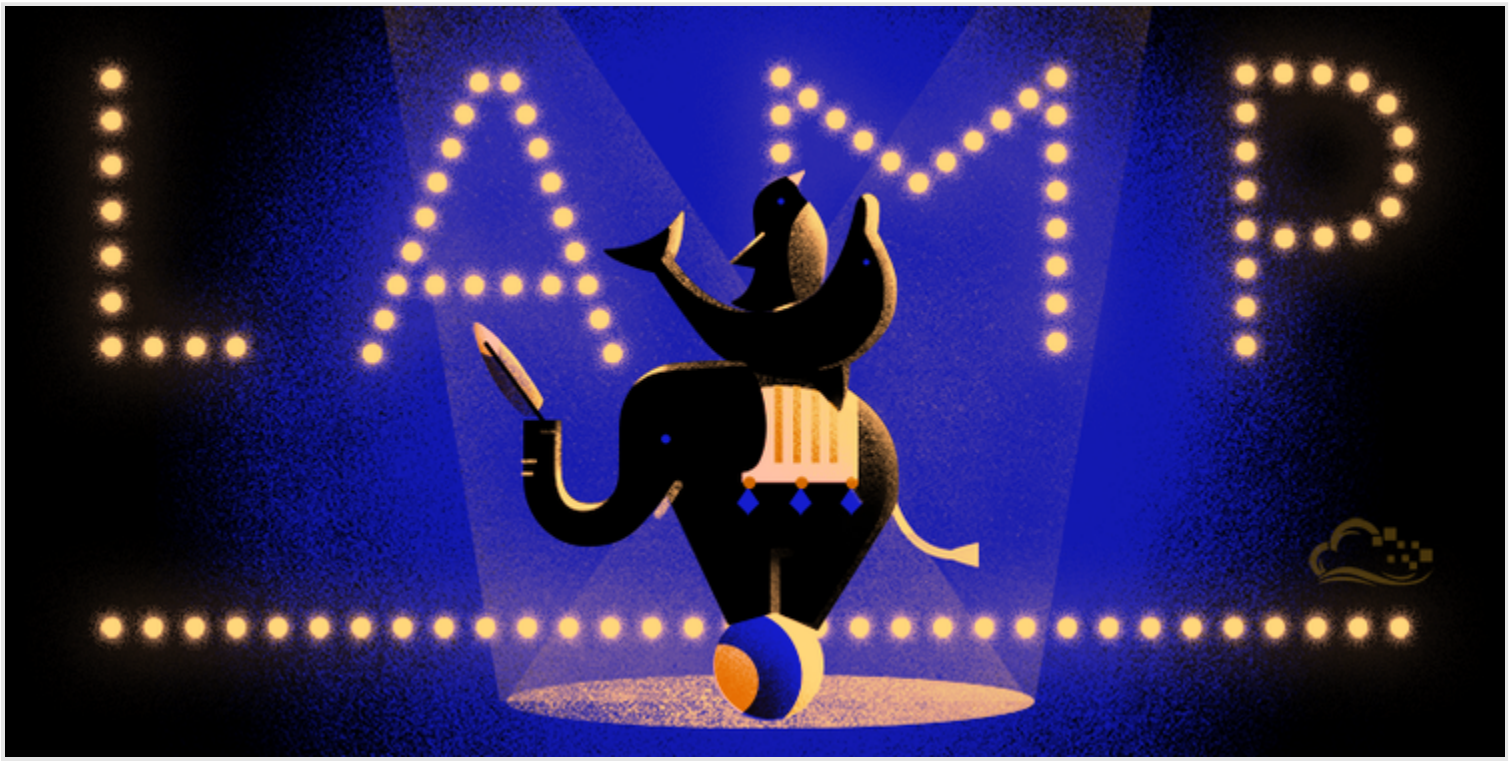


How To Install Linux, Apache, MySQL, PHP (LAMP) stack on Ubuntu 18.04

Posted April 27, 2018 433.7k LAMP STACK DATABASES PHP MYSQL APACHE UBUNTU 18.04

By: Mark Drake

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

A previous version of this tutorial was written by Brennan Bearnes.

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 will install a LAMP stack on an Ubuntu 18.04 server.

Prerequisites

In order to complete this tutorial, you will need to have an Ubuntu 18.04 server with a non-root `sudo`-enabled user account and a basic firewall. This can be configured using our initial server setup guide for Ubuntu 18.04.

Step 1 — Installing Apache and Updating the 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.

Install Apache using Ubuntu's package manager, `apt`:

```
$ sudo apt update
$ sudo apt install apache2
```

Since this is a `sudo` command, these operations are 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.

Adjust the Firewall to Allow Web Traffic

Next, assuming that you have followed the initial server setup instructions and enabled the UFW firewall, make sure that your firewall allows HTTP and HTTPS traffic. You can check 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 HTTP and HTTPS 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
```

You will see the default Ubuntu 18.04 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.

There are a few different ways to do this from the command line. First, you could use the `iproute2` tools to get your IP 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. This is done by asking a specific server what your IP address is:

```
$ sudo apt install curl
$ curl http://icanhazip.com
```

Regardless of the method you use to get your IP address, type it into your web browser's address bar to view the default Apache page.

Step 2 — Installing MySQL

Now that you have your 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 your site can store information.

Again, use `apt` to acquire and install this software:

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

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

This command, too, will show you a list of the packages that will be installed, along with the amount of disk space they'll take up. Enter `Y` to continue.

When the installation is complete, run a simple security script that comes pre-installed with MySQL which will remove some dangerous defaults and lock down access to your database system. Start the interactive script by running:

```
$ sudo mysql_secure_installation
```

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

> **Note:** 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 answer “yes”, 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
```

Regardless of whether you chose to set up the `VALIDATE PASSWORD PLUGIN`, your server will next 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 (although the one you are configuring now is a MySQL-specific account). Make sure this is a strong, unique password, and do not leave it blank.

If you enabled password validation, you'll be shown the password strength for the root password you just entered and your server will ask 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, 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 you 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 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
```

At this point, your database system is now set up and you can move on to installing PHP, the final component of the LAMP stack.

Step 3 — Installing PHP

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

Once again, leverage the `apt` system to install PHP. In addition, include some helper packages this time so that PHP code can run under the Apache server and talk to your MySQL database:

```
$ sudo apt install php libapache2-mod-php php-mysql
```

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

In most cases, you will 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 the web server to prefer PHP files over others, so 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 /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>
```

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`. Confirm the save by typing `Y` and then hit `ENTER` to verify the file save location.

After this, restart the Apache web server in order for your changes to be recognized. Do this by typing this:

```
$ sudo systemctl restart apache2
```

You 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 Tue 2018-04-23 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
```

```
           ├─13627 /usr/sbin/apache2 -k start
           ├─13628 /usr/sbin/apache2 -k start

           ├─13629 /usr/sbin/apache2 -k start
           └─13630 /usr/sbin/apache2 -k start
```

Press `Q` to exit this status output.

To enhance the functionality of PHP, you have the option to install some additional modules. To see the available options for PHP modules and libraries, pipe the results of `apt search` into `less`, a pager which lets you scroll through the output of other commands:

```
$ apt search php- | less
```

Use the arrow keys to scroll up and down, and press `Q` to quit.

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

```
bandwidthd-pgsql/bionic 2.0.1+cvs20090917-10ubuntu1 amd64
  Tracks usage of TCP/IP and builds html files with graphs

bluefish/bionic 2.2.10-1 amd64
  advanced Gtk+ text editor for web and software development

cacti/bionic 1.1.38+ds1-1 all
  web interface for graphing of monitoring systems

ganglia-webfrontend/bionic 3.6.1-3 all
  cluster monitoring toolkit - web front-end

golang-github-unknwon-cae-dev/bionic 0.0~git20160715.0.c6aac99-4 all
  PHP-like Compression and Archive Extensions in Go

haserl/bionic 0.9.35-2 amd64
  CGI scripting program for embedded environments

kdevelop-php-docs/bionic 5.2.1-1ubuntu2 all
  transitional package for kdevelop-php

kdevelop-php-docs-l10n/bionic 5.2.1-1ubuntu2 all
  transitional package for kdevelop-php-l10n
…
:
```

To learn more about what each module does, you could search the internet for more information about them. Alternatively, look at the long description of the package by typing:

```
$ apt show package_name
```

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

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

```
$ apt show php-cli
```

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

```
Output
…
Description: 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 Ubuntu's default

 PHP version (currently 7.2).
…
```

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

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

```
$ sudo apt 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 install` command, like this:

```
$ sudo apt install package1 package2 ...
```

At this point, your LAMP stack is installed and configured. Before making any more changes or deploying an application, though, it would be helpful to proactively test out your PHP configuration in case there are any issues that should be addressed.

Step 4 — Testing PHP Processing on your Web Server

In order to test that your system is configured properly for PHP, create a very basic PHP script called `info.php`. In order for Apache to find this file and serve it correctly, it must be saved to a very specific directory, which is called the "web root".

In Ubuntu 18.04, this directory is located at `/var/www/html/`. Create the file at that location by running:

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

This will open a blank file. Add 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 you can test whether your web server is able to correctly display content generated by this PHP script. To try this out, visit this page in your web browser. You'll need your server's public IP address again.

The address you will want to visit is:

```
http://your_server_ip/info.php
```

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

PHP Version 7.2.3-1ubuntu1

System	Linux LAMP-1804-test 4.15.0-15-generic #16-Ubuntu SMP Wed Apr 4 13:58:14 UTC 2018 x86_64
Build Date	Mar 14 2018 22:03:58
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.2/apache2
Loaded Configuration File	/etc/php/7.2/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.2/apache2/conf.d
Additional .ini files parsed	/etc/php/7.2/apache2/conf.d/10-mysqlnd.ini, /etc/php/7.2/apache2/conf.d/10-opcache.ini, /etc/php/7.2/apache2/conf.d/10-pdo.ini, /etc/php/7.2/apache2/conf.d/20-calendar.ini, /etc/php/7.2/apache2/conf.d/20-ctype.ini, /etc/php/7.2/apache2/conf.d/20-curl.ini, /etc/php/7.2/apache2/conf.d/20-exif.ini, /etc/php/7.2/apache2/conf.d/20-fileinfo.ini, /etc/php/7.2/apache2/conf.d/20-ftp.ini, /etc/php/7.2/apache2/conf.d/20-gd.ini, /etc/php/7.2/apache2/conf.d/20-gettext.ini, /etc/php/7.2/apache2/conf.d/20-iconv.ini, /etc/php/7.2/apache2/conf.d/20-intl.ini, /etc/php/7.2/apache2/conf.d/20-json.ini, /etc/php/7.2/apache2/conf.d/20-mysqli.ini, /etc/php/7.2/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.2/apache2/conf.d/20-phar.ini, /etc/php/7.2/apache2/conf.d/20-posix.ini, /etc/php/7.2/apache2/conf.d/20-readline.ini, /etc/php/7.2/apache2/conf.d/20-shmop.ini, /etc/php/7.2/apache2/conf.d/20-sockets.ini, /etc/php/7.2/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.2/apache2/conf.d/20-sysvsem.ini, /etc/php/7.2/apache2/conf.d/20-sysvshm.ini, /etc/php/7.2/apache2/conf.d/20-tokenizer.ini, /etc/php/7.2/apache2/conf.d/20-xmlrpc.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 v7.2.3-1ubuntu1, Copyright (c) 1999-2018, by Zend Technologies

Configuration

apache2handler

Apache Version	Apache/2.4.29 (Ubuntu)
Apache API Version	20120211
Server Administrator	webmaster@localhost
Hostname:Port	162.243.26.126:80
User/Group	www-data(33)/33
Max Requests	Per Child: 0 - Keep Alive: on - Max Per Connection: 100
Timeouts	Connection: 300 - Keep-Alive: 5
Virtual Server	Yes
Server Root	/etc/apache2
Loaded Modules	core mod_so mod_watchdog http_core mod_log_config mod_logio mod_version mod_unixd mod_access_compat mod_alias mod_auth_basic mod_authn_core mod_authn_file mod_authz_core mod_authz_host mod_authz_user mod_autoindex mod_deflate mod_dir mod_env mod_filter mod_mime prefork mod_negotiation mod_php7 mod_reqtimeout mod_setenvif mod_status

This page provides some basic 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 you can see this page in your browser, 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, run the following command:

```
$ 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.

As an immediate next step, you should ensure that connections to your web server are secured, by serving them via HTTPS. The easiest option here is to use Let's Encrypt to secure your site with a free TLS/SSL certificate.

Some other popular options are:

- Install Wordpress the most popular content management system on the internet.
- Set Up PHPMyAdmin to help manage your MySQL databases from web browser.
- Learn how to use SFTP to transfer files to and from your server.

By: Mark Drake

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

An Introduction to Queries in MySQL

An Introduction to Queries in PostgreSQL

29 Comments

Leave a comment...

Log In to Comment

meridiansw *April 29, 2018*

1 Awesome! Worked well for my brand new installation of Ubuntu 18.04 (on localhost).

Thank you.

awadheshkumar934 *October 16, 2018*

0 Awesome!

m7esain *April 29, 2018*

0 it's not asking to set the mysql root password, don't know why, also im not able to use mysql unless i use it with sudo

aalaap *May 3, 2018*

2 I had a perfectly working LAMP setup on 17.10, but the upgrade to 18.04 broke the PHP module for Apache. PHP scripts weren't being interpreted, as if there was no PHP installed.

```
 $ a2query -m php
No module matches php
```

All I had to do was (re)enable the module and restart Apache.

```
 $ sudo a2enmod php7.2
Considering dependency mpm_prefork for php7.2:
Considering conflict mpm_event for mpm_prefork:
Considering conflict mpm_worker for mpm_prefork:
Module mpm_prefork already enabled
Considering conflict php5 for php7.2:
Enabling module php7.2.
To activate the new configuration, you need to run:
systemctl restart apache2
$ sudo systemctl restart apache2
```

simplyearl *November 2, 2018*

0 Just wanted to say I was in the exact same scenario and your comment saved me. Many thanks!

run4him *May 22, 2018*

1 Awesome! Thank you! Instructions were very clear.

timlepes *May 24, 2018*

0 In Step 4 (Testing PHP Processing on your Web Server), you instruct us to create a file /var/www/html/info.php

After doing so, and refreshing my browser page, I did not get the PHP info page.

I renamed /var/www/html/info.php to /var/www/html/index.php and it worked.

I think you made a typo in the file name. Thanks.

arun007 *May 27, 2018*

1 In Ubuntu 18.04 apache is a dependency for php. So
sudo apt install php
will install apache and libapache2-mod-php7.2 automatically

kylej *September 22, 2018*

0 thank you for the info arun; i confirmed:

```
sudo apt install php
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

```
The following additional packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapache2-mod-php7.2 libapr1 libaprutil
  php7.2-json php7.2-opcache php7.2-readline ssl-cert
Suggested packages:
  www-browser apache2-doc apache2-suexec-pristine | apache2-suexec-custom php-pear openss
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils libapache2-mod-php7.2 libapr1 libaprutil
  php7.2-common php7.2-json php7.2-opcache php7.2-readline ssl-cert
0 upgraded, 20 newly installed, 0 to remove and 0 not upgraded.
Need to get 5723 kB of archives.
After this operation, 24.5 MB of additional disk space will be used.
```

easier to install this way, thanks again

---

fivedogs59 *June 7, 2018*

0 hello to all,
when i go in phpmyadmin i riceve this msg:

Warning in ./libraries/plugin_interface.lib.php#551
count(): Parameter must be an array or an object that implements Countable

What is the problem?

---

satzkumar84 *June 27, 2018*

0 Amazing step by step explanation provided. Worked exactly the way it is described. Thank you so much

---

Cancelor *June 29, 2018*

0 I noticed the command
sudo nano /etc/apache2/apache2.conf
was not used after installing Apache.
Without this the ServerName directive is not present and I get the message
"Could not reliably determine the server's fully qualified domain name"

The tutorial for 16.04 does include this.

---

vsvinit0 *August 25, 2018*

1 Worked like a charm!

I was struggling a lot to install LAMP stack on my new Linux Mint machine. I have been gone through a lot of articles but didn't get much help. But this article helped me a lot with a clear understanding on how to install LAMP on Linux.

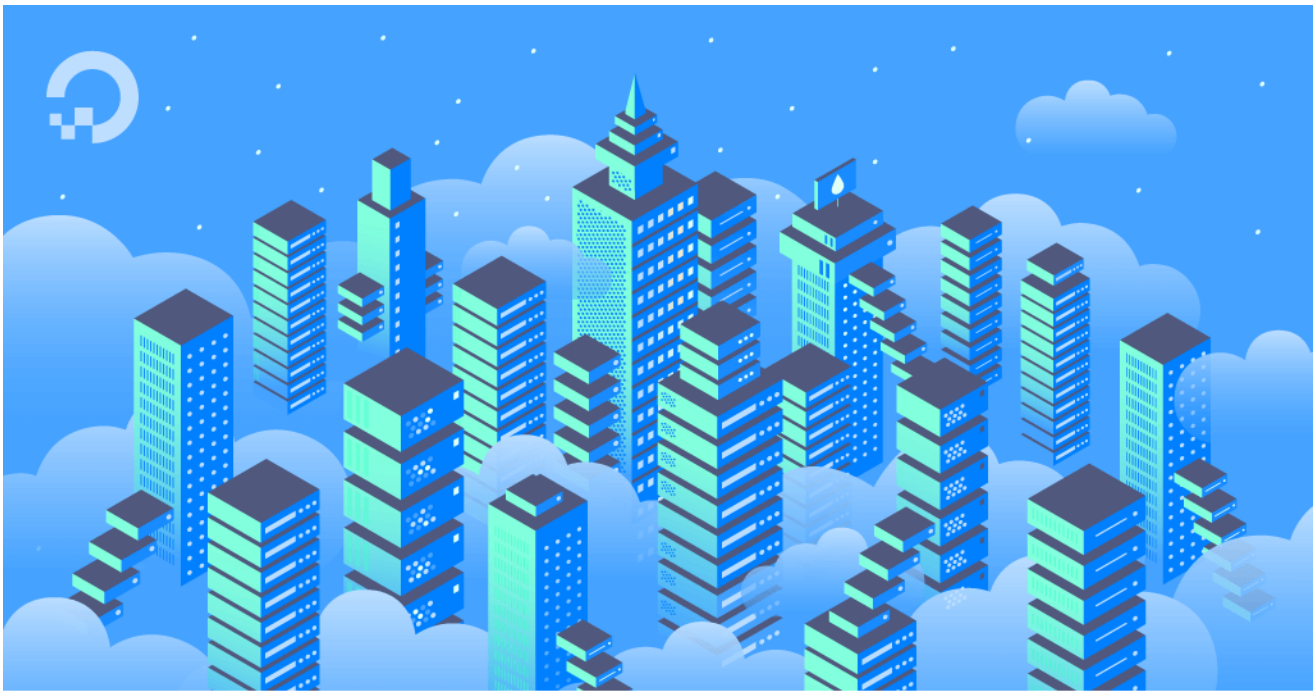
Anyway thanks a lot.

ghostpizza *September 17, 2018*

0 Hello nice tut, thank you but any news about tut for Apache VIrtual Hosts configuration on Ubuntu 18? Thank you

takaia *December 5, 2018*

0 https://www.digitalocean.com/community/tutorials/how-to-install-the-apache-web-server-on-ubuntu-18-04#step-5-%E2%80%94-setting-up-virtual-hosts-recommended



How To Install the Apache Web Server on Ubuntu 18.04

by Justin Ellingwood
by Kathleen Juell

The Apache HTTP server is the most widely-used web server in the world. It provides many powerful features including dynamically loadable modules, robust media support, and extensive integration with other popular software. In this guide, we'll explain how to install an Apache web

PathToEternity *September 17, 2018*

1 After I ran:

```
sudo systemctl status apache2
```

I had trouble getting out of the output. Discovered I had to press *q* to quit out.

krupesh *September 20, 2018*

1 Thankyou so much for this easy, step by step tutorial . I rarely read this type of user friendly docs on web.

helgatheviking *October 6, 2018*

0 Can this be automated?

jeremc *October 13, 2018*

0 If you create a new droplet you can select in the "One-click apps" section LAMP. It should do the same.

helgatheviking *October 13, 2018*

0 Thanks! I see now that the one-click apps are updated to use ubuntu 18.04... especially pleased that the WordPress app is using that by default too.

DzhuSLab *October 18, 2018*

0 In Step 2 — Installing MySQL:

' If you enabled password validation, you'll be shown the password strength for the root password you just entered and your server will ask 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 '

But it doesn't work because in this step you need to enter y (yes) instead as the question is ' Do you wish to continue with the password provided? (Press y|Y for Yes, any other key for No) '

Everything else is fine. Thank you.

nikhil6867 *October 31, 2018*

0 hi am unable to get the apache config page when i put my ip in browser

Load More Comments