Lab 11.1b: Backing up MySQL
IN719 Systems Administration

Introduction

Backing up data from a database is a special case. Ultimately database systems store their data on disk in file, so on the one hand we could just back up those files. We call this a *physical backup*. However, those file are likely to be changing very frequently and this will interfere with backups. We could resolve this but shutting down the database while the backup is running, but our use cases may not allow that downtime. We could go further by setting up a replica database on another server and back up the data from the replica. This is a good strategy, but obviously involves additional complication and resources. Note that most database systems provide some tooling to support physical backups.

Another strategy is to perform a *logical backup*. Rather than copying the raw data from disk, a logical backup writes an SQL script that creates the database tables and populates them with data. It uses the locking capabilities of the database server itself to get a consistent snapshot of the data even as it's being updated. Logical backups are also very flexible both to create ands to restore. The main drawback of them is that backups and restores of large databases are very slow and the backup files themselves are very large. Since we are not backing up large amounts of data this will not be a problem for us.

More information about this topic, in particular for MariaDB systems like ours, is available at `https://mariadb.com/kb/en/backup-and-restore-overview/`.

1 Using mysqldump

Since we set up a backup process for files and directories in part (a) of this lab, we just need to see how to produce the logical backup file for our database. We will do this with the `mysqldump` utility. This will produce an SQL script that will restore our database structure and data. For example, the command

```
mysqldump --all-databases --add-drop-table > /home/someuser/db-backup.sql
```

will save your database information in the file `db-backup.sql` in someuser's home directory. You can back up this file by adding it to the script you produced in part (a). If you want to see more options for how to use `mysqldump`, follow the link above. You should do some reading to be sure that you know how to load a database from a dump file.

You will need to run this process regularly as part of your backup. You can either add the `mysqldump` command to the relevant shell script you created earlier, or you could set up a seperate cron job to handle it. Both options work fine as long as everyone on your team is clear on what process to follow and it's properly documented.