How To Install Linux, Nginx, MySQL, PHP (LEMP stack) on Ubuntu 18.04

53

Posted May 23, 2018 143k LEMP DATABASES NGINX MYSQL PHP UBUNTU 18.04

By: Justin Ellingwood By: Mark Drake

Not using **Ubuntu 18.04**? Choose a different version:

Introduction

The LEMP software stack is a group of software that can be used to serve dynamic web pages and web applications. This is an acronym that describes a **L**inux operating system, with an Nginx (pronounced like "**E**ngine-X") web server. The backend data is stored in the **M**ySQL database and the dynamic processing is handled by **P**HP.

This guide demonstrates how to install a LEMP stack on an Ubuntu 18.04 server. The Ubuntu operating system takes care of the first requirement. We will describe how to get the rest of the components up and running.

Prerequisites

Before you complete this tutorial, you should have a regular, non-root user account on your server with `sudo` privileges. Set up this account by completing our initial server setup guide for Ubuntu 18.04.

Once you have your user available, you are ready to begin the steps outlined in this guide.

Step 1 – Installing the Nginx Web Server

In order to display web pages to our site visitors, we are going to employ Nginx, a modern, efficient web server.

All of the software used in this procedure will come from Ubuntu's default package repositories. This means we can use the `apt` package management suite to complete the necessary installations.

Since this is our first time using `apt` for this session, start off by updating your server's package index. Following that, install the server:

```
$ sudo apt update
$ sudo apt install nginx
```

On Ubuntu 18.04, Nginx is configured to start running upon installation.

If you have the `ufw` firewall running, as outlined in the initial setup guide, you will need to allow connections to Nginx. Nginx registers itself with `ufw` upon installation, so the procedure is rather straightforward.

It is recommended that you enable the most restrictive profile that will still allow the traffic you want. Since you haven't configured SSL for your server in this guide, you will only need to allow traffic on port `80`.

Enable this by typing:

```
$ sudo ufw allow 'Nginx HTTP'
```

You can verify the change by running:

```
$ sudo ufw status
```

This command's output will show that HTTP traffic is allowed:

```
Output
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
Nginx HTTP                 ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
Nginx HTTP (v6)            ALLOW       Anywhere (v6)
```

With the new firewall rule added, you can test if the server is up and running by accessing your server's domain name or public IP address in your web browser.

If you do not have a domain name pointed at your server and you do not know your server's public IP address, you can find it by running the following command:

```
$ ip addr show eth0 | grep inet | awk '{ print $2; }' | sed 's/\/.*$//'
```

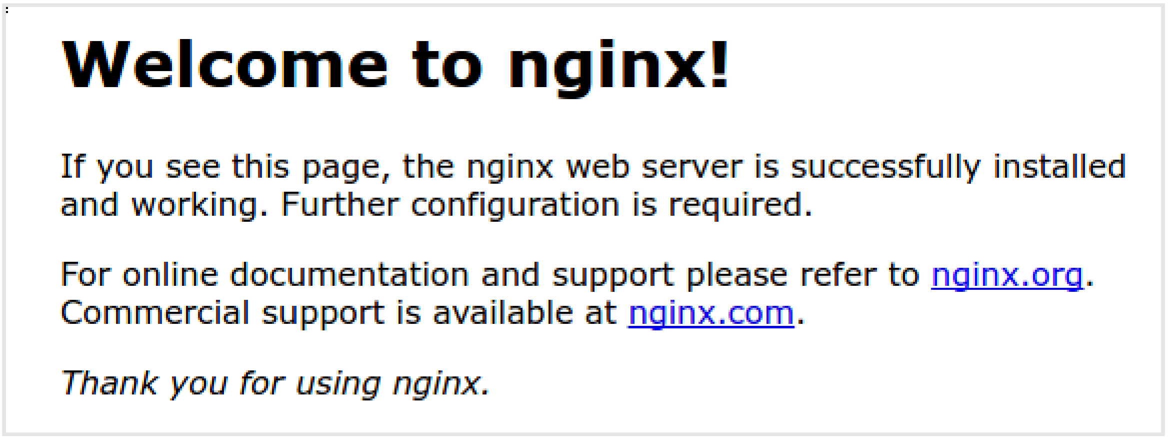
This will print out a few IP addresses. You can try each of them in turn in your web browser.

As an alternative, you can check which IP address is accessible, as viewed from other locations on the internet:

```
$ curl -4 icanhazip.com
```

Type the address that you receive in your web browser and it will take you to Nginx's default landing page:

```
http://server_domain_or_IP
```



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

If you see the above page, you have successfully installed Nginx.

Step 2 – Installing MySQL to Manage Site Data

Now that you have a web server, you need to install MySQL (a database management system) to store and manage the data for your site.

Install MySQL by typing:

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

The MySQL database software is now installed, but its configuration is not yet complete.

To secure the installation, MySQL comes with a script that will ask whether we want to modify some insecure defaults. Initiate the script by typing:

```
$ sudo mysql_secure_installation
```

This script will ask if you want to configure the `VALIDATE PASSWORD PLUGIN`.

Warning: Enabling this feature is something of a judgment call. If enabled, passwords which don't match the specified criteria will be rejected by MySQL with an error. This will cause issues if you use a weak password in conjunction with software which automatically configures MySQL user credentials, such as the Ubuntu packages for phpMyAdmin. It is safe to leave validation disabled, but you should always use strong, unique passwords for database credentials.

Answer `Y` for yes, or anything else to continue without enabling.

```
VALIDATE PASSWORD PLUGIN 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 plugin?

Press y|Y for Yes, any other key for No:
```

If you've enabled validation, the script will also ask you to select a level of password validation. Keep in mind that if you enter **2** – for the strongest level – you will receive errors when attempting to set any password which does not contain numbers, upper and lowercase letters, and special characters, or which is based on common dictionary words.

```
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                  file

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

Next, you'll be asked to submit and confirm a root password:

```
Please set the password for root here.

New password:

Re-enter new password:
```

For the rest of the questions, you should press `Y` and hit the `ENTER` key at each prompt. 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 we have made.

Note that 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.

If using the `auth_socket` plugin to access MySQL fits with your workflow, you can proceed to Step 3. If, however, you prefer to use a password when connecting 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:

```
mysql> SELECT user,authentication_string,plugin,host FROM mysql.user;
```

```
Output
+------------------+-------------------------------------------+-----------------------+-----------+
| user             | authentication_string                     | plugin                | host      |
+------------------+-------------------------------------------+-----------------------+-----------+
| root             |                                           | auth_socket           | localhost |
| mysql.session    | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE | mysql_native_password | localhost |
| mysql.sys        | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE | mysql_native_password | localhost |
| debian-sys-maint | *CC744277A401A7D25BE1CA89AFF17BF607F876FF | mysql_native_password | localhost |
+------------------+-------------------------------------------+-----------------------+-----------+
4 rows in set (0.00 sec)
```

In this example, you can see that the **root** user does in fact authenticate using the `auth_socket` plugin. 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:

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

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

```
mysql> 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:

```
mysql> SELECT user,authentication_string,plugin,host FROM mysql.user;
```

```
Output
+------------------+-------------------------------------------+-----------------------+-----------+
| user             | authentication_string                     | plugin                | host      |
+------------------+-------------------------------------------+-----------------------+-----------+
| root             | *3636DACC8616D997782ADD0839F92C1571D6D78F | mysql_native_password | localhost |
| mysql.session    | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE | mysql_native_password | localhost |
| mysql.sys        | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE | mysql_native_password | localhost |
| debian-sys-maint | *CC744277A401A7D25BE1CA89AFF17BF607F876FF | mysql_native_password | localhost |
+------------------+-------------------------------------------+-----------------------+-----------+
4 rows in set (0.00 sec)
```

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:

```
mysql> exit
```

Note: After configuring your **root** MySQL user to authenticate with a password, you'll no longer be able to access MySQL with the `sudo mysql` command used previously. Instead, you must run the following:

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

After entering the password you just set, you will see the MySQL prompt.

At this point, your database system is now set up and you can move on to installing PHP.

Step 3 – Installing PHP and Configuring Nginx to Use the PHP Processor

You now have Nginx installed to serve your pages and MySQL installed to store and manage your data. However, you still don't have anything that can generate dynamic content. This is where PHP comes into play.

Since Nginx does not contain native PHP processing like some other web servers, you will need to install `php-fpm`, which stands for "fastCGI process manager". We will tell Nginx to pass PHP requests to this software for processing.

Note: Depending on your cloud provider, you may need to add Ubuntu's `universe` repository, which includes free and open-source software maintained by the Ubuntu community, before installing the `php-fpm` package. You can do this by typing:

```
$ sudo add-apt-repository universe
```

Install the `php-fpm` module along with an additional helper package, `php-mysql`, which will allow PHP to communicate with your database backend. The installation will pull in the necessary PHP core files. Do this by typing:

```
$ sudo apt install php-fpm php-mysql
```

You now have all of the required LEMP stack components installed, but you still need to make a few configuration changes in order to tell Nginx to use the PHP processor for dynamic content.

This is done on the server block level (server blocks are similar to Apache's virtual hosts). To do this, open a new server block configuration file within the `/etc/nginx/sites-available/` directory. In this example, the new server block configuration file is named `example.com`, although you can name yours whatever you'd like:

```
$ sudo nano /etc/nginx/sites-available/example.com
```

By editing a new server block configuration file, rather than editing the default one, you'll be able to easily restore the default configuration if you ever need to.

Add the following content, which was taken and slightly modified from the default server block configuration file, to your new server block configuration file:

/etc/nginx/sites-available/example.com

```
server {
        listen 80;
        root /var/www/html;
        index index.php index.html index.htm index.nginx-debian.html;
        server_name example.com;

        location / {
                try_files $uri $uri/ =404;
        }

        location ~ \.php$ {
                include snippets/fastcgi-php.conf;
                fastcgi_pass unix:/var/run/php/php7.2-fpm.sock;
        }

        location ~ /\.ht {
                deny all;
        }
}
```

Here's what each of these directives and location blocks do:

- `listen` — Defines what port Nginx will listen on. In this case, it will listen on port `80`, the default port for HTTP.
- `root` — Defines the document root where the files served by the website are stored.
- `index` — Configures Nginx to prioritize serving files named `index.php` when an index file is requested, if they're available.
- `server_name` — Defines which server block should be used for a given request to your server. **Point this directive to your server's domain name or public IP address.**
- `location /` — The first location block includes a `try_files` directive, which checks for the existence of files matching a URI request. If Nginx cannot find the appropriate file, it will return a 404 error.
- `location ~ \.php$` — This location block handles the actual PHP processing by pointing Nginx to the `fastcgi-php.conf` configuration file and the `php7.2-fpm.sock` file, which declares what socket is associated with `php-fpm`.
- `location ~ /\.ht` — The last location block deals with `.htaccess` files, which Nginx does not process. By adding the `deny all` directive, if any `.htaccess` files happen to find their way into the document root they will not be served to visitors.

After adding this content, save and close the file. Enable your new server block by creating a symbolic link from your new server block configuration file (in the `/etc/nginx/sites-available/` directory) to the `/etc/nginx/sites-enabled/` directory:

```
$ sudo ln -s /etc/nginx/sites-available/example.com /etc/nginx/sites-enabled/
```

Then, unlink the default configuration file from the `/sites-enabled/` directory:

```
$ sudo unlink /etc/nginx/sites-enabled/default
```

Note: If you ever need to restore the default configuration, you can do so by recreating the symbolic link, like this:

```
$ sudo ln -s /etc/nginx/sites-available/default /etc/nginx/sites-enabled/
```

Test your new configuration file for syntax errors by typing:

```
$ sudo nginx -t
```

If any errors are reported, go back and recheck your file before continuing.

When you are ready, reload Nginx to make the necessary changes:

```
$ sudo systemctl reload nginx
```

This concludes the installation and configuration of your LEMP stack. However, it’s prudent to confirm that all of the components can communicate with one another.

Step 4 – Creating a PHP File to Test Configuration

Your LEMP stack should now be completely set up. You can test it to validate that Nginx can correctly hand `.php` files off to the PHP processor.

To do this, use your text editor to create a test PHP file called `info.php` in your document root:

```
$ sudo nano /var/www/html/info.php
```

Enter the following lines into the new file. This is valid PHP code that will return information about your server:

/var/www/html/info.php

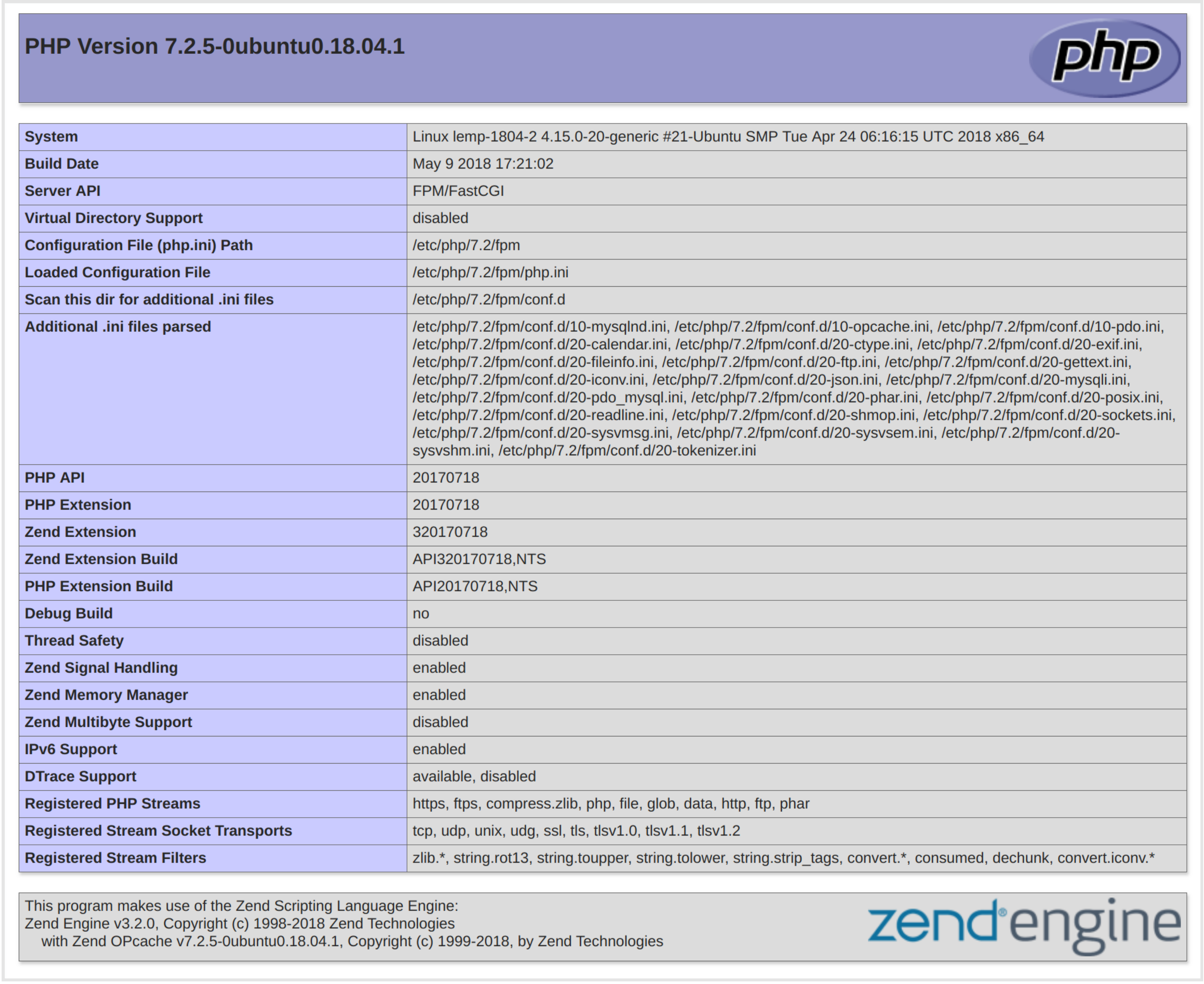
```
<?php
phpinfo();
```

When you are finished, save and close the file.

Now, you can visit this page in your web browser by visiting your server's domain name or public IP address followed by `/info.php`:

```
http://your_server_domain_or_IP/info.php
```

You should see a web page that has been generated by PHP with information about your server:



PHP Version 7.2.5-0ubuntu0.18.04.1

System	Linux lemp-1804-2 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24 06:16:15 UTC 2018 x86_64
Build Date	May 9 2018 17:21:02
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.2/fpm
Loaded Configuration File	/etc/php/7.2/fpm/php.ini
Scan this dir for additional .ini files	/etc/php/7.2/fpm/conf.d
Additional .ini files parsed	/etc/php/7.2/fpm/conf.d/10-mysqlnd.ini, /etc/php/7.2/fpm/conf.d/10-opcache.ini, /etc/php/7.2/fpm/conf.d/10-pdo.ini, /etc/php/7.2/fpm/conf.d/20-calendar.ini, /etc/php/7.2/fpm/conf.d/20-ctype.ini, /etc/php/7.2/fpm/conf.d/20-exif.ini, /etc/php/7.2/fpm/conf.d/20-fileinfo.ini, /etc/php/7.2/fpm/conf.d/20-ftp.ini, /etc/php/7.2/fpm/conf.d/20-gettext.ini, /etc/php/7.2/fpm/conf.d/20-iconv.ini, /etc/php/7.2/fpm/conf.d/20-json.ini, /etc/php/7.2/fpm/conf.d/20-mysqli.ini, /etc/php/7.2/fpm/conf.d/20-pdo_mysql.ini, /etc/php/7.2/fpm/conf.d/20-phar.ini, /etc/php/7.2/fpm/conf.d/20-posix.ini, /etc/php/7.2/fpm/conf.d/20-readline.ini, /etc/php/7.2/fpm/conf.d/20-shmop.ini, /etc/php/7.2/fpm/conf.d/20-sockets.ini, /etc/php/7.2/fpm/conf.d/20-sysvmsg.ini, /etc/php/7.2/fpm/conf.d/20-sysvsem.ini, /etc/php/7.2/fpm/conf.d/20-sysvshm.ini, /etc/php/7.2/fpm/conf.d/20-tokenizer.ini
PHP API	20170718
PHP Extension	20170718
Zend Extension	320170718
Zend Extension Build	API320170718,NTS
PHP Extension Build	API20170718,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, convert.iconv.*

This program makes use of the Zend Scripting Language Engine:
Zend Engine v3.2.0, Copyright (c) 1998-2018 Zend Technologies
    with Zend OPcache v7.2.5-0ubuntu0.18.04.1, Copyright (c) 1999-2018, by Zend Technologies

If you see a page that looks like this, you've set up PHP processing with Nginx successfully.

After verifying that Nginx renders the page correctly, it's best to remove the file you created as it can actually give unauthorized users some hints about your configuration that may help them try to break in. You can always regenerate this file if you need it later.

For now, remove the file by typing:

```
$ sudo rm /var/www/html/info.php
```

With that, you now have a fully-configured and functioning LEMP stack on your Ubuntu 18.04 server.

Conclusion

A LEMP stack is a powerful platform that will allow you to set up and serve nearly any website or application from your server.

There are a number of next steps you could take from here. For example, you should ensure that connections to your server are secured. To this end, you could secure your Nginx installation with Let's

Encrypt. By following this guide, you will acquire a free TLS/SSL certificate for your server, allowing it to serve content over HTTPS.

By: Justin Ellingwood   By: Mark Drake

Upvote (53)   Subscribe   Share

Related Tutorials

How To Troubleshoot Issues in MySQL

How To Set Up a Remote Database to Optimize Site Performance with MySQL on Ubuntu 18.04

How To Install and Configure pgAdmin 4 in Server Mode

How to Deploy a Symfony 4 Application to Production with LEMP on Ubuntu 18.04

An Introduction to Queries in MySQL

51 Comments

Leave a comment...

Log In to Comment

iamjdk *May 24, 2018*

1 Awesome guidelines,
Now, write an article about installing WordPress with LEMP Stock on Ubuntu 18.04.
Thanks!

emcie4 *May 30, 2018*

0 I Always love having my websites run on the latest software and this seems to be the latest setup. Good article I will install WordPress and see how it goes from here.

mandi *May 30, 2018*

0 nice one what else to do now and which editor to use

iberno *June 1, 2018*

0 Thanks... Very usefull!

dons *June 4, 2018*

0 Why not just `sudo apt-get install mysql-server`? It loads version 5.7.22.

By the way: smart of you to point out `auth_socket`` in 18.04.

mdrake MOD *June 4, 2018*

0 Thank you for your comment @dons. I had initially specified the MySQL version number in the install command, which was an oversight on my part. I've since updated the command as you suggested to install the latest version of MySQL.

ozkanemre *June 4, 2018*

0 awesome guide thank you

NewCustomer *June 18, 2018*

0 how to install wordpress in ubuntu 18.04??

arthurnangai *July 19, 2018*

1 getting a 502 Bad Gateway with the info.php page

hafidzfairiz *July 21, 2018*

1 For me, change this line at /etc/nginx/sites-available/default is worked:

old:

```
fastcgi_pass unix:/var/run/php/php7.0-fpm.sock
```

new:

```
fastcgi_pass unix:/var/run/php/php7.2-fpm.sock
```

Or whatever with your php-fpm version.

ostap34 *July 27, 2018*

0 How about add line to /etc/hosts

```
127.0.0.1   example.com
```

?

natoboram *July 30, 2018*

0 **502 Bad Gateway**

Here's how to fix it!

```
sudo nano /etc/php/7.2/fpm/pool.d/www.conf
```

Check what kind of Unix Socket you are using! For me, it was written :

```
listen = /run/php/php7.2-fpm.sock
```

Now you need to tell nginx that *this* is the one socket you're using!

```
sudo nano /etc/nginx/sites-enabled/default
```

You need to edit `fastcgi_pass unix`.

```
        location ~ \.php$ {
                include snippets/fastcgi-php.conf;

                # With php-fpm (or other unix sockets):
                fastcgi_pass unix:/run/php/php7.2-fpm.sock;
        }
```

---

odtrtest *August 1, 2018*

1 In Step 4 - the info.php file is downloading instead of displaying it. did i missed something in configurations. please help how to fix this. thanks.

---

samu9dalle *August 3, 2018*

0 Have the same issue, need help

---

tonton *August 4, 2018*

1 Hi,

please try to unlink default vhost configuration :

```
tonton@ubuntu-s-1vcpu-1gb-xxx-01:/etc/nginx/sites-enabled$ sudo unlink default
```

and restart Nginx by doing : systemctl restart nginx.service

You should be able now to display your phpinfo (don't not forget to remove it after your test ;-))

Please let me know if it's OK for you \o/

---

websiteshelpz *September 9, 2018*

0 hello tonton
It says "no such file or directory" but the directory is already there.

---

oilyuhvoilyhvc *August 2, 2018*

0 For people who wants use MariaDB instead MySQL, the steps are the same, but instead of installing the package `mysql-server-5.7`, install the package `mariadb-server`. It worked for me.

samu9dalle *August 3, 2018*

2 In Step 4 - the info.php file is downloading instead of displaying it. did i missed something in configurations. please help how to fix this. thanks.

---

tonton *August 4, 2018*

3 Hello,

This is probably due to the default vhost configuration. Could you try to unlink it :

```
tonton@ubuntu-s-1vcpu-1gb-xxx-01:/etc/nginx/sites-enabled$ sudo unlink default
```

As a good practice always check Nginx configuration before restarting it :

```
tonton@ubuntu-s-1vcpu-1gb-xxx-01:/etc/nginx/sites-enabled$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful


tonton@ubuntu-s-1vcpu-1gb-xxx-01:/etc/nginx/sites-enabled$ sudo systemctl restart nginx.s
```

And let me know if it helped you :)

---

samu9dalle *August 5, 2018*

1 yes it worked, thanks you

---

websiteshelpz *September 9, 2018*

0 did you modify the code provided by tonton or entered as it is. It is not working for me, saying no such file or directory

---

vincentondigitalocean *December 4, 2018*

0 it worked for me as well

check following:

```
ls /etc/nginx/sites-enabled/
```

if it says `default` and `example.com` you have to run

```
sudo unlink /etc/nginx/sites-enabled/default
```

glenocean *August 21, 2018*

1 I've tried all of the fixes suggested here, but info.php keeps on downloading instead of being rendered.

Amazing that there is no response from the author or other DO staff after more than two weeks.

cdcasey *September 9, 2018*

0 For some reason I got an error saying that php-fpm could not be found. I had to download it from packages.ubuntu.com and install it manually.

cdcasey *September 9, 2018*

1 I'm testing on a local VM before moving to a droplet. Looks like I had to enable the universe repo.

Avanthas *October 17, 2018*

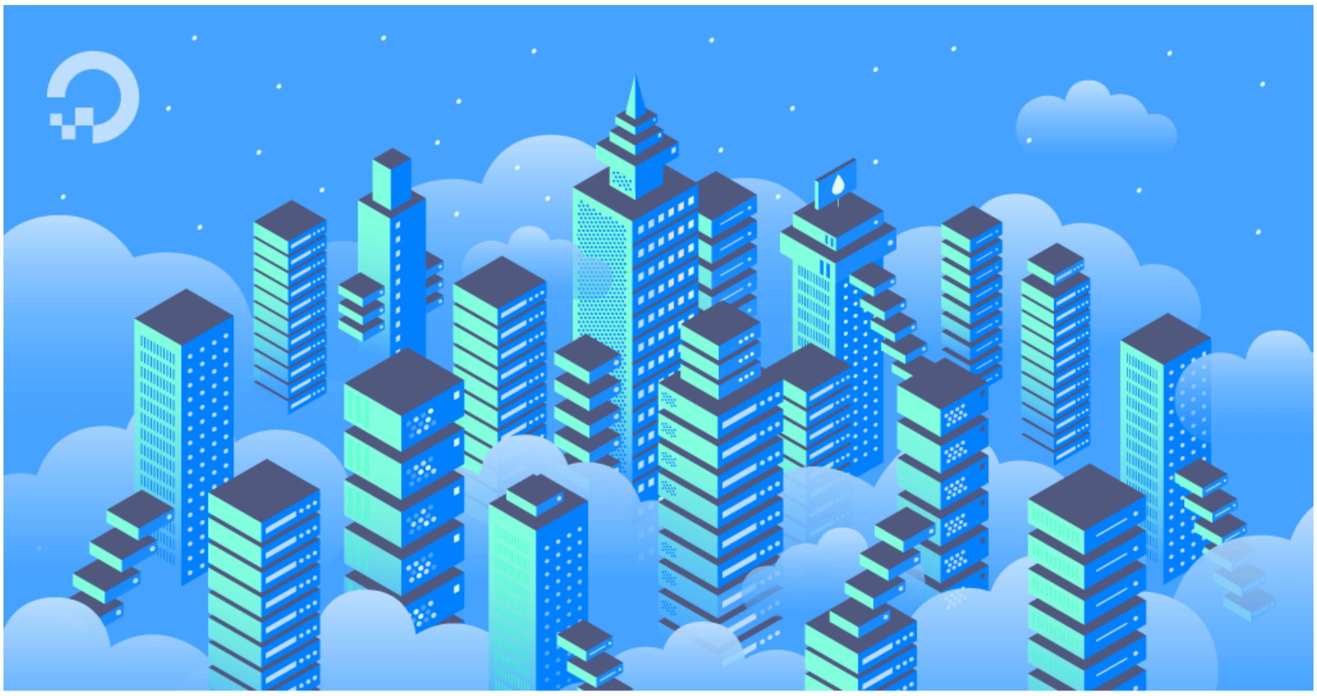
0 That's correct!
It can be added using `sudo add-apt-repository universe`

KristapsL *September 13, 2018*

0 Hello, I am having an issue where if I test my php configuration, instead of showing the php.info file it downloads it? What is the issue? I followed through the steps multiple times

VincentLoy *September 15, 2018*

0 Check response here



How To Install Linux, Nginx, MySQL, PHP (LEMP stack) on Ubuntu 18.04

by Justin Ellingwood

by Mark Drake

This tutorial details the process for installing and configuring the components that constitute a LEMP stack on an Ubuntu 18.04 server, including Nginx, MySQL, and PHP. It also includes instructions for testing that these components can communicate effectively and serve your content

---

mdonahue21 *October 18, 2018*

0 Are you switching to the `/etc/nginx/sites-enabled` directory before running the commands listed above?

---

linux4one *December 16, 2018*

0 check following link also install lemp stack on ubuntu

---

artmonger *September 24, 2018*

0 When I go to myipaddress/info.php it downloads the file instead of displaying the page. Why?

Load More Comments