CONTENTS

Hazel Virdó and Mark Drake

English

Not using Ubuntu 20.04?
Choose a different version or distribution.

Ubuntu 20.04

Introduction

MySQL is an open-source database management system, commonly installed as part of the popular LAMP (Linux, Apache, MySQL, PHP/Python/Perl) stack. It implements the relational model and uses Structured Query Language (better known as SQL) to manage its data.

DigitalOcean focus on maintaining, upgrading, and backups.

Prerequisites

To follow this tutorial, you will need:

- One Ubuntu 20.04 server with a non-root administrative user and a firewall configured with UFW. To set this up, follow our initial server setup guide for Ubuntu 20.04.

Step 1 – Installing 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 repository is version 8.0.27.

To install it, update the package index on your server if you've not done so recently:

Copy

```
$ sudo apt update
```

Then install the `mysql-server` package:

Copy

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

Ensure that the server is running using the `systemctl start` command:

Copy

```
$ sudo systemctl start mysql.service
```

For fresh installations of MySQL, you'll want to run the DBMS's included security script. This script changes some of the less secure default options for things like remote root logins and sample users.

Warning: As of July 2022, an error will occur when you run the `mysql_secure_installation` script without some further configuration. The reason is that this script will attempt to set a password for the installation's **root** MySQL account but, by default on Ubuntu installations, this account is not configured to connect using a password.

Prior to July 2022, this script would silently fail after attempting to set the **root** account password and continue on with the rest of the prompts. However, as of this writing the script will return the following error after you enter and confirm a password:

```
Output
 ... Failed! Error: SET PASSWORD has no significance for user 'root'@'localhost' as t

New password:
```

This will lead the script into a recursive loop which you can only get out of by closing your terminal window.

Because the `mysql_secure_installation` script performs a number of other actions that are useful for keeping your MySQL installation secure, it's still recommended that you run it before you begin using MySQL to manage your data. To avoid entering this recursive loop, though, you'll need to first adjust how your **root** MySQL user authenticates.

First, open up the MySQL prompt:

Copy

```
$ sudo mysql
```

Then run the following `ALTER USER` command to change the **root** user's authentication method to one that uses a password. The following example changes the authentication method to `mysql_native_password`:

Copy

```
mysql> exit
```

Following that, you can run the `mysql_secure_installation` script without issue.

Once the security script completes, you can then reopen MySQL and change the **root** user's authentication method back to the default, `auth_socket`. To authenticate as the **root** MySQL user using a password, run this command:

Copy

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

Then go back to using the default authentication method using this command:

Copy

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED WITH auth_socket;
```

This will mean that you can once again connect to MySQL as your **root** user using the `sudo mysql` command.

Run the security script with `sudo`:

Copy

```
$ sudo mysql_secure_installation
```

This will take you through a series of prompts where you can make some changes to your MySQL installation's security options. The first prompt will ask whether you'd like to set up the Validate Password Plugin, which can be used to test the password strength of new MySQL users before deeming them valid.

```
Output
Securing the MySQL server deployment.

Connecting to MySQL using a blank password.

VALIDATE PASSWORD COMPONENT 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 component?

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

Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG:
  2
```

Regardless of whether you choose to set up the Validate Password Plugin, the next prompt will be to set a password for the MySQL **root** user. Enter and then confirm a secure password of your choice:

```
Output
Please set the password for root here.


New password:

Re-enter new password:
```

Note that even though you've set a password for the **root** MySQL user, this user is not currently configured to authenticate with a password when connecting to the MySQL shell.

If you used the Validate Password Plugin, you'll receive feedback on the strength of your new password. Then the script will ask if you want to continue with the password you

```
Do you wish to continue with the password provided?(Press y|Y for Yes, any other key
```

From there, you can press `Y` and then `ENTER` to accept the defaults for all the subsequent questions. This will remove some anonymous users and the test database, disable remote root logins, and load these new rules so that MySQL immediately respects the changes you have made.

Once the script completes, your MySQL installation will be secured. You can now move on to creating a dedicated database user with the MySQL client.

Step 3 – Creating a Dedicated MySQL User and Granting Privileges

Upon installation, MySQL creates a **root** user account which you can use to manage your database. This user has full privileges over the MySQL server, meaning it has complete control over every database, table, user, and so on. Because of this, it's best to avoid using this account outside of administrative functions. This step outlines how to use the **root** MySQL user to create a new user account and grant it privileges.

In Ubuntu systems running MySQL `5.7` (and later versions), the **root** MySQL user is set to authenticate using the `auth_socket` plugin by default rather than with a password. This plugin requires that the name of the operating system user that invokes the MySQL client matches the name of the MySQL user specified in the command, so you must invoke `mysql` with `sudo` privileges to gain access to the **root** MySQL user:

Copy

```
$ sudo mysql
```

> **Note:** If you installed MySQL with another tutorial and enabled password authentication for **root**, you will need to use a different command to access the MySQL shell. The following will run your MySQL client with regular user privileges, and you will only gain administrator privileges within the database by authenticating:

Copy

Once you have access to the MySQL prompt, you can create a new user with a `CREATE USER` statement. These follow this general syntax:

Copy

```
mysql> CREATE USER ' username '@' host ' IDENTIFIED WITH  authentication_plugin  BY '
```

After `CREATE USER`, you specify a username. This is immediately followed by an `@` sign and then the hostname from 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 host in single quotes isn't always necessary, but doing so can help to prevent errors.

You have several options when it comes to choosing your user's authentication plugin. The `auth_socket` plugin mentioned previously can be convenient, as it provides strong security without requiring valid users 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.

As an alternative, you can leave out the `WITH authentication_plugin` portion of the syntax entirely to have the user authenticate with MySQL's default 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 a user that authenticates with `caching_sha2_password`. Be sure to change `sammy` to your preferred username and `password` to a strong password of your choosing:

Copy

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

Note: There is a known issue with some versions of PHP that causes problems with

The `PRIVILEGE` value in this example syntax defines what actions the user is allowed to perform on the specified `database` and `table`. You can grant multiple privileges to the same user in one command by separating each with a comma. You can also grant a user privileges globally by entering asterisks (`*`) in place of the database and table names. In SQL, asterisks are special characters used to represent “all” databases or tables.

To illustrate, the following command grants a user global privileges to `CREATE`, `ALTER`, and `DROP` databases, tables, and users, as well as the power to `INSERT`, `UPDATE`, and `DELETE` data from any table on the server. It also grants the user the ability to query data with `SELECT`, create foreign keys with the `REFERENCES` keyword, and perform `FLUSH` operations with the `RELOAD` privilege. However, you should only grant users the permissions they need, so feel free to adjust your own user’s privileges as necessary.

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

Run this `GRANT` statement, replacing `sammy` with your own MySQL user’s name, to grant these privileges to your user:

Copy

Warning: Some users may want to grant their MySQL user the `ALL PRIVILEGES` privilege, which will provide them with broad superuser privileges akin to the **root** user's privileges, like so:

Copy

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

Such broad privileges **should not be granted lightly**, as anyone with access to this MySQL user will have complete control over every database on the server.

Following this, it's good practice to run the `FLUSH PRIVILEGES` command. This will free up any memory that the server cached as a result of the preceding `CREATE USER` and `GRANT` statements:

Copy

```
mysql> FLUSH PRIVILEGES;
```

Then you can exit the MySQL client:

Copy

```
mysql> exit
```

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

Copy

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

Regardless of how you installed it, MySQL should have started running automatically. To test this, check its status.

Copy

```
$ systemctl status mysql.service
```

You'll see output similar to the following:

```
Output
● mysql.service - MySQL Community Server
     Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabl
     Active: active (running) since Tue 2020-04-21 12:56:48 UTC; 6min ago
   Main PID: 10382 (mysqld)
     Status: "Server is operational"
      Tasks: 39 (limit: 1137)
     Memory: 370.0M
     CGroup: /system.slice/mysql.service
             └─10382 /usr/sbin/mysqld
```

If MySQL isn't running, you can start it with `sudo systemctl start mysql`.

For an additional check, you can try connecting to the database using the `mysqladmin` tool, which is a client that lets you run administrative commands. For example, this command says to connect as a MySQL user named **sammy** (`-u sammy`), prompt for a password (`-p`), and return the version. Be sure to change `sammy` to the name of your dedicated MySQL user, and enter that user's password when prompted:

Copy

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

You should see output similar to this:

```
owners.

Server version        8.0.19-0ubuntu5
Protocol version    10
Connection        Localhost via UNIX socket
UNIX socket        /var/run/mysqld/mysqld.sock
Uptime:            10 min 44 sec

Threads: 2  Questions: 25  Slow queries: 0  Opens: 149  Flush tables: 3  Open tables:
```

This means MySQL is up and running.

Conclusion

You now have a basic MySQL setup installed on your server. Here are a few examples of next steps you can take:

- Set up a LAMP stack or a LEMP stack
- Practice running queries with SQL
- Manage your MySQL installation with phpMyAdmin

hi! i write do.co/docs now, but i used to be the senior tech editor publishing tutorials here in the community.

Mark Drake Author

Manager, Developer Education

Technical Writer @ DigitalOcean

Still looking for an answer? Ask a question

Search for more help

Was this helpful? Yes No

Comments

10 Comments

Leave a comment...

This textbox defaults to using `Markdown` to format your answer.

melvinpg7 • August 11, 2021

Hello Mark, Thanks for the share of this tutorial, it helped me a lot, First time I do this without any issues and very clear all your steps on this article. Appreciate it. Thank You.

Reply

vlada972010 • May 1, 2020

Hello,

How I can install MySQL 5? This is very important to me for using Magento 1 version.

Thank you.

Show replies Reply

Dexel • November 20, 2023

```
# mysql
```

ERROR 1045 (28000): Access denied for user ‘root’@‘localhost’ (using password: NO)

```
# mysql -u root
```

ERROR 1045 (28000): Access denied for user ‘root’@‘localhost’ (using password: NO)

/var/run/mysqld/mysqld.sock (2)

Show replies Reply

d.j.bidossessi • October 15, 2023

i like it. This is so clear. Thanks you so much.

Reply

OnStandBy • March 17, 2023

Do anybody else have this issue upon running `sud apt install mysql-server` on a fresh droplet?

```
Setting up mysql-server-8.0 (8.0.32-0ubuntu0.22.10.2) ...
update-alternatives: using /etc/mysql/mysql.cnf to provide /etc/mysql/my.cnf
Renaming removed key_buffer and myisam-recover options (if present)
mysqld will log errors to /var/log/mysql/error.log
2023-03-17T17:42:33.732349Z 0 [ERROR] [MY-011065] [Server] Unable to determi
2023-03-17T17:42:33.733195Z 0 [ERROR] [MY-010946] [Server] Failed to start m
Warning: Unable to start the server.
Created symlink /etc/systemd/system/multi-user.target.wants/mysql.service →
Job for mysql.service failed.
See "systemctl status mysql.service" and "journalctl -xeu mysql.service" for
invoke-rc.d: initscript mysql, action "start" failed.
● mysql.service - MySQL Community Server
     Loaded: loaded (/lib/systemd/system/mysql.service; enabled; preset: ena
     Active: activating (auto-restart) (Result: oom-kill) since Fri 2023-03-
    Process: 2883 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (cod
    Process: 2891 ExecStart=/usr/sbin/mysqld (code=killed, signal=KILL)
   Main PID: 2891 (code=killed, signal=KILL)
     Status: "Server startup in progress"
        CPU: 624ms
dpkg: error processing package mysql-server-8.0 (--configure):
 installed mysql-server-8.0 package post-installation script subprocess retu
Setting up libcgi-pm-perl (4.54-1) ...
Setting up libhtml-template-perl (2.97-2) ...
```

```
Processing triggers for man-db (2.10.2-2) ...
No apport report written because the error message indicates its a followup
             Processing triggers for libc-bin (2.36-0ubuntu4) ...
Errors were encountered while processing:
 mysql-server-8.0
 mysql-server
needrestart is being skipped since dpkg has failed
E: Sub-process /usr/bin/dpkg returned an error code (1)
```

I don’t see where I went wrong in copying-pasting 2 commands

Show replies Reply

Jimmy Schwarz • January 25, 2023

Note, this will not work on the $4 instance but works fine on the $6 ones. If you have a $4 and upgrade to $6 it will work as well.

Show replies Reply

Vivek • October 13, 2022

I recently ran into the issue of not being able to log in after Step 2- mysql_secure_installation in WSL2

```
sudo mysql
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'pass
exit
sudo mysql_secure_installation
# various prompts within this step

mysql -u root -p
```

connection

```
mysql -u root -p --protocol=tcp
```

OR use sudo

```
sudo mysql -u root -p
[sudo] password for vivek:  <---- the sudo user password
Enter password: <---- mysql password
```

Show replies Reply

jaredsmith • June 1, 2022

“sudo mysql_secure_installation” on a brand-new Ubuntu 20.04 provided by Digital Ocean gave the following error after attempting to set the root password:

“Failed! Error: SET PASSWORD has no significance for user ‘root’@‘localhost’ as the authentication method used doesn’t store authentication data in the MySQL server. Please consider using ALTER USER instead if you want to change authentication parameters.”

So to workaround this, use CTRL-C to cancel the mysql_secure_installation script and do the following:

```
sudo mysql
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password by '<roo
quit
```

Then you can run “sudo mysql_secure_installation” again and this time, decline the option to change the root password so you can perform the other steps of

Show replies Reply

7a90d00da452a7235a973cf444f933 • February 16, 2022

This will not work on a server with 512mb RAM. To fix, add some swap space.

But you can add some swap fairly easily. I used the instructions at this link, but there were some other good simple tutorials if this no longer works: https://salslab.com/a/running-mysql-on-a-vps-with-512mb-ram/

Show replies Reply

Ganesh Dhumal • January 19, 2022

Great Article it worked for me

Reply

Load more comments

Sign up

Popular Topics

Ubuntu

Linux Basics

JavaScript

Python

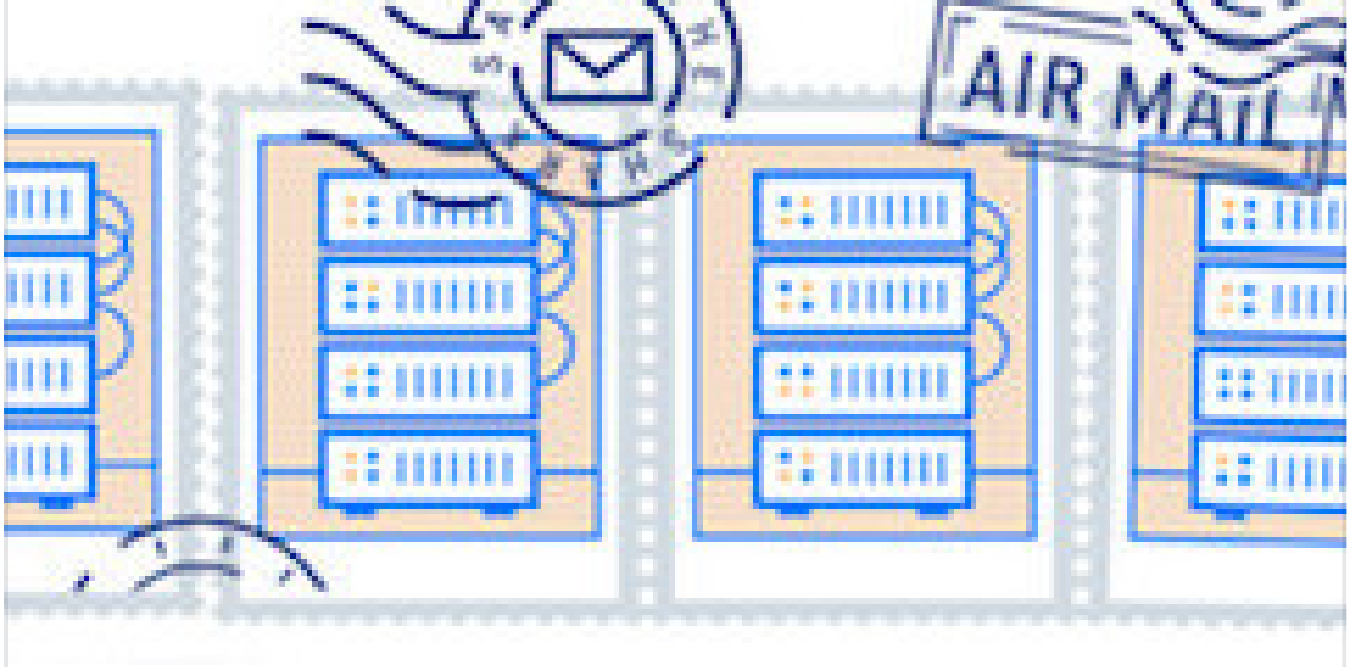
MySQL

Docker

Kubernetes

All tutorials →

Talk to an expert →



Get our biweekly newsletter

Sign up for Infrastructure as a Newsletter.

Sign up →

Hollie's Hub for Good

Working on improving health and education, reducing inequality, and spurring economic growth? We'd like to help.

Learn more →

Featured Tutorials

Kubernetes Course Learn Python 3 Machine Learning in Python

Getting started with Go Intro to Kubernetes

DigitalOcean Products

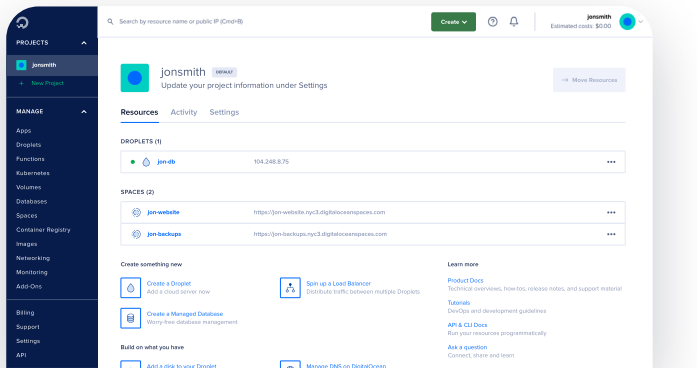
Cloudways Virtual Machines Managed Databases Managed Kubernetes

Block Storage Object Storage Marketplace VPC Load Balancers

Welcome to the developer cloud

DigitalOcean makes it simple to launch in the cloud and scale up as you grow — whether you're running one virtual machine or ten thousand.

Learn more



Company

Solutions

Contact

 Sitemap. Cookie Preferences