Documentation: Database Backup and Recovery

Objective

To demonstrate the process of backing up a MySQL database and restoring it in case of a failure using automated scripts.

Tools Used

- Database: MySQL

- Script Language: Bash (Linux Shell)

- Utilities: mysqldump, mysql

- Storage: Local backup directory

Directory Structure

/project-directory

■■■ db_backup_restore.sh # Combined backup and restore script

■■■ /backups # Directory to store .sql backup files

Backup Process

Step 1: Define Configuration

- Set database name, user, password, and backup directory.

- Generate a filename with a timestamp for uniqueness.

Step 2: Execute Backup

- Use mysqldump to export the database to an SQL file.

- Save the file with the current date and time.

- Copy the latest version to a fixed name for recovery convenience.

Output Example:

Starting backup for database: mydb

Backup successful: /backups/mydb-backup-2025-07-20_14-30-00.sql

Recovery Process

Step 1: Locate Backup File

- Check for the existence of the most recent backup SQL file (mydb-latest-backup.sql).

Step 2: Restore Database

- Use mysql to import the backup file into the database.

- Ensure database service is running before execution.

Output Example:

Starting recovery for database: mydb

Recovery successful!

Best Practices

- Automate backups using a cron job for regular intervals (daily, weekly).

- Test recovery periodically to ensure backups are usable.

- Secure sensitive data by restricting permissions and optionally encrypting backups.

- Use remote storage or cloud backup for disaster recovery plans.

- Keep documentation for team usage and handovers.

Sample Cron Job (Daily at 2 AM)

0 2 * * * /bin/bash /path/to/db_backup_restore.sh backup >> /path/to/logfile.log 2>&1

Notes

- This documentation and script are specific to MySQL.

- Replace placeholders in the script with actual credentials and paths before use.

- Ensure execution permissions (chmod +x) are set on the script file.