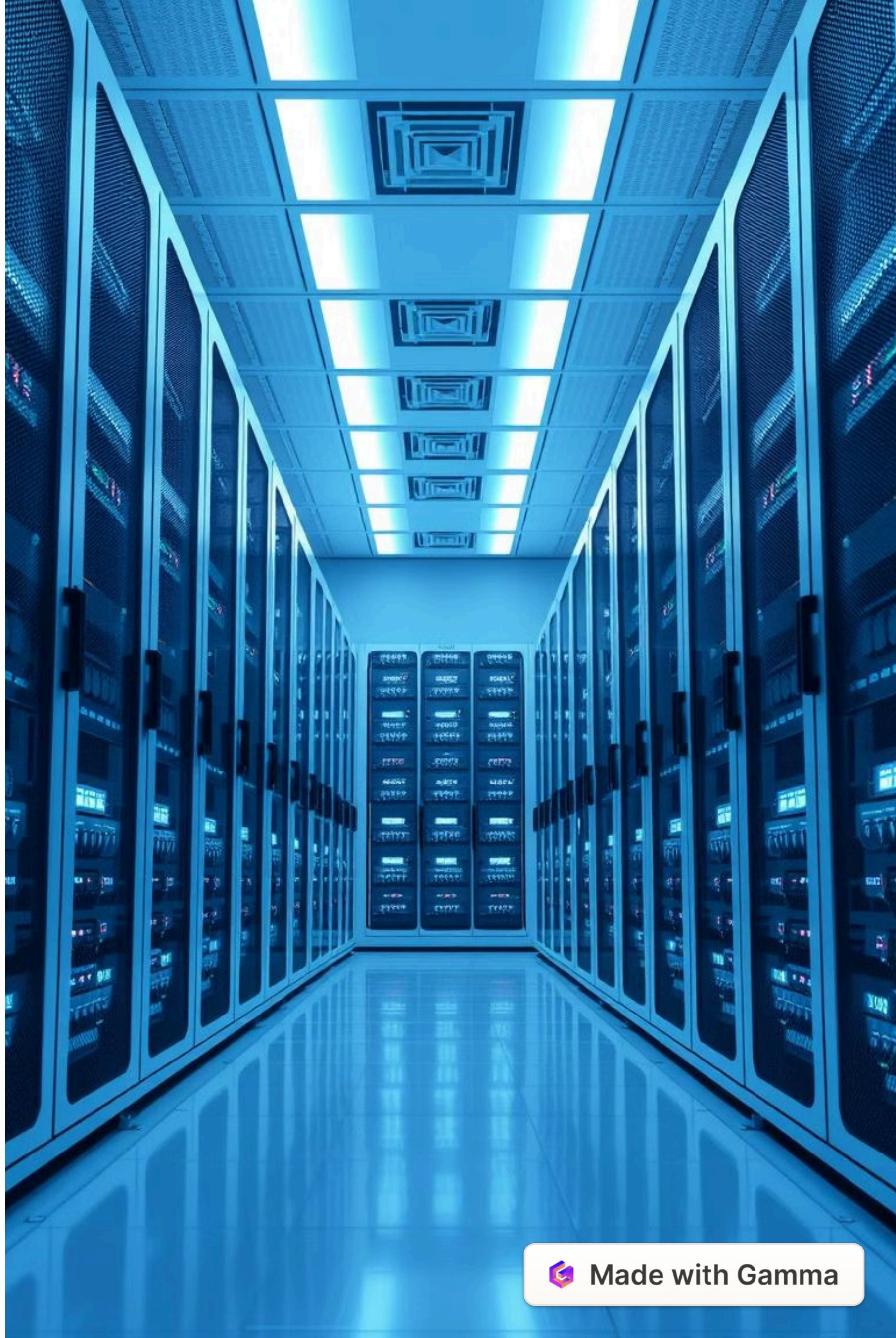
MariaDB Backups

1. Logical Backup

A **logical backup** in MariaDB refers to a backup method that exports database objects such as tables, schemas, and data in a human-readable format, typically SQL statements. Logical backups do not include raw data files but instead generate scripts that can be used to recreate the database.



Key Features of Logical Backups

1. **Human-Readable** – The backup consists of SQL `CREATE`, `INSERT`, and `ALTER` statements.
2. **Portability** – Can be restored on different MariaDB/MySQL versions or other database systems.
3. **Selective Backup** – Allows backing up specific databases or tables.
4. **Flexible Restore** – You can modify the SQL before restoring.
5. **Slower Than Physical Backup** – Since it involves exporting data as SQL, it can be slower for large databases.

Logical Backup Command & details

Best Practice to create backup with local user by assigning suitable permission.

1. **mysqldump -u root -p arya_devops > /db_backup/backup.dump**.
2. check backup.dump also we can send this file to another system and restore data there as well.
3. To restore data we required one database first.
4. **Mysql -u root -p database-name < /badb_backupckup/backup.dump.**

2. Physical Backup

A **physical backup** in MariaDB involves copying the actual database files, such as `.frm`, `.ibd`, and binary logs, rather than exporting SQL statements. This method is faster than logical backups, especially for large databases.

Key Features of Physical Backup

✔️ **Faster than Logical Backups** – Directly copies data files instead of exporting SQL statements.
✔️ **Consistent Snapshot** – Ensures data consistency, especially with InnoDB tables.
✔️ **Efficient for Large Databases** – Avoids the performance overhead of logical export/import.
✔️ **Binary-Compatible** – Can be restored without conversion, but must match OS and MariaDB

Physical Backup Command & details

1. For Full backup of directory /var/lib/mysql the directory where backup is going to be store that must be empty or not present.
2. Syntax - mariabackup --backup --target-dir=/path/to/backup --user=root --password=your_password
3. **mariabackup --backup --target-dir /physical-backup/ --user root --password redhat.**
4. Now to restore the backup the mariadb service should be stop.
5. **systemctl stop mariadb**
6. now delete the main database file **rm -rvf /var/liv/mysql/***
7. **mariabackup --copy-back --target-dir=/physical-backup. or mariabackup --move-back --target-dir=/physical-backup**
8. Now if we start the that will not start because all copy files and owner root:root but that should be mysql.
9. **chown -R mysql:mysql /var/lib/mysql/**
10. **systemctl start mariadb.service**

3. Replication Backup in MariaDB

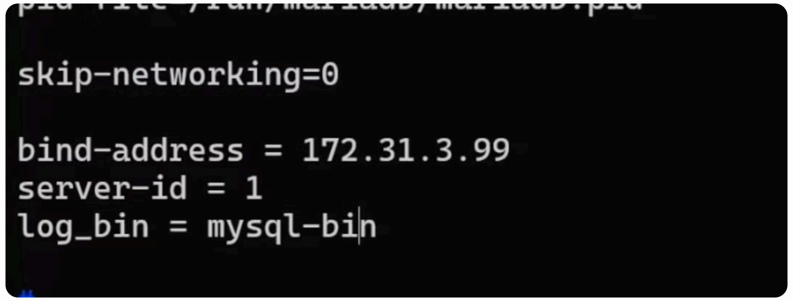
Replication in MariaDB allows a secondary server (slave/replica) to maintain a copy of the primary server (master). **Replication backup** ensures data availability, disaster recovery, and scalability. Instead of traditional backups, replication enables **real-time or near-real-time data synchronization**.

1. MasterConfiguration: ● The master server maintains a binary log (binlog), which records all changes to the database (like INSERT, UPDATE, DELETE). ● Each transaction is written to the binary log in sequence as events.
2. SlaveConfiguration: ● The slave server reads the binary log of the master and applies the same changes to its own data.

ErrorHandling:
● If the slave loses connection with the master, it can resume replication by continuing from the last processed event in the relay log when the connection is restored.

Configure the Master Node (Server)

1. Configure the **Master** node and grant the **slave** node access to it. First, we need to edit the **mysql-server.cnf** configuration file. $ sudo vim /etc/my.cnf.d/mysql-server.cnf
2. Add the following data in the config file



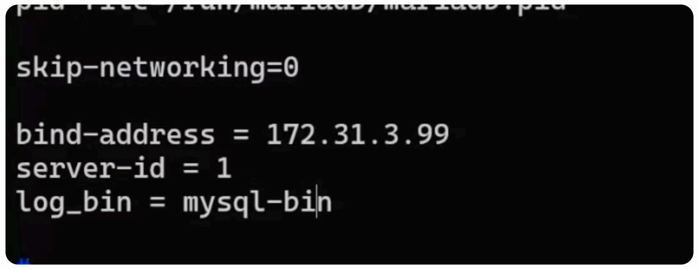
```
pid-file=/run/mariadb/mariadb.pid

skip-networking=0

bind-address = 172.31.3.99
server-id = 1
log_bin = mysql-bin
```

3. Once done, save the changes and exit. Then restart the **MySQL** server. $ **sudo sysemctl restart mysqld**
4. sudo mysql -u root.
5. create user 'replica'@'slaveIP' identified by 'P@ssword321';
6. GRANT REPLICATION SLAVE ON . TO 'replica'@'slaveIP';
7. FLUSH PRIVILEGES;
8. exit
9. Verify the status of the master. mysql> **SHOW MASTER STATUS\G**

Configure the Slave Node (Server)

1. Edit the **mysql-server.cnf** configuration file.
2. sudo vim /etc/my.cnf.d/mysql-server.cnf

As before, paste these lines under the [mysqld] section. Change the IP address to correspond to the slave's IP. Also, assign a different **server-id**. Here we have assigned it the value of **2**.



```
skip-networking=0

bind-address = 172.31.3.99
server-id = 1
log_bin = mysql-bin
```

3. Save the changes and exit the file. Then restart the database server. **sudo systemctl restart mysqld**

4. stop slave;

5. Then execute the following command to configure the slave node to replicate databases from the master

6. CHANGE MASTER TO

→> MASTER_ HOST='172.31. 3.99',

→> MASTER_USER='replica'

-> MASTER_PASSWORD='P@ssword321'

- MASTER_LOG_FILE= 'mysqL-bin. 000001'

→> MASTER_LOG_POS=1105;

7. START SLAVE;

8. Now create the database on master and check the same on salve machine data will replicate in real time.