Lab 11.02: Backing up MySQL
IN719 Systems Administration

Introduction

Backing up data from a database is a special case. We could just back up the database files, but if the database server is running when we perform the backup the results will be unreliable. Shutting down the database server would allow us to perform a physical backup of the files, but the resulting system downtime may not be acceptable. If we want to perform a backup of a database without requiring downtime, then we need to use a specialised database backup tool. One such tool is *mysqldump*. We will use it in this lab.

1 Using mysqldump

We can easily 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. If this file is backed up by Bacula we will be able to restore the database as it was when the file was produced.

There are many options for `mysqldump` that are documented at `https://dev.mysql.com/doc/refman/5.7/en/using-mysqldump.html`.

2 Set up a cron job

Since we must automate our backup processes, it's necessary to place a command like the one above in cron. A suitable crontab entry would be something like this

```
20 1 * * * mysqldump --all-databases --add-drop-table > /home/someuser/db-backup.sql
```

will write a backup file at 1:20 each morning. Note that, since it will overwrite the output file every time it is run, it's important to back up the file after it is written.

More information about cron is available at `http://www.tldp.org/LDP/lame/LAME/linux-admin-made-easy/using-cron.html`.

3 Using Puppet

Since we prefer to manage our configuration with Puppet, we should set up a Puppet `cron` resource as part of our MySQL module. In this way we ensure that every MySQL server runs a backup job.

See `https://docs.puppet.com/puppet/3.5/type.html#cron` for more information.