Install MySQL Server on the Ubuntu operating system

Last updated on: 2019-12-20

Authored by: Jered Heeschen

MySQL is an open-source relational database that is free and widely used. It is a good choice if you know that you need a database but don't know much about all the available options.

This article describes a basic installation of a MySQL database server on the Ubuntu operating system. You might need to install other packages to let applications use MySQL, like extensions for PHP. Check your application documentation for details.

Install MySQL

Install the MySQL server by using the Ubuntu operating system package manager:

```
sudo apt-get update
sudo apt-get install mysql-server
```

The installer installs MySQL and all dependencies.

If the secure installation utility does not launch automatically after the installation completes, enter the following command:

`sudo mysql_secure_installation utility`

This utility prompts you to define the mysql root password and other security-related options, including removing remote access to the root user and setting the root password.

Allow remote access

If you have iptables enabled and want to connect to the MySQL database from another machine, you must open a port in your server's firewall (the default port is 3306). You don't need to do this if the application that uses MySQL is running on the same server.

Run the following command to allow remote access to the mysql server:

```
sudo ufw enable
sudo ufw allow mysql
```

Start the MySQL service

After the installation is complete, you can start the database service by running the following command. If the service is already started, a message informs you that the service is already running:

```
sudo systemctl start mysql
```

Launch at reboot

To ensure that the database server launches after a reboot, run the following command:

```
sudo systemctl enable mysql
```

Configure interfaces

MySQL, by default is no longer bound to (listening on) any remotely accessible interfaces. Edit the "bind-address" directive in /etc/mysql /mysql.conf.d/mysqld.cnf:

```
bind-address            = 127.0.0.1 ( The default. )
bind-address            = XXX.XXX.XXX.XXX ( The ip address of your Public Net interface. )
bind-address            = ZZZ.ZZZ.ZZZ.ZZZ ( The ip address of your Service Net interface. )
bind-address            = 0.0.0.0 ( All ip addresses. )
```

Restart the mysql service.

```
sudo systemctl restart mysql
```

Start the mysql shell

There is more than one way to work with a MySQL server, but this article focuses on the most basic and compatible approach, the `mysql` shell.

1. At the command prompt, run the following command to launch the `mysql` shell and enter it as the root user:

   ```
   /usr/bin/mysql -u root -p
   ```

2. When you're prompted for a password, enter the one that you set at installation time, or if you haven't set one, press **Enter** to submit no password.

   The following `mysql` shell prompt should appear:

   ```
   mysql>
   ```

Set the root password

If you logged in by entering a blank password, or if you want to change the root password that you set, you can create or change the password.

1. For versions earlier than MySQL 5.7, enter the following command in the `mysql` shell, replace `password` with your new password:

   ```
   UPDATE mysql.user SET Password = PASSWORD('password') WHERE User = 'root';
   ```

For version MySQL 5.7 and later, enter the following command in the `mysql` shell, replacing `password` with your new password:

```
UPDATE mysql.user SET authentication_string = PASSWORD('password') WHERE User = 'root';
```

2. To make the change take effect, reload the stored user information with the following command:

```
FLUSH PRIVILEGES;
```

Note: We're using all-caps for SQL commands. If you type those commands in lowercase, they'll work. By convention, the commands are written in all-caps to make them stand out from field names and other data that's being manipulated.

If you need to reset the root password later, see Reset a MySQL root password.

View users

MySQL stores the user information in its own database. The name of the database is **mysql**. Inside that database the user information is in a table, a dataset, named **user**. If you want to see what users are set up in the MySQL user table, run the following command:

```
SELECT User, Host, authentication_string FROM mysql.user;
```

The following list describes the parts of that command:

- **SELECT** tells MySQL that you are asking for data.
- **User**, **Host**, **authentication_string** tells MySQL what fields you want it to look in. Fields are categories for the data in a table. In this case, you are looking for the username, the host associated with the username, and the encrypted password entry.
- **FROM mysql.user** " tells MySQL to get the data from the **mysql** database and the **user** table.
- A semicolon (;) ends the command.

Note: All SQL queries end in a semicolon. MySQL does not process a query until you type a semicolon.

User hosts

The following example is the output for the preceding query:

```
SELECT User, Host, authentication_string FROM mysql.user;
+------------------+-----------+-------------------------------------------+
| User             | Host      | authentication_string                     |
+------------------+-----------+-------------------------------------------+
| root             | localhost | *756FEC25AC0E1823C9838EE1A9A6730A20ACDA21 |
| mysql.session    | localhost | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE |
| mysql.sys        | localhost | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE |
| debian-sys-maint | localhost | *27E7CA2445405AB10C656AFD0F86AF76CCC57692 |
+------------------+-----------+-------------------------------------------+
```

Users are associated with a host, specifically, the host from which they connect. The root user in this example is defined for **localhost**, for the IP address of **localhost**, and the hostname of the server. You usually need to set a user for only one host, the one from which you typically connect.

If you're running your application on the same computer as the MySQL server, the host that it connects to by default is **localhost**. Any new users that you create must have **localhost** in their **host** field.

If your application connects remotely, the **host** entry that MySQL looks for is the IP address or DNS hostname of the remote computer (the one from which the client is coming).

Anonymous users

In the example output, one entry has a host value but no username or password. That's an *anonymous user*. When a client connects with no username specified, it's trying to connect as an anonymous user.

You usually don't want any anonymous users, but some MySQL installations include one by default. If you see one, you should either delete the user (refer to the username with empty quotes, like ' ') or set a password for it.

Create a database

There is a difference between a *database server* and a *database*, even though those terms are often used interchangeably. MySQL is a database server, meaning it tracks databases and controls access to them. The database stores the data, and it is the database that applications are trying to access when they interact with MySQL.

Some applications create a database as part of their setup process, but others require you to create a database yourself and tell the application about it.

To create a database, log in to the `mysql` shell and run the following command, replacing `demodb` with the name of the database that you want to create:

```
CREATE DATABASE demodb;
```

After the database is created, you can verify its creation by running a query to list all databases. The following example shows the query and example output:

```
SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| demodb             |
| mysql              |
+--------------------+
3 rows in set (0.00 sec)
```

Add a database user

When applications connect to the database using the root user, they usually have more privileges than they need. You can add users that applications can use to connect to the new database. In the following example, a user named **demouser** is created.

1. To create a new user, run the following command in the `mysql` shell:

```
INSERT INTO mysql.user (User,Host,authentication_string,ssl_cipher,x509_issuer,x509_subject)
VALUES('demouser','localhost',PASSWORD('demopassword'),'','','');
```

2. When you make changes to the **user** table in the **mysql** database, tell MySQL to read the changes by flushing the privileges, as follows:

```
FLUSH PRIVILEGES;
```

3. Verify that the user was created by running a SELECT query again:

```
SELECT User, Host, authentication_string FROM mysql.user;
```

```
+------------------+-----------+-------------------------------------------+
| User             | Host      | Password                                  |
+------------------+-----------+-------------------------------------------+
| root             | localhost | *756FEC25AC0E1823C9838EE1A9A6730A20ACDA21 |
| mysql.session    | localhost | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE |
| mysql.sys        | localhost | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE |
| debian-sys-maint | localhost | *27E7CA2445405AB10C656AFD0F86AF76CCC57692 |
| demouser         | localhost | *0756A562377EDF6ED3AC45A00B356AAE6D3C6BB6 |
+------------------+-----------+-------------------------------------------+
```

Grant database user permissions

Right after you create a new user, it has no privileges. The user can log in, but can't be used to make any database changes.

1. Give the user full permissions for your new database by running the following command:

```
GRANT ALL PRIVILEGES ON demodb.* to demouser@localhost;
```

2. Flush the privileges to make the change official by running the following command:

```
FLUSH PRIVILEGES;
```

3. To verify that those privileges are set, run the following command:

```
SHOW GRANTS FOR 'demouser'@'localhost';
2 rows in set (0.00 sec)
```

MySQL returns the commands needed to reproduce that user's permissions if you were to rebuild the server. `USAGE on \*.\*` means the users gets no privileges on anything by default. That command is overridden by the second command, which is the grant you ran for the new database.

```
+-----------------------------------------------------------------------------------------------------------------
------+
| Grants for demouser@localhost
     |
+-----------------------------------------------------------------------------------------------------------------
-----+
| GRANT USAGE ON *.* TO 'demouser'@'localhost' IDENTIFIED BY PASSWORD
'*0756A562377EDF6ED3AC45A00B356AAE6D3C6BB6' |
| GRANT ALL PRIVILEGES ON `demodb`.* TO 'demouser'@'localhost'
     |
+-----------------------------------------------------------------------------------------------------------------
-----+
2 rows in set (0.00 sec)
```

Summary

If you're just creating a database and a user, you are done. The concepts covered here should give you a solid start from which to learn more.

Related articles

- Configure MySQL server on the Ubuntu operating system
- Reset a MySQL root password

SHARE THIS INFORMATION: