RDBMS - Week 5

Assignment Report

Siddharth Kumar

September 3, 2025

Contents

Chapter 1

Mariadb Master / Slave Replication

Discussion:
Row-Based Replication (RBR) logs actual row changes, making it the most reliable for complex statements and deterministic results, though it produces larger binlogs. Statement-Based Replication (SBR) logs SQL statements, which keeps logs small but risks errors with non-deterministic functions. Mixed-Based Replication (MBR) uses SBR by default and switches to RBR when needed. In practice, RBR is generally recommended for consistency and safety.

1.1 Master Node setup

```
1  #/etc/mysql/mariadb.conf.d/50-server.cnf
2  server-id=1
3  log_bin=/var/log/mysql/mariadb-bin.log
4  binlog_format=ROW
5  bind-address=0.0.0.0
6
7
8  sudo systemctl restart mariadb
9
10 mysql -u root -p
11 "
12 CREATE USER 'slave'@'%' IDENTIFIED BY '1234';
13 GRANT REPLICATION SLAVE ON *.* TO 'slave'@'%';
14 FLUSH PRIVILEGES;
15 "
16
17 "FLUSH TABLES WITH READ LOCK;
18 SHOW MASTER STATUS\G"
19 #Note File and Position values (e.g., mariadb-bin.000002,
     764).
```

Listing 1.1: Master setup

Explanation
- Line 2: Assigns a unique `server-id` to the MariaDB server (must be unique in a cluster).
- Line 3: Enables binary logging and sets the log file path for replication.
- Line 4: Configures binary log format to `ROW`, which logs row-level changes.
- Line 5: Allows MariaDB to listen on all interfaces, so replicas can connect.
- Line 8: Restarts the MariaDB service to apply configuration changes.
- Line 10: Logs into MariaDB as the root user.
- Lines 12–14: Creates a replication user and grants it the necessary privileges.
- Line 17: Locks all tables to ensure consistency while noting the replication start point.
- Line 18: Displays the current binary log file name and position for replication setup.
- Line 19: Example values (`mariadb-bin.000002`, position 764) to be used by the slave.

1.2 Slave Node setup

```
 1 server-id=2
 2 relay-log=/var/log/mysql/mariadb-relay-bin.log
 3 log_bin=/var/log/mysql/mariadb-bin.log
 4 binlog_format=ROW
 5 
 6 systemctl restart mariadb
 7 
 8 
 9 mysql -u root -p
10 "
11 CHANGE MASTER TO
12 MASTER_HOST='192.168.64.36',
13 MASTER_USER='slave',
14 MASTER_PASSWORD='1234',
15 MASTER_LOG_FILE='mysql-bin.000002',
16 MASTER_LOG_POS=764;
17 START SLAVE;
18 "
```

Listing 1.2: Configuring slave

Explanation
- Line 1: Assigns a unique `server-id` to the replica (different from the master).
- Line 2: Specifies the relay log file location, where the replica stores events received from the master.
- Line 3: Enables binary logging on the replica so it can act as a master to other replicas if needed.
- Line 4: Sets the binary log format to `ROW` for precise row-level replication.
- Line 6: Restarts the MariaDB service to apply configuration changes.
- Line 9: Logs into MariaDB as the root user.
- Lines 11–16: Configures replication with the `CHANGE MASTER TO` command:
• `MASTER_HOST` sets the master server's IP address.
• `MASTER_USER` specifies the replication user created on the master.
• `MASTER_PASSWORD` provides the password for that user.
• `MASTER_LOG_FILE` indicates the binary log file name from the master.
• `MASTER_LOG_POS` sets the exact log position to start replication.
- Line 17: Starts the slave replication process.

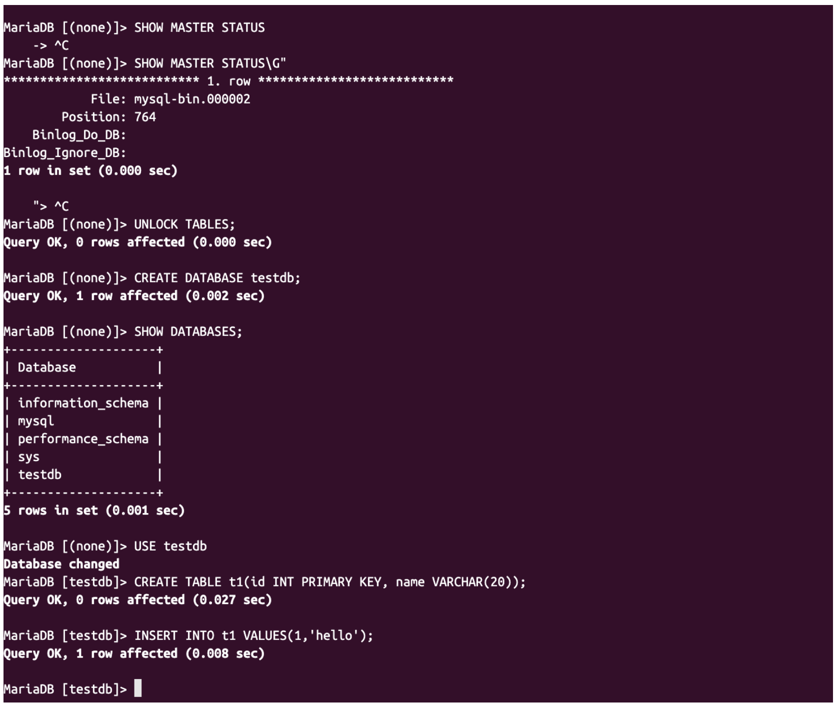
1.3 Testing Replication

```
 1 #on master
 2 mysql -u root -p
 3 "CREATE DATABASE testdb;
 4 USE testdb;
 5 CREATE TABLE t1(id INT PRIMARY KEY, name VARCHAR(20));
 6 INSERT INTO t1 VALUES(1,'hello');"
 7 
 8 #on slave
 9 mysql -u root -p
10 "SELECT * FROM testdb.t1;"
```

Listing 1.3: Testing salt-slave working

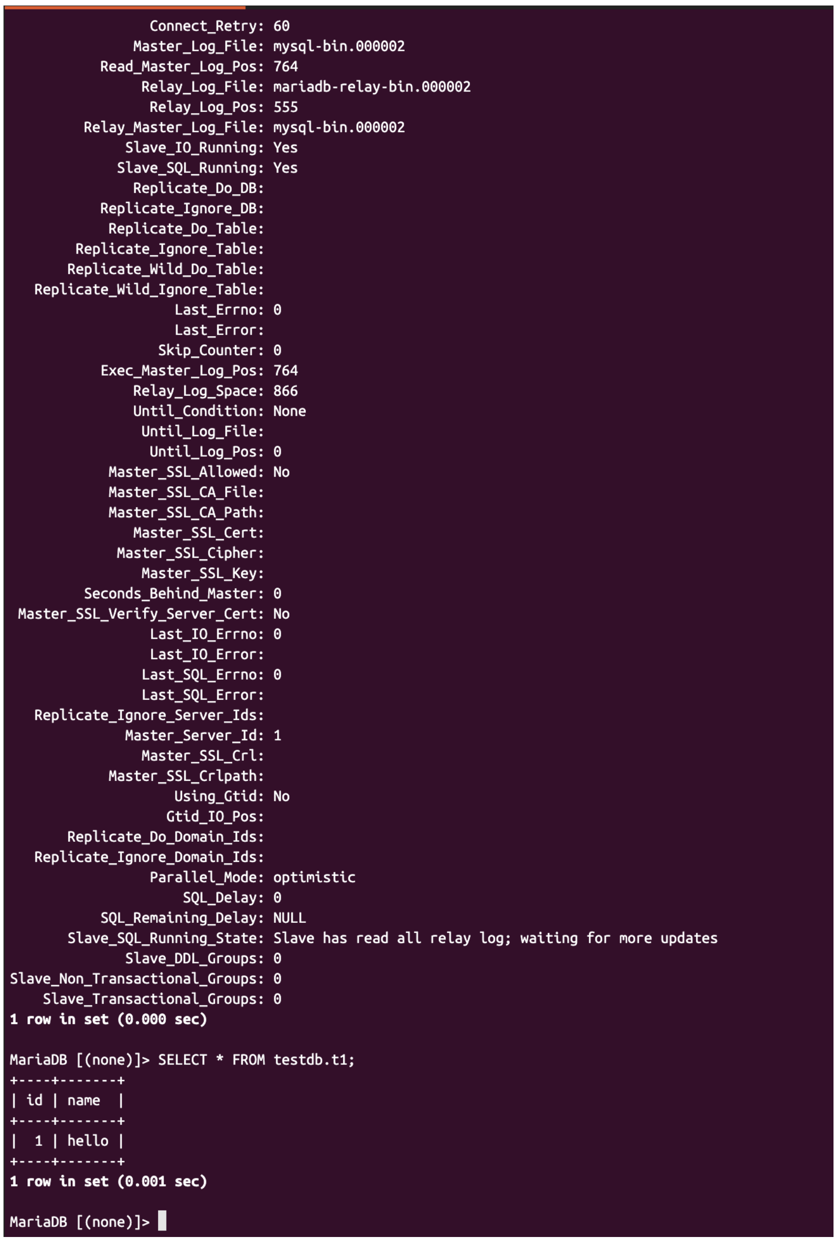
Explanation
- Line 2: Logs into MariaDB on the master as the root user.
- Lines 3–6: On the master, create a new database and table, then insert a sample row to test replication.
• `CREATE DATABASE testdb;` creates a new database.
• `USE testdb;` selects the database.
• `CREATE TABLE t1 (...);` defines a table with `id` and `name` columns.
• `INSERT INTO t1 VALUES (1, 'hello');` inserts one test record.
- Line 9: Logs into MariaDB on the slave as the root user.
- Line 10: On the slave, query the replicated table to confirm data is synced from the master.

1.4 Screenshots



```
MariaDB [(none)]> SHOW MASTER STATUS
    -> ^C
MariaDB [(none)]> SHOW MASTER STATUS\G"
*************************** 1. row ***************************
            File: mysql-bin.000002
        Position: 764
    Binlog_Do_DB:
Binlog_Ignore_DB:
1 row in set (0.000 sec)

    "> ^C
MariaDB [(none)]> UNLOCK TABLES;
Query OK, 0 rows affected (0.000 sec)

MariaDB [(none)]> CREATE DATABASE testdb;
Query OK, 1 row affected (0.002 sec)

MariaDB [(none)]> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| testdb             |
+--------------------+
5 rows in set (0.001 sec)

MariaDB [(none)]> USE testdb
Database changed
MariaDB [testdb]> CREATE TABLE t1(id INT PRIMARY KEY, name VARCHAR(20));
Query OK, 0 rows affected (0.027 sec)

MariaDB [testdb]> INSERT INTO t1 VALUES(1,'hello');
Query OK, 1 row affected (0.008 sec)

MariaDB [testdb]>
```

Figure 1.1: Master STATUS and Creating table for testing



```
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000002
          Read_Master_Log_Pos: 764
               Relay_Log_File: mariadb-relay-bin.000002
                Relay_Log_Pos: 555
        Relay_Master_Log_File: mysql-bin.000002
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 764
              Relay_Log_Space: 866
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Master_SSL_Allowed: No
           Master_SSL_CA_File:
           Master_SSL_CA_Path:
              Master_SSL_Cert:
            Master_SSL_Cipher:
               Master_SSL_Key:
        Seconds_Behind_Master: 0
 Master_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Master_Server_Id: 1
               Master_SSL_Crl:
           Master_SSL_Crlpath:
                   Using_Gtid: No
                  Gtid_IO_Pos:
      Replicate_Do_Domain_Ids:
  Replicate_Ignore_Domain_Ids:
                Parallel_Mode: optimistic
                    SQL_Delay: 0
          SQL_Remaining_Delay: NULL
      Slave_SQL_Running_State: Slave has read all relay log; waiting for more updates
             Slave_DDL_Groups: 0
Slave_Non_Transactional_Groups: 0
    Slave_Transactional_Groups: 0
1 row in set (0.000 sec)

MariaDB [(none)]> SELECT * FROM testdb.t1;
+----+-------+
| id | name  |
+----+-------+
|  1 | hello |
+----+-------+
1 row in set (0.001 sec)

MariaDB [(none)]>
```

Figure 1.2: Slave STATUS and replication from master

Chapter 2

Setup a Mariadb Galera Cluster

Discussion:
Split-brain issues are prevented by Galera's quorum mechanism, which uses majority voting to decide cluster health, and by ensuring that only one node is ever bootstrapped to avoid divergence. For optimization, Galera benefits from tuning several parameters to fully utilize system capacity: selecting the appropriate SST method (e.g., `mariabackup` for large datasets) and IST for fast incremental recovery, configuring `gcache.size` large enough to retain recent transactions, adjusting `wsrep_slave_threads` to match available CPU cores for parallel replication, and allocating sufficient InnoDB buffer pool memory to fit active data. Flow control thresholds (`gcs.fc_limit`, `fc_master_threshold`) should be tuned to minimize pauses under heavy load while still protecting lagging nodes.

If a node is down for two days, it may rejoin via Incremental State Transfer (IST) if all required changes remain in the donor's gcache. Otherwise, a full State Snapshot Transfer (SST) is required, which is slower and resource-intensive. To reduce the risk of costly SST, administrators should size gcache based on expected downtime and workload volume. Overall, careful parameter tuning ensures Galera makes full use of system resources, reduces replication lag, and avoids unnecessary downtime during node recovery. .

2.1 Install MariaDB, Galera, Mariadb-Backup (on each node)

```
1 sudo apt update
2 sudo apt install dirmngr software-properties-common apt-transport-https ca-certificates curl -y
3 curl -LsS https://r.mariadb.com/downloads/mariadb_repo_setup | sudo bash
4 sudo apt update
5 sudo apt install mariadb-server mariadb-client galera-4 mariadb-backup -y
6 sudo mariadb-secure-installation
```

Listing 2.1: Installing on db1

Explanation
- Line 1: Updates the local package index to ensure the latest versions are available.
- Line 2: Installs required tools and dependencies for adding repositories and handling HTTPS sources.
- Line 3: Downloads and runs the MariaDB repository setup script to configure the official MariaDB repository.
- Line 4: Updates the package index again, this time including the new MariaDB repository.
- Line 5: Installs MariaDB server, client, Galera cluster package, and backup tools.
- Line 6: Runs the MariaDB secure installation script to configure security options (root password, removing test DB, disabling remote root login, etc.).

2.2 Create Galera config (same file on all nodes, adjust node IP/name)

```
 1 #/etc/mysql/mariadb.conf.d/60-galera.cnf
 2 [mysqld]
 3 wsrep_on=ON
 4 binlog_format=row
 5 default_storage_engine=InnoDB
 6 innodb_autoinc_lock_mode=2
 7
 8 # Cluster name
 9 wsrep_cluster_name="mycluster"
10 # List all node addresses (use comma-separated IPs)
11 wsrep_cluster_address="gcomm
       ://192.168.64.38,192.168.64.39,192.168.64.40"
12
13 # Galera provider library
14 wsrep_provider="/usr/lib/galera/libgalera_smm.so"
15
16 # This node's address (IP:port)
17 wsrep_node_address="192.168.64.38"   # change per node
18 wsrep_node_name="db1"# change per node
19 wsrep_sst_method = mariabackup
20 wsrep_sst_auth = sstuser:1234
21 # Galera provider options for performance / cache
22 wsrep_provider_options="gcache.size=2G; gcs.fc_limit=128;"
```

Listing 2.2: Galera config

Explanation

- Line 1: Specifies the configuration file `60-galera.cnf` under `/etc/mysql/mariadb.conf.d/`.
- Line 2: Opens the `[mysqld]` section for server configuration.
- Line 3: Enables Galera replication by setting `wsrep_on = ON`.
- Line 4: Sets the binary log format to `ROW`, required for Galera consistency.
- Line 5: Uses `InnoDB` as the default storage engine (Galera requires InnoDB).
- Line 6: Configures `innodb_autoinc_lock_mode = 2`, which is recommended for Galera replication.
- Line 9: Defines the cluster name (`mycluster`). All nodes must share the same name.
- Line 11: Specifies the list of cluster node addresses using `gcomm://`.
- Line 14: Defines the Galera provider library path (`libgalera_smm.so`).
- Line 17: Sets the current node's IP address (change for each node).
- Line 18: Assigns a unique node name (e.g., `db1`, `db2`, etc.).
- Line 19: Chooses the state snapshot transfer (SST) method as `mariabackup`.

- Line 20: Provides the authentication credentials (`sstuser:1234`) used for SST.
- Line 22: Configures provider options such as cache size and flow control limits.

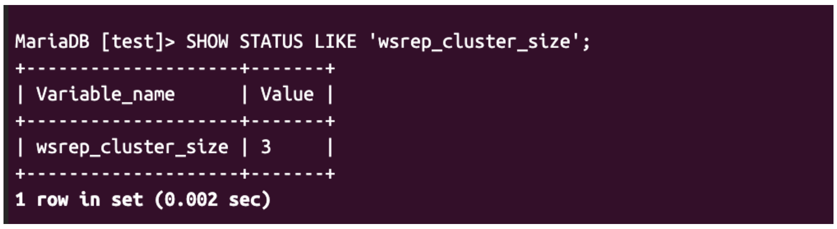
2.3 Creating SST user and Bootstraping db1

```
1 -- in mysql shell:
2 CREATE USER 'sstuser'@'%' IDENTIFIED BY '1234';
3 GRANT RELOAD, LOCK TABLES, REPLICATION CLIENT, REPLICATION
     SLAVE, PROCESS, FILE ON *.* TO 'sstuser'@'%';
4 FLUSH PRIVILEGES;
5
6 # Stop mariadb if running, then bootstrap:
7 sudo systemctl stop mariadb
8 # bootstrap cluster
9 sudo galera_new_cluster
```

Listing 2.3: creating SST user on db1

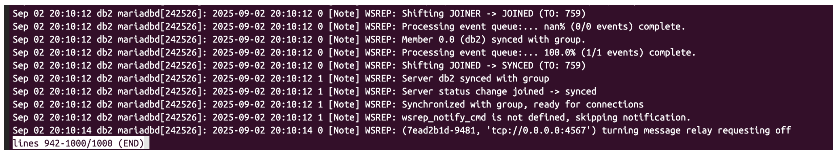
Explanation
- Line 2: Creates a special user `sstuser` with password `1234`, used for State Snapshot Transfer (SST).
- Line 3: Grants the required privileges (`RELOAD`, `LOCK TABLES`, `REPLICATION CLIENT`, `REPLICATION SLAVE`, `PROCESS`, `FILE`) so that SST can copy data between nodes.
- Line 4: Applies the new privileges immediately.
- Line 7: Stops the MariaDB service to prepare for cluster bootstrap.
- Line 9: Bootstraps the first Galera cluster node using `galera_new_cluster`. This initializes the cluster and marks this node as the primary component.

2.4 Screenshots



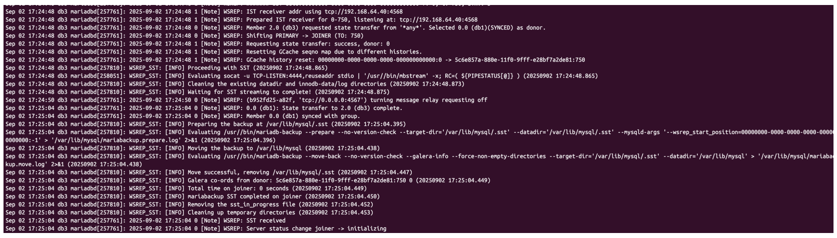
```
MariaDB [test]> SHOW STATUS LIKE 'wsrep_cluster_size';
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| wsrep_cluster_size | 3     |
+--------------------+-------+
1 row in set (0.002 sec)
```

Figure 2.1: Cluster size



```
Sep 02 20:10:12 db2 mariadbd[242526]: 2025-09-02 20:10:12 0 [Note] WSREP: Shifting JOINER -> JOINED (TO: 759)
Sep 02 20:10:12 db2 mariadbd[242526]: 2025-09-02 20:10:12 0 [Note] WSREP: Processing event queue:... nan% (0/0 events) complete.
Sep 02 20:10:12 db2 mariadbd[242526]: 2025-09-02 20:10:12 0 [Note] WSREP: Member 0.0 (db2) synced with group.
Sep 02 20:10:12 db2 mariadbd[242526]: 2025-09-02 20:10:12 0 [Note] WSREP: Processing event queue:... 100.0% (1/1 events) complete.
Sep 02 20:10:12 db2 mariadbd[242526]: 2025-09-02 20:10:12 0 [Note] WSREP: Shifting JOINED -> SYNCED (TO: 759)
Sep 02 20:10:12 db2 mariadbd[242526]: 2025-09-02 20:10:12 1 [Note] WSREP: Server db2 synced with group
Sep 02 20:10:12 db2 mariadbd[242526]: 2025-09-02 20:10:12 1 [Note] WSREP: Server status change joined -> synced
Sep 02 20:10:12 db2 mariadbd[242526]: 2025-09-02 20:10:12 1 [Note] WSREP: Synchronized with group, ready for connections
Sep 02 20:10:12 db2 mariadbd[242526]: 2025-09-02 20:10:12 1 [Note] WSREP: wsrep_notify_cmd is not defined, skipping notification.
Sep 02 20:10:14 db2 mariadbd[242526]: 2025-09-02 20:10:14 0 [Note] WSREP: (7ead2b1d-9481, 'tcp://0.0.0.0:4567') turning message relay requesting off
lines 942-1000/1000 (END)
```

Figure 2.2: Replication



```
Sep 02 17:24:48 db3 mariadbd[257761]: 2025-09-02 17:24:48 1 [Note] WSREP: IST receiver addr using tcp://192.168.64.40:4568
Sep 02 17:24:48 db3 mariadbd[257761]: 2025-09-02 17:24:48 1 [Note] WSREP: Prepared IST receiver for 0-750, listening at: tcp://192.168.64.40:4568
Sep 02 17:24:48 db3 mariadbd[257761]: 2025-09-02 17:24:48 0 [Note] WSREP: Member 2.0 (db3) requested state transfer from '*any*'. Selected 0.0 (db1)(SYNCED) as donor.
Sep 02 17:24:48 db3 mariadbd[257761]: 2025-09-02 17:24:48 0 [Note] WSREP: Shifting PRIMARY -> JOINER (TO: 750)
Sep 02 17:24:48 db3 mariadbd[257761]: 2025-09-02 17:24:48 1 [Note] WSREP: Requesting state transfer: success, donor: 0
Sep 02 17:24:48 db3 mariadbd[257761]: 2025-09-02 17:24:48 1 [Note] WSREP: Resetting GCache seqno map due to different histories.
Sep 02 17:24:48 db3 mariadbd[257761]: 2025-09-02 17:24:48 1 [Note] WSREP: GCache history reset: 00000000-0000-0000-0000-000000000000:0 -> 5c6e857a-880e-11f0-9fff-e28bf7a2de81:750
Sep 02 17:24:48 db3 mariadbd[257810]: WSREP_SST: [INFO] Proceeding with SST (20250902 17:24:48.865)
Sep 02 17:24:48 db3 mariadbd[258051]: WSREP_SST: [INFO] Evaluating socat -u TCP-LISTEN:4444,reuseaddr stdio | '/usr//bin/mbstream' -x; RC=( ${PIPESTATUS[@]} ) (20250902 17:24:48.865)
Sep 02 17:24:48 db3 mariadbd[257810]: WSREP_SST: [INFO] Cleaning the existing datadir and innodb-data/log directories (20250902 17:24:48.873)
Sep 02 17:24:48 db3 mariadbd[257810]: WSREP_SST: [INFO] Waiting for SST streaming to complete! (20250902 17:24:48.875)
Sep 02 17:24:50 db3 mariadbd[257761]: 2025-09-02 17:24:50 0 [Note] WSREP: (b952fd25-a82f, 'tcp://0.0.0.0:4567') turning message relay requesting off
Sep 02 17:25:04 db3 mariadbd[257761]: 2025-09-02 17:25:04 0 [Note] WSREP: 0.0 (db1): State transfer to 2.0 (db3) complete.
Sep 02 17:25:04 db3 mariadbd[257761]: 2025-09-02 17:25:04 0 [Note] WSREP: Member 0.0 (db1) synced with group.
Sep 02 17:25:04 db3 mariadbd[257810]: WSREP_SST: [INFO] Preparing the backup at /var/lib/mysql/.sst (20250902 17:25:04.395)
Sep 02 17:25:04 db3 mariadbd[257810]: WSREP_SST: [INFO] Evaluating /usr//bin/mariadb-backup --prepare --no-version-check --target-dir='/var/lib/mysql/.sst' --datadir='/var/lib/mysql/.sst' --mysqld-args '--wsrep_start_position=00000000-0000-0000-0000-0000
0000000:-1' > '/var/lib/mysql/mariabackup.prepare.log' 2>&1 (20250902 17:25:04.396)
Sep 02 17:25:04 db3 mariadbd[257810]: WSREP_SST: [INFO] Moving the backup to /var/lib/mysql (20250902 17:25:04.438)
Sep 02 17:25:04 db3 mariadbd[257810]: WSREP_SST: [INFO] Evaluating /usr//bin/mariadb-backup --move-back --no-version-check --galera-info --force-non-empty-directories --target-dir='/var/lib/mysql/.sst' --datadir='/var/lib/mysql' > '/var/lib/mysql/mariaba
kup.move.log' 2>&1 (20250902 17:25:04.438)
Sep 02 17:25:04 db3 mariadbd[257810]: WSREP_SST: [INFO] Move successful, removing /var/lib/mysql/.sst (20250902 17:25:04.447)
Sep 02 17:25:04 db3 mariadbd[257810]: WSREP_SST: [INFO] Galera co-ords from donor: 5c6e857a-880e-11f0-9fff-e28bf7a2de81:750 0 (20250902 17:25:04.449)
Sep 02 17:25:04 db3 mariadbd[257810]: WSREP_SST: [INFO] Total time on joiner: 0 seconds (20250902 17:25:04.449)
Sep 02 17:25:04 db3 mariadbd[257810]: WSREP_SST: [INFO] mariabackup SST completed on joiner (20250902 17:25:04.450)
Sep 02 17:25:04 db3 mariadbd[257810]: WSREP_SST: [INFO] Removing the sst_in_progress file (20250902 17:25:04.452)
Sep 02 17:25:04 db3 mariadbd[257810]: WSREP_SST: [INFO] Cleaning up temporary directories (20250902 17:25:04.453)
Sep 02 17:25:04 db3 mariadbd[257761]: 2025-09-02 17:25:04 0 [Note] WSREP: SST received
Sep 02 17:25:04 db3 mariadbd[257761]: 2025-09-02 17:25:04 0 [Note] WSREP: Server status change joiner -> initializing
```

Figure 2.3: SST on work

Chapter 3

Automatic failure setup

Discussion:
In Galera, only nodes in the Synced state should receive traffic. A common solution is to use Linux Virtual Server (LVS) in combination with Keepalived. Keepalived leverages VRRP (Virtual Router Redundancy Protocol) to assign a floating Virtual IP (VIP) that always points to a healthy node. With unicast VRRP configuration, cluster nodes explicitly know their peers and avoid multicast dependencies, making the setup more stable in modern networks. LVS provides efficient load balancing or failover by directing client traffic to the active node with the VIP. Health checks ensure that if a node fails or desynchronizes, the VIP is shifted automatically to another healthy node, maintaining continuous availability without manual intervention. To prevent false failovers, consistent checks are applied.

3.1 Installing and setting up Keepalived

```
 1 sudo apt update
 2 sudo apt install keepalived -y
 3 sudo systemctl enable keepalived
 4 
 5 #/etc/keepalived/keepalived.conf
 6 vrrp_script chk_mariadb {
 7     script "mariadb-admin -u root ping"
 8     interval 2
 9 weight 20
10 }
11 
12 vrrp_instance VI_1 {
13     state MASTER
14     interface enp0s1
15     virtual_router_id 51
16     priority 101
17     advert_int 1
18     authentication {
19 auth_type PASS
20 auth_pass 42
21     }
22 
23 # Unicast configuration
24     unicast_src_ip 192.168.64.38
25     unicast_peer {
26 192.168.64.39
27 192.168.64.40
28     }
29 
30 virtual_ipaddress {
31 192.168.64.50
32     }
33 
34 # Health check
35     track_script {
36 chk_mariadb
37     }
38 }
```

Listing 3.1: kEEPALIVED Master setup

Explanation

- Line 1: Updates the local package index.
- Line 2: Installs Keepalived, which provides VRRP for high availability.
- Line 3: Enables the Keepalived service to start automatically on boot.
- Line 5: Specifies the Keepalived configuration file location.

- Lines 6–10: Defines a health check script `chk_mariadb` that pings MariaDB every 2 seconds. If successful, the node gains weight (+20) in priority.
- Lines 12–22: Creates a VRRP instance (`VI_1`) with the following settings:
• `state MASTER` sets this node as the initial master.
• `interface enp0s1` specifies the network interface to use.
• `virtual_router_id 51` uniquely identifies this VRRP group.
• `priority 101` determines master preference (higher = more preferred).
• `advert_int 1` sets the VRRP advertisement interval (1 second).
• Authentication is configured with type `PASS` and password `42`.
- Lines 24–28: Configures unicast communication between VRRP peers (`192.168.64.39` and `192.168.64.40`).
- Lines 30–32: Defines the virtual IP address (`192.168.64.50`) that will float between nodes. Applications connect to this IP for high availability.
- Lines 35–37: Tracks the `chk_mariadb` script so that VRRP priority changes based on MariaDB health. If MariaDB fails, another node will take over.

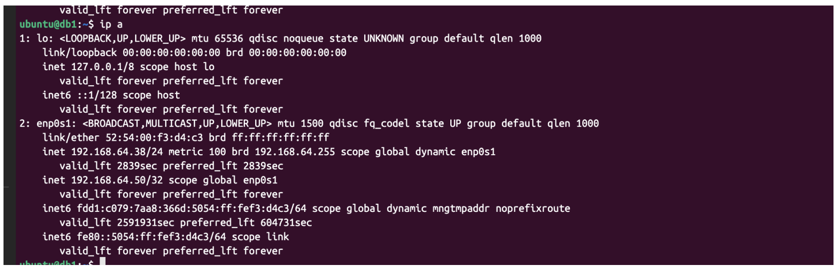
3.2 Testing VIP

```
1 #on db1(Master)
2 sudo systemctl stop mariadb
3
4 #on db2
5 ip addr show enp0s1
```

Listing 3.2: checking VIP on enp0s1

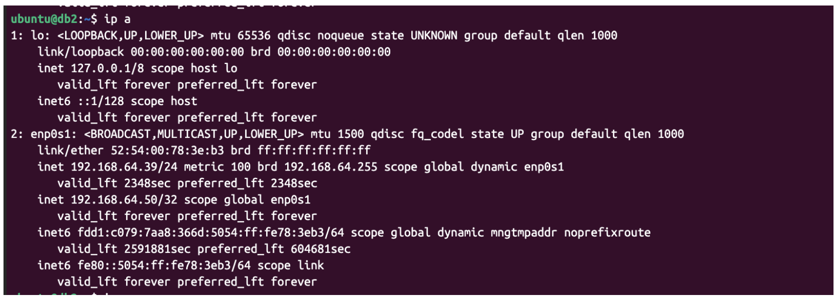
Explanation
- Line 1: Indicates that the following command is run on `db1`, which is the master node.
- Line 2: Stops the MariaDB service on the master to simulate a failure or to prepare for failover testing.
- Line 4: Switches context to `db2`, a replica/backup node.
- Line 5: Displays the IP addresses assigned to the network interface `enp0s1`, used to verify if the virtual IP has failed over from `db1` to `db2`.

3.3 Screenshots



```
       valid_lft forever preferred_lft forever
ubuntu@db1:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 52:54:00:f3:d4:c3 brd ff:ff:ff:ff:ff:ff
    inet 192.168.64.38/24 metric 100 brd 192.168.64.255 scope global dynamic enp0s1
       valid_lft 2839sec preferred_lft 2839sec
    inet 192.168.64.50/32 scope global enp0s1
       valid_lft forever preferred_lft forever
    inet6 fdd1:c079:7aa8:366d:5054:ff:fef3:d4c3/64 scope global dynamic mngtmpaddr noprefixroute
       valid_lft 2591931sec preferred_lft 604731sec
    inet6 fe80::5054:ff:fef3:d4c3/64 scope link
       valid_lft forever preferred_lft forever
ubuntu@db1:~$
```

Figure 3.1: Master having VIP



```
ubuntu@db2:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 52:54:00:78:3e:b3 brd ff:ff:ff:ff:ff:ff
    inet 192.168.64.39/24 metric 100 brd 192.168.64.255 scope global dynamic enp0s1
       valid_lft 2348sec preferred_lft 2348sec
    inet 192.168.64.50/32 scope global enp0s1
       valid_lft forever preferred_lft forever
    inet6 fdd1:c079:7aa8:366d:5054:ff:fe78:3eb3/64 scope global dynamic mngtmpaddr noprefixroute
       valid_lft 2591881sec preferred_lft 604681sec
    inet6 fe80::5054:ff:fe78:3eb3/64 scope link
       valid_lft forever preferred_lft forever
```

Figure 3.2: Back-up node having VIP when Master is down

Chapter 4

Altering table

Discussion:
In Galera, small alters can use Total Order Isolation (TOI) for automatic cluster-wide replication, while large alters are better handled with Rolling Schema Upgrade (RSU), applied node by node. For very large tables in asynchronous replication, MariaDB also provides Non-Blocking Operations (NBO), which allow certain schema changes to proceed without holding exclusive metadata locks for the entire duration of the operation. This minimizes blocking of concurrent queries and reduces application impact. Applications must remain compatible with both old and new schemas during transitions.

4.1 RSU (Rolling Schema Upgrade)

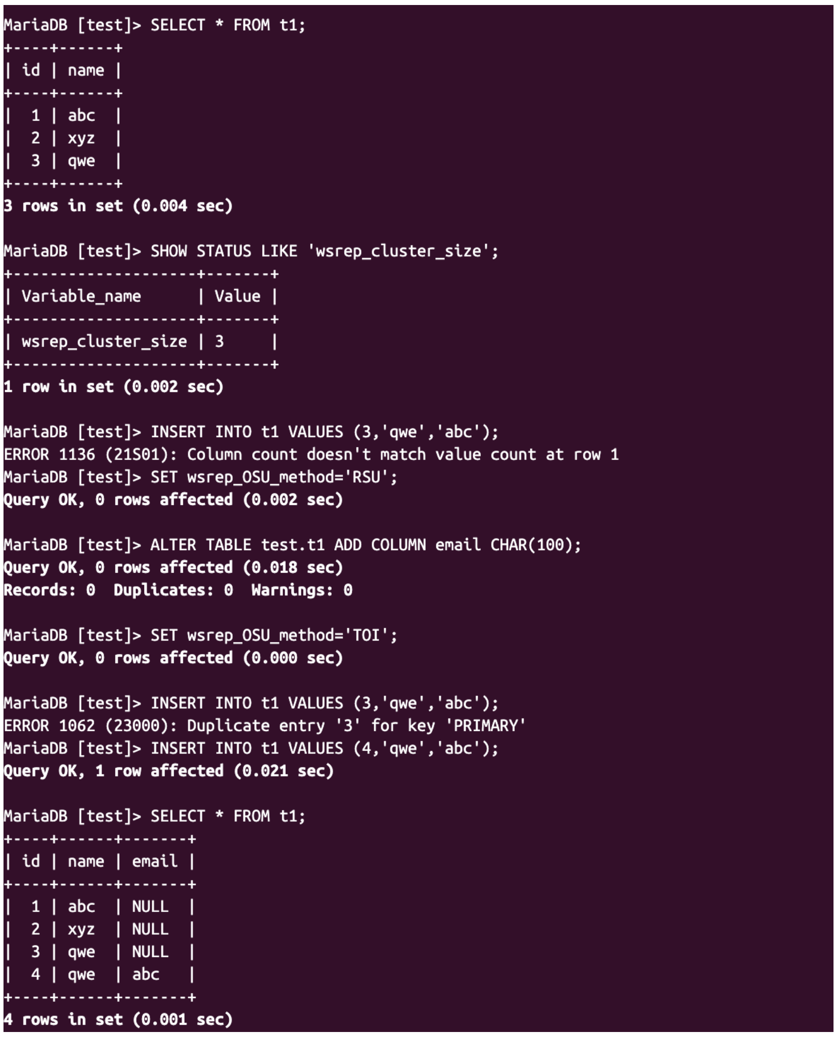
```
1 #Adding column
2 SET wsrep_OSU_method='RSU';
3 ALTER TABLE test.t1 ADD COLUMN email CHAR(100);
4 SET wsrep_OSU_method='TOI';
5
6 #Adding index
7 SET wsrep_OSU_method='RSU';
8 ALTER TABLE test.t1 ADD INDEX idx_name (name);
9 SET wsrep_OSU_method='TOI';
```

Listing 4.1: Doing RSU on db1

Explanation

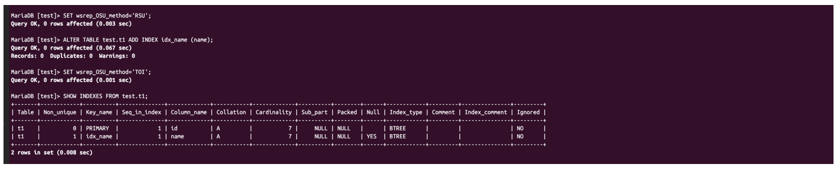
- Line 2: Sets the `wsrep_OSU_method` to `RSU` (Rolling Schema Upgrade). In this mode, schema changes are applied only on the local node and not replicated to the cluster automatically.
- Line 3: Adds a new column `email` to the table `test.t1`. Since RSU is used, this must be repeated manually on other nodes.
- Line 4: Switches the `wsrep_OSU_method` back to `TOI` (Total Order Isolation), the default Galera mode where schema changes are replicated cluster-wide.
- Line 7: Again sets the schema update method to `RSU` for the index creation.
- Line 8: Adds an index named `idx_name` on the column `name`. This change, like the column addition, affects only the local node in RSU mode.
- Line 9: Returns to `TOI` mode so that future schema changes are replicated across the cluster.

4.2 Screenshots



```
MariaDB [test]> SELECT * FROM t1;
+----+------+
| id | name |
+----+------+
|  1 | abc  |
|  2 | xyz  |
|  3 | qwe  |
+----+------+
3 rows in set (0.004 sec)

MariaDB [test]> SHOW STATUS LIKE 'wsrep_cluster_size';
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| wsrep_cluster_size | 3     |
+--------------------+-------+
1 row in set (0.002 sec)

MariaDB [test]> INSERT INTO t1 VALUES (3,'qwe','abc');
ERROR 1136 (21S01): Column count doesn't match value count at row 1
MariaDB [test]> SET wsrep_OSU_method='RSU';
Query OK, 0 rows affected (0.002 sec)

MariaDB [test]> ALTER TABLE test.t1 ADD COLUMN email CHAR(100);
Query OK, 0 rows affected (0.018 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [test]> SET wsrep_OSU_method='TOI';
Query OK, 0 rows affected (0.000 sec)

MariaDB [test]> INSERT INTO t1 VALUES (3,'qwe','abc');
ERROR 1062 (23000): Duplicate entry '3' for key 'PRIMARY'
MariaDB [test]> INSERT INTO t1 VALUES (4,'qwe','abc');
Query OK, 1 row affected (0.021 sec)

MariaDB [test]> SELECT * FROM t1;
+----+------+-------+
| id | name | email |
+----+------+-------+
|  1 | abc  | NULL  |
|  2 | xyz  | NULL  |
|  3 | qwe  | NULL  |
|  4 | qwe  | abc   |
+----+------+-------+
4 rows in set (0.001 sec)
```

Figure 4.1: Column added on db1



```
MariaDB [test]> SET wsrep_OSU_method='RSU';
Query OK, 0 rows affected (0.003 sec)

MariaDB [test]> ALTER TABLE test.t1 ADD INDEX idx_name (name);
Query OK, 0 rows affected (0.067 sec)
Records: 0  Duplicates: 0  Warnings: 0

MariaDB [test]> SET wsrep_OSU_method='TOI';
Query OK, 0 rows affected (0.001 sec)

MariaDB [test]> SHOW INDEXES FROM test.t1;
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+
| Table | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Ignored |
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+
| t1    |          0 | PRIMARY  |            1 | id          | A         |           7 |     NULL | NULL   |      | BTREE      |         |               | NO      |
| t1    |          1 | idx_name |            1 | name        | A         |           7 |     NULL | NULL   | YES  | BTREE      |         |               | NO      |
+-------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+---------+---------------+---------+
2 rows in set (0.008 sec)
```

Figure 4.2: Index added on db1