

mysql e le Web app

- Spesso *mysql* è la prima scelta come database di supporto per una web app: qui si considerano i framework SpringBoot (Java) e Laravel (PHP)
- L'intento di queste brevi note è tutt'altro che esaustivo e l'argomento è (tecnicamente) complesso e ricco di difficoltà
- La soluzione più semplice è installare un pacchetto completo, come XAMPP (Apache, MariaDB, PHP, Perl, phpMyAdmin), disponibile per Windows, ma anche OSX e Unix
- Sotto Unix, però, è probabile che ci si affidi al gestore di pacchetti standard (*brew* per OSX, *apt* per Debian, etc.) per la *distribuzione* e che questa abbia cura dei dettagli
- Vi sono però comunque, per l'amministratore di un sistema Unix, numerosi trabocchetti e altrettanti *caveat*, che qui si è cercato di organizzare e raccogliere
- Per una visione più ampia e sistematica, dal Web, in italiano:

[http://guide.debianizzati.org/index.php/Installare_un_ambiente_LAMP: Linux, Apache2, SSL, MySQL, PHP5 - Stretch](http://guide.debianizzati.org/index.php/Installare_un_ambiente_LAMP:_Linux,_Apache2,_SSL,_MySQL,_PHP5_-_Stretch)

Avviare mysql come servizio

- *Mysql* o il suo fork *mariadb* girano come servizi
- In ambiente Windows conviene installare il pacchetto XAMPP (o analogo) e lanciare *mariadb* all'interno di esso
- In ambiente Unix, *mariadb* potrebbe essere stato installato da shell:

```
$ brew install mariadb # OSX
```

```
$ apt install mariadb-server mariadb-client # Ubuntu Linux
```

(in realtà installare mariadb su Ubuntu potrebbe avere dei prerequisiti, vedi:
<https://computingforgeeks.com/install-mariadb-10-on-ubuntu-18-04-and-centos-7/>)

- Sempre da shell, si avvia il servizio (daemon) con:

```
$ brew services run mariadb # OSX, stop anziché run per spegnerlo
```

```
$ sudo systemctl start mariadb # Ubuntu Linux, stop anziché start per spegnerlo
```

(è possibile vi siano difficoltà all'avvio come utente comune e serve *sudo*)

- Il servizio (daemon) *mysql* risponde (per default) sulla porta 3306 o anche, in Unix e secondo configurazione, su socket "locale" del kernel
 - più avanti (*phpmyadmin*) si vedrà che quest'aspetto può risultare critico

La shell *mysql*

- Varie shell grafiche (spesso commerciali) e l'applicazione web GUI *phpmyadmin* consentono di interagire con il servizio *mysql*
- Lo strumento più di base è però una shell testuale, *mysql*, avviata dall'interprete dei comandi (*bash* o *command.com* di Windows)

```
$ sudo mysql -u root
```

```
Welcome to the MariaDB monitor. Commands end with ; or \g. Your MariaDB connection id is 195
```

- L'utente *root* (di *mysql*) è predefinito (nei file dell'installazione)
- Qui si immagina che *root* (di *mysql*) non abbia password
 - NB: in questo caso, sotto Unix, l'accesso con *mysql* sarà consentito solo al super-user Unix *root* (da cui il comando *sudo* nell'esempio sopra).
- Vedremo oltre come definire altri utenti, con password
- Per l'utente *pippo*, con password *segreto*, il login si fa con le opzioni *-u* e *-p*, con password che segue (senza spazi) *-p*, oppure scritta al prompt:

```
$ mysql -u pippo -psegreto # oppure
```

```
$ mysql -u pippo -p
```

```
Enter password:
```

La root di mysql e quella di Unix (saltare)

- Come detto, non vanno confusi l'utente *root* di *mysql* (un elemento della tabella degli utenti del DB) e il (super-)user *root* di Unix
 - hanno lo stesso nome, ma corrispondono a concetti diversi
- Tuttavia, le versioni più recenti di *mysql* sono configurate per consentire l'accesso come *root* di mysql solo al super-user di Unix:

```
$ mysql -uroot
ERROR 1698 (28000): Access denied for user 'root'@'localhost'
$ sudo mysql -u root
Password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
...
```

- è possibile, ma forse inutile, rimuovere questa limitazione, si veda:
<https://askubuntu.com/questions/766334/cant-login-as-mysql-user-root-from-normal-user-account-in-ubuntu-16-04/784347#784347>
- Potrebbe anche risultare utile la pagina:
<https://serverfault.com/questions/483964/mysql-root-problems-access-denied-for-root-user>
- In ogni caso, il materiale in questa slide non dovrebbe servirvi, salvo complicazioni

La password per *root*

- L'utente *mysql* predefinito *root* può avere o no password
- Alcune installazioni (p.es. Debian) introducono questa password, altre non lo prevedono
 - per Debian si può fare riferimento a: <https://wiki.debian.org/MySQL>
- L'uso di password per *root* (NB: *root* di *mysql*) è raccomandato in produzione, ma fonte di problemi per i primi passi
- Se si vuole eliminare la password di root introdotta per sbaglio o, in generale, ri-inizializzare i metadati di *mysql*, si veda:
<https://dev.mysql.com/doc/refman/8.0/en/data-directory-initialization.html>
 - In sintesi, si useranno i comandi (dalla dir di base di *mysql*):

```
$ bin/mysql --initialize-insecure --user=mysql # Unix, ma non supportato da ogni distribuzione ...
```

```
> bin\mysql --initialize-insecure --console # Windows
```

- Oppure, un ottimo [tutorial](https://www.digitalocean.com/community/tutorials/how-to-reset-your-mysql-or-mariadb-root-password):

La shell *mysql* – autocompletion

mysql offre la *autocompletion* con tasto "Freccia-su", rispetto alla *history* dei comandi

Anche il tasto *Tab* autocompleta: i comandi *mysql*, gli statement *SQL* e i nomi di DB, tabelle, attributi;

- ciò accade però solo se si seleziona un DB di default (qualsiasi) con *use*:

```
$ mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g. Your MariaDB connection id is 195...
MariaDB [(none)]> use mysql; -- mysql è un database "di servizio" (contiene p.es. gli utenti)
Reading table information for completion of table and column names.
You can turn off this feature to get a quicker startup with -A. Database changed
MariaDB [mysql]> -- notare il cambio di prompt, che ora mostra il nuovo DB di default mysql
```

- Inoltre, come si capisce dalla risposta a *use mysql*, l'autocompletamento è il default, salvo che si avvii *mysql* con *-A* (ovvero *--no-auto-rehash*)

```
$ mysql -u root --no-auto-rehash # autocompletion disabilitata, la attiveremo "a mano"
MariaDB [(none)]> use mysql; -- occorre prima selezionare un (qualsiasi) DB di default
MariaDB [mysql]> rehash;      -- rehash attiva la autocompletion. rehash si può anche porre in /etc/my.cnf
```

- Riferimenti: <https://www.cyberciti.biz/tips/mysql-auto-completion-for-database-table-names.html>
<https://dev.mysql.com/doc/mysql-shell/en/mysql-shell-autocompletion.html>

La shell *mysql* – Aggiungere utenti

- Ora aggiungiamo un nuovo utente *admin* con password *admin*, concedendogli tutti i privilegi (*admin* avrà sostanzialmente i poteri di *root*):

```
MariaDB [(none)]> CREATE USER 'admin'@'localhost' IDENTIFIED BY 'admin'; -- user e password
Query OK, 0 rows affected (0.004 sec)

MariaDB [(none)]> GRANT ALL ON *.* TO 'admin'@'localhost' WITH GRANT OPTION;
Query OK, 0 rows affected (0.005 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.005 sec)

MariaDB [(none)]> quit;

$ mysql -u admin -padmin
MariaDB [(none)]> use mysql;
MariaDB [mysql]>
```

- Come si vede, anche un utente Unix comune può accedere a *mysql* come utente *admin*, con gli stessi privilegi di *root* di *mysql*
- Come si ricorderà, operare come *root* di *mysql* richiede invece l'accesso con *sudo*)

Caveat: autenticazione e "wildcard" % in mysql (saltare)

- Nel creare un utente *mysql*, si potrebbe essere tentati di assegnargli l'host % (), ragionando che *localhost* rientra nel "wildcard" %

```
MariaDB> CREATE USER pippo'@%' IDENTIFIED BY pippo'; -- %è qualsiasi host
```

- in realtà, questo rischia di creare problemi, anche a causa del fatto che, nella configurazione standard "out-of-the-box" di *mysql*, per l'host *localhost* è previsto un utente senza nome ("", altro wildcard) ma senza privilegi
- si tratta di questioni piuttosto astruse, che non vale la pena di approfondire qui, le citiamo solo a mo' di *caveat* per possibili problemi
- se se ne verificassero, si può fare riferimento al web, per esempio si veda: <https://stackoverflow.com/questions/11634084>

mysql: file di configurazione *my.cnf*

Senza pretesa di esaustività, ecco a destra un esempio di file di configurazione.

Il file di configurazione *my.cnf* può essere in *.../etc*, nella *home* dell'utente o in altre posizioni specificate nella documentazione.

```
# File my.cnf
# This group is read both both by the client and the server
# use it for options that affect everything
[client-server]

# default: socket in /tmp/mysql.sock
socket=/usr/local/var/mysql/mysql.sock

# option group for mysql text client
[mysql]
auto-rehash

# option group for any client
[client]

# option group for server
[mysqld]

# include all files from the config directory
!includedir /usr/local/etc/my.cnf.d
```

In generale, su quest'argomento, si consulti la documentazione online: <https://dev.mysql.com/doc/refman/en/option-files.html>

Mysql e *phpmyadmin*

- *phpmyadmin* è una web app che fornisce una GUI per interagire con il servizio *mysql*
- è ovviamente più *user-friendly* della shell solo testo *mysql*
- con Windows è già parte del pacchetto XAMPP
- sotto Linux o OSX si può installare come pacchetto; richiede siano installati *mysql* e *Apache*, nonché un intervento sulla configurazione di Apache:

```
$ brew info phpmyadmin # OSX, similmente per Linux
```

To enable phpMyAdmin in Apache, add the following to httpd.conf and restart Apache:

```
Alias /phpmyadmin /usr/local/share/phpmyadmin
<Directory /usr/local/share/phpmyadmin/>
    Options Indexes FollowSymLinks MultiViews
    AllowOverride All
    <IfModule mod_authz_core.c>
        Require all granted
    </IfModule>
    <IfModule !mod_authz_core.c>
        Order allow,deny
        Allow from all
    </IfModule>
</Directory>
```

Then open <http://localhost/phpmyadmin>

The configuration file is `/usr/local/etc/phpmyadmin.config.inc.php`

Phpmyadmin: collegamento a *mysql*

Come detto, il servizio *mysql* ha due modalità di accesso:

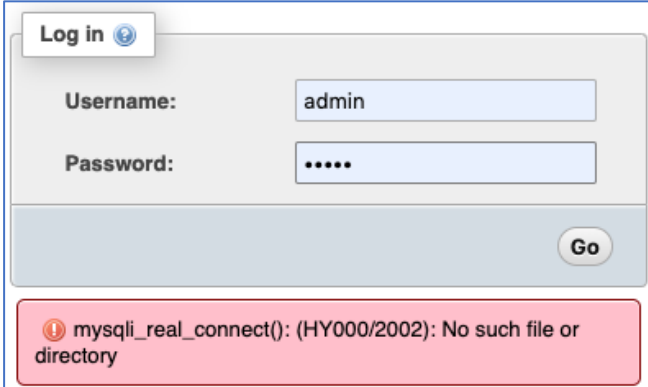
- per *localhost*, via una socket locale del kernel, p.es. `.../var/mysql/mysql.sock`
- da altri host IP (compreso 127.0.0.1), attraverso il port 3306

Se *phpmyadmin* è stato installato "a mano" (p.es. in Unix), anziché con pacchetti come XAMPP, è possibile che sia pre-configurato per collegarsi a *localhost*.

In questo caso, il login non sarà possibile (sarebbero richiesti altri interventi)

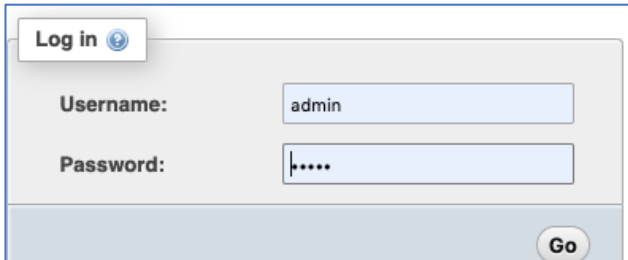
Conviene invece intervenire sul file di configurazione `.../etc/phpmyadmin.config.inc.php` come mostrato qui a fianco (commentando il rigo con *localhost*)

Così *phpmyadmin* si collegherà a *mysql* via port 3306 e il login sarà possibile



The screenshot shows the phpMyAdmin login interface. At the top is a "Log in" button with a user icon. Below it are two input fields: "Username:" with the value "admin" and "Password:" with masked characters ".....". A "Go" button is at the bottom right. A red error message box at the bottom states: "mysqli_real_connect(): (HY000/2002): No such file or directory".

```
// $cfg['Servers'][$i]['host'] = 'localhost';  
$cfg['Servers'][$i]['host'] = '127.0.0.1';
```



This screenshot shows the same phpMyAdmin login interface as above, but without the error message. The "Username:" field contains "admin" and the "Password:" field contains masked characters ".....". The "Go" button is visible at the bottom right.

Fonti: <https://stackoverflow.com/questions/41881123> <https://askubuntu.com/questions/1105970>

phpMyAdmin: file di configurazione (in, o sotto, .../etc)

Senza alcuna pretesa di esaustività, si mostrano qui le differenze tra il file di configurazione di default e quello effettivamente impiegato:

```
/usr/local/etc $ diff phpmyadmin.config.inc.php.default phpmyadmin.config.inc.php
17c17
< $cfg['blowfish_secret'] = ''; /* YOU MUST FILL IN THIS FOR COOKIE AUTH! */
---
> $cfg['blowfish_secret'] = 'abcdefghijklmnopqrstuvwxyz012345'; /* YOU MUST FILL IN THIS FOR COOKIE AUTH! */
31c31,32
< $cfg['Servers'][$i]['host'] = 'localhost';
---
> //$cfg['Servers'][$i]['host'] = 'localhost';
> $cfg['Servers'][$i]['host'] = '127.0.0.1';
33c34
< $cfg['Servers'][$i]['AllowNoPassword'] = false;
---
> $cfg['Servers'][$i]['AllowNoPassword'] = true;
75a77
> $cfg['TempDir'] = '/tmp';
```

phpMyAdmin stand-alone

- phpMyAdmin è in effetti un'applicazione PHP
- si può quindi scaricare come .zip dal sito o via GitHub o, essendo appunto un app PHP, si può installare con *composer*, come qui:

```
~ $ composer create-project phpmyadmin/phpmyadmin
Installing phpmyadmin/phpmyadmin (4.8.5)
  - Installing phpmyadmin/phpmyadmin (4.8.5): Downloading (100%)
Created project in /Users/gp/phpmyadmin
...
~ $ cd /Users/gp/phpmyadmin
phpmyadmin $ composer update # aggiorna phpmyadmin
...
```

- ora si può eseguire *phpmyadmin*, indipendentemente da Apache, come app PHP **stand-alone**:

```
~ phpmyadmin $ php -S localhost:7777 # o altro port a piacere
```

- anche in questo caso conviene adattare il file *config.inc.php* per consentire l'interazione con *mysql* attraverso 127.0.0.1 e il port 3306

Login di *phpmyadmin*

- Il login con utente *root* e senza password probabilmente non funzionerà (come non è permesso con la shell testuale *mysql*), cf:
<https://askubuntu.com/questions/763336/cannot-enter-phpmyadmin-as-root-mysql-5-7>
- Dovrebbe invece poter entrare l'utente *admin* (con password *admin*) già creato con la shell *mysql*, con i passi che qui si riportano, per chiarezza:



```
MariaDB [(none)]> CREATE USER 'admin'@'localhost' IDENTIFIED BY 'admin';  
Query OK, 0 rows affected (0.004 sec)  
  
MariaDB [(none)]> GRANT ALL ON *.* TO 'admin'@'localhost' WITH GRANT OPTION;  
Query OK, 0 rows affected (0.005 sec)  
  
MariaDB [(none)]> FLUSH PRIVILEGES;  
Query OK, 0 rows affected (0.005 sec)
```

- Il sistema stesso chiederà poi, al login, di attivare un DB di servizio (*phpmyadmin*):

⚠ The phpMyAdmin configuration storage is not completely configured, some extended features have been deactivated. [Find out why.](#)
Or alternately go to 'Operations' tab of any database to set it up there.

si può procedere con un clic su [Find out why](#)

Spring Boot: preparare il Database

- Prima di poter avviare l'applicazione dobbiamo creare e collegare un database al nostro progetto.
- Creiamo quindi un nuovo database *esempio_db* e (se non lo si era già fatto) un nuovo utente con privilegi per *esempio_db*
- Non serve creare alcuna tabella con i dati, per ora – Spring Boot si occuperà di creare tabelle e relazioni di *esempio_db* automaticamente utilizzando le classi Java che creeremo.

```
$ mysql -u root    # sotto Unix, potrebbe essere necessario premettere sudo (come spiegato prima)
Welcome to the MariaDB monitor.  Commands end with ; or \g. Your MariaDB connection id is 195...

MariaDB [(none)]> rehash      -- per autocompletion! Meglio ancora in /etc/my.cnf

MariaDB [(none)]> CREATE DATABASE esempio_db;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> CREATE USER 'pippo'@'localhost' IDENTIFIED BY 'pippo';    -- user e password
Query OK, 0 rows affected (0.004 sec)

MariaDB [(none)]> GRANT ALL ON esempio_db.* TO 'pippo'@'localhost';    -- occhio (anche sopra) ai 4 apici
Query OK, 0 rows affected (0.005 sec)
# Attenti a introdurre utenti per l'host "wildcard" %, che significa "tutti gli host", ma non localhost !
# 'pippo'@'localhost' e 'pippo'@'%' sono utenti diversi (con diritti e password possibilmente diversi)!
# vedi prossima slide
```

mysql e Laravel

- Occorre creare "a mano", con *phpmyadmin* o *mysql*, il database di supporto per l'app; p.es. *prova_db* per l'app *prova*

```
$ mysql -u root # sotto Unix, potrebbe essere necessario premettere sudo (come spiegato prima)
MariaDB [(none)]> CREATE DATABASE prova_db;
Query OK, 1 row affected (0.001 sec)
```

- Per creare le tabelle si userà l'ambiente Laravel (non discusso qui)
- Il cuore dell'interazione tra *mysql* e *Laravel* è il file *.env* dell'app Laravel (generata automaticamente nella directory *prova*)

```
~/prova $ grep DB_ .env
grep DB .env
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel
DB_USERNAME=root
DB_PASSWORD=
```

```
DB_DATABASE=prova_db
DB_USERNAME=admin
DB_PASSWORD=admin
```

- Va modificato il nome del DB in *prova_db*
- user *root* senza password funziona se *mysql* non restringe questo accesso al super-user (come però accade ormai per default)
- potrebbe essere **meglio**, se lo si è definito, (cf. slide precedenti) usare un utente, p.es. *admin*, con i privilegi di *root*