysql-bin.000001',
MASTER_LOG_POS=154;
4□Start Replication
sql
START SLAVE;
5Verify Replication
Check status:
sql
SHOW SLAVE STATUS\G
Ensure:
   • Slave_IO_Running: Yes
   • Slave_SQL_Running: Yes
   • No errors in Last_Error.
★ Important Considerations
 Replication Issues?
      Restart replication:
sql
STOP SLAVE;
START SLAVE;
```

• If errors persist, reset replication: sql **STOP SLAVE; RESET SLAVE;** CHANGE MASTER TO ...; # Reconfigure with correct log position **START SLAVE**; Backup & Recovery • Take a backup before setup: bash mysqldump -u root -p --all-databases > backup.sql Restore backup on Slave if needed: bash mysql -u root -p < backup.sql Summary 1. Master: Enable binary logging, create a replication user, get log position. 2. Slave: Configure MySQL, connect to master, start replication.

3. Verify using SHOW SLAVE STATUS\G.

## **MASTER - MASTER REPLICATION**

Setting Up Master-Master (Bidirectional) Replication in MySQL

Master-Master replication (also called active-active replication) allows both servers to act as master and replicate changes to each other. This ensures high availability and load balancing.

**★** Steps to Set Up Master-Master Replication in MySQL

```
Let's assume:
```

• Server 1 (Master A): 192.168.1.100

• Server 2 (Master B): 192.168.1.200

• Database to replicate: do\_db

- Step 1: Configure MySQL on Master A
  - 1. Edit MySQL config file (my.cnf or my.ini) on Server 1 (192.168.1.100):

ini

[mysqld]

server-id=1

log-bin=mysql-bin

binlog-do-db=do\_db # Replicate only this DB

auto-increment-increment=2

auto-increment-offset=1

#### Why auto-increment settings?

It ensures that primary key values generated on each master do not conflict.

2. Restart MySQL:

bash

systemctl restart mysql # Linux

net stop mysql && net start mysql # Windows

3. Create a replication user:

sql

CREATE USER 'replica\_user'@'%' IDENTIFIED WITH mysql\_native\_password BY 'password';

GRANT REPLICATION SLAVE ON \*.\* TO 'replica\_user'@'%';

FLUSH PRIVILEGES;

4. Check Master A's binary log position:

sql

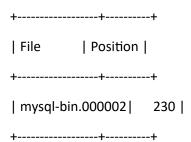
SHOW MASTER STATUS;

Example output:
diff
++
File   Position
++
mysql-bin.000001  154
<del>+</del>
Note down <b>File</b> and <b>Position</b> values.
Step 2: Configure MySQL on Master B
1. Edit my.cnf or my.ini on Server 2 (192.168.1.200):
ini
[mysqld]
server-id=2
log-bin=mysql-bin
binlog-do-db=do_db
auto-increment-increment=2
auto-increment-offset=2
2. Restart MySQL:
bash
systemctl restart mysql
3. Create the same replication user:
sql
CREATE USER 'replica_user'@'%' IDENTIFIED WITH mysql_native_password BY 'password';
GRANT REPLICATION SLAVE ON *.* TO 'replica_user'@'%';
FLUSH PRIVILEGES;
4. Check Master B's binary log position:

SHOW MASTER STATUS;

Example output:

diff



Note down File and Position values.

## Step 3: Set Up Replication on Both Masters

Set Master B as the Slave of Master A

On **Master B (192.168.1.200)**, run:

sql

**CHANGE MASTER TO** 

MASTER\_HOST='192.168.1.100',

MASTER\_USER='replica\_user',

MASTER\_PASSWORD='password',

MASTER\_LOG\_FILE='mysql-bin.000001',

MASTER\_LOG\_POS=154;

START SLAVE;

Set Master A as the Slave of Master B

On Master A (192.168.1.100), run:

sql

**CHANGE MASTER TO** 

MASTER\_HOST='192.168.1.200',

```
MASTER_USER='replica_user',

MASTER_PASSWORD='password',

MASTER_LOG_FILE='mysql-bin.000002',

MASTER_LOG_POS=230;

START SLAVE;
```

## Step 4: Verify Master-Master Replication

On both servers, check replication status:

sql

#### SHOW SLAVE STATUS\G

#### Ensure:

- Slave\_IO\_Running: Yes
- Slave\_SQL\_Running: Yes
- No errors in Last\_Error.

## Important Considerations

### Conflict Avoidance

- Use auto-increment-increment=2 and auto-increment-offset to prevent primary key conflicts.
- Ensure applications do not update the same row on both servers simultaneously.

## Handling Failures

• If a master crashes, **STOP SLAVE** on the surviving master:

sql

## STOP SLAVE;

• Restart the crashed server and re-sync.

# Summary

- 1. Configure Master A and Master B with server-id, binary logs, and auto-increment.
- 2. Create a replication user on both servers.

- 3. Get binary log positions using SHOW MASTER STATUS;.
- 4. Set up replication between Master A  $\rightarrow$  Master B and Master B  $\rightarrow$  Master A.
- 5. **Verify replication** using SHOW SLAVE STATUS\G.