How To Backup and Restore MySQL Database

Scripts And Commands with Examples

Asfaw Gedamu

MYSQL DATABASE BACKUP AND RECOVERY

- Backup a MySQL database
- Restore a MySQL database from backup
- Backup multiple MySQL databases
- Backup a specific table
- Restore all databases
- Backup databases in compress format

MySQL offers various methods for backup and recovery. Here's a brief backup and recovery walkthrough with scripts and examples:

1. Backup a MySQL Database:

Method 1: Using mysqldump (Logical Backup):

```
Syntax: mysqldump -u <username> -p<password> <database_name> > backup.sql


[mysql@localhost]# mysqldump –u root –pmysql sampledb > /tmp/dbdump.sql


-- MySQL dump 10.13  Distrib 8.0.30, for Linux (x86_64)
-- Host: localhost    Database: mydatabase
-- ------------------------------------------------------
-- Server version         8.0.30
-- Table structure for table `mytable1`

CREATE TABLE `mytable1` (
  ...
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

-- Dumping data for table `mytable1`

LOCK TABLES `mytable1` WRITE;
/*!40000 ALTER TABLE `mytable1` DISABLE KEYS */;
INSERT INTO `mytable1` VALUES (...);
/*!40000 ALTER TABLE `mytable1` ENABLE KEYS */;
UNLOCK TABLES;
-- Table structure for table `mytable2`
...
```

This example takes a backup of sampledb database and dumps the output to dbdump.sql

Method 2: Using mysqladmin (Physical Backup):

```
Syntax: mysqladmin -u <username> -p<password> shutdown
cp -r /data/mysql/<database_name> /backups/<database_name>
mysqladmin -u <username> -p<password> start
```

This shuts down the MySQL server, copies the physical directory of the database_name to a backup location, and then restarts the server.

2. Restore a MySQL Database from Backup:

Method 1: Using mysql (Logical Restore):

```
Syntax: mysql -u <username> -p<password> <database_name> < backup.sql
```

This applies the SQL statements in the backup.sql file to the existing database_name or creates a new database with the same name.

```
[mysql@localhost]# mysqldump –u root –pmysql sampledb > dbdump.sql
Query OK, 100 rows affected (0.01 sec)
Query OK, 50 rows affected (0.02 sec)
...
```

For example, if you want to take backup of both sampledb and newdb database, execute the mysqldump as shown below:

```
[root@localhost]# mysqldump –u root –p --databases sampled newdb > /tmp/dbdump.sql
```

We cannot take backup of information_schema and performance_schema databases as these are metadata databases.

```
-- Backup all databases:
root# mysqldump -u root -p --all-databases > all-database.sql
```

Method 2: Using mysqlbinlog (Point-in-Time Recovery):

```
mysqlbinlog -u <username> -p<password> --start-position=<log_position> --stop-position=<end_position> > recovery.sql
mysql -u <username> -p<password> < recovery.sql
```

This uses the binary logs to extract specific changes within a time range and then applies them to recover the database to a specific point in time.

3. Backup Multiple MySQL Databases:

Method 1: Using mysqldump:

```
Syntax: mysqldump -u <username> -p<password> --databases db1 db2 db3 > multi_databases.sql

---This backs up multiple databases (db1, db2, db3) into a single multi_databases.sql file.
```

For example, if you want to take backup of both sampledb and newdb database, execute the mysqldump as shown below:

```
[root@localhost]# mysqldump –u root –p --databases sampled newdb > /tmp/dbdump.sql
```

We cannot take backup of information_schema and performance_schema databases as these are metadata databases.

```
-- Backup all databases:
root# mysqldump -u root -p --all-databases > all-database.sql
```

Method 2: Using a Script:

Write a script to loop through a list of databases and call mysqldump for each one.

```
#!/bin/bash
# Database credentials
MYSQL_USER="your_mysql_username"
MYSQL_PASSWORD="your_mysql_password"
```

```
# List of databases to back up (modify as needed)
DATABASES="db1 db2 db3"
# Backup directory (adjust as needed)
BACKUP_DIR="/path/to/backups"
# Loop through the databases
for db_name in $DATABASES; do
  # Create the backup file path
  backup_file="$BACKUP_DIR/$db_name.sql"
  # Create the backup
  echo "Backing up database $db_name..."
  mysqldump -u "$MYSQL_USER" -p"$MYSQL_PASSWORD" "$db_name" > "$backup_file"
done


Backing up database db1...
Query OK, 100 rows affected (0.01 sec)

Backing up database db2...
Query OK, 50 rows affected (0.02 sec)
...
```

To use the script:

1. **Save it as a file** (e.g., backup_databases.sh).
2. **Make it executable:** chmod +x backup_databases.sh
3. **Run the script:** ./backup_databases.sh

Important notes:

- **Replace the placeholders** with your actual MySQL username, password, backup directory path, and list of databases.
- **Ensure the backup directory** exists and is writable.
- **Test the script** thoroughly on a non-production environment before using it in production.

- **Consider error handling:** Add error handling to the script to catch potential issues during the backup process.

4. Backup a Specific Table:

```
Syntax: mysqldump -u <username> -p<password> <database_name> <table_name> > table_backup.sql

---In this example, we backup only the ta2 table from sampledb database.


root$ mysqldump –u root –p sample ta2 > /tmp/nwedb_ta2.sql


-- MySQL dump 10.13  Distrib 8.0.29, for Linux (x86_64)
--
-- Host: localhost    Database: mydatabase
-- ------------------------------------------------------
-- Server version         8.0.29
-- Dumping data for table mytable

LOCK TABLES `mytable` WRITE;
/*!40000 ALTER TABLE `mytable` DISABLE KEYS */;
INSERT INTO `mytable` VALUES (...);
/*!40000 ALTER TABLE `mytable` ENABLE KEYS */;
UNLOCK TABLES;
```

This backs up only the table_name from the database_name into table_backup.sql.

5. Restore All Databases:

Method 1: Using a Script:

Write a script to loop through the backup files of each database and call mysql to restore them individually.

```
#!/bin/bash
# Database credentials
MYSQL_USER="your_mysql_username"
MYSQL_PASSWORD="your_mysql_password"
# Directory containing backup files (adjust as needed)
```

```
BACKUP_DIR="/path/to/backups"
# Loop through backup files
for file in "$BACKUP_DIR"/*.sql; do
  # Extract database name from filename (assuming filename format is database_name.sql)
  db_name=$(basename "$file" .sql)
  # Restore the database
  echo "Restoring database $db_name..."
  mysql -u "$MYSQL_USER" -p"$MYSQL_PASSWORD" "$db_name" < "$file"
done



Restoring database db1...
Query OK, 100 rows affected (0.01 sec)
Restoring database db2...
Query OK, 50 rows affected (0.02 sec)
```

To use the script:

1. **Save it as a file** (e.g., restore_databases.sh).
2. **Make it executable:** chmod +x restore_databases.sh
3. **Run the script:** ./restore_databases.sh

Important notes:

- **Replace the placeholders** with your actual MySQL username, password, and backup directory path.
- **Ensure the backup files** are in the specified directory and have the .sql extension.
- **Test the script** thoroughly on a non-production environment before using it in production.
- **Consider error handling:** Add error handling to the script to catch potential issues during the restoration process.

6. Backup Databases in Compressed Format:

```
Syntax: mysqldump -u <username> -p<password> <database_name> | gzip -c > backup.sql.gz
```

```
---With bzip2:
mysqldump --all-databases | bzip2 -c > databasebackup.sql.bz2


---With gzip:
mysqldump --all-databases | gzip> databasebackup.sql.gz
```

This pipes the output of mysqldump directly to gzip for compression, creating a compressed backup file backup.sql.gz.

Note: Please remember to choose the appropriate method based on your specific needs and environment. Always test your recovery process on a non-production system before relying on it in a critical situation.

These are just some basic examples, and the specific commands may vary depending on your MySQL version and configuration. Refer to the official MySQL documentation for detailed information on backup and recovery options: https://dev.mysql.com/doc/mysql-