How To Install Linux, Apache, MySQL, PHP (LAMP) stack On CentOS 7

By Mitchell Anicas

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 (using MariaDB), and dynamic content is processed by **P**HP.

In this guide, we’ll get a LAMP stack installed on an CentOS 7 VPS. CentOS will fulfill our first requirement: a Linux operating system.

Note: The LAMP stack can be installed automatically on your Droplet by adding this script to its User Data when launching it. Check out this tutorial to learn more about Droplet User Data.

Prerequisites

Before you begin with this guide, you should have a separate, non-root user account set up on your server. You can learn how to do this by completing steps 1-4 in the initial server setup for CentOS 7.

Step One — Install Apache

The Apache web server is currently the most popular web server in the world, which makes it a great default choice for hosting a website.

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

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

```
sudo yum install httpd
```

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.

Afterwards, your web server is installed.

Once it installs, you can start Apache on your VPS:

```
sudo systemctl start httpd.service
```

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 CentOS 7 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.

The last thing you will want to do is enable Apache to start on boot. Use the following command to do so:

```
sudo systemctl enable httpd.service
```

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 one or two lines back. They are both 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 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:

```
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 Two — Install MySQL (MariaDB)

Now that we have our web server up and running, it is time to install MariaDB, a MySQL drop-in replacement. MariaDB is a community-developed fork of the MySQL relational database management system. Basically, it will organize and provide access to databases where our site can store information.

Again, we can use `yum` 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 yum install mariadb-server mariadb
```

When the installation is complete, we need to start MariaDB with the following command:

```
sudo systemctl start mariadb
```

Now that our MySQL database is running, 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:

```
sudo mysql_secure_installation
```

The prompt will ask you for your current root password. Since you just installed MySQL, you most likely won't have one, so leave it blank by pressing enter. Then the prompt will ask you if you want to set a root password. Go ahead and enter `Y`, and follow the instructions:

```
Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MariaDB
root user without the proper authorization.
```

```
New password: password
Re-enter new password: password
Password updated successfully!
Reloading privilege tables..
 ... Success!
```
For the rest of the questions, you should simply hit the “ENTER” key through each prompt to accept the default values. This will remove some sample users and databases, disable remote root logins, and load these new rules so that MySQL immediately respects the changes we have made.

The last thing you will want to do is enable MariaDB to start on boot. Use the following command to do so:

```
sudo systemctl enable mariadb.service
```
At this point, your database system is now set up and we can move on.

Step Three — 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 `yum` system to install our components. We’re going to include the php-mysql package as well:

```
sudo yum install php php-mysql
```
This should install PHP without any problems. We need to restart the Apache web server in order for it to work with PHP. You can do this by typing this:

```
sudo systemctl restart httpd.service
```

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 type this into your system:

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

```
php-bcmath.x86_64 : A module for PHP applications for using the bcmath
library
php-cli.x86_64 : Command-line interface for PHP
php-common.x86_64 : Common files for PHP
php-dba.x86_64 : A database abstraction layer module for PHP applications
php-devel.x86_64 : Files needed for building PHP extensions
php-embedded.x86_64 : PHP library for embedding in applications
php-enchant.x86_64 : Enchant spelling extension for PHP applications
php-fpm.x86_64 : PHP FastCGI Process Manager
php-gd.x86_64 : A module for PHP applications for using the gd graphics
library
. . .
```

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

```
yum info 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-fpm` module does, we could type this:

```
yum info php-fpm
```

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

```
. . .
Summary     : PHP FastCGI Process Manager
URL         : http://www.php.net/
License     : PHP and Zend and BSD
Description : PHP-FPM (FastCGI Process Manager) is an alternative PHP
FastCGI
            : implementation with some additional features useful for
sites of
            : any size, especially busier sites.
```

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

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

```
sudo yum install php-fpm
```

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

```
sudo yum install package1 package2 ...
```

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

Step Four — 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 CentOS 7, this directory is located at `/var/www/html/`. We can create the file at that location by typing:

```
sudo vi /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:

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

When you are finished, save and close the file.

If you are running a firewall, run the following commands to allow HTTP and HTTPS traffic:

```
sudo firewall-cmd --permanent --zone=public --add-service=http
sudo firewall-cmd --permanent --zone=public --add-service=https
sudo firewall-cmd --reload
```

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 5.4.16

System	Linux lamp-c7 3.10.0-123.el7.x86_64 #1 SMP Mon Jun 30 12:09:22 UTC 2014 x86_64
Build Date	Jun 10 2014 02:54:27
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc
Loaded Configuration File	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
Additional .ini files parsed	/etc/php.d/curl.ini, /etc/php.d/fileinfo.ini, /etc/php.d/json.ini, /etc/php.d/mysql.ini, /etc/php.d/mysqli.ini, /etc/php.d/pdo.ini, /etc/php.d/pdo_mysql.ini, /etc/php.d/pdo_sqlite.ini, /etc/php.d/phar.ini, /etc/php.d/sqlite3.ini, /etc/php.d/zip.ini
PHP API	20100412
PHP Extension	20100525
Zend Extension	220100525
Zend Extension Build	API220100525,NTS
PHP Extension Build	API20100525,NTS

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.

Reference

https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-centos-7