# **clarPractical 13 Install MySQL on Ubuntu 18.04**

### Introduction

[MySQL](https://www.mysql.com/) is an open-source database management system, commonly installed as part of the popular [LAMP](https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-ubuntu-18-04) (Linux, Apache, MySQL, PHP/Python/Perl) stack. It uses a relational database and SQL (Structured Query Language) to manage its data.

The short version of the installation is simple: update your package index, install the mysql-serverpackage, and then run the included security script.

sudo apt update

sudo apt install mysql-server

sudo mysql\_secure\_installation

This tutorial will explain how to install MySQL version 5.7 on an Ubuntu 18.04 server. However, if you're looking to update an existing MySQL installation to version 5.7, you can read [this MySQL 5.7 update guide](https://www.digitalocean.com/community/tutorials/how-to-prepare-for-your-mysql-5-7-upgrade) instead.

## Prerequisites

To follow this tutorial, you will need:

* One Ubuntu 18.04 server set up by following [this initial server setup guide](https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-18-04), including a non-root user with sudo privileges and a firewall.

## Step 1 — Installing MySQL

On Ubuntu 18.04, only the latest version of MySQL is included in the APT package repository by default. At the time of writing, that's MySQL 5.7

To install it, update the package index on your server with apt:

**sudo apt update**

Then install the default 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. Because this leaves your installation of MySQL insecure, we will address this next.

## Step 2 — Configuring MySQL

For fresh installations, you'll want to run the included security script. This changes some of the less secure default options for things like remote root logins and sample users. On older versions of MySQL, you needed to initialize the data directory manually as well, but this is done automatically now.

Run the security script:

**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 strength of your MySQL password. Regardless of your choice, the next prompt will be to set a password for the MySQL root user. Enter and then confirm a secure password of your choice.

(Password for assignment : Ubuntu1@#)

(password is ubuntu)

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.

To initialize the MySQL data directory, you would use mysql\_install\_db for versions before 5.7.6, and mysqld --initialize for 5.7.6 and later. However, if you installed MySQL from the Debian distribution, as described in Step 1, the data directory was initialized automatically; you don't have to do anything. If you try running the command anyway, you'll see the following error:

Output

mysqld: Can't create directory '/var/lib/mysql/' (Errcode: 17 - File exists)

. . .

2018-04-23T13:48:00.572066Z 0 [ERROR] Aborting

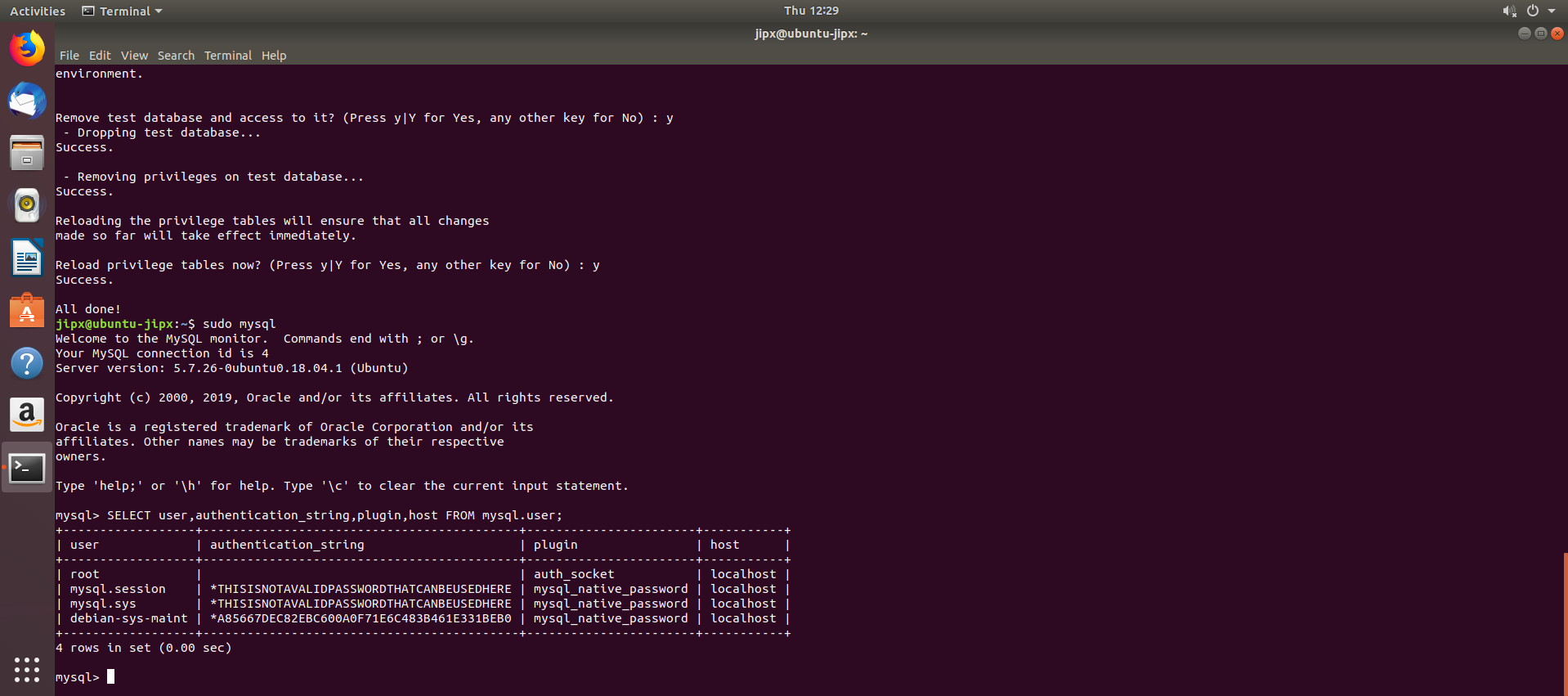
Note that even though you’ve set a password for the root MySQL user, this user is not configured to authenticate with a password when connecting to the MySQL shell. If you’d like, you can adjust this setting by following Step 3.

## Step 3 — (Optional) Adjusting User Authentication and 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 allows for some greater security and usability in many cases, but it can also complicate things when you need to allow an external program (e.g., phpMyAdmin) to access the user.

In order to use a password to connect to MySQL as root, you will need to switch its authentication method from auth\_socket to mysql\_native\_password. To do this, open up the MySQL prompt from your terminal:

**sudo mysql**

Next, check which authentication method each of your MySQL user accounts use with the following command:  
**SELECT user,authentication\_string,plugin,host FROM mysql.user;**

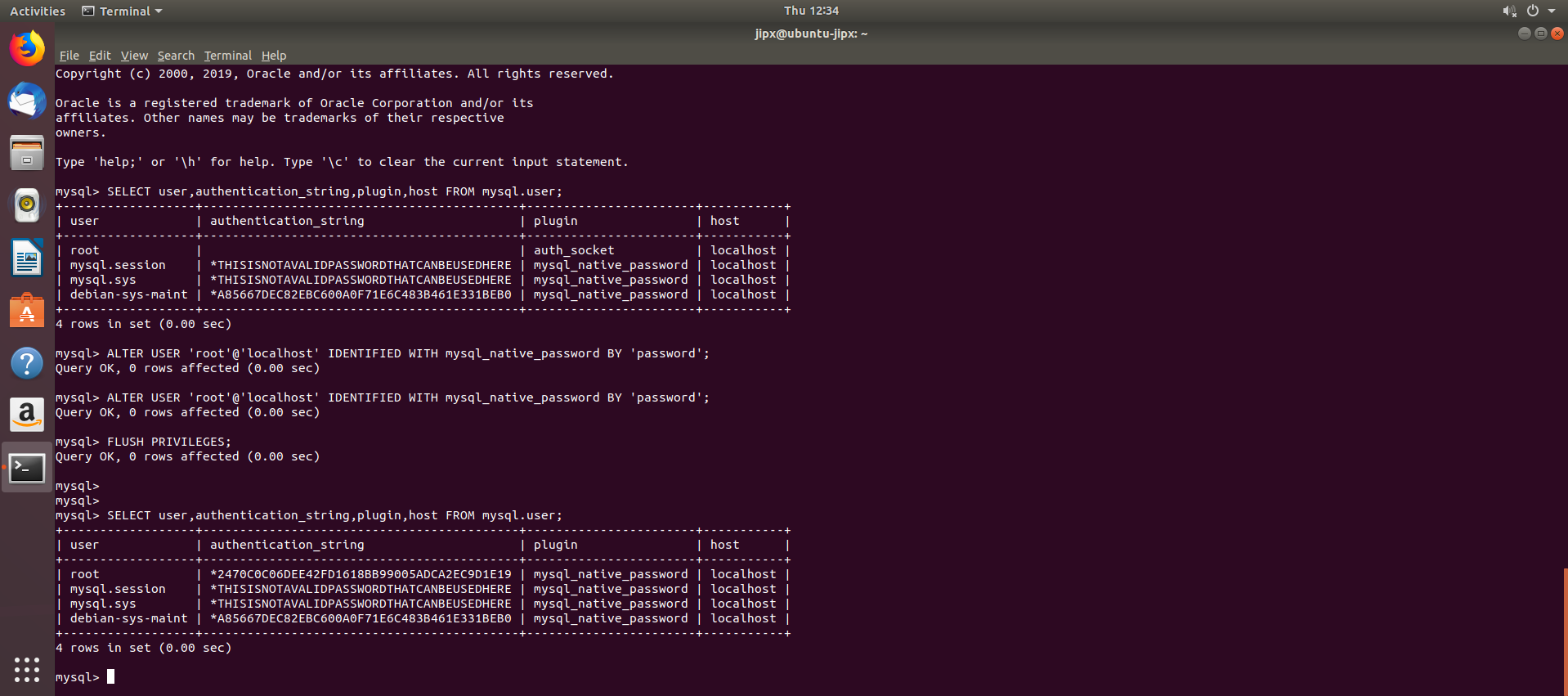
In this example, you can see that the root user does in fact authenticate using the auth\_socketplugin. To configure the root account to authenticate with a password, run the following ALTER USER command. Be sure to change password to a strong password of your choosing, and note that this command will change the root password you set in Step 2:  
**ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY 'Ubuntu1@#';**

Then, run FLUSH PRIVILEGES which tells the server to reload the grant tables and put your new changes into effect:

**FLUSH PRIVILEGES;**

Check the authentication methods employed by each of your users again to confirm that root no longer authenticates using the auth\_socket plugin:

**SELECT user,authentication\_string,plugin,host FROM mysql.user;**



You can see in this example output that the root MySQL user now authenticates using a password. Once you confirm this on your own server, you can exit the MySQL shell:

**Exit**

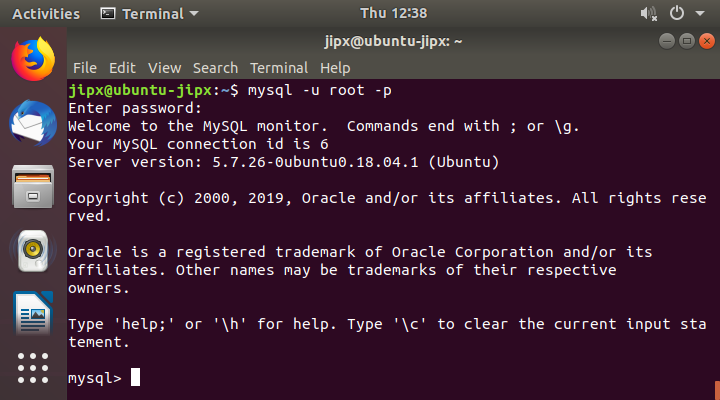
**(Optional)**

(This part onwards need to do)

Log in to mysql: (I stop here)

sudo systemctl status mysql ( checks whether mysql is active )

**sudo mysql -u root -p (Login mysql using username of root)**



Create a new user and give it a strong password:

**CREATE USER 'sammy'@'localhost' IDENTIFIED BY 'Ubuntu1@#';**

Then, grant your new user the appropriate privileges. For example, you could grant the user privileges to all tables within the database, as well as the power to add, change, and remove user privileges, with this command:

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

Note that, at this point, you do not need to run the FLUSH PRIVILEGES command again. This command is only needed when you modify the grant tables using statements like INSERT, UPDATE, or DELETE. Because you created a new user, instead of modifying an existing one, FLUSH PRIVILEGES is unnecessary here.

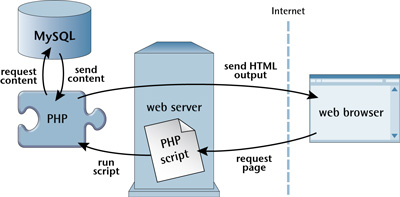
Following this, exit the MySQL shell:

**exit**

Finally, let's test the MySQL installation.

mysql -u sammy -p

Create a user for Wordpress to communicate with MYSQL database.  
**To create a database user and database for your WordPress installation**

Your WordPress installation needs to store information, such as blog posts and user comments, in a database. This procedure helps you create your blog's database and a user that is authorized to read and save information to it.  
  


1. Start the database server.

**sudo systemctl start mysql**

1. Log in to the database server as the root user. Enter your database root password when prompted; this may be different than your root system password, or it may even be empty if you have not secured your database server.

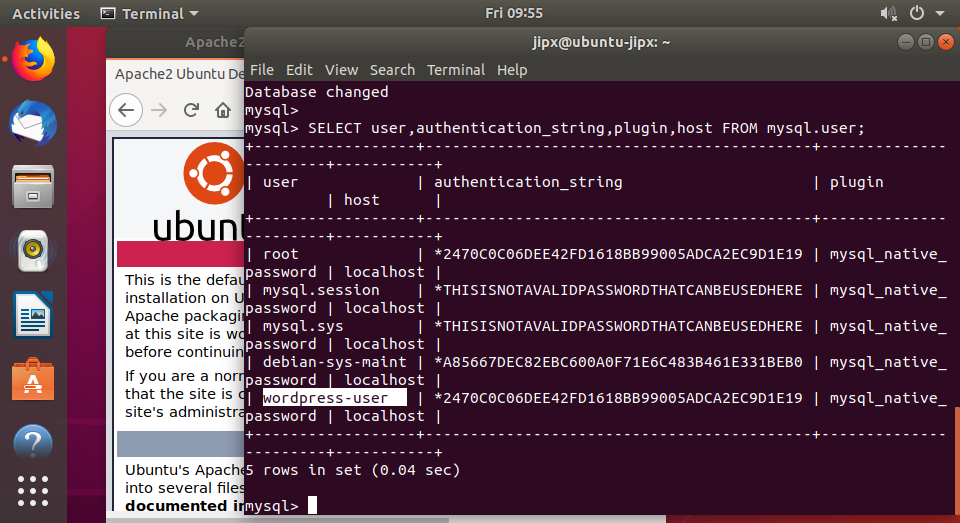
If you have not secured your database server yet, it is important that you do so. For more information, see: [To secure the database server](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/install-LAMP.html#SecuringMySQLProcedure).

**sudo mysql -u root -p**

1. Create a user and password for your MySQL database. Your WordPress installation uses these values to communicate with your MySQL database. Enter the following command, substituting a unique user name and password.

**CREATE USER '*wordpress-user*'@'localhost' IDENTIFIED BY '*Ubuntu1@#*';**

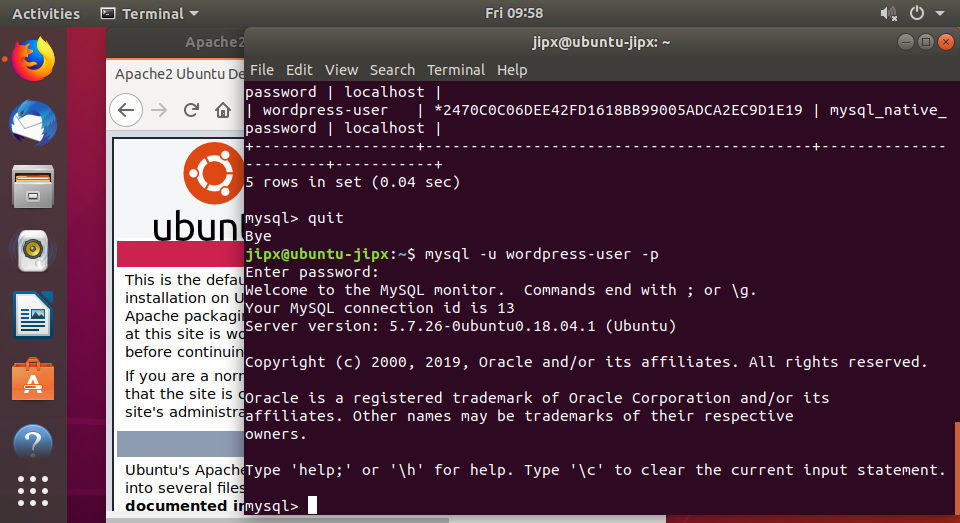
Make sure that you create a strong password for your user. Do not use the single quote character ( ' ) in your password, because this will break the preceding command. For more information about creating a secure password, go to <http://www.pctools.com/guides/password/>. Do not reuse an existing password, and make sure to store this password in a safe place.

Verify the user “wordpress-user” is created:  


select user,authentication\_string from mysql.user;

Alternatively you can log in to mysql using “wordpress-user” (Password for wordpress is Ubuntu1@#)

mysql -u wordpress-user -pUbuntu1@#



1. Create your database. Give your database a descriptive, meaningful name, such as wordpress-db.

**Note**

The punctuation marks surrounding the database name in the command below are called backticks. The backtick (`) key is usually located above the Tab key on a standard keyboard. Backticks are not always required, but they allow you to use otherwise illegal characters, such as hyphens, in database names.

**CREATE DATABASE `*wordpress-db*`;**

1. Grant full privileges for your database to the WordPress user that you created earlier.

**GRANT ALL PRIVILEGES ON `*wordpress-db*`.\* TO "*wordpress-user*"@"localhost";**

1. Flush the database privileges to pick up all of your changes.

**FLUSH PRIVILEGES;**

1. Exit the mysql client.

**exit**

Document the above:

|  |  |  |
| --- | --- | --- |
| DB\_NAME | DB\_USER | DB\_PASSWORD |
| wordpress-db | wordpress-user | Ubuntu1@# |

## Step 4 — Testing MySQL

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

systemctl status mysql

You'll see output similar to the following:

Output

● mysql.service - MySQL Community Server

Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: en

Active: active (running) since Wed 2018-04-23 21:21:25 UTC; 30min ago

Main PID: 3754 (mysqld)

Tasks: 28

Memory: 142.3M

CPU: 1.994s

CGroup: /system.slice/mysql.service

└─3754 /usr/sbin/mysqld

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

**(OPTIONAL)**

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 to MySQL as root (-u root), prompt for a password (-p), and return the version.

**sudo mysqladmin -p -u root version**

You should see output similar to this:

Output

mysqladmin Ver 8.42 Distrib 5.7.21, for Linux on x86\_64

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.

Server version 5.7.21-1ubuntu1

Protocol version 10

Connection Localhost via UNIX socket

UNIX socket /var/run/mysqld/mysqld.sock

Uptime: 30 min 54 sec

Threads: 1 Questions: 12 Slow queries: 0 Opens: 115 Flush tables: 1 Open tables: 34 Queries per second avg: 0.006

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:

* [Implement some additional security measures](https://www.digitalocean.com/community/tutorials/how-to-secure-mysql-and-mariadb-databases-in-a-linux-vps)
* [Relocate the data directory](https://www.digitalocean.com/community/tutorials/how-to-move-a-mysql-data-directory-to-a-new-location-on-ubuntu-16-04)
* [Manage your MySQL servers with SaltStack](https://www.digitalocean.com/community/tutorials/saltstack-infrastructure-creating-salt-states-for-mysql-database-servers)

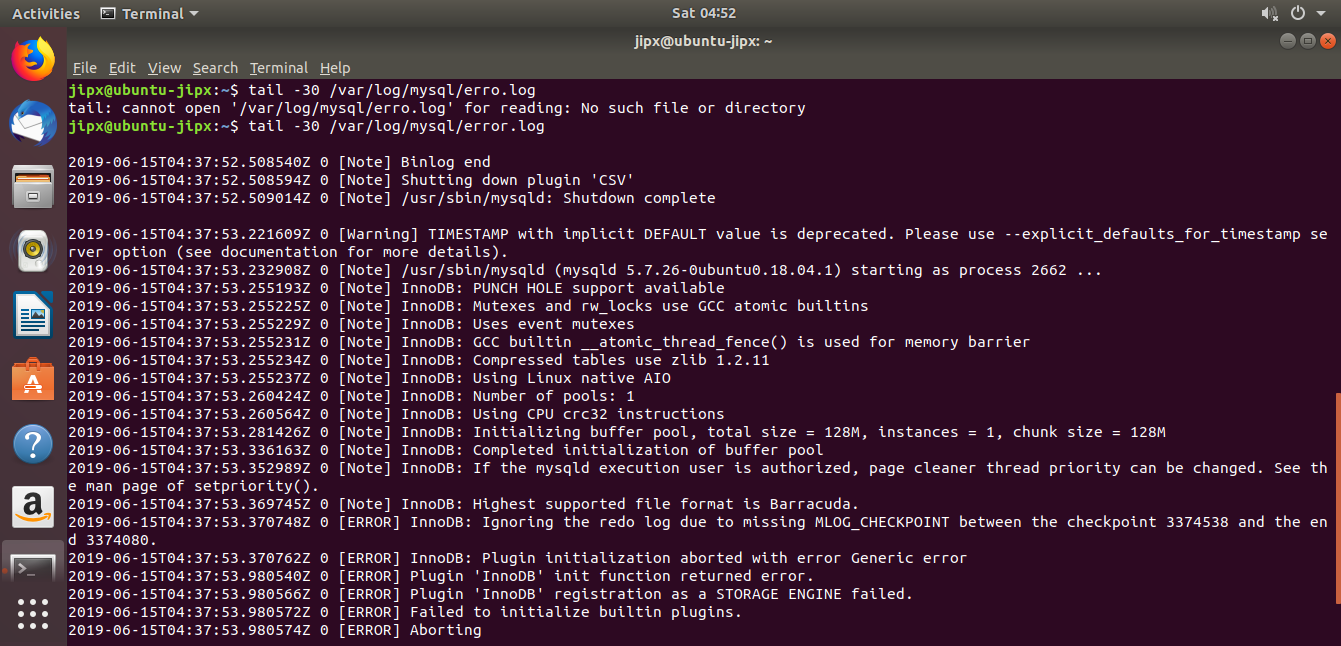
<https://www.digitalocean.com/community/tutorials/how-to-install-mysql-on-ubuntu-18-04>

Troubleshooting

|  |
| --- |
| jipx@ubuntu-jipx:~$ sudo systemctl status mysql  [sudo] password for jipx:  ● mysql.service - MySQL Community Server  Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)  Active: failed (Result: exit-code) since Sat 2019-06-15 04:37:54 UTC; 19min ago  Process: 2660 ExecStart=/usr/sbin/mysqld --daemonize --pid-file=/run/mysqld/mysqld.pid (code=exited, status=1/FAILURE)  Process: 2641 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=exited, status=0/SUCCESS)  Jun 15 04:37:54 ubuntu-jipx systemd[1]: mysql.service: Failed with result 'exit-code'.  Jun 15 04:37:54 ubuntu-jipx systemd[1]: Failed to start MySQL Community Server.  Jun 15 04:37:54 ubuntu-jipx systemd[1]: mysql.service: Service hold-off time over, scheduling restart.  Jun 15 04:37:54 ubuntu-jipx systemd[1]: mysql.service: Scheduled restart job, restart counter is at 5.  Jun 15 04:37:54 ubuntu-jipx systemd[1]: Stopped MySQL Community Server.  Jun 15 04:37:54 ubuntu-jipx systemd[1]: mysql.service: Start request repeated too quickly.  Jun 15 04:37:54 ubuntu-jipx systemd[1]: mysql.service: Failed with result 'exit-code'.  Jun 15 04:37:54 ubuntu-jipx systemd[1]: **Failed to start MySQL Community Server.**  jipx@ubuntu-jipx:~$ |

Root Cause:  
WordPress needs 1GB RAM minimum in production. You might be able to run on 512MB, but if just a couple of people visit your site at the same time and WordPress is checking for updates, then you can easily run out of RAM.

|  |
| --- |
| MYSQL Error Logfile location on **Linux:** /var/log/mysql/error.log  tail -30 /var/log/mysql/error.log |



Solution

|  |
| --- |
| 1. Move [redo log files](https://dev.mysql.com/doc/refman/5.7/en/innodb-redo-log.html) ib\_logfile0 and ib\_logfile1 to another directory. In this example, the files are move to /root:  # mv /var/lib/mysql/ib\_log\* /root 2. Start MySQL/MariaDB:  sudo systemctl start mysql  Reference:<https://support.plesk.com/hc/en-us/articles/115001738733> |

|  |
| --- |
| If you need to re-install mysql from scratch, you may like to **[Completely remove mysql 5.7](https://askubuntu.com/questions/1052950/completely-remove-mysql-5-7)** Try this:  sudo apt purge mysql-\*  Afterwards run:  sudo apt autoremove |

* + - 1. What have you learnt?

Today, I have learned how to install mysql onto my ubuntu and create a word press account in it. I also learned from there I could locate my wordpress page by visiting <http://localhost/p2006264> . From there, I was able to customize and edit my word press website

2. Difficulties encountered and how you solved the problems?

I encountered one very prominent difficulty when setting up mysql. Sometimes when I ran the command sudo mysql, It will not allow access to it. This problem can be resolved if I went back to the previous snapshot.