**How To Install Linux, Nginx, MySQL, PHP (LEMP) stack On CentOS 7**

**Introduction**

A LEMP software 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 **E**Nginx web server (which replaces the Apache component of a LAMP stack). 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 LEMP stack installed on an CentOS 7 VPS. CentOS 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 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](https://www.digitalocean.com/community/articles/initial-server-setup-with-centos-7).

**Note about SELinux:** If you run into issues with Nginx not running, make sure the SELinux context of your Nginx configuration files is correct or change the SELinux mode to permissive or disabled.

**Step One — Install Nginx**

In order to display web pages to our site visitors, we are going to employ Nginx, a modern, efficient web server.

To add the CentOS 7 Nginx yum repository, open terminal and use the following command:

sudo rpm -Uvh http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-centos-7-0.el7.ngx.noarch.rpm

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 that you have permission to run commands with root privileges.

Now that the Nginx repository is installed on your server, install Nginx using the following yum command:

sudo yum install nginx

Afterwards, your web server is installed.

Once it is installed, you can start Nginx on your VPS:

sudo systemctl start nginx.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://server\_domain\_name\_or\_IP/

You will see the default CentOS 7 Nginx 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.

Before continuing, you will want to do is enable Nginx to start on boot. Use the following command to do so:

sudo systemctl enable nginx.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 instuctions:

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 authorisation.

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 and php-fpm packages as well:

sudo yum install php php-mysql php-fpm

**Configure the PHP Processor**

We now have our PHP components installed, but we need to make a slight configuration change to make our setup more secure.

Open the main php-fpm configuration file with root privileges:

sudo vi /etc/php.ini

What we are looking for in this file is the parameter that sets cgi.fix\_pathinfo. This will be commented out with a semi-colon (;) and set to "1" by default.

This is an extremely insecure setting because it tells PHP to attempt to execute the closest file it can find if a PHP file does not match exactly. This basically would allow users to craft PHP requests in a way that would allow them to execute scripts that they shouldn't be allowed to execute.

We will change both of these conditions by uncommenting the line and setting it to "0" like this:

cgi.fix\_pathinfo=0

Save and close the file when you are finished.

Next, open the php-fpm configuration file www.conf:

sudo vi /etc/php-fpm.d/www.conf

Find the line that specifies the listen parameter, and change it so it looks like the following:

listen = /var/run/php-fpm/php-fpm.sock

Then save and quit.

Now, we just need to start our PHP processor by typing:

sudo systemctl start php-fpm

This will implement the change that we made.

Next, enable php-fpm to start on boot:

sudo systemctl enable php-fpm.service

**Step Four — Configure Nginx to Process PHP Pages**

Now, we have all of the required components installed. The only configuration change we still need to do is tell Nginx to use our PHP processor for dynamic content.

We do this on the server block level (server blocks are similar to Apache's virtual hosts). Open the default Nginx server block configuration file by typing:

sudo vi /etc/nginx/conf.d/default.conf

Currently, with the comments removed, the Nginx default server block looks like this:

server {

listen 80;

server\_name localhost;

location / {

root /usr/share/nginx/html;

index index.html index.htm;

}

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root /usr/share/nginx/html;

}

}

We need to make some changes to this file for our site.

* First, we need to add an index.php option as the first value of our index directive to allow PHP index files to be served when a directory is requested
* We also need to modify the server\_name directive to point to our server's domain name or public IP address
* The actual configuration file includes some commented out lines that define error processing routines. We will uncomment those to include that functionality
* For the actual PHP processing, we will need to uncomment a portion of another section. We will also need to add a try\_files directive to make sure Nginx doesn't pass bad requests to our PHP processor

The changes that you need to make are in red in the text below:

server {

listen 80;

server\_name server\_domain\_name\_or\_IP;

root /usr/share/nginx/html;

index index.php index.html index.htm;

location / {

try\_files $uri $uri/ =404;

}

error\_page 404 /404.html;

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root /usr/share/nginx/html;

}

location ~ \.php$ {

try\_files $uri =404;

fastcgi\_pass unix:/var/run/php-fpm/php-fpm.sock;

fastcgi\_index index.php;

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

include fastcgi\_params;

}

}

When you've made the above changes, you can save and close the file.

Restart Nginx to make the necessary changes:

sudo systemctl restart nginx

**Step Five — 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 /usr/share/nginx/html/. We can create the file at that location by typing:

sudo vi /usr/share/nginx/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(); ?>

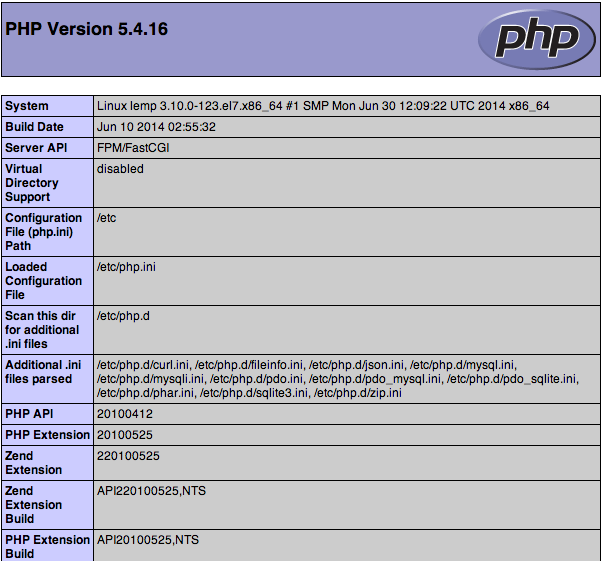
When you are finished, save and close the file.

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:



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 /usr/share/nginx/html/info.php

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

**Conclusion**

Now that you have a LEMP 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.

# How To Install and Secure phpMyAdmin with Nginx on a CentOS 7 Server

### Introduction

Relational database management systems like MySQL and MariaDB are needed for a significant portion of web sites and applications. However, not all users feel comfortable administering their data from the command line.

To solve this problem, a project called phpMyAdmin was created in order to offer an alternative in the form of a web-based management interface. In this guide, we will demonstrate how to install and secure a phpMyAdmin configuration on a CentOS 7 server. We will build this setup on top of the Nginx web server, which has a good performance profile and can handle heavy loads better than some other web servers.

## Prerequisites

Before we begin, there are a few requirements that need to be settled.

To ensure that you have a solid base to build this system upon, you should run through our [initial server setup guide for CentOS 7](https://www.digitalocean.com/community/articles/initial-server-setup-with-centos-7). Among other things, this will walk you through setting up a non-root user with sudo access for administrative commands.

The second prerequisite that must be fulfilled in order to start on this guide is to install a LEMP (Linux, Nginx, MariaDB, and PHP) stack on your CentOS 7 server. This is the platform that we will use to serve our phpMyAdmin interface (MariaDB is also the database management software that we are wishing to manage). If you do not yet have a LEMP installation on your server, follow our tutorial on [installing LEMP on CentOS 7](https://www.digitalocean.com/community/tutorials/how-to-install-linux-nginx-mysql-php-lemp-stack-on-centos-7).

When your server is in a properly functioning state after following these guides, you can continue on with the rest of this page.

## Step One — Install phpMyAdmin

With our LEMP platform already in place, we can begin right away with installing the phpMyAdmin software. Unfortunately, phpMyAdmin is not available in CentOS 7's default repository.

To get the packages we need, we'll have to add an additional repo to our system. The EPEL repo (**E**xtra **P**ackages for **E**nterprise **L**inux) contains many additional packages, including the phpMyAdmin package we are looking for.

Luckily, the procedure for adding the EPEL repository has gotten a lot easier. There is actually a package called epel-release that reconfigures our package manager to use the EPEL repos.

We can install that now by typing:

sudo yum install epel-release

Now that you have access to the EPEL repository, you can install phpMyAdmin through yum:

sudo yum install phpmyadmin

The installation will now complete. For the Nginx web server to find and serve the phpMyAdmin files correctly, we just need to create a symbolic link from the installation files to our Nginx document root directory by typing this:

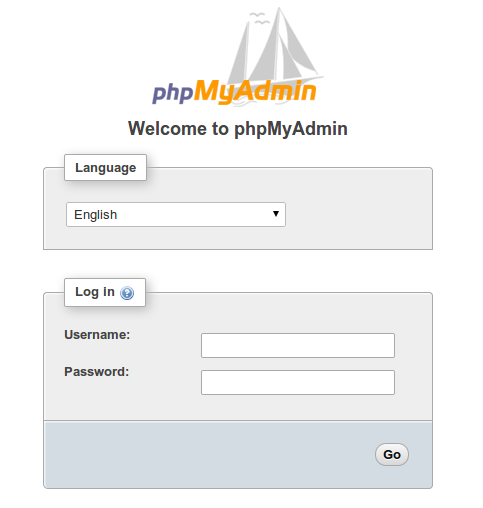
sudo ln -s /usr/share/phpMyAdmin /usr/share/nginx/html

We should also restart our PHP processor to be sure that it can load the additional PHP modules that we installed:

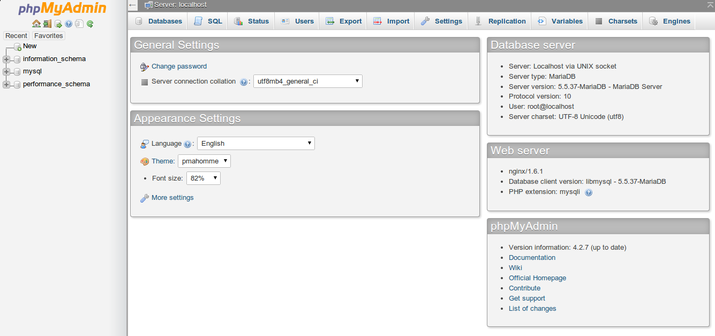
sudo systemctl restart php-fpm

With that, our phpMyAdmin installation is now operational. To access the interface, go to your server's domain name or public IP address followed by /phpMyAdmin, in your web browser:

http://server\_domain\_or\_IP/phpMyAdmin



To sign in, use a username/password pair of a valid MariaDB user. The root user and the MariaDB administrative password is a good choice to get started. You will then be able to access the administrative interface:



Click around to get familiar with the interface. In the next section, we will take steps to secure our new interface.

### ****Forward pada Postfix iRedMail****

Dalam melakukan setting email, mungkin ada permintaan supaya suatu email diforward ke email lain. Forward ini bisa berarti move atau berarti copy. Untuk mengetahui setting di Postfix iRedmail terkait forward, perhatikan setting pada main.cf berikut ini:  
  
virtual\_alias\_maps = proxy:mysql:/etc/postfix/mysql/virtual\_alias\_maps.cf, proxy:mysql:/etc/postfix/mysql/domain\_alias\_maps.cf, proxy:mysql:/etc/postfix/mysql/catchall\_maps.cf, proxy:mysql:/etc/postfix/mysql/domain\_alias\_catchall\_maps.cf  
  
Jika kita melihat isi dari /etc/postfix/mysql/virtual\_alias\_maps.cf maka isinya adalah seperti berikut ini:  
user        = vmail  
password    = passW8sBqJ  
hosts       = 127.0.0.1  
port        = 3306  
dbname      = vmail  
query       = SELECT alias.goto FROM alias,domain WHERE alias.address='%s' AND alias.domain='%d' AND alias.domain=domain.domain AND alias.active=1 AND domain.backupmx=0 AND domain.active=1

Jika melihat tabel alias maka isinya seperti berikut ini  


Jika kita ingin menambahkan forward pada suatu email maka kita dapat mengeditnya seperti berikut ini:  


Pada contoh di atas, email ferdi akan masuk ke mailbox ferdi dan juga diforward ke email arisg.

Ada juga email tidak mempunyai mailbox sendiri dan langsung diforward ke orang lain misalnya seperti contoh berikut ini  
http://www.proweb.co.id/pic/forwardonly-1152.png