How to install MySQL on Ubuntu 20.04

In this article, TipsMake will show how to install MySQL version 8.0 on Ubuntu 20.04 server. By completing it, you'll have an active relational database that can be used to build your next website or app.

MySQL is an open source database administration system, usually installed as part of the popular LAMP (Linux, Apache, MySQL, PHP / Python / Perl) stack. It implements a relational model and uses structured query language (also known as SQL - Structured Query Language) to manage data.

To follow along with this guide, you will need an Ubuntu 20.04 server with a non-root admin user and a firewall configured using UFW.

How to install MySQL on Ubuntu 20.04

Step 1 - Install MySQL

On Ubuntu 20.04, you can install MySQL using the APT package repository. At the time of this writing, the version of MySQL available in the default Ubuntu repositories is version 8.0.19.

To install it, update the package index on the server, if you haven't done so recently:

```
sudo apt update
```

Then install the mysql-server package:

```
sudo apt install mysql-server
```

This will install MySQL, but will not prompt you to set a password or make any other configuration changes. Since this makes your MySQL installation insecure, we will address this issue next.

Step 2 - Configure MySQL

For new installations of MySQL, you will want to run the built-in security script of the DBMS. This script changes some of the less secure defaults for things like remote root credentials and sample users.

Run the security script with sudo:

```
sudo mysql_secure_installation
```

This will take you through a series of prompts where you can make some changes to the security options of your MySQL installation. The first prompt will ask if you want to set up a Validate Password Plugin, which can be used to check the password strength of a new MySQL user before it is deemed valid.

If you choose to set up the Validate Password Plugin, any MySQL user that you create password authentication will be asked to have a password that meets the policy of your choice. The strongest policy level - which you can choose by entering **2** - will require a password to be at least 8 characters long and include a combination of uppercase, lowercase, numeric and special characters:

```
Securing the MySQL server deployment. Connecting to MySQL using a blank password
```

Regardless of whether you choose to set up the Validate Password Plugin or not, the next prompt will be to set a password for the MySQL root user. Enter and then confirm a secure password of your choice:

```
Please set the password for root here. New password: Re-enter new password:
```

Note that although you have already set a password for the MySQL root user, this user is not currently configured for password authentication when connecting to the MySQL shell.

If you used the Validate Password Plugin, you will get feedback on the strength of your new password. The script will then ask if you want to continue with the password you just entered or if you want to enter a new password. Assuming you are satisfied with the strength of the password you just entered, type **Y** to continue the script:

```
Estimated strength of the password: 100 Do you wish to continue with the passwor
```

From there, you can press **Y** then **ENTER** to accept the default value for all subsequent questions. This will remove some anonymous users and the test database, disable remote root login, and load these new rules so that MySQL immediately obeys the changes you made.

Once the script is complete, the MySQL installation will be secure. Now, you can move on to creating a dedicated database user with the MySQL client.

Step 3 - Create a dedicated MySQL user and grant privileges

Once installed, MySQL creates a root user account that you can use to manage your database. This user has full privileges over the MySQL server, which means he / she has full control over any databases, tables, users, etc . Therefore, it is best to avoid using an account. This is in addition to when administrative functions are needed. This step outlines how to use the MySQL root user to create a new user account and grant privileges for that account.

In Ubuntu systems running MySQL 5.7 (and later versions), MySQL root users are required to authenticate using the **auth_socket** plugin by default, not with a password.

This plugin requires that the name of the operating system user calling the MySQL client must match the name of the MySQL user specified in the command, so you must call **mysql** with **sudo** privileges in order to gain access to the MySQL root user:

```
sudo mysql
```

Note : If you installed MySQL with another tutorial and enabled password authentication for root, you will need to use another command to access the MySQL shell. The following command will run your MySQL client with regular user privileges, and you will only obtain admin privileges in the database by authenticating:

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

Once you have access to the MySQL prompt, you can create a new user with the **CREATE USER** statement **.** They follow the following general syntax:

```
CREATE USER 'username'@'host' IDENTIFIED WITH authentication_plugin BY 'password
```

After **CREATE USER** , you specify a username. Immediately followed by the **@** sign and then the name of the host to which this user will connect. If you only plan to access this user locally from your Ubuntu server, you can specify localhost. Wrapping both the username and the server in quotation marks is not always necessary, but doing so can help avoid errors.

You have several options when choosing a user authentication plugin. The **auth_socket** plugin mentioned earlier can be convenient, as it provides strong security, without requiring a valid user to enter a password to access the database. But it also prevents remote connections, which can complicate things when external programs need to interact with MySQL.

Instead, you can completely remove the **WITH** validation plugin part of the syntax so that users authenticate with the default MySQL plugin, **caching_sha2_password.** The MySQL documentation recommends this plugin for users who want to log in with a password due to its strong security features.

Run the following command to create an authenticated user using **caching_sha2_password.** Make sure to change **sammy** to your username and **password** to a strong password of your choice:

```
CREATE USER 'sammy'@'localhost' IDENTIFIED BY 'password';
```

Note : Some PHP versions cause problems with **caching_sha2_password.** If you intend to use this database with a PHP application - for example, **phpMyAdmin** - you may want to create a user that will authenticate with theolder, but still secure **mysql_native_password** plugin, instead:

```
CREATE USER 'sammy'@'localhost' IDENTIFIED WITH mysql_native_password BY 'passwo
```

If in doubt, you can always create an authenticated user with **caching_sha2_plugin** , and then change it later with this command:

```
ALTER USER 'sammy'@'localhost' IDENTIFIED WITH mysql_native_password BY 'passwor
```

After creating a new user, you can grant them the appropriate privileges. The general syntax for granting user privileges is as follows:

```
GRANT PRIVILEGE ON database.table TO 'username'@'host';
```

The **PRIVILEGE** value in this example syntax defines the actions a user is allowed to perform on the specified database and table. You can grant multiple privileges to the same user in a command by separating each privilege with commas. You can also grant user privileges by entering an asterisk (*) in the location of the database and the table name. In SQL, the asterisks are special characters used to represent "all" a database or table.

For the sake of illustration, the following command grants the user privileges to CREATE, ALTER, and DROP databases, tables, and users, as well as permissions to INSERT, UPDATE, and DELETE data from any table on the server. It also gives the user the ability to query data with SELECT, create Foreign Key with the REFERENCES keyword, and perform FLUSH operations with RELOAD privileges. However, you should only give the user the permissions they need, so feel free to adjust user privileges if needed.

You can find the complete list of available privileges in the official MySQL documentation.

```
https://dev.mysql.com/doc/refman/8.0/en/privileges-provided.html#privileges-provi
```

Run this GRANT statement, replacing **sammy** with your own MySQL username, to grant these privileges to the desired user:

```
GRANT CREATE, ALTER, DROP, INSERT, UPDATE, DELETE, SELECT, REFERENCES, RELOAD on
```

Note that this command also includes the **WITH GRANT OPTION** . This will allow MySQL users to grant any permissions they have to other users on the system.

Warning : Some users may want to grant their MySQL user the ALL PRIVILEGES privilege, which will give them extensive superuser privileges just like the root user privilege, like so:

```
GRANT ALL PRIVILEGES ON *.* TO 'sammy'@'localhost' WITH GRANT OPTION;
```

Granting such extensive privileges should not be taken lightly, since anyone with access to this MySQL user will have complete control over every database on the server.

Here you should run the FLUSH PRIVILEGES command. This will free any memory the server has cached as a result of the previous CREATE USER and GRANT statements:

```
FLUSH PRIVILEGES;
```

You can then exit the MySQL client:

```
exit
```

In the future, to log in as a new MySQL user, you would use a command like the following:

```
mysql -u sammy -p
```

The **-p** flag will cause the MySQL client to prompt you for the MySQL user password for authentication.

Finally, check out the MySQL installation.

Step 4 - Test MySQL

Regardless of how you have it installed, MySQL must start running automatically. For testing, check its status.

```
systemctl status mysql.service
```

You should see output similar to the following:

```
mysql.service - MySQL Community Server Loaded: loaded (/lib/systemd/system/mysql
??10382 /usr/sbin/mysqld
```

If MySQL is not running, you can start it with **sudo systemctl start mysql** .

To test it further, you can try connecting to the database using the **mysqladmin** tool , which is a client that allows you to run admin commands. For example, this command tells you to connect as a MySQL user named **sammy** (**-u sammy**), prompt for your password (**-p**), and return the version. Make sure to change **sammy to** the name of your dedicated MySQL user, and enter that user's password when prompted:

```
sudo mysqladmin -p -u sammy version
```

You should see output similar to the following:

```
mysqladmin Ver 8.0.19-0ubuntu5 for Linux on x86_64 ((Ubuntu)) Copyright (c) 2000,
```

This means that MySQL is working.

You finished reading the article "**How to install MySQL on Ubuntu 20.04**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.
