


 Subscribe Share Contents ▾

How To Install Linux, Apache, MariaDB, PHP (LAMP) stack on Debian 9



Posted September 4, 2018

 32.5k

LAMP STACK

APACHE

MARIADB

PHP

DATABASES

DEBIAN 9

By: Mark Drake

Not using **Debian 9**? Choose a different version:

CentOS 7



Debian 8



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**ariaDB database, and dynamic content is processed by **P**HP.

In this guide, we will install a LAMP stack on a Debian 9 server.

Prerequisites

In order to complete this tutorial, you will need to have a Debian 9 server with a non-root `sudo`-enabled user account and a basic firewall. This can be configured using our [initial server setup guide for Debian 9](#).

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 Debian'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.

Next, assuming that you have followed the initial server setup instructions by installing and enabling the UFW firewall, make sure that your firewall allows HTTP and HTTPS traffic.

When installed on Debian 9, UFW comes loaded with app profiles which you can use to tweak your firewall settings. View the full list of application profiles by running:

```
$ sudo ufw app list
```

The `WWW` profiles are used to manage ports used by web servers:

Output

Available applications:

```
. . .
WWW
WWW Cache
WWW Full
WWW Secure
. . .
```

If you inspect the `WWW Full` profile, it shows that it enables traffic to ports `80` and `443` :

```
$ sudo ufw app info "WWW Full"
```

Output

```
Profile: WWW Full
Title: Web Server (HTTP,HTTPS)
Description: Web Server (HTTP,HTTPS)
```

Ports:

```
80,443/tcp
```


Allow incoming HTTP and HTTPS traffic for this profile:

```
$ sudo ufw allow in "WWW 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:

`http://your_server_ip`

You will see the default Debian 9 Apache web page, which is there for informational and testing purposes. It should look something like this:



Apache2 Debian Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. 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

Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Debian 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 Debian 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.

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

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 MariaDB

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

MariaDB is a community-built fork of MySQL. In Debian 9, the default MySQL server is MariaDB 10.1, and the `mysql-server` package, which is normally used to install MySQL, is a transitional package that will actually install MariaDB. However, it's recommended that you install MariaDB using the program's actual package, `mariadb-server`.

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

```
$ sudo apt install mariadb-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, and 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 MariaDB which will remove some insecure default settings and lock down access to your database system. Start the interactive script by running:

```
$ sudo mysql_secure_installation
```

This will take you through a series of prompts where you can make some changes to your MariaDB installation's security options. The first prompt will ask you to enter the current database **root** password. This is an administrative account in MariaDB 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 MariaDB-specific account). Because you just installed MariaDB and haven't made any configuration changes yet, this password will be blank, so just press **ENTER** at the prompt.

The next prompt asks you whether you'd like to set up a database **root** password. Type **N** and then press **ENTER**. In Debian, the **root** account for MariaDB is tied closely to automated system maintenance, so we should not change the configured authentication methods for that account. Doing so would make it possible for a package update to break the database system by removing access to the administrative account. Later, we will cover how to optionally set up an additional administrative account for password access if socket authentication is not appropriate for your use case.

From there, you can press **Y** and then **ENTER** to accept the defaults for all the subsequent questions. This will remove some anonymous users and the test database, disable remote **root** logins, and load these new rules so that MariaDB immediately respects the changes you have made.

In new installs on Debian systems, the **root** MariaDB user is set to authenticate using the `unix_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) administrative rights.

Because the server uses the **root** account for tasks like log rotation and starting and stopping the server, it is best not to change the **root** account's authentication details. Changing the account credentials in the `/etc/mysql/debian.cnf` may work initially, but package updates could potentially overwrite those changes. Instead of modifying the **root** account, the package maintainers recommend creating a separate administrative account if you need to set up password-based access.

To do so, we will be creating a new account called **admin** with the same capabilities as the **root** account, but configured for password authentication. To do this, open up the MariaDB prompt from your terminal:

```
$ sudo mariadb
```

Now, we can create a new user with **root** privileges and password-based access. Change the username and password to match your preferences:

```
MariaDB [(none)]> GRANT ALL ON *.* TO 'admin'@'localhost' IDENTIFIED BY 'password' WITH GRANT OPTION
```

Flush the privileges to ensure that they are saved and available in the current session:

```
MariaDB [(none)]> FLUSH PRIVILEGES;
```

Following this, exit the MariaDB shell:

```
MariaDB [(none)]> exit
```

Now, any time you want to access your database as your new administrative user, you'll need to authenticate as that user with the password you just set using the following command:

```
$ mariadb -u admin -p
```

At this point, your database system is 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 MariaDB 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 MariaDB 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>
```

```
<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 - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2018-09-04 18:23:03 UTC; 9s ago
     Process: 22209 ExecStop=/usr/sbin/apachectl stop (code=exited, status=0/SUCCESS)
     Process: 22216 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
    Main PID: 22221 (apache2)
       Tasks: 6 (limit: 4915)
    CGroup: /system.slice/apache2.service
            └─22221 /usr/sbin/apache2 -k start
            └─22222 /usr/sbin/apache2 -k start
            └─22223 /usr/sbin/apache2 -k start
            └─22224 /usr/sbin/apache2 -k start
            └─22225 /usr/sbin/apache2 -k start
            └─22226 /usr/sbin/apache2 -k start
```

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:

Output

Sorting...

Full Text Search...

bandwidthd-pgsql/stable 2.0.1+cv520090917-10 amd64

Tracks usage of TCP/IP and builds html files with graphs

bluefish/stable 2.2.9-1+b1 amd64

advanced Gtk+ text editor for web and software development

cacti/stable 0.8.8h+ds1-10 all

web interface for graphing of monitoring systems

cakephp-scripts/stable 2.8.5-1 all

rapid application development framework for PHP (scripts)

ganglia-webfrontend/stable 3.6.1-3 all

cluster monitoring toolkit - web front-end

haserl/stable 0.9.35-2+b1 amd64

CGI scripting program for embedded environments

kdevelop-php-docs/stable 5.0.3-1 all

transitional package for kdevelop-php

kdevelop-php-docs-l10n/stable 5.0.3-1 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 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 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 called the *web root*.

In Debian 9, 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:

`/var/www/html/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 7.0.30-0+deb9u1

System	Linux LAMPtesterdebian9-02 4.9.0-7-amd64 #1 SMP Debian 4.9.110-1 (2018-07-05) x86_64
Build Date	Jun 14 2018 13:50:25
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-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	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, sslv2, 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 v3.0.0, Copyright (c) 1998-2017 Zend Technologies
with Zend OPcache v7.0.30-0+deb9u1, Copyright (c) 1999-2017, by Zend Technologies

zend®engine

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.

By: Mark Drake

 Upvote (5)  Subscribe  Share



We just made it easier for you to deploy faster.

[TRY FREE](#)

Related Tutorials

[How To Sync and Share Your Files with Seafile on Debian 9](#)

[How To Deploy a PHP Application with Kubernetes on Ubuntu 16.04](#)

[How To Install and Configure an Apache ZooKeeper Cluster on Ubuntu 18.04](#)

[How To Sync and Share Your Files with Seafile on Ubuntu 18.04](#)

[How to Back Up, Import, and Migrate Your Apache Kafka Data on Ubuntu 18.04](#)

4 Comments

Leave a comment...

Log In to Comment

^ [itf1e0fc1dd55fcea8e70199c7](#) *October 23, 2018*

- o \$ sudo ufw app info "WWW Full" # returns an error
use this instead:
\$ sudo ufw allow 443
\$ sudo ufw allow 80

^ [mdrake](#) MOD *October 23, 2018*

- o Hello [@itf1e0fc1dd55fcea8e70199c7!](#)

I just went through this tutorial and I didn't see any errors when I ran `sudo ufw app info "WWW Full"`. However, that command is only used to show information about the `WWW Full` UFW profile, and running it won't change your server's firewall rules. To do that, you'll need to run the next command shown in the guide:

```
$ sudo ufw allow in "WWW Full"
```

That said, this command performs the exact same action as running both `sudo ufw allow 443` and `sudo ufw allow 80`, so either will work within the context of this tutorial.

^ [itf1e0fc1dd55fcea8e70199c7](#) *October 24, 2018*

- o I made a mistake there.
It was not 'sudo ufw app info "WWW Full"' but that 'sudo ufw allow in "WWW Full"' that did not work. The error I got was "ERROR: Need 'to' or 'from' clause"

^ [HeebieJeebies](#) November 2, 2018

0 Just a quick correction...

The line that reads:

```
GRANT ALL ON *. TO 'admin'@'localhost' IDENTIFIED BY 'password' WITH GRANT OPTION;*
```

should read

```
GRANT ALL PRIVILEGES ON *. TO 'admin'@'localhost' IDENTIFIED BY 'password' WITH GRANT OPTION;*
```



This work is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.



Copyright © 2019 DigitalOcean™ Inc.

[Community](#) [Tutorials](#) [Questions](#) [Projects](#) [Tags](#) [Newsletter](#) [RSS](#) 

[Distros & One-Click Apps](#) [Terms, Privacy, & Copyright](#) [Security](#) [Report a Bug](#) [Write for DOnations](#) [Shop](#)