MYSQL BASIC ADMINISTRATION

Logs? Configs? `mysqladmin`...

MY.CNF

- The main MySQL configuration file is `my.cnf`
- On many Linux systems, when MySQL is installed from package, this file is located in `/etc/my.cnf`
- Users can also have a `.my.cnf` in their home directory, which will be parsed for client options when using the `mysql` command.
- The format for the `my.cnf` file is similar to a windows ini file format. Sections are headed with `[sectionname]` and settings are simple `name=value` pairings.
- `my.cnf` settings are simply command line argument defaults

IMPORTANT MY.CNF SETTINGS

- There are lots of settings available in my.cnf. Some important ones include:
  - `datadir`: Filesystem path to data files
  - `log-error`: Filesystem path for error log file
  - `max_allowed_packet`: Sets maximum packet size for data exchanges between server and client.
- To see all parameters for `mysqld`:
  - `mysqld --help --verbose`

LAB

1) Browse through your global `my.cnf` file. Look up some of the parameters in the MySQL documentation.

2) Set up a user `.my.cnf` file so that you can connect to your MovieCollection database as moviedba automatically, without needing to type a password.

MYSQLADMIN

- `mysqladmin` is a very useful command line administration tool for the MySQL system. Some of the operations an admin can perform include:
  - Database creation/deletion
  - Cache flushing
  - Server shutdown
  - Password management

LAB

1) Use `mysqladmin` to ping your server. An exit code of 0 means the server is running, 1 means it is not. The exit code can be viewed by typing "echo $?". Shutdown your server and ping again. Can you imagine how you could write a simple server monitoring script?

2) Explore the `processlist` subcommand and theorize as to it's output. In another window, connect to the server. Then run the `processlist` subcommand again. See your second session? Use the `kill` subcommand to destroy your second connection. Very useful for runaway queries. We will explore this further in a future lecture.

LOGS

- MySQL maintains a log file with useful information on the server's activity.
- The path name to the log file can be determined by examining the global my.cnf file.
- During a future lab, take a few minutes to examine your log file. Use google for more information on any cryptic log messages

NETWORKING

- By default, the MySQL server will grab every network interface on a machine and listen on port 3306.
- To control the IP addresses MySQL will listen for traffic on, adjust the “bind-address” parameter. Example:
  - `bind-address = 192.168.1.15`
- If this setting is not specified, or if it is set to `0.0.0.0`, MySQL will listen on all interfaces. Note that those are the only options - all interfaces or one.

ROOT PASSWORD RESET

- The class has already covered changing passwords, but how to reset the root password when it is lost?
- There are two ways of doing this:
  - Disable all access controls, restart the server and change it manually. Then restart again with access controls enabled. Not the best solution.
  - Write a short SQL script to reset the password, then add an init-file option to my.cnf.
- Let's see examples of both methods...

INSECURE ROOT PASSWORD RESET

- `/etc/init.d/mysqld stop`
- `mysqld_safe --skip-grant-tables &`
- `mysql mysql`
- `UPDATE user SET password = '' where user = 'root' AND host = 'localhost';`
- `mysqladmin shutdown`
- `/etc/init.d/mysqld start`

SECURE ROOT PASSWORD RESET

- Create a new text file with the following contents:
  - `SET PASSWORD FOR`

    `'root'@'localhost' = PASSWORD('new_password');`
- Name it `/var/lib/mysql/mysql-init`
- Add under `[mysqld]` to `/etc/my.cnf:`
  - `init-file=/var/lib/mysql/mysql-init`
- Restart the `mysqld` service
- Remove the `init-file` line from `/etc/my.cnf`
- Remove `/var/lib/mysql/mysql-init`

LAB

1) Practice resetting the root password on your database using both the secure and insecure method. Verify each time by using a password of “secure” for the secure method, and “insecure” for the insecure method. When you are finished, you can reset the password to whatever you wish.

2) Change your MySQL configuration to not accept any connections from the network, only local connections. Test with a neighbor.

BACKUPS

- All of this data...
- What would happen if someone were to SIGKILL your mysqld process? Or if a hard drive detonated?
- Sad, sad days, that's what!
- So instead of worrying about such things, we *backup* our data. Sometimes weekly, sometimes daily, sometimes constantly..

BACKUPS IN MYSQL

- There are several backup techniques available:
  - `mysqldump`: Tried and true - everything dumped as SQL
  - `mysqlhotcopy`: For MyISAM tables only - faster than `mysqldump`.
  - Filesystem backups/snapshots
  - Replication

MYSQLDUMP

- By far, `mysqldump` is the most common simple backup strategy, as the backup is portable (all just SQL) and fast for most databases.

- `mysqldump` is very easy to use. Often no arguments are necessary besides credentials and the database to dump.

- Once dumped, the backup can be used for replication setup, experimentation, or simply compressed and stored in archive as a point in time backup.

COPYING AND SNAPSHOTS

- Backups can also be made by copying data files. If the database is being stored on a filesystem or device that supports snapshotting, backups can be easily performed with minimal downtime:
  - On the server, execute:

    ```
    FLUSH TABLES WITH READ LOCK
    ```

  - Take a snapshot of the data directory filesystem (or if snapshots are unavailable, start copying data files)
  - Then run:

    ```
    UNLOCK TABLES
    ```

  - Backup from the snapshot at leisure, then destroy it.

REPLICATION

- Replication is common in larger environments, where the cost to run multiple servers is outweighed by the need for instantaneous backups and failover capability.

- Replication involves streaming every change from the master database to all slave databases. In this manner, every database is a complete copy of the master.

- Backups can be easily pulled from slaves without impacting production at all, with the extra advantage that if the master goes down, a slave can take over.

SETTING UP REPLICATION

- Setting up replication is actually quite easy with MySQL. Short form:
  - Enable binary logging; assign unique server id's
  - Stop all writes to Master, grab status and perform backup
  - Restore backup to Slave
  - Point Slave to Master using status information
  - Start replication on Slave. Verify.

BINARY LOGGING AND SERVER ID'S

- To enable the binary log, add a `log-bin` line to `my.cnf`
- Every server must have a unique server identifier specified in the `my.cnf` file:
  - *Master*
    - `[mysqld]`
    - `server-id=5`
    - `log-bin=master`
  - *Slave*
    - `[mysqld]`
    - `server-id=10`

LOCK, STATUS AND BACKUP

- `FLUSH TABLES WITH READ LOCK`
- `SET GLOBAL read_only = ON;`
- `SHOW MASTER STATUS;`
  - Write down log file and log position
- Perform backup
- `UNLOCK TABLES`
- `SET GLOBAL read_only = OFF;`

CREATE SLAVE ACCOUNT ON MASTER

- Create the slave account on the Master:
- ```
  GRANT REPLICATION SLAVE ON *.*
  TO ‘slave’@‘slave-server.mycompany.com’
  IDENTIFIED BY ‘slavepass’;
  ```
- This account will be used by the Slave to connect to the Master to transfer the binary logs.
  ```

RESTORE TO SLAVE

- Restore the Master backup to the Slave server:

- `mysql -u root -p < backup.sql`

SET UP SLAVE

- Point the Slave to the Master:
- `CHANGE MASTER TO`

```
MASTER_HOST = ‘master-server.mycompany.com’,

MASTER_USER = ‘slave’,

MASTER_PASSWORD = ‘slavepass’,

MASTER_LOG_FILE = ‘master.000001’,

MASTER_LOG_POS = 238;
```

START THE SLAVE AND VERIFY

- Start up the Slave threads:
  - `START SLAVE;`
- Verify using:
  - `SHOW SLAVE STATUS\G`

LAB

1) Use `mysqldump` to back up your MovieCollection database. Take a look at the backup file and note how it works. Read the documentation on `mysqldump` and play with some of the options.

2) Once you have a good, solid backup, `DROP` the MovieCollection database and restore from backup. Verify everything is correct.

3) Work with a partner to establish replication from one machine to the other.

```
slideshow.end();
```