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

By Justin Ellingwood and Mark Drake Become an author

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 $2; }' | sed 's/\/.*$//'`

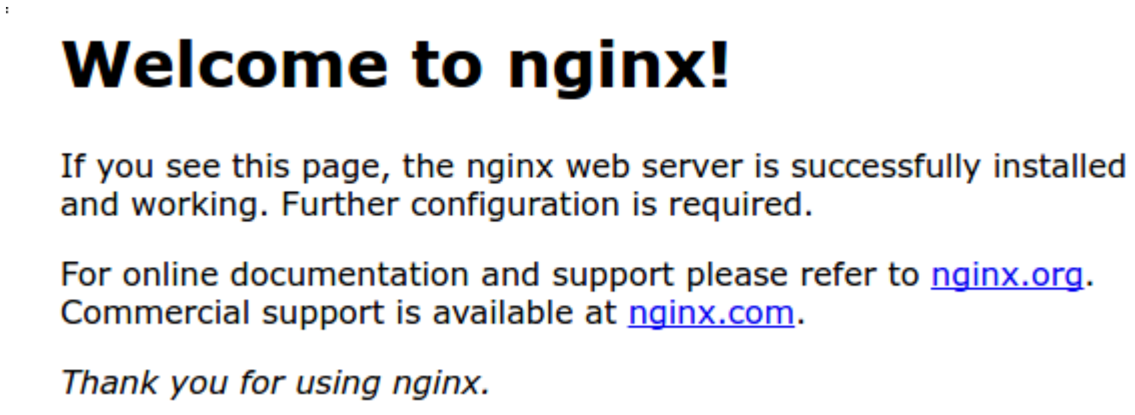
This will print out a few IP addresses. You can try each of them in turn in your web browser.

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

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:

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

```
Output

+------------------+-------------------------------------------+------------------
| user             | authentication_string                     | plugin
+------------------+-------------------------------------------+------------------
| root             |                                           | auth_socket
| mysql.session    | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE | mysql_native_pass
| mysql.sys        | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE | mysql_native_pass
| debian-sys-maint | *CC744277A401A7D25BE1CA89AFF17BF607F876FF | mysql_native_pass
+------------------+-------------------------------------------+------------------
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:

```
• ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'pass
```

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;
```

```
Output

+------------------+-------------------------------------------+-----------------
| user             | authentication_string                     | plugin
+------------------+-------------------------------------------+-----------------
| root             | *3636DACC8616D997782ADD0839F92C1571D6D78F | mysql_native_pass
| mysql.session    | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE | mysql_native_pass
| mysql.sys        | *THISISNOTAVALIDPASSWORDTHATCANBEUSEDHERE | mysql_native_pass
| debian-sys-maint | *CC744277A401A7D25BE1CA89AFF17BF607F876FF | mysql_native_pass
+------------------+-------------------------------------------+-----------------
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:

```
• 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

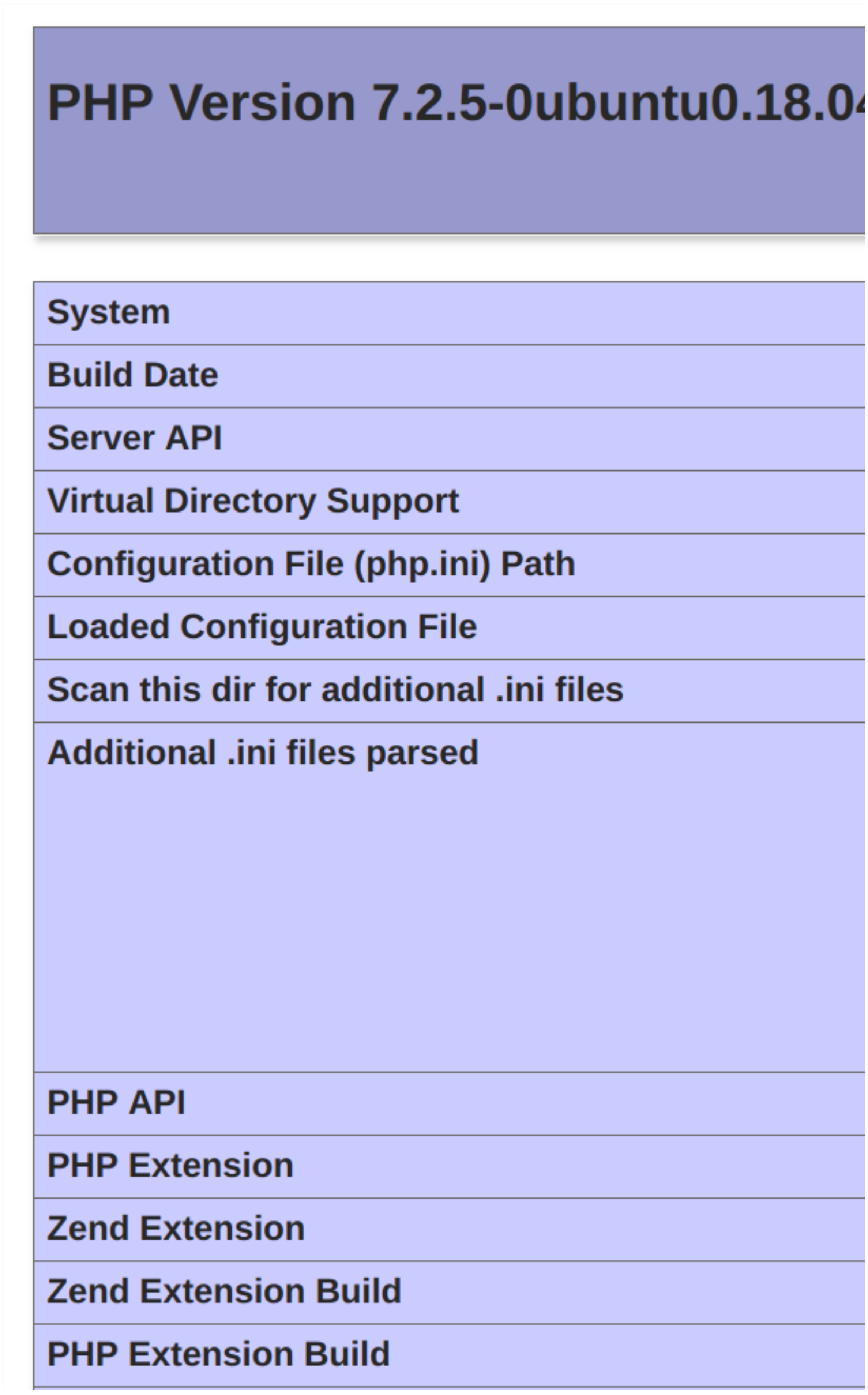
```
<?php
phpinfo();
```

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 7.2.5-0ubuntu0.18.0

System
Build Date
Server API
Virtual Directory Support
Configuration File (php.ini) Path
Loaded Configuration File
Scan this dir for additional .ini files
Additional .ini files parsed
PHP API
PHP Extension
Zend Extension
Zend Extension Build
PHP Extension Build

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.