How to install/configure MySQL Database

Nurcan Ozturk

Abstract: I explain how I installed MySQL database on my machine heppc6.uta.edu and the web-interfece of MySQL (phpMyAdmin, running on heppc1.uta.edu).

Part I: MySQL database

Introduction: MySQL is a relational database management system. It is a very fast, multi-threaded, multi-user, and robust SQL (Structured Query Language) database server. MySQL is open source software. It is provided by MySQL AB, a Swedish company owned and run by the MySQL founders and main developers.

Since the core of the **Magda**-**Ma**nager for **G**rid-based **Da**ta is a MySQL database (the principal entities of Magda are implemented as MySQL tables), we wanted to install MySQL first to better understand its functionality and features.

Requirements: MySQL needs at least Linux kernel versions 2.0 or higher.

Where to learn more: The MySQL home page, http://www.mysql.com, provides the latest information about MySQL. The most recent version of the MySQL reference manual is available at http://www.mysql.com/documentation/index.html

How to download/install MySQL database:

1. Go to the page http://www.mysql.com to see which platforms are supported and to decide which version and type of distribution to use. I describe here how to install MySQL 3.23 (stable release, recommended) on Linux.

2. Go to downloads and Linux downloads links. The recommended way to install MySQL on Linux is by using an RPM (RetHat Package Manager) file. Download both Server and Client programs rpm files. They will appear in your local directory as
   MySQL-3.23.43-1.i386.rpm
   MySQL-client-3.23.43-1.i386.rpm

3. • To see all the files in an RPM package, run:
   shell> rpm –qpl MySQL-3.23.43-1.i386.rpm

   • To perform the installation, run **(as root):**
   shell> rpm –i MySQL-3.23.43-1.i386.rpm MySQL-client-3.23.43-1.i386.rpm

4. The RPM places data in '/var/lib/mysql'. The RPM also creates the appropriate entries in '/etc/rc.d' to start the server automatically at boot time.

5. Once you install MySQL , the **safe_mysqld** script in /usr/bin directory starts the server with **mysqld** daemon and the **mysql_install_db** script in the same directory creates six tables (user, db, host, table_priv, colums_priv and func) in the **mysql database** which is controlled by the administrator. It is required because it stores all the information about user access privileges.

   The server should always be running. To check if the server is running:

   ```
   shell> /usr/bin/mysqladmin version
   shell> /usr/bin/mysqladmin variables
   ```

6. When you install MySQL database, a **test database** is automatically created that any user has the all privileges on it to try things out. Type:

   ```
   shell> mysql –u test
   mysql>
   ```

   Since anything created in the test database can be removed by anyone else with access to it, it is always good idea to set up a root (administrator) password and open user accounts and grant privileges to users. To do this:

   ```
   shell> mysql –u root –p
   Enter password: skip by enter
   mysql> SET PASSWORD FOR root=PASSWORD('new_password');
   ```

   MySQL commands should be followed by a semicolon. You can connect to MySQL as root (administrator), even from another host . Type:

   ```
   shell> mysql –u root –h heppc6 –p
   Enter password:******
   mysql>
   ```

7. Now you can create user accounts (as administrator):

   ```
   mysql> GRANT  USAGE ON database_name.*  TO  utahep@localhost
          IDENTIFIED BY  'utahep';
   ```

   and grant privileges to users:

   ```
   mysql> GRANT  SELECT ON database_name.* TO  utahep@localhost
          IDENTIFIED BY  'utahep';
   ```

   This means that you created a user 'utahep' who can connect with a password 'utahep' from the localhost having only read privileges (means selective privileges) on the database which is 'database_name'. If you want to grant full

privileges (create, drop, grant, select, insert, update, delete, index, alter, etc.) to the user on the database:

```
mysql> GRANT  ALL ON database_name.*  TO  utahep@localhost
       IDENTIFIED BY   'utahep';
```

If you want the user to connect from any other host you must issue GRANT statements for both utahep@localhost and utahep@"%" ("%" means any host name).

Do not give normal users read access to the tables in the **mysql database.** Passwords are stored encrypted in the **mysql.user** table in this database such that any user can read them and potentially unencrypt the passwords.

8. The user 'utahep' can now connect to MySQL from any host (no need for '–h heppc6 ' when connecting from localhost):

```
shell> mysql –u utahep –h heppc6 –p
Enter password:******
mysql> SHOW databases;
mysql> USE database_name            (no need for semicolon)
```

create a table, and load data into the table in the database (which is database_name) that the user has privileges in to do so. It is very easy to create a table, you just write a text file 'table_name.txt' containing one record per line, with values seperated by tabs and given in the order in which the columns will be listed in the table. I do not explain details here, since it is easier to do these things on the web, after you install the phpMyAdmin program. See the instructions below.

Part II: phpMyAdmin

Introduction: The phpMyAdmin program (the web-interface of MySQL) is used to handle the administration of MySQL over the WWW.

PhpMyAdmin can administer a whole MySQL server (needs root user/administrator) but also a single database. To accomplish the latter you will need a properly configured MySQL user who has only read/write privileges on the desired database. Currently phpMyAdmin can:

- create and drop databases
- create, copy, drop and alter tables
- delete, edit and add fields
- execute any SQL statement, including batch-queries
- manage keys on fields
- load text files into tables
- create and read dumps of tables

- export and import data to CVS values
- communicate in more than 20 different languages

Requriements: PHP3 or PHP4 (PHP > 3.0.8), MySQL , a web browser.
(**PHP** is a tool that lets you create dynamic web pages. PHP-enabled web pages are treated just like regular HTML pages and you can create and edit them the same way you normally create regular HTML pages. For more information see **http://php.net**)

How to download/install phpMyAdmin 2.2.1:

1. Go to the web page **http://phpadmin.sourceforge.net**. Download the appropriate file for your system (tar.bz2, tar.gz or zip files). Ours is phpMyAdmin-2.2.1-php.tar.gz.

2. Untar the distribution :
   shell> tar xzvf phpMyAdmin_2.2.1-php.tar.gz

3. Open the file **config.inc.php** in your favorite editor and change the values for host, user, and password to fit your environment. Read Documentation.txt file for the details.

   Make sure that the configuration file does not have any non-ascii characters (like EOF). Initially our server gave strange error messages because of these extra characters.

4. Open the file **www.your-host.com/<your-install-dir>/index.php** in your browser. PhpMyAdmin should now display a welcome screen and your database, or a login dialog if using advanced authentication.
   Ours is htpp://heppc1.uta.edu/kaushik/phpadmin/index.php

5. We use Advanced Authentication:
   - phpMyAdmin needs a standard_user that has only the SELECT privileges on the mysql.db and mysql.user tables. You must specify the details for the standard_user in the config.inc.php file. If you need help read the Documentation.txt file. To create this user, connect to MySQL:

     shell> mysql –u root –p mysql
     Enter password:******
     mysql> GRANT USAGE ON mysql.* TO ‘standard_user’@localhost
     IDENTIFIED BY ‘password’;
     mysql>GRANT SELECT ON mysql.user TO ‘standard_user’@localhost;
     mysql> GRANT SELECT ON mysql.db TO ‘standard_user’@localhost;

- Then each of the other users should be granted of a set of privileges on a set of particular databases but should not have any global privileges (administrator privileges: reload, shutdown, etc.).

The Gridview program which displays the U.S. ATLAS Grid Testbed status, written by Kaushik De, is now filling the table TestBedArchive20 in the Gridview database every 30 minutes. Anybody can take a look using the same user/password currently set for the standard ATLAS web user/password on
htpp://heppc1.uta.edu/kaushik/phpadmin/index.php .

Part III: Replication in MySQL

Introduction: It is always a good idea to replicate all your databases in the case of a machine failure on the primary MySQL server. A MySQL server installation on one machine can act as the master, while the second MySQL server on another machine can act as the slave. The master server keeps a binary log of updates and an index file to the binary logs to keep track of binary rotation. Another benefit of using replication is that you can get live backups of your system by doing a backup on a slave instead of doing it on a master.

Requirements: The second MySQL server must be installed and running on another machine (it is 'heppc5.uta.edu' in the following).

How to set up replication:

1. Set up a special replication user on the master with only FILE privilege and permission to connect from all the slaves.

   ```
   shell> mysql –u root –p mysql
   Enter password:******
   mysql> GRANT FILE ON *.* TO repl@"%" IDENTIFIED BY 'password';
   ```

   You created a user named 'repl' which can access your master from any host (this is ideally correct but it did not work out until I specified the name of master and the slave machine as well, so I repeated the above for repl@heppc6.uta.edu and repl@heppc5.uta.edu).

2. Shut down MySQL server on the master.

   ```
   shell> /usr/bin/mysqladmin –u root –p shutdown
   Enter password:******
   ```

3. Go to your data directory (/var/lib/mysql) where you store all the data files and create a snapshot of all the data on your master server. To do this:

   ```
   shell> tar –cvf /tmp/mysql-snapshot.tar /var/lib/mysql
   ```

4. Create a ‘my.cnf’ file on the master in the /etc directory and write the following into this file:

   [mysqld]
   log-bin=heppc6-bin.001
   log-bin-index=heppc6-bin.index
   server-id=1
   binlog-do-db= name of the database which master should log updates
   to the binary log
   binlog-ignore-db=name of the database which master should not log updates
   to the binary log

   You usually choose ‘1’ as the server-id of the master and ‘2, 3 ,..’ as that of the slaves.

5. Restart the server on master:

   shell> /usr/bin/safe_mysqld --log-bin

   This will create a binary log file (for instance, heppc6-bin.001 above) containing all SQL commands that update data and a binary index file (for instance, heppc6-bin.index above) to be able to know which different binary log files have been used in the /var/lib/mysql directory. Whenever you restart the server, mysqld will append an extension to the binary log filename and all used binary log files will be stored in the binary index file.

6. Create a ‘my.cnf’ file on the slave in the /etc directory and write the following into this file:

   [mysqld]
   master-host=heppc6.uta.edu
   master-user=repl
   master-password=password
   master-port=3306
   server-id=2
   replicate-do-db= name of the database which you want to replicate
   replicate-ignore-db=name of the database which you do not want to replicate

7. Copy the snapshot data into your data directory (/var lib/mysql) on the slave and untar it:

   shell> tar –xvf mysql-snapshot.tar

8. Restart the slave:

   shell> /usr/bin/safe_mysqld &

Now, the slave should connect to the master and catch up on any updates which happened since the snapshot was taken. Once the slave is replicating, you will find a file called master.info in the /var/lib/mysql directory on the slave. This file is used by the slave to keep track of how much of the master's binary log has processed.

You can also check the status of the master and the slave in the mysql database:

```
shell>  mysql –u root –p mysql
Enter password:******
mysql> show master status;
mysql> show slave status;
```

Please read the MySQL reference manual for more information on replication options.

Please let me know of any questions.

Acknowledgements: Many thanks to Mark Sosebee, Kaushik De and Patrick Mcguigan for their help.