

MySQL on Linux (Beginners Tutorial)

Mokhtar Ebrahim

18–22 minutes

In this post, we will cover many aspects of MySQL on Linux. First of all, how to install it, how to perform basic CRUD operations, how to import & export data, play with the MySQL engine itself such as setting the root user password, and much more.

MySQL is one of the most popular relational database engines in the world. It has earned its fame for being open source and quite stable.

It is also compatible with most known programming languages. Of course, it's possible to install it and use it on most [Linux distributions](#) that exist, for example, Ubuntu and CentOS. Let's start.

Install MySQL on Linux Ubuntu and CentOS

The first step to use MySQL on Linux is obviously to install it in our system.

Ubuntu and CentOS are two of the most used Linux distributions. In the case of Ubuntu, it is quite popular among novices who come from other operating systems like Windows.

Recently, Canonical, the company behind the development of Ubuntu, has been profiling Ubuntu to be used on servers.

On the other hand, CentOS was born as a clone of Red Hat Enterprise Linux, and it has always been used in the servers area as its main objective.

So if you use Ubuntu, you can install MySQL by typing this command in a terminal session:

\$ sudo apt install mysql-server

```
angelo@virtual:~$ sudo apt install mysql-server
[sudo] password for angelo:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libaio1 libcgf-fast-perl libcgf-pm-perl libencode-locale-perl libevent-core-2.1-6 libfcgi-perl libhtml-parser-perl libhtml-tagset-perl libhtml-template-perl libhttp-date-perl
  libhttp-message-perl libio-html-perl liblwp-mediatypes-perl libtimedate-perl liburi-perl mysql-client-5.7 mysql-client-core-5.7 mysql-common mysql-server-5.7 mysql-server-core-5.7
Suggested packages:
  libdata-dump-perl libipc-sharedcache-perl libwww-perl mailx tinyc
The following NEW packages will be installed:
  libaio1 libcgf-fast-perl libcgf-pm-perl libencode-locale-perl libevent-core-2.1-6 libfcgi-perl libhtml-parser-perl libhtml-tagset-perl libhtml-template-perl libhttp-date-perl
  libhttp-message-perl libio-html-perl liblwp-mediatypes-perl libtimedate-perl liburi-perl mysql-client-5.7 mysql-client-core-5.7 mysql-common mysql-server mysql-server-5.7
mysql-server-core-5.7
0 upgraded, 21 newly installed, 0 to remove and 54 not upgraded.
Need to get 21.1 MB of archives.
After this operation, 162 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Then, after typing the user’s password, it will start the download and subsequent installation.

On the other hand, CentOS does not include MySQL by default in its repositories. However, it is easy to add the MySQL repository to install it. First, switch to the root user and then add the repository.

\$ su

\$ yum install https://dev.mysql.com/get/mysql80-community-release-el7-2.noarch.rpm

```
[root@localhost angelo]# yum install https://dev.mysql.com/get/mysql80-community-release-el7-2.noarch.rpm
Loaded plugins: fastestmirror
mysql80-community-release-el7-2.noarch.rpm                                     | 25 kB  00:00:00
Examining /var/tmp/yum-root-SLrIDK/mysql80-community-release-el7-2.noarch.rpm: mysql80-community-release-el7-2.noarch
Marking /var/tmp/yum-root-SLrIDK/mysql80-community-release-el7-2.noarch.rpm to be installed
Resolving Dependencies
--> Running transaction check
--> Package mysql80-community-release.noarch 0:el7-2 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

Package Arch Version Repository Size
Installing:
mysql80-community-release noarch el7-2 /mysql80-community-release-el7-2.noarch 31 k
Transaction Summary
Install 1 Package
Total size: 31 k
Installed size: 31 k
Is this ok [y/d/N]:
```

Type Y to start the process.

Now, you can install MySQL on CentOS. Just type the following command:

\$ yum install mysql-community-server

Package	Arch	Version	Repository	Size
Installing:				
mysql-community-libs	x86_64	8.0.17-1.el7	mysql80-community	3.0 M
replacing mariadb-libs.x86_64 1:5.5.60-1.el7_5	x86_64	8.0.17-1.el7	mysql80-community	2.1 M
mysql-community-server	x86_64	8.0.17-1.el7	mysql80-community	415 M
Installing for dependencies:				
mysql-community-client	x86_64	8.0.17-1.el7	mysql80-community	32 M
mysql-community-common	x86_64	8.0.17-1.el7	mysql80-community	589 k
net-tools	x86_64	2.0-0.24.20131004git.el7	base	386 k
perl	x86_64	4:5.16.3-294.el7_6	updates	8.0 M
perl-Carp	noarch	1.26-244.el7	base	19 k
perl-Encode	x86_64	2.51-7.el7	base	1.5 M
perl-Exporter	noarch	5.68-3.el7	base	28 k
perl-File-Path	noarch	2.09-2.el7	base	26 k
perl-File-Temp	noarch	0.23.01-3.el7	base	56 k
perl-Filter	x86_64	1.49-3.el7	base	76 k
perl-Getopt-Long	noarch	2.40-3.el7	base	56 k
perl-HTTP-Tiny	noarch	0.033-3.el7	base	38 k
perl-PathTools	x86_64	3.40-5.el7	base	82 k
perl-Pod-Escapes	noarch	1:1.04-294.el7_6	updates	51 k
perl-Pod-Perldoc	noarch	3.20-4.el7	base	87 k
perl-Pod-Simple	noarch	1:3.28-4.el7	base	216 k
perl-Pod-Usage	noarch	1.63-3.el7	base	27 k
perl-Scalar-List-Utils	x86_64	1.27-248.el7	base	36 k
perl-Socket	x86_64	2.010-4.el7	base	49 k
perl-Storable	x86_64	2.45-3.el7	base	77 k
perl-Text-ParseWords	noarch	3.29-4.el7	base	14 k

perl-Time-HiRes	x86_64	4:1.9725-3.el7	base	45 k
perl-Time-Local	noarch	1.2300-2.el7	base	24 k
perl-constant	noarch	1.27-2.el7	base	19 k
perl-libs	x86_64	4:5.16.3-294.el7_6	updates	688 k
perl-macros	x86_64	4:5.16.3-294.el7_6	updates	44 k
perl-parent	noarch	1:0.225-244.el7	base	12 k
perl-podlators	noarch	2.5.1-3.el7	base	112 k
perl-threads	x86_64	1.87-4.el7	base	49 k
perl-threads-shared	x86_64	1.43-6.el7	base	39 k

Transaction Summary

Install 3 Packages (+30 Dependent packages)

Total download size: 465 M

Is this ok [y/d/N]:

Check if MySQL is installed

It is possible that during the installation on both systems that something went wrong. So it is a good idea to check if MySQL is installed or not.

On both systems (Ubuntu & CentOS), if you want to check if MySQL is correctly installed, you can show the version its version using the following command:

```
$ mysql --version
```

```
angelo@virtual:~$ mysql --version
mysql Ver 14.14 Distrib 5.7.27, for Linux (x86_64) using EditLine wrapper
angelo@virtual:~$
```

By showing the current version of MySQL, that means MySQL is correctly installed and ready to use.

Where does MySQL installed on Linux?

Generally, each Linux distribution has its way of unpacking each of the programs we install. However, Linux has some conventions.

Normally, MySQL binaries are in the following locations:

```
/usr/bin
```

On the other hand, it is good to know where the databases are stored in the system. Normally the location is as follows:

```
/var/lib/mysql/
```

Finally, if you want to know where all the installed MySQL files are located, you can run the following command:

For Ubuntu:

```
$ dpkg -L mysql-server
```

```
angelo@virtual:~$ dpkg -L mysql-server
```

```
angelo@virtual:~$ dpkg-query -W -f='${Package}\n${Architecture}\n${Version}\n${FileList}\n' mysql-server
/
/usr
/usr/share
/usr/share/doc
/usr/share/doc/mysql-server
/usr/share/doc/mysql-server/copyright
/usr/share/doc/mysql-server/NEWS.Debian.gz
/usr/share/doc/mysql-server/changelog.Debian.gz
angelo@virtual:~$
```

Where is the MySQL configuration file (my.cnf)?

MySQL runs as a system service. In Unix family systems, services are configured through text files. The my.cnf file is the file where all MySQL configurations are defined.

If you use Ubuntu, you can find the MySQL configuration file in the following path:

`/etc/mysql/my.cnf`

On the other hand, if you use CentOS, the path is as follows:

`/etc/my.cnf`

One thing to keep in mind is that the configuration file may include other configuration files. This is to separate the configurations and make them more understandable.

If you want to see its content, you can use the [cat command](#):

`$ cat /etc/mysql/my.cnf`

```
angelo@virtual:~$ cat /etc/mysql/my.cnf
#
# The MySQL database server configuration file.
#
# You can copy this to one of:
# - "/etc/mysql/my.cnf" to set global options,
# - "~/.my.cnf" to set user-specific options.
#
# One can use all long options that the program supports.
# Run program with --help to get a list of available options and with
# --print-defaults to see which it would actually understand and use.
#
# For explanations see
# http://dev.mysql.com/doc/mysql/en/server-system-variables.html
#
# * IMPORTANT: Additional settings that can override those from this file!
#   The files must end with '.cnf', otherwise they'll be ignored.
#

!includedir /etc/mysql/conf.d/
!includedir /etc/mysql/mysql.conf.d/
angelo@virtual:~$
```

You can edit its content using a text editor such as **nano** or **vim**. In the case of Ubuntu, most of the configurations are in

`/etc/mysql/conf.d/`

Remember that if you are going to edit it, create a backup of the file first.

Start MySQL & enable on startup

Now MySQL is installed but still not running. So how to start MySQL?

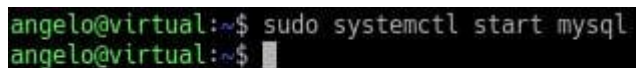
Since previous versions of Linux, the *systemd* is in charge of managing the system services. As we may already know, MySQL is a system service.

To start running the MySQL service, type the following command in the terminal:

```
$ sudo systemctl start mysql
```

If you are using CentOS, use this command as the root user:

```
$ systemctl start mysqld
```



```
angelo@virtual:~$ sudo systemctl start mysql
angelo@virtual:~$
```

As we can see, no output. That means the service started successfully.

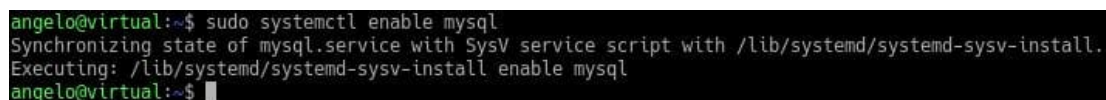
If you reboot the system, you will notice that MySQL is not running. So we need to enable it to run on system startup.

To do this, use the `systemctl` command but add the `enable` option:

```
$ sudo systemctl enable mysql
```

In the case you are using CentOS:

```
$ systemctl enable mysqld
```



```
angelo@virtual:~$ sudo systemctl enable mysql
Synchronizing state of mysql.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable mysql
angelo@virtual:~$
```

Now, if you reboot the system, you will note that MySQL is running. Let's see how to ensure that.

Check if MySQL is running

You can check if MySQL is running or not using the systemctl command but adding the status option:

```
$ sudo systemctl status mysql
```

For CentOS, the command should run as root user and as follows:

```
$ systemctl status mysqld
```

```
angelo@virtual:~$ sudo systemctl status mysql
● mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2019-08-22 00:14:54 UTC; 26min ago
     Main PID: 14044 (mysqld)
        Tasks: 27 (limit: 660)
      CGroup: /system.slice/mysql.service
              └─14044 /usr/sbin/mysqld --daemonize --pid-file=/run/mysqld/mysqld.pid

Aug 22 00:14:53 virtual systemd[1]: Starting MySQL Community Server...
Aug 22 00:14:54 virtual systemd[1]: Started MySQL Community Server.
angelo@virtual:~$
```

The command generates a screen output with a green signal indicating that the service is running.

Otherwise, it appears in gray, indicating that it is inactive or stopped.

```
angelo@virtual:~$ sudo systemctl status mysql
● mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
   Active: inactive (dead) since Thu 2019-08-22 00:42:55 UTC; 2s ago
     Main PID: 14044 (code=exited, status=0/SUCCESS)

Aug 22 00:14:53 virtual systemd[1]: Starting MySQL Community Server...
Aug 22 00:14:54 virtual systemd[1]: Started MySQL Community Server.
Aug 22 00:42:54 virtual systemd[1]: Stopping MySQL Community Server...
Aug 22 00:42:55 virtual systemd[1]: Stopped MySQL Community Server.
angelo@virtual:~$
```

Check which port MySQL is running

An important part of managing MySQL service is knowing which port the instance is running on. By default, the port used by MySQL is **3306**. **However**, it is always advisable to check before starting work.

To check which port is used by MySQL on Linux, you can use one of these two commands:

```
$ lsof -n | grep 'mysql.*TCP'
```

Or, you can use this one too:

```
$ netstat -tlnp | grep mysql
```



```
[root@localhost angelo# netstat -tln | grep mysql
tcp6      0      0 :::33060          :::*               LISTEN      9377/mysqld
tcp6      0      0 :::3306          :::*               LISTEN      9377/mysqld
[root@localhost angelo]#
```

Anyway, it is pretty useful to know which port is used by MySQL in order to open that port in case you are using a firewall like [iptables](#) or CSF firewall.

Restart MySQL

If you make any changes to MySQL settings, they will not become effective until you reboot MySQL or restart the MySQL service.

To restart MySQL just use the following command:

```
$ sudo systemctl restart mysql
```

For CentOS use this one as the root user:

```
$ systemctl restart mysqld
```

```
angelo@virtual:~$ sudo systemctl restart mysql
angelo@virtual:~$
```

If the command does not generate any screen output, it means that the service was successfully restarted.

Set root password for MySQL

The MySQL root user is the one who has all the privileges over all databases. So it is a must to define a very strong password for this user.

Some of the operations that the root user can do:

- Create new users.
- Manage the permissions of users.

To do this, we will use the **mysql_secure_installation** script, where we can not only define the password of the root user, but we can also add other configurations.

So, in Ubuntu, run the following command to use the

mysql_secure_installation script.

```
$ sudo mysql_secure_installation
```

On CentOS, the command is the same but run it as the root user.

```
$ mysql_secure_installation
```

There the first question will be about to activate the Validate Password plugin. It is a good idea to enable it. Then, choose the level of a password validation policy.

```
angelo@virtual:~$ sudo mysql_secure_installation
Securing the MySQL server deployment.

Connecting to MySQL using a blank password.

VALIDATE PASSWORD PLUGIN can be used to test passwords
and improve security. It checks the strength of password
and allows the users to set only those passwords which are
secure enough. Would you like to setup VALIDATE PASSWORD plugin?

Press y|Y for Yes, any other key for No: y

There are three levels of password validation policy:

LOW      Length >= 8
MEDIUM  Length >= 8, numeric, mixed case, and special characters
STRONG  Length >= 8, numeric, mixed case, special characters and dictionary file

Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG: 0
Please set the password for root here.
```

Next, we can set the root user password:

```
New password:
Re-enter new password:

Estimated strength of the password: 50
Do you wish to continue with the password provided?(Press y|Y for Yes, any other key for No) : y
By default, a MySQL installation has an anonymous user,
allowing anyone to log into MySQL without having to have
a user account created for them. This is intended only for
testing, and to make the installation go a bit smoother.
You should remove them before moving into a production
environment.
```

After that, the plugin will evaluate the strength of the password.

Type Y to continue.

Then, it will ask for other configuration questions that you have to answer. The answers depend on each particular case.

```
Remove anonymous users? (Press y|Y for Yes, any other key for No) : y
Success.

Normally, root should only be allowed to connect from
'localhost'. This ensures that someone cannot guess at
the root password from the network.

Disallow root login remotely? (Press y|Y for Yes, any other key for No) : y
Success.

By default, MySQL comes with a database named 'test' that
anyone can access. This is also intended only for testing,
and should be removed before moving into a production
environment.

Remove test database and access to it? (Press y|Y for Yes, any other key for No) : y
- Dropping test database...
Success.

- Removing privileges on test database...
Success.

Reloading the privilege tables will ensure that all changes
```



```
made so far will take effect immediately.  
Reload privilege tables now? (Press y|Y for Yes, any other key for No) : y  
Success.
```

Congratulations! The root password has been correctly defined.

Check MySQL version

MySQL is an application with quite active development. Besides this, the large Linux distributions constantly send security patches. So it is good to know which version of MySQL is installed.

On the other hand, on the internet, they often make publications about discovered security vulnerabilities and which versions are affected.

By knowing which version we have, we will be able to know if we have any vulnerabilities and take precautions.

To do this, simply run the following command:

```
$ mysql --version
```

```
angelo@virtual:~$ mysql --version  
mysql Ver 14.14 Distrib 5.7.27, for Linux (x86_64) using EditLine wrapper  
angelo@virtual:~$
```

There you can see the current version that is installed so that you can update it or keep it according to our needs.

Create a user on MySQL

Working with MySQL is a delicate matter, not because of the complexity of the service itself, but because of the security policies, we have to implement to protect the data.

A basic security policy is to create specific users for each application or database.

To do this, you have to first access the MySQL shell as the root user:

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

On CentOS:

```
$ mysql -u root -p
```

```
angelo@virtual:~$ sudo mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.7.27-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> █
```

When you run the command, you will be asked for the MySQL root user password. Then, you will be able to use MySQL commands.

To create a new user run the following sentence:

```
> CREATE USER 'user'@'localhost' IDENTIFIED BY 'password';
```

```
mysql> CREATE USER 'user'@'localhost' IDENTIFIED BY 'password';
Query OK, 0 rows affected (0.00 sec)

mysql> █
```

Replace 'user' with the username of your choice. 'localhost' is the host where you will create the user. And in 'password' enter the password for this new user. Similarly, it is recommended that it be a strong password.

Grant all privileges

The newly created user does not have any privilege on any database. In fact, you would not be able to access the MySQL shell either. Then, the root user has to grant certain privileges to this new user.

There are several privileges, and their assignment will depend on the server administrator. Some of them are:

- **ALL PRIVILEGES:** Assigns all permissions on the chosen database.
- **CREATE:** This privilege allows you to create new tables or databases.

- DROP: In this case, it allows you to delete tables and databases.
- DELETE: It allows deleting records from the tables.
- INSERT: With this privilege, you can create records in the tables.
- SELECT: This privilege is required to read the records from the tables.
- UPDATE: It allows you to update the records of a table.
- GRANT OPTION: With this privilege, the user can remove privileges from certain users.

The syntax for granting a privilege is as follows:

```
> GRANT [privilege1, privilege2] ON [database].[table] TO  
'[user]'@'localhost';
```

For example:

```
> GRANT CREATE ON test.* TO 'user'@'localhost';
```

```
mysql> GRANT CREATE ON test.* TO 'user'@'localhost';  
Query OK, 0 rows affected (0.03 sec)  
  
mysql> █
```

In this case, the user can only create tables in the database called test.

On the contrary, if you want to grant all permissions on all tables in a single call for the test database, the best option is ALL PRIVILEGES. This is an example:

```
> GRANT ALL PRIVILEGES ON test.* TO 'user'@'localhost';
```

```
mysql> GRANT ALL PRIVILEGES ON test.* TO 'user'@'localhost';  
Query OK, 0 rows affected (0.00 sec)  
  
mysql> █
```

In this way, privileges are assigned to a user in MySQL.

List all MySQL databases

Over time, our MySQL instance will be replete with many

databases. Then, it is better at some point to list them all.

To do this, access the MySQL console with the following command as the root user or using sudo:

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

Once we entered the MySQL console, just use the following command to list all existing databases:

```
> SHOW DATABASES;
```

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql      |
| performance_schema |
| sys       |
+-----+
4 rows in set (0.05 sec)

mysql> █
```

In the screen output, we can see all the databases.

Create a MySQL table

One of the most basic operations when creating a new MySQL instance is to create a table.

However, to create a table, we have to create a database.

Once we have entered the MySQL console, we create the database with the following command:

```
> CREATE DATABASE [database_name];
```

Then, we need to select the newly created database.

```
> USE [database_name];
```

In this case, I will create a database called “Example”.

```
mysql> CREATE DATABASE Example;
Query OK, 1 row affected (0.01 sec)

mysql> USE Example;
Database changed
mysql> █
```

Now we can create the table. The basic MySQL syntax to create a table is the following:

```
> CREATE TABLE [table_name] (column1_name data_type(length)
[NOT NULL] [DEFAULT value] [AUTO_INCREMENT],
column2_name data_type(length) [NOT NULL] [DEFAULT value]
[AUTO_INCREMENT] ... );
```

In the following example, I will create a table called “Person” with three columns. The first one will be “Id” that will be of integer type of length ten that cannot be null.

Generally, the first column is the one dedicated to the Primary Key. Besides, the options in [] are optional.

The second column will be “name” that will be varchar of length ten, and the last column will be “last_name” that will also be varchar of length 10.

```
> CREATE TABLE Person (Id int(10) NOT NULL, name
varchar(10), last_name varchar(10));
```

```
mysql> CREATE TABLE Person (Id int(10) NOT NULL, name varchar(10), last_name varchar(10));
Query OK, 0 rows affected (0.06 sec)

mysql> █
```

Awesome! You just created a MySQL table. However, the table is empty since it’s just created, and there is no data inserted yet. Now let’s see how to insert data into that table.

CRUD operations

CRUD operations to a database table are other very basic data manipulation operations. So, we need to learn how to do them.

CRUD stands for:

C: Create a table.

R: Read data from the database.

U: Update database.

D: Delete data from the database.

Create a new record (Insert)

Once the table is created, it is time to insert records into it. To do this, it is necessary to know which types the columns of the table are.

For this example, we will use the same table that we have created before. The basic syntax to create a record with the **insert** statement is as the following:

```
> INSERT INTO table_name (column1, column2, column3,..)  
VALUES ( value1, value2, value3,..);
```

So, it would be something like this:

```
> INSERT INTO Person (Id, name, last_name) VALUES  
(1,'Richard','Winters');
```

```
mysql> INSERT INTO Person (Id, name, last_name) VALUES (1,'Richard','Winters');  
Query OK, 1 row affected (0.00 sec)  
  
mysql> █
```

Read data from the Table (Select)

Once the data has been saved, it is time to read it. The statement is **select**. The basic syntax is as follows:

```
> SELECT column1, column2, ... FROM table_name;
```

If you want to read all the data in the table, use this one:

```
> SELECT * FROM table_name;
```

```
mysql> SELECT * FROM Person;  
+----+-----+-----+  
| Id | name   | last_name |  
+----+-----+-----+  
|  1 | Richard | Winters   |  
+----+-----+-----+  
1 row in set (0.00 sec)  
  
mysql> █
```

However, it is usually required to set a condition to read the data we need. You may have millions or billions of records, and it's highly

advisable not to return all this amount of data at once.

To do this, add the Where clause along with the condition.

> SELECT * FROM table_name WHERE [condition]

For example, if I wanted to read all the fields where the id is 1, I would use the following statement:

> SELECT * FROM Person WHERE Id=1;

```
mysql> SELECT * FROM Person WHERE Id=1;
+----+-----+-----+
| Id | name   | last_name |
+----+-----+-----+
|  1 | Richard | Winters   |
+----+-----+-----+
1 row in set (0.00 sec)

mysql> █
```

Update data

You may have inserted an incorrect data, so you need to update it.

To do this, we need to use the Update statement.

First, you need to know which record needs to be updated; it is necessary to add a small condition. Otherwise, it will update all records. So you need to be careful.

The basic syntax is as follows:

> UPDATE table_name SET column1 = value1, column2 = value2
WHERE [condition];

Continuing with the previous example, if you want to update the “last_name” column of the data whose Id is “1”, you would do it in the following way:

> UPDATE Person SET last_name = 'Clark' WHERE Id=1;

```
mysql> UPDATE Person SET last_name = 'Clark' WHERE Id=1;
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> █
```

Now, run the select statement again to see the changes:

> SELECT * FROM Person;

```
mysql> SELECT * FROM Person;
```

```

+----+-----+-----+
| Id | name  | last_name |
+----+-----+-----+
| 1  | Richard | Clark    |
+----+-----+-----+
1 row in set (0.00 sec)

mysql> █

```

Delete a record

If you want to delete records, you can use the delete statement. Also, you have to use a condition to know which record to delete.

The basic syntax of the delete command is as follows:

> DELETE FROM table_name WHERE condition;

For example, if you want to delete a record whose Id has a value of 1, the statement would be as follows:

> DELETE from Person WHERE Id=1;

```

mysql> DELETE from Person WHERE Id=1;
Query OK, 1 row affected (0.02 sec)

mysql> █

```

Import SQL file

MySQL allows you to import a database backup in SQL format. It is a great advantage if we want to move a database to a new server.

The first requirement is to create a database where the SQL file will be imported.

> CREATE DATABASE example2;

Then, we can exit the console using the following command:

> exit;

After that, we can start importing the file with the following command as the root user:

\$ sudo mysql -u username -p [database] < [sql_file_path]

```

angelo@virtual:~$ sudo mysql -u root -p example2 < file.sql
Enter password:
angelo@virtual:~$ █

```

Note that it is necessary to write the full path where the SQL file is located. In our case, the file “file.sql” is located on the home folder.

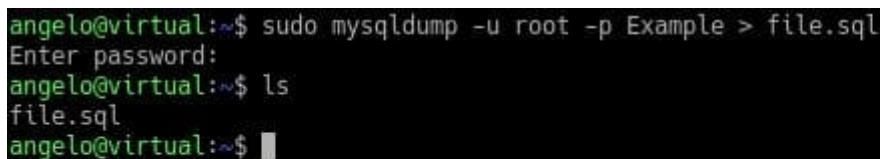
Export database

If MySQL can import a database, then it can also export a database to an SQL file.

To do this, without being in the MySQL console, we run the following command:

```
$ sudo mysqldump -u username -p database_to_export > file.sql
```

You can check the results using the [ls command](#):



```
angelo@virtual:~$ sudo mysqldump -u root -p Example > file.sql
Enter password:
angelo@virtual:~$ ls
file.sql
angelo@virtual:~$
```

When running the command, you will be prompted for the MySQL root user password.

Also, remember that this command must be run by the root user of the system or use sudo. Finally, it is possible to choose the path of the output file.

Backup database

MySQL has a tool called mysqldump that allows you to make backups of the databases in a MySQL instance. It is a very useful and agile tool.

Making a backup is similar to exporting a database as we have seen before. However, it is also possible with this tool to make a backup of all existing databases.

For it, it is enough to run the following command as a root user or with sudo:

```
$ sudo mysqldump -u root -p --all-databases > alldatabases.sql
```



```
angelo@virtual:~$ sudo mysqldump -u root -p --all-databases > alldatabases.sql
```

```
angelo@virtual:~$ sudo mysqldump -u root -p --all-databases > alldatabases.sql
Enter password:
angelo@virtual:~$ ls
alldatabases.sql  file.sql
angelo@virtual:~$
```

Make MySQL case insensitive

When developing independent applications, the names of the tables are usually a problem. By default, MySQL is case sensitive in table names.

Therefore, in some cases, you need to revert this behavior.

To make this change, it is necessary to add a directive in the MySQL configuration file. First, stop the MySQL service.

```
$ sudo systemctl stop mysql
```

If we use CentOS, the command is as follows:

```
$ systemctl stop mysqld
```

```
angelo@virtual:~$ sudo systemctl stop mysql
angelo@virtual:~$
```

After this, edit the following specific configuration file. So open it with root user permissions:

On Ubuntu:

```
$ sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
```

If we use CentOS, the path will be different. The path will be

/etc/my.cnf

.

Then, under the section [mysqld] add the following directive:

```
lower_case_table_names = 1
```

```
[mysqld]
#
# * Basic Settings
#
user                = mysql
pid-file            = /var/run/mysqld/mysqld.pid
socket              = /var/run/mysqld/mysqld.sock
port                = 3306
basedir             = /usr
datadir             = /usr
tmpdir              = /tmp
log-err              = /var/log/mysqld.log
log-queries          = 0
log-warnings         = 1
```

```
datadir = /var/lib/mysql
tmpdir = /tmp
lc-messages-dir = /usr/share/mysql
skip-external-locking

lower_case_table_names = 1
```

Then save the changes and close the file.

Next, we need to start the MySQL service again.

```
$ sudo systemctl start mysql
```

Or on CentOS:

```
$ systemctl start mysqld
```

And that is enough. You can create tables with upper and lower case, and MySQL will treat them the same way.

Where are MySQL logs on Linux?

[Log files](#) are sequential records of all events related to a particular program or system service.

It is very important to check them from time to time in search for any abnormal behaviors in the execution of the program. Also, when the MySQL service fails or cannot start, the error is recorded directly in the log.

On Ubuntu, the MySQL logs are saved in

```
/var/log/mysql/error.log
```

.

In the case of CentOS, the file is located in

```
/var/log/mysqld.log
```

.

To review the contents of the file, you can use the [tail command](#).

For example:

```
$ tail /var/log/mysql/error.log
```

```
angelo@virtual:~$ tail /var/log/mysql/error.log
2019-08-22T01:37:30.620373Z 0 [Note] InnoDB: Buffer pool(s) dump completed at 190822 1:37:30
2019-08-22T01:37:32.142430Z 0 [Note] InnoDB: Shutdown completed; log sequence number 2652287
2019-08-22T01:37:32.146255Z 0 [Note] InnoDB: Removed temporary tablespace data file: "ibtmp1"
2019-08-22T01:37:32.146284Z 0 [Note] Shutting down plugin 'MEMORY'
2019-08-22T01:37:32.146297Z 0 [Note] Shutting down plugin 'CSV'
2019-08-22T01:37:32.146309Z 0 [Note] Shutting down plugin 'sha256_password'
```

```

2019-08-22T01:37:32.146317Z 0 [Note] Shutting down plugin 'mysql_native_password'
2019-08-22T01:37:32.146601Z 0 [Note] Shutting down plugin 'binlog'
2019-08-22T01:37:32.147032Z 0 [Note] /usr/sbin/mysqld: Shutdown complete

angelo@virtual:~$ █

```

Or, if you use CentOS:

```
$ tail /var/log/mysqld.log
```

By running it, you will be able to see everything that has happened with the service.

Also, you can use -f option with the tail command to continuously see the latest updates for the log records.

```
$ tail -f /var/log/mysqld.log
```

Uninstall MySQL

In case you want to uninstall MySQL from Linux, just invoke the package manager of each distribution.

In the case of Ubuntu, we use the APT package manager. To uninstall it, the following command is enough:

```
$ sudo apt remove mysql
```

```

angelo@virtual:~$ sudo apt remove mysql-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libatoi libcgi-fast-perl libcgi-pm-perl libencode-locale-perl libevent-core-2.1-6 libfcgi-perl libhtml-parser-perl libhtml-tagset-perl libhtml-template-perl libhttp-date-perl
  libhttp-message-perl libio-html-perl liblwp-mediatypes-perl libtimedate-perl liburi-perl mysql-client-5.7 mysql-client-core-5.7 mysql-common mysql-server-5.7 mysql-server-core-5.7
Use 'sudo apt autoremove' to remove them.
The following packages will be REMOVED:
  mysql-server
0 upgraded, 0 newly installed, 1 to remove and 54 not upgraded.
After this operation, 110 kB disk space will be freed.
Do you want to continue? [Y/n] █

```

On CentOS, the package manager is YUM, so as the root user, use the following command:

```
$ yum remove mysql-community-server
```

```

[root@localhost angelo]# yum remove mysql-community-server
Loaded plugins: fastestmirror
Resolving Dependencies
--> Running transaction check
--> Package mysql-community-server.x86_64 0:8.0.17-1.el7 will be erased
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package                               Arch          Version           Size
-----
Removing:
mysql-community-server                x86_64        8.0.17-1.el7      1.8 G
=====
Transaction Summary
=====
Remove 1 Package

Installed size: 1.8 G
Is this ok [y/N]: █

```

Conclusion

MySQL is a very popular relational database. Its power & ease of use as we saw makes it the favorite engine for a large number of developers. However, in this post, we have covered some basics of MySQL on Linux.

I hope you find the tutorial useful. Keep coming back.



Mokhtar is the founder of LikeGeeks.com. He works as a Linux system administrator since 2010. He is responsible for maintaining, securing, and troubleshooting Linux servers for multiple clients around the world. He loves writing shell and Python scripts to automate his work.