



// Tutorial //

How To Create a New User and Grant Permissions in MySQL

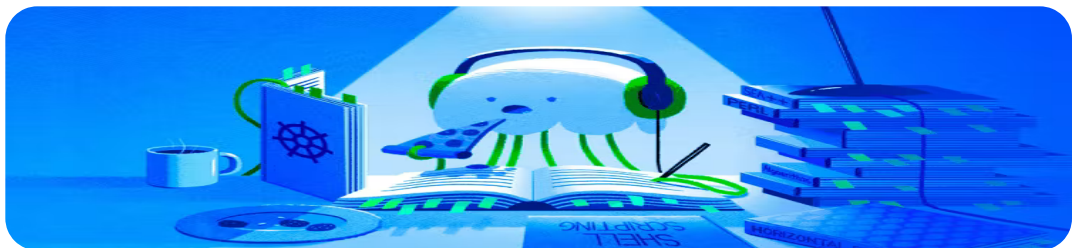
Published on June 12, 2012 · Updated on March 18, 2022

MySQL

By [Etel Sverdlov](#)

Developer and author at DigitalOcean.

English ▾



Introduction

[MySQL](#) is an open-source relational database management system. It is commonly deployed as part of the *LAMP stack* (which stands for **L**inux, **A**pache, **M**ySQL, and **P**HP) and, as of this writing, is the [most popular open-source database](#) in the world.

This guide outlines how to create a new MySQL user and grant them the permissions needed to perform a variety of actions.

Prerequisites

In order to follow along with this guide, you'll need access to a MySQL database. This guide assumes that this database is installed on a virtual private server running Ubuntu 20.04, though the principles it outlines should be applicable regardless of how you access your database.

If you don't have access to a MySQL database and would like to set one up yourself, you can follow one of our guides on [How To Install MySQL](#). Again, regardless of your server's underlying operating system, the methods for creating a new MySQL user and granting them permissions will generally be the same.

You could alternatively spin up a MySQL database managed by a cloud provider. For details on how to spin up a DigitalOcean Managed Database, see our [product documentation](#).

Please note that any portions of example commands that you need to change or customize will be highlighted like this throughout this guide.

Creating a New User

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. This means that you need to precede the `mysql` command with `sudo` to invoke it with the privileges of the **root** Ubuntu user in order to gain access to the **root** MySQL user:

```
$ sudo mysql
```

Copy

Note: If your **root** MySQL user is configured to authenticate with a password, 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 with the correct password:

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

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:

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

Copy

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:

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

Copy

Note: There is a known issue with some versions of PHP that causes problems with `caching_sha2_password`. If you plan to use this database with a PHP application — phpMyAdmin, for example — you may want to create a user that will authenticate with the older, though still secure, `mysql_native_password` plugin instead:

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

Copy

If you aren't sure, you can always create a user that authenticates with `caching_sha2_password` and then `ALTER` it later on with this command:

```
mysql> ALTER USER 'sammy'@'localhost' IDENTIFIED WITH mysql_native_password BY 'password';
```

Copy

After creating your new user, you can grant them the appropriate privileges.

Granting a User Permissions

CONTENTS

Introduction

Prerequisites

Creating a New User

Granting a User Permissions

Conclusion

RELATED

How To Install Linux, Apache, MySQL, PHP (LAMP) stack on Ubuntu

[Tutorial](#)

How To Install Rails, Apache, and MySQL on Ubuntu with Passenger

[Tutorial](#)

The general syntax for granting user privileges is as follows:

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

[Copy](#)

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:

```
mysql> GRANT CREATE, ALTER, DROP, INSERT, UPDATE, DELETE, SELECT, REFERENCES, RELOAD ( Copy T
```



Note that this statement also includes `WITH GRANT OPTION`. This will allow your MySQL user to grant any permissions that it has to other users on the system.

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:

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

[Copy](#)

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.

Many guides suggest running the `FLUSH PRIVILEGES` command immediately after a `CREATE USER` or `GRANT` statement in order to reload the grant tables to ensure that the new privileges are put into effect:

```
mysql> FLUSH PRIVILEGES;
```

[Copy](#)

However, according to the [official MySQL documentation](#), when you modify the grant tables indirectly with an account management statement like `GRANT`, the database will reload the grant tables immediately into memory, meaning that the `FLUSH PRIVILEGES` command isn't necessary in our case. On the other hand, running it won't have any negative effect on the system.

If you need to revoke a permission, the structure is almost identical to granting it:

```
mysql> REVOKE type_of_permission ON database_name.table_name FROM 'username'@'host' Copy
```

Note that when revoking permissions, the syntax requires that you use `FROM`, instead of `TO` which you used when granting the permissions.

You can review a user's current permissions by running the `SHOW GRANTS` command:

```
mysql> SHOW GRANTS FOR 'username'@'host';
```

[Copy](#)

Just as you can delete databases with `DROP`, you can use `DROP` to delete a user:

```
mysql> DROP USER 'username'@'localhost';
```

[Copy](#)

After creating your MySQL user and granting them privileges, you can exit the MySQL client:

```
mysql> exit
```

[Copy](#)

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

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

[Copy](#)

The `-p` flag will cause the MySQL client to prompt you for your MySQL user's password in order to authenticate.

Conclusion

By following this tutorial, you've learned how to add new users and grant them a variety of permissions in a MySQL database. From here, you could continue to explore and experiment with different permissions settings for your MySQL user, or you may want to learn more about some higher-level MySQL configurations.

For more information about the basics of MySQL, you can check out the following tutorials:

- [How To Create and Manage Databases in MySQL and MariaDB on a Cloud Server](#)
- [How To Set Up Replication in MySQL](#)
- [How To Configure MySQL Group Replication on Ubuntu 20.04](#)

Want to learn more? Join the DigitalOcean Community!

Join our DigitalOcean community of over a million developers for free! Get help and share knowledge in our Questions & Answers section, find tutorials and tools that will help you grow as a developer and scale your project or business, and subscribe to topics of interest.

Sign up →

About the authors



[Etel Sverdlov](#) Author

Developer and author at DigitalOcean.

Still looking for an answer?

[Ask a question](#)

[Search for more help](#)

Was this helpful?

Yes

No



Comments

10 Comments

B I U ↶ ↷ ↺ ↻ H₁ H₂ H₃ ☰ 1. “,” ⓘ ☐ <>



Leave a comment...

Login to Comment

ejkuhn • September 26, 2016

Thanks for the great tutorial. One comment: apparently FLUSH PRIVILEGES is unnecessary in most cases:

<http://dbahire.com/stop-using-flush-privileges/>

[Reply](#)**asad** • June 27, 2014

You guys have an article for everything...so psycho its awesome!

[Reply](#)**petre.octavian** • December 12, 2013

Hi,

I have done all the above but I can't seem to log with the new user with:

```
mysql -u [username]-p
```

I have the error:

```
ERROR 1045 (28000): Access denied for user 'prosper202'@'localhost' (using password: YES)
```

What may be the problem?

[Reply](#)**petre.octavian** • December 12, 2013

Hi,

I have done all the above but I can't seem to log with the new user with:

```
mysql -u [username]-p
```

I have the error:

I have the error:

ERROR 1045 (28000): Access denied for user 'prosper202'@'localhost' (using password: YES)

What may be the problem?

[Reply](#)

[Etel Sverdlov](#) • March 7, 2013

Updated

[Reply](#)

[chris96043](#) • February 14, 2013

You can grant multiple privileges in one command by separating them with commas: eg: "GRANT UPDATE, SELECT ON [database name].[table name] TO '[username]'@'localhost';"

[Reply](#)

[nabakhteveli17](#) • December 7, 2021

Thank you!

[Reply](#)

[sekrete](#) • August 8, 2021

You don't need to FLUSH PRIVILEGES after CREATE USER, GRANT and so on. Please verify. You'll need it only if you manually edit privileges tables.

[Reply](#)

[digitaloceana4d99985a4a534](#) • December 25, 2020

Excellent tutorial, thanks!

You should also mention that you can use the 'ALTER' command to make updates to existing users.

[Reply](#)

[Colin Wagaba](#) • September 7, 2020

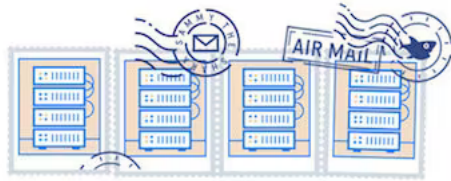
Very helpful, thanks.

[Reply](#)

[Load More Comments](#)

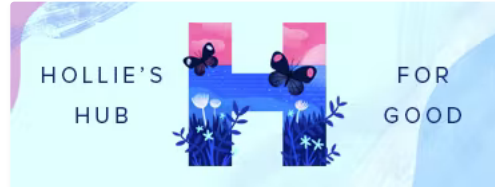


This work is licensed under a Creative Commons Attribution-NonCommercial- ShareAlike 4.0 International License.



GET OUR BIWEEKLY NEWSLETTER

Sign up for Infrastructure as a Newsletter.



HOLLIE'S HUB FOR GOOD

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



BECOME A CONTRIBUTOR

You get paid; we donate to tech nonprofits.

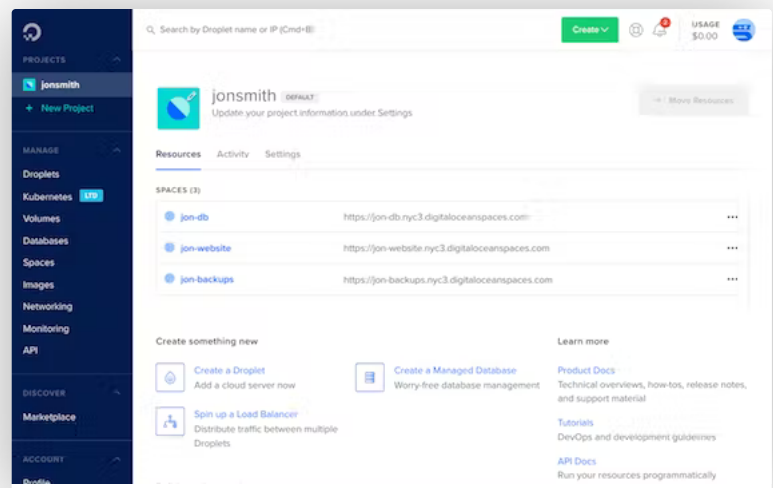
Featured on [Community](#) [Kubernetes Course](#) [Learn Python 3](#) [Machine Learning in Python](#) [Getting started with Go](#) [Intro to Kubernetes](#)

[DigitalOcean Products](#) [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](#)



© 2022 DigitalOcean, LLC. All rights reserved.

[Company](#)

[About](#)

[Leadership](#)

[Products](#)

[Pricing](#)

[Products Overview](#)

[Droplets](#)

[Community](#)

[Tutorials](#)

[Q&A](#)

Blog	Kubernetes	Tools and Integrations
Careers	Managed Databases	Tags
Partners	Spaces	Write for DigitalOcean
Referral Program	Marketplace	Presentation Grants
Press	Load Balancers	Hatch Startup Program
Legal	Block Storage	Shop Swag
Security & Trust Center	API Documentation	Research Program
	Documentation	Open Source
	Release Notes	Code of Conduct
Contact		
Get Support		
Trouble Signing In?		
Sales		
Report Abuse		
System Status		
Share your ideas		