**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 install a LAMP stack on an Ubuntu 18.04 server.

Prerequisites

In order to complete this tutorial, you’ll need to have an Ubuntu 18.04 server with a non-root sudo-enabled user account and a basic firewall configured. To set this up, you can follow our [initial server setup guide for Ubuntu 18.04](https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-18-04).

Step 1 — Installing Apache and Updating the Firewall

The Apache web server is a popular open source web server that can be used along with PHP to host dynamic websites. It’s well-documented and has been in wide use for much of the history of the web.

First, make sure your apt cache is updated with:

* sudo apt update

Copy

If this is your first time using sudo in this session, you’ll be prompted to provide your regular user’s password to validate your permissions.

Once the cache has been updated, you can install Apache with:

* sudo apt install apache2

Copy

After entering this command, 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 confirm, 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

Copy

Output

Available applications:

Apache

Apache Full

Apache Secure

OpenSSH

If you look at the Apache Full profile details, you’ll see that it enables traffic to ports 80 and 443:

* sudo ufw app info "Apache Full"

Copy

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

To allow incoming HTTP and HTTPS traffic for this server, run:

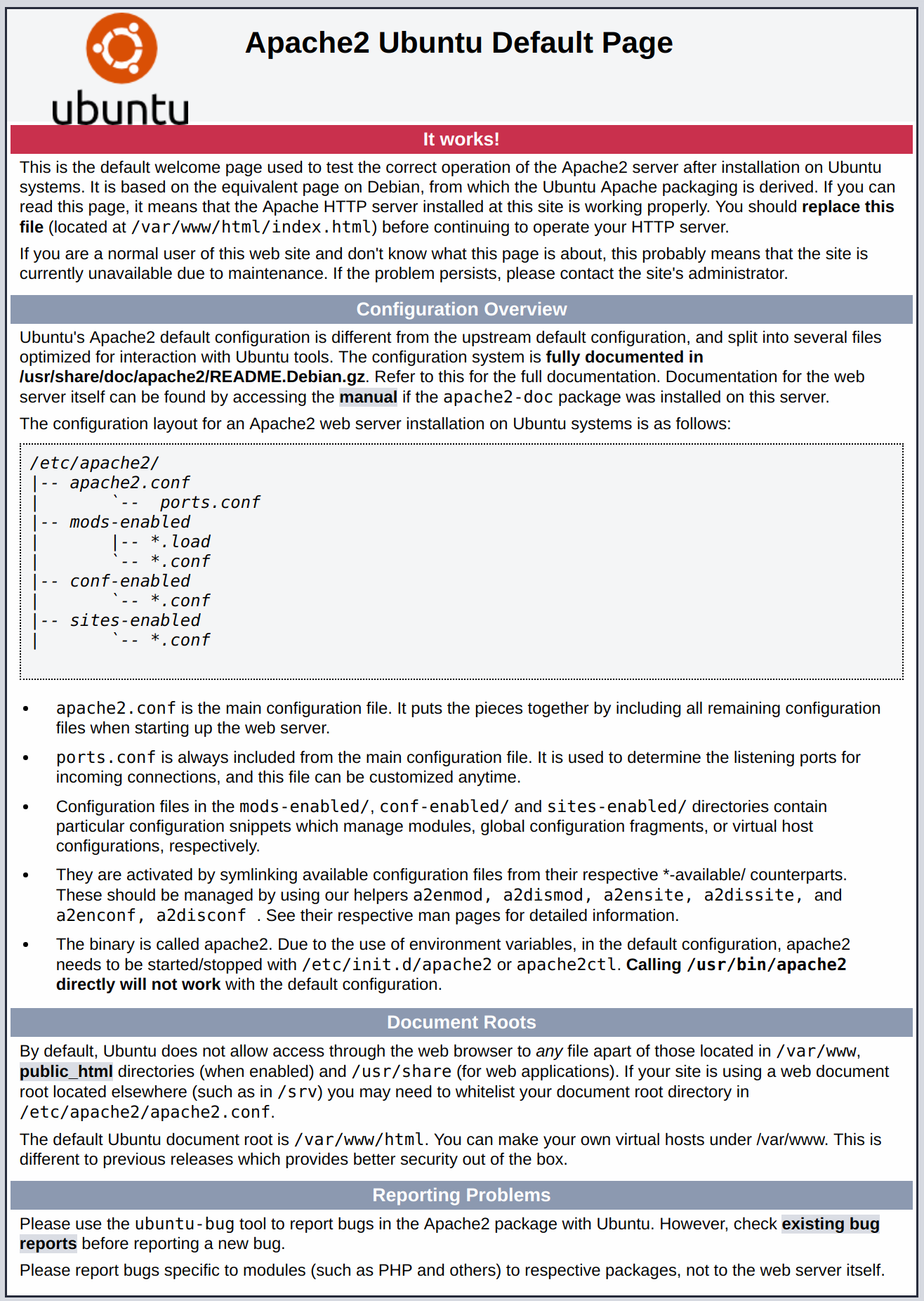
* sudo ufw allow "Apache Full"

Copy

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:



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

Copy

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

Copy

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

Copy

**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

Copy

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 not to be confused with the **system root**. The **database root** user is an administrative user with full privileges over the database system. Even though the default authentication method for the MySQL root user dispenses the use of a password, **even when one is set**, you should define a strong password here as an additional safety measure. We’ll talk about this in a moment.

If you’ve 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.

When you’re finished, test if you’re able to log in to the MySQL console by typing:

* sudo mysql

Copy

This will connect to the MySQL server as the administrative database user **root**, which is inferred by the use of sudo when running this command. You should see output like this:

Output

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 5

Server version: 5.7.34-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

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

affiliates. Other names may be trademarks of their respective

owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>

To exit the MySQL console, type:

* exit

Copy

Notice that you didn’t need to provide a password to connect as the **root** user, even though you have defined one when running the mysql\_secure\_installation script. That is because the default authentication method for the administrative MySQL user is unix\_socket instead of password. Even though this might look like a security concern at first, it makes the database server more secure because the only users allowed to log in as the **root** MySQL user are the system users with sudo privileges connecting from the console or through an application running with the same privileges. In practical terms, that means you won’t be able to use the administrative database **root** user to connect from your PHP application. Setting a password for the **root** MySQL account works as a safeguard, in case the default authentication method is changed from unix\_socket to password.

For increased security, it’s best to have dedicated user accounts with less expansive privileges set up for every database, especially if you plan on having multiple databases hosted on your server. Please refer to our guide on [How To Create a New User and Grant Permissions on MySQL](https://www.digitalocean.com/community/tutorials/how-to-create-a-new-user-and-grant-permissions-in-mysql) for detailed instructions on how to create MySQL users and configure database access rights.

Your MySQL server is now installed and secured. Next, we’ll install PHP, the final component in 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 so that it can display the results to your visitors.

Once again, leverage the apt system to install PHP. In addition to the php package, you’ll also need libapache2-mod-php to integrate PHP into Apache, and the php-mysql package to allow PHP to connect to MySQL databases. Run the following command to install all three packages and their dependencies:

* sudo apt install php libapache2-mod-php php-mysql

Copy

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

Changing Apache’s Directory Index (Optional)

In some cases, you’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 the web server to prefer PHP files over others, to make Apache look for an index.php file first. If you don’t do that, an index.html file placed in the document root of the application will always take precedence over an index.php file.

To make this change, open the dir.conf configuration file in a text editor of your choice. Here, we’ll use nano:

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

Copy

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>

Copy

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>

Copy

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. You can do that with the following command:

* sudo systemctl restart apache2

Copy

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

* sudo systemctl status apache2

Copy

Sample Output

● apache2.service - The Apache HTTP Server

Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)

Drop-In: /lib/systemd/system/apache2.service.d

└─apache2-systemd.conf

Active: active (running) since Thu 2021-07-15 09:22:59 UTC; 1h 3min ago

Main PID: 3719 (apache2)

Tasks: 55 (limit: 2361)

CGroup: /system.slice/apache2.service

├─3719 /usr/sbin/apache2 -k start

├─3721 /usr/sbin/apache2 -k start

└─3722 /usr/sbin/apache2 -k start

Jul 15 09:22:59 ubuntu1804 systemd[1]: Starting The Apache HTTP Server...

Jul 15 09:22:59 ubuntu1804 apachectl[3694]: AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1. Set the 'ServerName' di

Jul 15 09:22:59 ubuntu1804 systemd[1]: Started The Apache HTTP Server.

Press Q to exit this status output.

Installing PHP Extensions (Optional)

To extend 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

Copy

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

Copy

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

Copy

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

Copy

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 ...

Copy

At this point, your LAMP stack is installed and configured. Before you do anything else, we recommend that you set up an Apache virtual host where you can store your server’s configuration details.

Step 4 — Setting Up a Virtual Host (Recommended)

When using the Apache web server, you can use *virtual hosts* (similar to server blocks in Nginx) to encapsulate configuration details and host more than one domain from a single server. We will set up a domain called **your\_domain**, but you should **replace this with your own domain name**. To learn more about setting up a domain name with DigitalOcean, see our [Introduction to DigitalOcean DNS](https://www.digitalocean.com/community/tutorials/an-introduction-to-digitalocean-dns).

Apache on Ubuntu 18.04 has one server block enabled by default that is configured to serve documents from the /var/www/html directory. While this works well for a single site, it can become unwieldy if you are hosting multiple sites. Instead of modifying /var/www/html, let’s create a directory structure within /var/www for **your\_domain** site, leaving /var/www/html in place as the default directory to be served if a client request doesn’t match any other sites.

Create the directory for **your\_domain** as follows:

* sudo mkdir /var/www/your\_domain

Copy

Next, assign ownership of the directory with the $USER environment variable, which references the current logged user:

* sudo chown -R $USER:$USER /var/www/your\_domain

Copy

The permissions of your web root directory should be correct if you haven’t modified its umask value, but you can make sure by typing:

* sudo chmod -R 755 /var/www/your\_domain

Copy

Next, create a sample index.html page using nano or your favorite editor:

* nano /var/www/your\_domain/index.html

Copy

Inside, add the following sample HTML:

/var/www/your\_domain/index.html

<html>

<head>

<title>Welcome to Your\_domain!</title>

</head>

<body>

<h**1**>Success! The your\_domain server block is working!</h**1**>

</body>

</html>

Copy

Save and close the file when you are finished.

In order for Apache to serve this content, it’s necessary to create a virtual host file with the correct directives. Instead of modifying the default configuration file located at /etc/apache2/sites-available/000-default.conf directly, let’s make a new one at /etc/apache2/sites-available/your\_domain.conf:

* sudo nano /etc/apache2/sites-available/your\_domain.conf

Copy

Paste in the following configuration block, which is similar to the default, but updated for our new directory and domain name:

/etc/apache2/sites-available/your\_domain.conf

<VirtualHost \*:8**0**>

ServerAdmin webmaster@localhost

ServerName your\_domain

ServerAlias www.your\_domain

DocumentRoot /var/www/your\_domain

ErrorLog ${APACHE\_LOG\_DIR}/error.log

CustomLog ${APACHE\_LOG\_DIR}/access.log combined

</VirtualHost>

Copy

Notice that we’ve updated the DocumentRoot to our new directory and ServerAdmin to an email that the **your\_domain** site administrator can access. We’ve also added two directives: ServerName, which establishes the base domain that should match for this virtual host definition, and ServerAlias, which defines further names that should match as if they were the base name.

Save and close the file when you are finished.

Let’s enable the file with the a2ensite tool:

* sudo a2ensite your\_domain.conf

Copy

Disable the default site defined in 000-default.conf:

* sudo a2dissite 000-default.conf

Copy

Next, let’s test for configuration errors:

* sudo apache2ctl configtest

Copy

You should see the following output:

Output

Syntax OK

Restart Apache to implement your changes:

* sudo systemctl restart apache2

Copy

Apache should now be serving your domain name. You can test this by navigating to http://your\_domain, where you should see something like this:

Apache virtual host example

With that, your virtual host is fully set up. 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 5 — Testing PHP Processing on your Web Server

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

Create the file at the web root you created in the previous step by running:

* sudo nano /var/www/your\_domain/info.php

Copy

This will open a blank file. Add the following text, which is valid PHP code, inside the file:

/var/www/your\_domain/info.php

<?php

phpinfo();

Copy

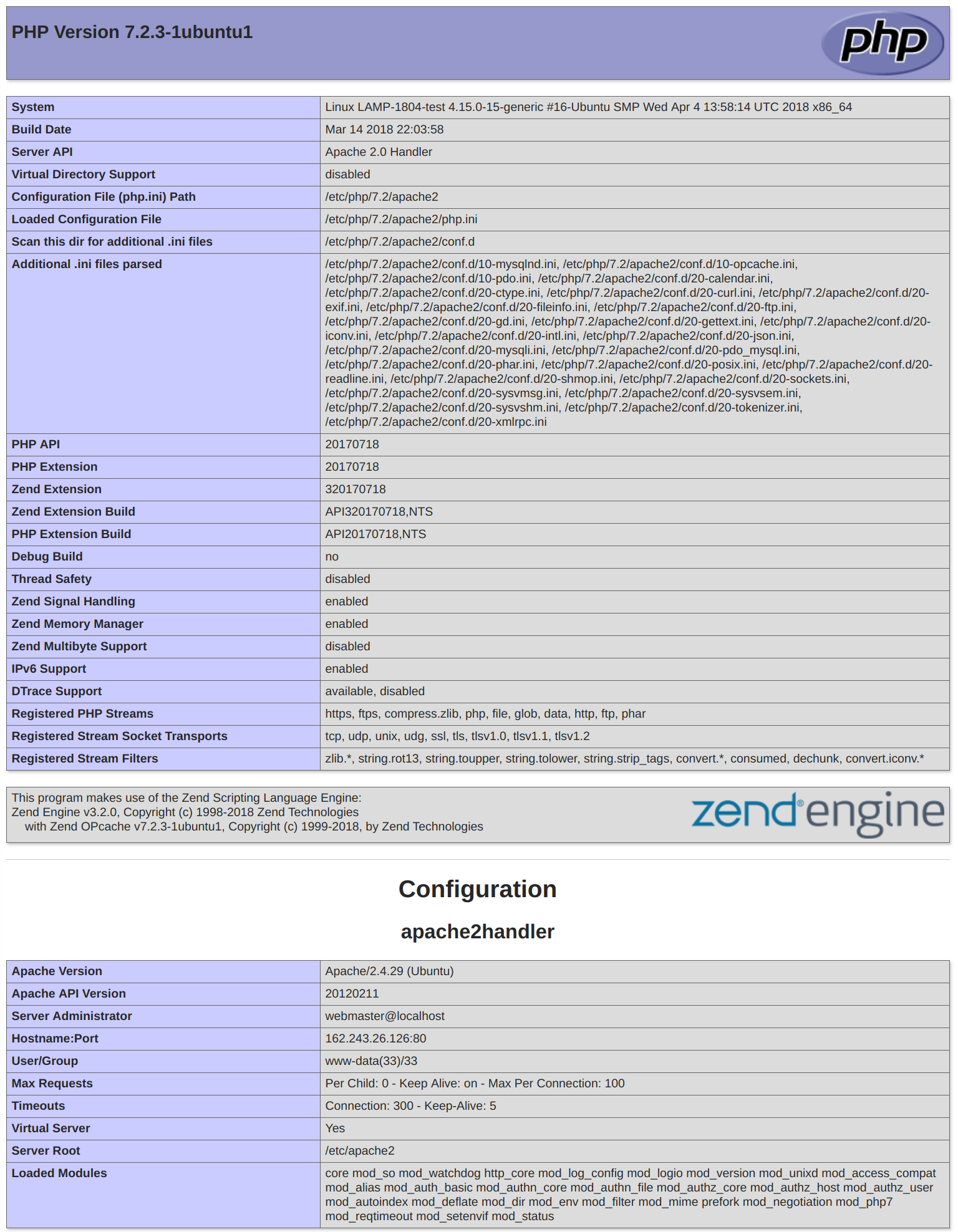
When you are finished, save and close the file.

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 or domain name again.

The address you will want to visit is:

http://your\_domain/info.php

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



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/your\_domain/info.php

Copy

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

**Conclusion:**

Digital Ocean has many droplets that you could installed the popular Applications just one click.

Try Digital Ocean now with Free $100 credits for 60 days. By click link below:

<https://m.do.co/c/fbc7a836fe39>