MySQL Replication

MySQL replication is a process that allows data from one MySQL database server (the primary or source server) to be copied automatically to one or more MySQL database servers (replica or slave servers). Replication is used to improve data availability, load balancing, and for backup and failover purposes.

Types of MySQL Replication

Asynchronous Replication:

- Description**:** The primary server sends events to the replica, but does not wait for the replica to acknowledge receipt.
- Use Cases: High performance, low latency applications where some delay in data consistency is acceptable.

Semi-Synchronous Replication:

- Description: The primary server waits for at least one replica to acknowledge that it has received the events, but not necessarily that it has applied them.
- Use Cases: Balances performance with increased data safety compared to asynchronous replication.

Synchronous Replication (Group Replication):

- Description: All replicas must acknowledge receipt and application of events before the primary server considers the transaction complete.
- Use Cases: High availability and data consistency scenarios where strong data guarantees are required.

==========

Replication Setups

> See here for more info, and the ANS KB for setup. The below example is from my own solution.

Master - Master

In a Master-Master replication setup, two MySQL servers are configured to replicate data to each other. Both servers can accept write operations, and changes are propagated bidirectionally.

Both servers act as master and slave to each other, allowing writes on both nodes.

If one server fails, the other can continue to serve both read and write requests.

Data is duplicated on both servers, providing a backup in case one fails.

Advantages:

1. Increased Availability:
   - Both servers can handle writes, providing high availability.
2. Load Balancing:
   - Distribute read and write operations across both servers.
3. Redundancy:
   - Data is replicated on both servers, providing a failover option.

Disadvantages:

1. Complexity:
   - More complex to set up and maintain compared to Master-Slave.
2. Conflict Resolution:
   - Potential for conflicts if the same data is modified on both servers simultaneously.
3. Data Consistency:
   - Possible data inconsistency issues if replication lag occurs.

Example Configuration

Master - Slave (asynchronous) configuration.

Configure Server A (master)

1. Set the `my.cnf` file for Server A

```
[mysqld]
server_id = 1
log_bin = /var/log/mysql/mysql-bin.log
```

2. Create a user for replication on Server A

```
CREATE USER 'replication'@'%' IDENTIFIED BY 'password';
GRANT REPLICATION SLAVE ON *.* TO 'repl'@'%';
FLUSH PRIVILEGES;
```

> As you can see here `CREATE USER 'replication'@'%' IDENTIFIED BY 'password';`, I've set the host as wildcard (%), You'll ideally want to create 2 users here, one with the internal IP of Server A, and one with the internal IP of server B (internal IF on same network). For example:
>
> ```
> CREATE USER 'replication'@'server_a_ip' IDENTIFIED BY 'password';
> CREATE USER 'replication'@'server_b_ip' IDENTIFIED BY 'password';
> ```

3. Lock all tables to get MySQL into a consistent state on server A

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

Note down the `File` and `Position` values. You will use these in Server A's configuration.

4. Create a database dump

For replication to work, we'll first need to import a dump of the locked databases from the master onto the slave.

```
mysqldump -u root -p --all-databases --master-data > dbdump.sql
```

--master-data	**Adds `CHANGE MASTER TO` Statement:** At the beginning of the dump file (`dbdump.sql` in your case), `--master-data` adds a comment containing the `CHANGE MASTER TO` statement. This statement includes the following information: • `MASTER_LOG_FILE`: The name of the binary log file that was being written to when the dump was initiated. • `MASTER_LOG_POS`: The position within that binary log file where the dump started. • This information helps in setting up a slave server to start replication from the exact point where the dump was taken, ensuring consistency between the master and the slave.

Configure Server B (Slave)

1. Set the my.cnf file for Server B

```
[mysqld]
server_id = 2
log_bin = /var/log/mysql/mysql-bin.log
```

2. Import database dump from Server A

```
mysql -u root -p < dbdump.sql
```

3. Configure replication on Server B using the below MySQL command:

```
CHANGE MASTER TO
    MASTER_HOST='server_A_ip',
    MASTER_USER='repl',
    MASTER_PASSWORD='password',
    MASTER_LOG_FILE='mysql-bin.000001',  -- use the File value from Server A
    MASTER_LOG_POS=120;                  -- use the Position value from Server A
START SLAVE;
```

3. Verify Replication Status:

Ensure that both servers are replication to each other without errors. You should see `Slave_IO_Running` and `Slave_SQL_Running` set to `Yes` on both servers.

```
SHOW SLAVE STATUS\G;
```

- Ensure `Slave_IO_Running` and `Slave_SQL_Running` are both set to `Yes`.
- Check for errors in the `Last_IO_Error` and `Last_SQL_Error` fields.

4. Unlock the tables on both servers

```
UNLOCK TABLES;
```

> Ensure to test once set up.

Master - Slave

In a Master-Slave replication setup, one MySQL server (the master) is responsible for all write operations, and one or more slave servers replicate the data from the master. Slave servers are typically read-only.

Data flows from the master to the slave(s) only.

Slaves can handle read operations, offloading the master.

Advantages:

1. **Simplicity:**
   - Easier to set up and maintain compared to Master-Master replication.
2. **Read Scalability:**
   - Offload read operations to slaves, improving performance.
3. **Data Safety:**
   - Slaves can be used for backups and disaster recovery without affecting the master.

Disadvantages:

1. **Single Point of Failure:**
   - The master is a single point of failure for write operations.
2. **Replication Lag:**
   - Potential for replication lag, which can lead to stale reads on slaves.
3. **Limited Write Scalability:**
   - All writes are handled by the master, which can become a bottleneck.

==========

Revision #7
Created 2024-07-06 02:09:27 UTC by Daniel
Updated 2024-07-07 15:14:19 UTC by Daniel