

# How to Install MySQL on Ubuntu 20.04

Learn how to install MySQL on Ubuntu 20.04 with our step-by-step guide. Includes setup, configuration, and security best practices for your database server.

---

# Contents

01	Introduction	3
02	Prerequisites	3
03	Install MySQL on Ubuntu 20.04	3
04	Manage the MySQL System Service	4
05	Secure the MySQL Server	5
06	Access MySQL	7
07	Create a Sample MySQL Database	8
08	Conclusion	10

# Introduction

MySQL is a popular open-source relational database used to store and manage data efficiently. Installing it on Ubuntu 20.04 provides a stable setup for building and securing database-driven applications.

This article covers the steps to install and configure MySQL on Ubuntu 20.04. For a similar setup on a different system, check out our article on [installing MySQL on Debian 12](#).

## Prerequisites

Before you begin:

- Have an [Ubuntu 20.04 server](#).
- Access the server using SSH as a non-root user with sudo privileges.
- [Update the server](#).

## Install MySQL on Ubuntu 20.04

MySQL is included in Ubuntu 20.04's default APT repositories. To install it, update your package index and install the latest MySQL server version using the APT package manager.

1. Update the server package index.

CONSOLE

```
$ sudo apt update
```

2. Install the MySQL server package.

CONSOLE

```
$ sudo apt install mysql-server -y
```

3. View the installed MySQL version on your server.

CONSOLE

```
$ mysql --version
```

Output:

```
mysql Ver 8.0.41-0ubuntu0.20.04.1 for Linux on x86_64 ((Ubuntu))
```

## Manage the MySQL System Service

MySQL runs as the `mysql` system service on Ubuntu 20.04. Enable it to start at boot and verify its status to manage the database processes.

1. Enable MySQL to automatically start at boot time.

CONSOLE

```
$ sudo systemctl enable mysql
```

Output:

```
Synchronizing state of mysql.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.  
Executing: /usr/lib/systemd/systemd-sysv-install enable mysql
```

2. Start the MySQL database server.

CONSOLE

```
$ sudo systemctl start mysql
```

3. View the MySQL server status and verify that it's running.

## CONSOLE

```
$ sudo systemctl status mysql
```

Output:

```
● mysql.service - MySQL Community Server
   Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset:
   enabled)
   Active: active (running) since Sun 2025-04-06 10:58:51 UTC; 5min ago
 Main PID: 17189 (mysqld)
   Status: "Server is operational"
   Tasks: 39 (limit: 9415)
  Memory: 365.9M
   CGroup: /system.slice/mysql.service
           └─17189 /usr/sbin/mysqld
```

## Secure the MySQL Server

Securing MySQL helps protect your databases from unauthorized access. Set a strong `root` password and remove insecure defaults using the steps below.

1. Run the following command to start the MySQL secure installation script.

## CONSOLE

```
$ sudo mysql_secure_installation
```

Follow these steps to secure your MySQL database server:

- **Enable password validation:** Enter `y` and press `Enter` to activate password validation.
- **Set strong password policy:** Enter `2` to enforce the use of strong passwords.
- **Set a new password:** Provide a strong password for the root user (it will be verified).
- **Confirm password:** Enter `y` to confirm the new root password.

- **Remove anonymous users:** Enter `y` to revoke access from unauthorized users.
- **Disable remote root login:** Enter `y` to prevent root user login from remote hosts.
- **Delete test databases:** Enter `y` to remove any test databases.
- **Reload privilege tables:** Enter `y` to apply the changes immediately.

2. Log in to the MySQL database server as `root`.

```
CONSOLE
$ sudo mysql
```

3. Set a strong password for the root user.

```
SQL
mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH
mysql_native_password BY 'your_strong_password';
```

Replace `your_strong_password` with a strong password depending on your password strength policy.

4. Reload the MySQL Privilege tables to apply changes.

```
SQL
mysql> FLUSH PRIVILEGES;
```

5. Exit the MySQL database console.

```
SQL
mysql> EXIT;
```

6. Restart the MySQL service to apply your configuration changes.

```
CONSOLE
```

```
$ sudo systemctl restart mysql
```

## Access MySQL

1. Log in to the MySQL database server as `root`.

CONSOLE

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

Enter the root user password you set earlier when prompted.

2. Create a new sample MySQL database. For example, `my_database`.

SQL

```
mysql> CREATE DATABASE my_database;
```

3. Create a new MySQL database user with a strong password. For example, `my_user` and replace `my_password` with your desired password depending on your password strength policy.

SQL

```
mysql> CREATE USER 'my_user'@'localhost' IDENTIFIED BY 'my_password';
```

4. Grant the database user `my_user` full privileges to your sample database `my_database`.

SQL

```
mysql> GRANT ALL PRIVILEGES ON my_database.* TO 'my_user'@'localhost';
```

5. Grant the user `my_user` permission to create databases on the server.

```
SQL
```

```
mysql> GRANT CREATE ON *.* TO 'my_user'@'localhost';
```

- Grant the user `my_user` permission to perform CRUD operations (SELECT, INSERT, UPDATE, DELETE) on all databases.

```
SQL
```

```
mysql> GRANT SELECT, INSERT, UPDATE, DELETE ON *.* TO  
'my_user'@'localhost';
```

- Refresh the MySQL privilege tables to apply the new user changes.

```
SQL
```

```
mysql> FLUSH PRIVILEGES;
```

- Exit the MySQL database console.

```
SQL
```

```
mysql> EXIT;
```

## Create a Sample MySQL Database

Non-privileged MySQL users, like `my_user`, can create databases and tables. In this example, `my_user` creates a database, adds a table, inserts data, and queries the records.

- Log in to the MySQL database console using the database user `my_user` you created earlier.

```
CONSOLE
```

```
$ mysql -u my_user -p
```

Enter the `my_user` password you set earlier when prompted.

## 2. List all databases available to the MySQL user

```
SQL
mysql> SHOW DATABASES;
```

Output:

```
+-----+
| Database          |
+-----+
| information_schema |
| my_database       |
| mysql             |
| performance_schema |
| sys               |
+-----+
```

## 3. Create another sample MySQL database. For example, `bookstore_db`.

```
SQL
mysql> CREATE DATABASE example_vultr;
```

## 4. Switch to the new database.

```
SQL
mysql> USE example_vultr;
```

## 5. Create a new sample table with the 3 columns to store different data types.

```
SQL
mysql> CREATE TABLE sample_table (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
```

```
description TEXT,  
created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
);
```

6. Insert sample data into the `sample_table` table.

SQL

```
mysql> INSERT INTO sample_table (name, description) VALUES  
('Sample Item 1', 'This is the first sample item.'),  
('Sample Item 2', 'This is the second sample item.'),  
('Sample Item 3', 'This is the third sample item.);
```

7. Select all table records to verify that the new table data is available.

SQL

```
mysql> SELECT * FROM sample_table;
```

Output:

```
+-----+-----+-----+-----+  
| id | name          | description                               | created_at          |  
+-----+-----+-----+-----+  
|  1 | Sample Item 1 | This is the first sample item.          | 2024-06-19 10:00:00 |  
|  2 | Sample Item 2 | This is the second sample item.         | 2024-06-19 10:00:00 |  
|  3 | Sample Item 3 | This is the third sample item.          | 2024-06-19 10:00:00 |  
+-----+-----+-----+-----+
```

## Conclusion

You have installed MySQL on your Ubuntu 20.04 server and secured it for authenticated access. MySQL integrates with modern web applications and can serve as a dedicated database server or backend for dynamic stacks like LAMP. For more details, visit the [official documentation](#).



VULTR

