By Directional Replication/Master-Master Replication in MySQL

Master-Master replication, also known as bidirectional replication, involves two or more MySQL servers, where each server acts as both a master and a slave. This setup allows both servers to accept writes and replicate data changes to each other, providing redundancy and high availability.

Key Features
Bidirectional Data Flow: Changes made on either server are replicated to the other, ensuring that both servers have the same data.

High Availability: If one server fails, the other can continue to handle both read and write requests.
Load Balancing: Read and write operations can be distributed across both servers to balance the load.
Considerations

Conflict Resolution: Write conflicts can occur if the same data is modified simultaneously on both servers. Strategies such as using different auto-increment offsets or conflict detection and resolution mechanisms can mitigate this.

Network Latency: Since changes must be replicated between servers, network latency can affect performance.

Data Consistency: Ensuring data consistency and integrity requires careful planning and monitoring.
Setting Up Master-Master Replication
Here's a step-by-step guide to setting up Master-Master replication in MySQL:

Step 1: Configure Both Servers
Ensure both MySQL servers are installed and running. In this example, we'll call the servers master1 and master2.

Step 2: Configure master1

1. **Edit MySQL Configuration File (my.cnf or my.ini):**

-- MYSQL--

```
[mysqld]
server-id=1
log_bin=mysql-bin
binlog-do-db=your_database_name
```

2. **.Restart MySQL:**

```
sudo systemctl restart mysql
```

3. **Create a Replication User:**

```
Mysql > CREATE USER 'replica'@'%' IDENTIFIED BY 'password';
Mysql > GRANT REPLICATION SLAVE ON *.* TO 'replica'@'%';
Mysql > FLUSH PRIVILEGES;
```

4. **Get the Binary Log File Position:**

```
Mysql > FLUSH TABLES WITH READ LOCK;
Mysql > SHOW MASTER STATUS;
```

Note down the File and Position values.

Step 3: Configure master2

1 .Edit MySQL Configuration File (my.cnf or my.ini):

```
[mysqld]
server-id=2
log_bin=mysql-bin
binlog-do-db=your_database_name
```

2. Restart MySQL:

```
sudo systemctl restart mysql
```

3. Create a Replication User:

```
Mysql > CREATE USER 'replica'@'%' IDENTIFIED BY 'password';
Mysql > GRANT REPLICATION SLAVE ON *.* TO 'replica'@'%';
Mysql > FLUSH PRIVILEGES;
```

4. Get the Binary Log File Position:

```
Mysql > FLUSH TABLES WITH READ LOCK;
Mysql > SHOW MASTER STATUS;
```

Note down the File and Position values.
Step 4: Configure Replication on Both Servers

On master1, Set master2 as the Slave:

```
Mysql > CHANGE MASTER TO
      MASTER_HOST='master2_ip_address',
      MASTER_USER='replica',
      MASTER_PASSWORD='password',
      MASTER_LOG_FILE='master2_binlog_file',
      MASTER_LOG_POS=master2_binlog_position;
Mysql > START SLAVE;
```

On master2, Set master1 as the Slave:

```
Mysql > CHANGE MASTER TO
      MASTER_HOST='master1_ip_address',
      MASTER_USER='replica',
      MASTER_PASSWORD='password',
      MASTER_LOG_FILE='master1_binlog_file',
      MASTER_LOG_POS=master1_binlog_position;
Mysql > START SLAVE;
```

Step 5: Verify Replication

1. **Check Slave Status on master1:**

Mysql > SHOW SLAVE STATUS\G;

Ensure Slave_IO_Running and Slave_SQL_Running are both Yes.

2. Check Slave Status on master2:
Mysql > SHOW SLAVE STATUS\G;

Ensure Slave_IO_Running and Slave_SQL_Running are both Yes.

Conflict Handling
To avoid auto-increment key conflicts, you can configure the auto_increment_increment and auto_increment_offset variables.

1. On master1:

[mysqld]
auto_increment_increment = 2
auto_increment_offset = 1

2. On master2:

[mysqld]
auto_increment_increment = 2
auto_increment_offset = 2

Maintenance and Monitoring

a) Regular Backups: Regularly backup your data to prevent data loss.
b) Monitoring Tools: Use tools like mysqlslap or third-party solutions for monitoring and alerting.
c) Logs: Monitor MySQL error logs and replication status for any issues.

Ajit Yadav (DBA)