How To Install Linux, Apache, MySQL, PHP (LAMP) stack on Ubuntu 18.04

Introduction

A "LAMP" stack is a group of open source software that is typically installed together to enable a server to host dynamic websites and web apps. This term is actually an acronym which represents the **L**inux operating system, with the **A**pache web server. The site data is stored in a **M**ySQL database, and dynamic content is processed by **P**HP.

In this guide, we'll get a LAMP stack installed on an Ubuntu 16.04 Droplet. Ubuntu will fulfill our first requirement: a Linux operating system.

Prerequisites

Before you begin with this guide, you should have a separate, non-root user account with `sudo` privileges set up on your server. You can learn how to do this by completing steps 1-4 in the initial server setup for Ubuntu 16.04.

Step 1: Install Apache and Allow in Firewall

The Apache web server is among the most popular web servers in the world. It's well-documented, and has been in wide use for much of the history of the web, which makes it a great default choice for hosting a website.

We can install Apache easily using Ubuntu's package manager, `apt`. A package manager allows us to install most software pain-free from a repository maintained by Ubuntu. You can learn more about how to use `apt` here.

For our purposes, we can get started by typing these commands:

- **`sudo apt-get update`**
- **`sudo apt-get install apache2`**

Since we are using a `sudo` command, these operations get executed with root privileges. It will ask you for your regular user's password to verify your intentions.

Once you've entered your password, `apt` will tell you which packages it plans to install and how much extra disk space they'll take up. Press **Y** and hit **Enter** to continue, and the installation will proceed.

Set Global ServerName to Suppress Syntax Warnings

Next, we will add a single line to the `/etc/apache2/apache2.conf` file to suppress a warning message. While harmless, if you do not set `ServerName` globally, you will receive the following warning when checking your Apache configuration for syntax errors:

- **`sudo apache2ctl configtest`**

```
Output
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress this message
Syntax OK
```

Open up the main configuration file with your text edit:

- **`sudo nano -c  /etc/apache2/apache2.conf`**

Inside, at the bottom of the file, add a `ServerName` directive, pointing to your primary domain name. If you do not have a domain name associated with your server, you can use your server's public IP address:

Note

If you don't know your server's IP address, skip down to the section on how to find your server's public IP address to find it.

/etc/apache2/apache2.conf

```
. . .
ServerName server_domain_or_IP
```

Save and close the file when you are finished.

Next, check for syntax errors by typing:

- **`sudo apache2ctl configtest`**

Since we added the global `ServerName` directive, all you should see is:

```
Output
Syntax OK
```

Restart Apache to implement your changes:

- **`sudo systemctl restart apache2`**

You can now begin adjusting the firewall.

Adjust the Firewall to Allow Web Traffic

Next, assuming that you have followed the initial server setup instructions to enable the UFW firewall, make sure that your firewall allows HTTP and HTTPS traffic. You can make sure that UFW has an application profile for Apache like so:

- `sudo ufw app list`

```
Output
Available applications:
  Apache
  Apache Full
  Apache Secure
  OpenSSH
```

If you look at the `Apache Full` profile, it should show that it enables traffic to ports 80 and 443:

- **`sudo ufw app info "Apache Full"`**

```
Output
Profile: Apache Full
Title: Web Server (HTTP,HTTPS)
Description: Apache v2 is the next generation of the omnipresent Apache web
server.

Ports:
  80,443/tcp
```

Allow incoming traffic for this profile:

- **`sudo ufw allow in "Apache Full"`**

You can do a spot check right away to verify that everything went as planned by visiting your server's public IP address in your web browser (see the note under the next heading to find out what your public IP address is if you do not have this information already):

```
http://your_server_IP_address
```

You will see the default Ubuntu 16.04 Apache web page, which is there for informational and testing purposes. It should look something like this:

Apache2 Ubuntu Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

- `apache2.conf` is the main configuration file. It puts the pieces together by including all remaining configuration files when starting up the web server.
- `ports.conf` is always included from the main configuration file. It is used to determine the listening ports for incoming connections, and this file can be customized anytime.
- Configuration files in the `mods-enabled/`, `conf-enabled/` and `sites-enabled/` directories contain particular configuration snippets which manage modules, global configuration fragments, or virtual host configurations, respectively.
- They are activated by symlinking available configuration files from their respective *-available/ counterparts. These should be managed by using our helpers **`a2enmod`**, **`a2dismod`**, **`a2ensite`**, **`a2dissite`**, and **`a2enconf`**, **`a2disconf`** . See their respective man pages for detailed information.
- The binary is called apache2. Due to the use of environment variables, in the default configuration, apache2 needs to be started/stopped with `/etc/init.d/apache2` or `apache2ctl`. **Calling `/usr/bin/apache2` directly will not work** with the default configuration.

Document Roots

By default, Ubuntu does not allow access through the web browser to *any* file apart of those located in `/var/www`, **public_html** directories (when enabled) and `/usr/share` (for web applications). If your site is using a web document root located elsewhere (such as in `/srv`) you may need to whitelist your document root directory in `/etc/apache2/apache2.conf`.

The default Ubuntu document root is `/var/www/html`. You can make your own virtual hosts under /var/www. This is different to previous releases which provides better security out of the box.

Reporting Problems

Please use the `ubuntu-bug` tool to report bugs in the Apache2 package with Ubuntu. However, check **existing bug reports** before reporting a new bug.

Please report bugs specific to modules (such as PHP and others) to respective packages, not to the web server itself.

If you see this page, then your web server is now correctly installed and accessible through your firewall.

How To Find your Server's Public IP Address

If you do not know what your server's public IP address is, there are a number of ways you can find it. Usually, this is the address you use to connect to your server through SSH.

From the command line, you can find this a few ways. First, you can use the `iproute2` tools to get your address by typing this:

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

This will give you two or three lines back. They are all correct addresses, but your computer may only be able to use one of them, so feel free to try each one.

An alternative method is to use the `curl` utility to contact an outside party to tell you how *it* sees your server. You can do this by asking a specific server what your IP address is:

- `sudo apt-get install curl`
- `curl http://icanhazip.com`

Regardless of the method you use to get your IP address, you can type it into your web browser's address bar to get to your server.

Step 2: Install MySQL

Now that we have our web server up and running, it is time to install MySQL. MySQL is a database management system. Basically, it will organize and provide access to databases where our site can store information.

Again, we can use `apt` to acquire and install our software. This time, we'll also install some other "helper" packages that will assist us in getting our components to communicate with each other:

- `sudo apt-get install mysql-server`

Note: In this case, you do not have to run `sudo apt-get update` prior to the command. This is because we recently ran it in the commands above to install Apache. The package index on our computer should already be up-to-date.

Again, you will be shown a list of the packages that will be installed, along with the amount of disk space they'll take up. Enter **Y** to continue.

During the installation, your server will ask you to select and confirm a password for the MySQL "root" user. This is an administrative account in MySQL that has increased privileges. Think of it as being similar to the root account for the server itself (the one you are configuring

now is a MySQL-specific account, however). Make sure this is a strong, unique password, and do not leave it blank.

When the installation is complete, we want to run a simple security script that will remove some dangerous defaults and lock down access to our database system a little bit. Start the interactive script by running:

- **`mysql_secure_installation`**

You will be asked to enter the password you set for the MySQL root account. Next, you will be asked 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:
```

You'll be asked 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
```

If you enabled password validation, you'll be shown a password strength for the existing root password, and asked you if you want to change that password. If you are happy with your current password, enter **n**for "no" at the prompt:

```
Using existing password for root.
```

```
Estimated strength of the password: 100
Change the password for root ? ((Press y|Y for Yes, any other key for No) : n
```

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.

At this point, your database system is now set up and we can move on.

Step 3: Install PHP

PHP is the component of our setup that will process code to display dynamic content. It can run scripts, connect to our MySQL databases to get information, and hand the processed content over to our web server to display.

We can once again leverage the `apt` system to install our components. We're going to include some helper packages as well, so that PHP code can run under the Apache server and talk to our MySQL database:

- **`sudo apt-get install php libapache2-mod-php php-mcrypt php-mysql`**

This should install PHP without any problems. We'll test this in a moment.

In most cases, we'll want to modify the way that Apache serves files when a directory is requested. Currently, if a user requests a directory from the server, Apache will first look for a file called `index.html`. We want to tell our web server to prefer PHP files, so we'll make Apache look for an `index.php` file first.

To do this, type this command to open the `dir.conf` file in a text editor with root privileges:

- **`sudo nano -c /etc/apache2/mods-enabled/dir.conf`**

It will look like this:

/etc/apache2/mods-enabled/dir.conf

```
<IfModule mod_dir.c>
    DirectoryIndex index.html index.cgi index.pl index.php index.xhtml
index.htm
</IfModule>
```

We want to move the PHP index file highlighted above to the first position after the `DirectoryIndex`specification, like this:

/etc/apache2/mods-enabled/dir.conf

```
<IfModule mod_dir.c>
    DirectoryIndex index.php index.html index.cgi index.pl index.xhtml
index.htm
</IfModule>
```

When you are finished, save and close the file by pressing **Ctrl-X**. You'll have to confirm the save by typing **Y** and then hit **Enter** to confirm the file save location.

After this, we need to restart the Apache web server in order for our changes to be recognized. You can do this by typing this:

- **`sudo systemctl restart apache2`**

We can also check on the status of the `apache2` service using `systemctl`:

- **`sudo systemctl status apache2`**

```
Sample Output
● apache2.service - LSB: Apache2 web server
   Loaded: loaded (/etc/init.d/apache2; bad; vendor preset: enabled)
  Drop-In: /lib/systemd/system/apache2.service.d
           └─apache2-systemd.conf
   Active: active (running) since Wed 2016-04-13 14:28:43 EDT; 45s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 13581 ExecStop=/etc/init.d/apache2 stop (code=exited,
status=0/SUCCESS)
  Process: 13605 ExecStart=/etc/init.d/apache2 start (code=exited,
status=0/SUCCESS)
    Tasks: 6 (limit: 512)
   CGroup: /system.slice/apache2.service
           ├─13623 /usr/sbin/apache2 -k start
           ├─13626 /usr/sbin/apache2 -k start
           ├─13627 /usr/sbin/apache2 -k start
           ├─13628 /usr/sbin/apache2 -k start
           ├─13629 /usr/sbin/apache2 -k start
           └─13630 /usr/sbin/apache2 -k start

Apr 13 14:28:42 ubuntu-16-lamp systemd[1]: Stopped LSB: Apache2 web server.
Apr 13 14:28:42 ubuntu-16-lamp systemd[1]: Starting LSB: Apache2 web server...
Apr 13 14:28:42 ubuntu-16-lamp apache2[13605]:  * Starting Apache httpd web
server apache2
Apr 13 14:28:42 ubuntu-16-lamp apache2[13605]: AH00558: apache2: Could not
reliably determine the server's fully qualified domain name, using 127.0.1.1.
Set the 'ServerNam
Apr 13 14:28:43 ubuntu-16-lamp apache2[13605]:  *
Apr 13 14:28:43 ubuntu-16-lamp systemd[1]: Started LSB: Apache2 web server.
```

Install PHP Modules

To enhance the functionality of PHP, we can optionally install some additional modules.

To see the available options for PHP modules and libraries, you can pipe the results of `apt-cache search`into `less`, a pager which lets you scroll through the output of other commands:

- `apt-cache search php- | less`
-

Use the arrow keys to scroll up and down, and **q** to quit.

The results are all optional components that you can install. It will give you a short description for each:

```
libnet-libidn-perl - Perl bindings for GNU Libidn
php-all-dev - package depending on all supported PHP development packages
php-cgi - server-side, HTML-embedded scripting language (CGI binary) (default)
php-cli - command-line interpreter for the PHP scripting language (default)
php-common - Common files for PHP packages
php-curl - CURL module for PHP [default]
php-dev - Files for PHP module development (default)
php-gd - GD module for PHP [default]
php-gmp - GMP module for PHP [default]
…
:
```

To get more information about what each module does, you can either search the internet, or you can look at the long description of the package by typing:

- `apt-cache show package_name`
-

There will be a lot of output, with one field called `Description-en` which will have a longer explanation of the functionality that the module provides.

For example, to find out what the `php-cli` module does, we could type this:

- `apt-cache show php-cli`
-

Along with a large amount of other information, you'll find something that looks like this:

```
Output
…
```

```
Description-en: command-line interpreter for the PHP scripting language
(default)
 This package provides the /usr/bin/php command interpreter, useful for
 testing PHP scripts from a shell or performing general shell scripting tasks.
 .
 PHP (recursive acronym for PHP: Hypertext Preprocessor) is a widely-used
 open source general-purpose scripting language that is especially suited
 for web development and can be embedded into HTML.
 .
 This package is a dependency package, which depends on Debian's default
 PHP version (currently 7.0).
…
```

If, after researching, you decide you would like to install a package, you can do so by using the `apt-get install` command like we have been doing for our other software.

If we decided that `php-cli` is something that we need, we could type:

- `sudo apt-get install php-cli`
-

If you want to install more than one module, you can do that by listing each one, separated by a space, following the `apt-get install` command, like this:

- `sudo apt-get install package1 package2 ...`
-

At this point, your LAMP stack is installed and configured. We should still test out our PHP though.

Step 4: Test PHP Processing on your Web Server

In order to test that our system is configured properly for PHP, we can create a very basic PHP script.

We will call this script `info.php`. In order for Apache to find the file and serve it correctly, it must be saved to a very specific directory, which is called the "web root".

In Ubuntu 16.04, this directory is located at `/var/www/html/`. We can create the file at that location by typing:

- `sudo nano -c /var/www/html/info.php`
-

This will open a blank file. We want to put the following text, which is valid PHP code, inside the file:

info.php

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

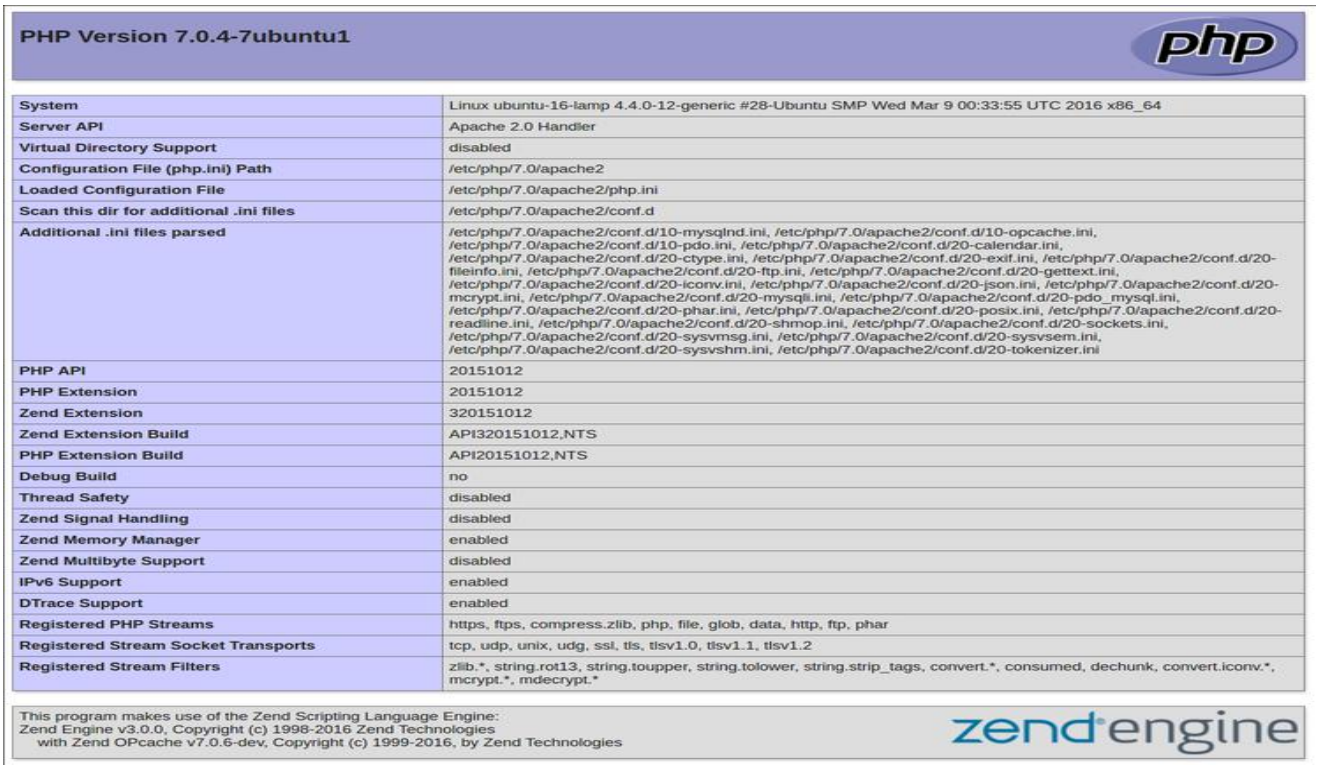
When you are finished, save and close the file.

Now we can test whether our web server can correctly display content generated by a PHP script. To try this out, we just have to visit this page in our web browser. You'll need your server's public IP address again.

The address you want to visit will be:

```
http://your_server_IP_address/info.php
```

The page that you come to should look something like this:



PHP Version 7.0.4-7ubuntu1

System	Linux ubuntu-16-lamp 4.4.0-12-generic #28-Ubuntu SMP Wed Mar 9 00:33:55 UTC 2016 x86_64
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.0/apache2
Loaded Configuration File	/etc/php/7.0/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.0/apache2/conf.d
Additional .ini files parsed	/etc/php/7.0/apache2/conf.d/10-mysqlnd.ini, /etc/php/7.0/apache2/conf.d/10-opcache.ini, /etc/php/7.0/apache2/conf.d/10-pdo.ini, /etc/php/7.0/apache2/conf.d/20-calendar.ini, /etc/php/7.0/apache2/conf.d/20-ctype.ini, /etc/php/7.0/apache2/conf.d/20-exif.ini, /etc/php/7.0/apache2/conf.d/20-fileinfo.ini, /etc/php/7.0/apache2/conf.d/20-ftp.ini, /etc/php/7.0/apache2/conf.d/20-gettext.ini, /etc/php/7.0/apache2/conf.d/20-iconv.ini, /etc/php/7.0/apache2/conf.d/20-json.ini, /etc/php/7.0/apache2/conf.d/20-mcrypt.ini, /etc/php/7.0/apache2/conf.d/20-mysqli.ini, /etc/php/7.0/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.0/apache2/conf.d/20-phar.ini, /etc/php/7.0/apache2/conf.d/20-posix.ini, /etc/php/7.0/apache2/conf.d/20-readline.ini, /etc/php/7.0/apache2/conf.d/20-shmop.ini, /etc/php/7.0/apache2/conf.d/20-sockets.ini, /etc/php/7.0/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.0/apache2/conf.d/20-sysvsem.ini, /etc/php/7.0/apache2/conf.d/20-sysvshm.ini, /etc/php/7.0/apache2/conf.d/20-tokenizer.ini
PHP API	20151012
PHP Extension	20151012
Zend Extension	320151012
Zend Extension Build	API320151012,NTS
PHP Extension Build	API20151012,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	enabled
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.*, mcrypt.*, mdecrypt.*

This program makes use of the Zend Scripting Language Engine:
Zend Engine v3.0.0, Copyright (c) 1998-2016 Zend Technologies
with Zend OPcache v7.0.6-dev, Copyright (c) 1999-2016, by Zend Technologies

This page basically gives you information about your server from the perspective of PHP. It is useful for debugging and to ensure that your settings are being applied correctly.

If this was successful, then your PHP is working as expected.

You probably want to remove this file after this test because it could actually give information about your server to unauthorized users. To do this, you can type this:

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

You can always recreate this page if you need to access the information again later.

Conclusion

Now that you have a LAMP stack installed, you have many choices for what to do next. Basically, you've installed a platform that will allow you to install most kinds of websites and web software on your server.