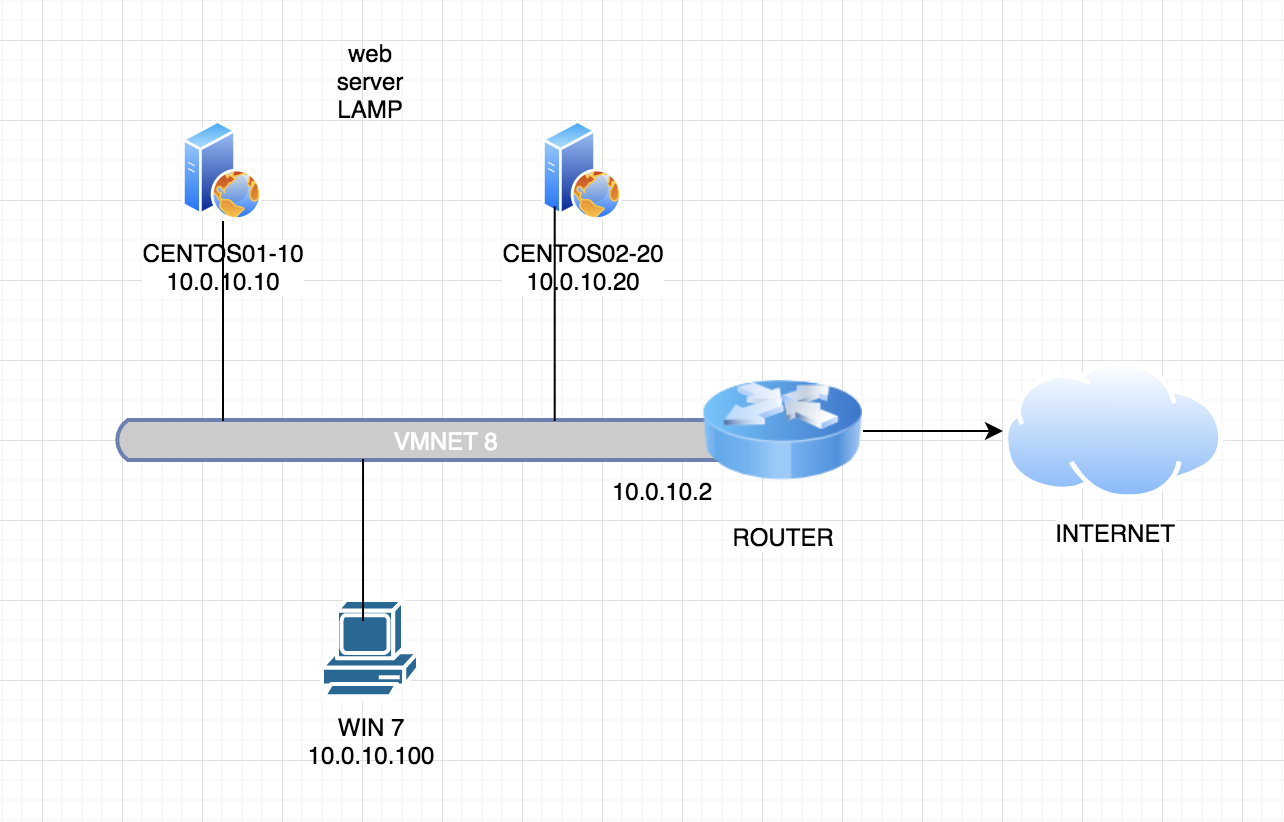
# 1. Install Apache, PHP And MySQL On CentOS 7 (LAMP)



In this tutorial, I use the hostname *centos01.fpt.local* with the IP address *10.0.10.10*. These settings might differ for you, so you have to replace them where appropriate.

I will add the EPEL repo here to install latest phpMyAdmin as follows:

*rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY\*  
yum -y install epel-release*

**2 Installing MySQL / MariaDB**

MariaDB is a MySQL fork of the original MySQL developer Monty Widenius. MariaDB is compatible with MySQL and I've chosen to use MariaDB here instead of MySQL. To install MySQL, we do install MariaDB like this:

*yum -y install mariadb-server mariadb*

Then we create the system startup links for MySQL (so that MySQL starts automatically whenever the system boots) and start the MySQL server:

*systemctl start mariadb.service  
systemctl enable mariadb.service*

Set passwords for the MySQL root account:

*mysql\_secure\_installation*

*[root@server1 ~]# mysql\_secure\_installation  
/usr/bin/mysql\_secure\_installation: line 379: find\_mysql\_client: command not found  
  
NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB  
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!  
  
In order to log into MariaDB to secure it, we'll need the current  
password for the root user.  If you've just installed MariaDB, and  
you haven't set the root password yet, the password will be blank,  
so you should just press enter here.  
  
Enter current password for root (enter for none):****<--ENTER*** *OK, successfully used password, moving on...  
  
Setting the root password ensures that nobody can log into the MariaDB  
root user without the proper authorisation.  
  
Set root password? [Y/n]   
New password:****<--yourmariadbpassword*** *Re-enter new password:****<--yourmariadbpassword*** *Password updated successfully!  
Reloading privilege tables..  
 ... Success!  
  
  
By default, a MariaDB installation has an anonymous user, allowing anyone  
to log into MariaDB without having to have a user account created for  
them.  This is intended only for testing, and to make the installation  
go a bit smoother.  You should remove them before moving into a  
production environment.  
  
Remove anonymous users? [Y/n]****<--ENTER*** *... Success!  
  
Normally, root should only be allowed to connect from 'localhost'.  This  
ensures that someone cannot guess at the root password from the network.  
  
Disallow root login remotely? [Y/n]****<--ENTER*** *... Success!  
  
By default, MariaDB comes with a database named 'test' that anyone can  
access.  This is also intended only for testing, and should be removed  
before moving into a production environment.  
  
Remove test database and access to it? [Y/n]****<--ENTER*** *- Dropping test database...  
 ... Success!  
 - Removing privileges on test database...  
 ... Success!  
  
Reloading the privilege tables will ensure that all changes made so far  
will take effect immediately.  
  
Reload privilege tables now? [Y/n]****<--ENTER*** *... Success!  
  
Cleaning up...  
  
All done!  If you've completed all of the above steps, your MariaDB  
installation should now be secure.  
  
Thanks for using MariaDB!  
[root@server1 ~]#*

**3 Installing Apache2**

CentOS 7 ships with apache 2.4. Apache2 is directly available as a CentOS 7.0 package, therefore we can install it like this:

*yum -y install httpd*

*[root@server1 ~]# yum install httpd  
Loaded plugins: fastestmirror, langpacks  
Loading mirror speeds from cached hostfile  
 \* base: ftp.plusline.de  
 \* extras: mirror.23media.de  
 \* updates: mirror.23media.de  
Package httpd-2.4.6-17.el7.centos.1.x86\_64 already installed and latest version  
Nothing to do  
[root@server1 ~]#*

By default apache will be installed, if-not then please install it as shown above

Now configure your system to start Apache at boot time...

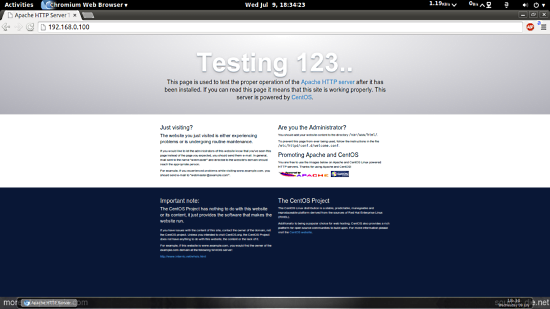
*systemctl start httpd.service*

*systemctl enable httpd.service*

In CentOS 7.0 uses Firewall-cmd, so I will customize it to allow external access to port *80* (http) and *443*(https).

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

Now direct your browser to *http://10.0.10.10*, and you should see the Apache2 placeholder page:

[](https://www.howtoforge.com/images/apache_php_mysql_on_centos_7.0_lamp/big/2.png)

**4 Installing PHP7.3**

PHP 7.3 is available for CentOS 7 and Fedora distributions from the Remi repository. Add it to your system by running

*sudo yum -y install http://rpms.remirepo.net/enterprise/remi-release-7.rpm*

*sudo yum -y install epel-release yum-utils*

## Step 2: Disable repo for PHP 5.4

By default, the enabled repository is for PHP 5.4. Disable this repo and enable on for PHP 7.3

sudo yum-config-manager --disable remi-php54  
sudo yum-config-manager --enable remi-php73

## Step 3: Install PHP 7.3 on CentOS 7 / Fedora

Once the repo has been enabled, install php 7.3 on CentOS 7 or Fedora using the command

sudo yum -y install php php-cli php-fpm php-mysqlnd php-zip php-devel php-gd php-mcrypt php-mbstring php-curl php-xml php-pear php-bcmath php-json

Check version installed

$ **php -v**  
 PHP 7.3.1 (cli) (built: Jan 8 2019 13:55:51) ( NTS )  
 Copyright (c) 1997-2018 The PHP Group  
 Zend Engine v3.3.1, Copyright (c) 1998-2018 Zend Technologies

I will update this guide once the final release of PHP 7.3 is available for production use.

Restart service httpd:

*#systemctl restart httpd*

**Testing PHP7.3 / Getting Details About Your PHP Installation**

The document root of the default website is /var/www/html. We will now create a small PHP file (info.php) in that directory and call it in a browser. The file will display lots of useful details about our PHP installation, such as the installed PHP version.

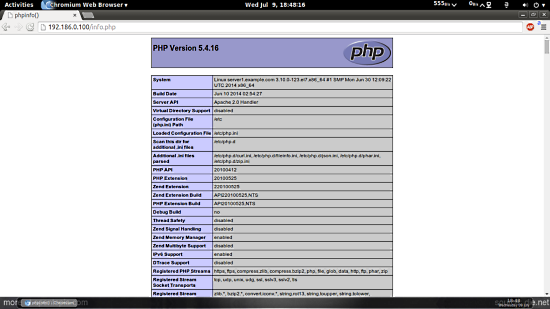
*vi /var/www/html/info.php*

<?php

phpinfo();

?>

Check your URL: **http://10.0.10.10/info.php**

[](https://www.howtoforge.com/images/apache_php_mysql_on_centos_7.0_lamp/big/4.png)

**7 phpMyAdmin installation**

phpMyAdmin is a web interface through which you can manage your MySQL databases.  
phpMyAdmin can now be installed as follows:

*yum install phpMyAdmin*

Now we configure phpMyAdmin. We change the Apache configuration so that phpMyAdmin allows connections not just from localhost (by commenting out the <Directory "*/usr/share/phpmyadmin*"> stanza):

*vi /etc/httpd/conf.d/phpMyAdmin.conf*

[...]  
Alias /phpMyAdmin /usr/share/phpMyAdmin

Alias /phpmyadmin /usr/share/phpMyAdmin

#<Directory /usr/share/phpMyAdmin/>

# <IfModule mod\_authz\_core.c>

# # Apache 2.4

# <RequireAny>

# Require ip 127.0.0.1

# Require ip ::1

# </RequireAny>

# </IfModule>

# <IfModule !mod\_authz\_core.c>

# # Apache 2.2

# Order Deny,Allow

# Deny from All

# Allow from 127.0.0.1

# Allow from ::1

# </IfModule>

#</Directory>

<Directory /usr/share/phpMyAdmin/>

Options none

AllowOverride Limit

Require all granted

</Directory>  
  
[...]

Next we change the authentication in phpMyAdmin from *cookie* to *http*:

*vi /etc/phpMyAdmin/config.inc.php*

[...]

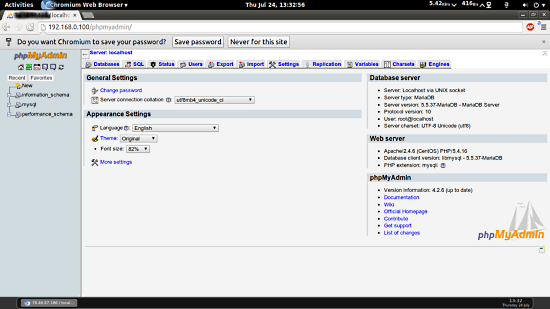
$cfg['Servers'][$i]['auth\_type'] = 'http'; // Authentication method (config, http or cookie based)?

[...]

Restart Apache:

*systemctl restart  httpd.service*

Afterwards, you can access phpMyAdmin under *http://10.0.10.10/phpmyadmin/*:

[](https://www.howtoforge.com/images/apache_php_mysql_on_centos_7.0_lamp/big/7.png)

**II. Install Wordpress on Centos**

### Introduction

WordPress is a free and open source website and blogging tool that uses PHP and MySQL. WordPress is currently the most popular CMS (Content Management System) on the Internet, and has over 20,000 plugins to extend its functionality. This makes WordPress a great choice for getting a website up and running quickly and easily.

In this guide, we will demonstrate how to get a WordPress instance set up with an Apache web server on CentOS 7.

## Prerequisites

Before you begin with this guide, there are a few steps that need to be completed first.

You will need a CentOS 7 server installed and configured with a non-root user that has sudo privileges. If you haven't done this yet, you can run through steps 1-4 in the [CentOS 7 initial server setup guide](https://www.digitalocean.com/community/tutorials/initial-server-setup-with-centos-7) to create this account.

Additionally, you'll need to have a LAMP (Linux, Apache, MySQL, and PHP) stack installed on your CentOS 7 server. If you don't have these components already installed or configured, you can use this guide to learn [how to install LAMP on CentOS 7](https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-centos-7).

When you are finished with these steps, you can continue with the installation of WordPress.

# **1.** Create a MySQL Database and User for WordPress

The first step that we will take is in preparation. WordPress uses a relational database to manage information for the site and its users. We have MariaDB (a fork of MySQL) installed already, which can provide this functionality, but we need to make a database and a user for WordPress to work with.

To get started, log into MySQL's root (administrative) account by issuing this command:

mysql -u root -p

You will be prompted for the password that you set for the root account when you installed MySQL. Once that password is submitted, you will be given a MySQL command prompt.

First, we'll create a new database that WordPress can control. You can call this whatever you would like, but I will be calling it wordpress for this example.

CREATE DATABASE wordpress;

**Note:** Every MySQL statement or command must end in a semi-colon (;), so check to make sure that this is present if you are running into any issues.

Next, we are going to create a new MySQL user account that we will use exclusively to operate on WordPress's new database. Creating one-function databases and accounts is a good idea, as it allows for better control of permissions and other security needs.

I am going to call the new account wordpressuser and will assign it a password of password. You should definitely use a different username and password, as these examples are not very secure.

CREATE USER 'wordpressuser'@'%' IDENTIFIED BY 'password';

At this point, you have a database and user account that are each specifically made for WordPress. However, the user has no access to the database. We need to link the two components together by granting our user access to the database.

GRANT ALL PRIVILEGES ON wordpress.\* TO wordpressuser@localhost IDENTIFIED BY 'password';

Now that the user has access to the database, we need to flush the privileges so that MySQL knows about the recent privilege changes that we've made:

FLUSH PRIVILEGES;

Once these commands have all been executed, we can exit out of the MySQL command prompt by typing:

exit

You should now be back to your regular SSH command prompt.

# **2.** Install WordPress

Before we download WordPress, there is one PHP module that we need to install to ensure that it works properly. Without this module, WordPress will not be able to resize images to create thumbnails. We can get that package directly from CentOS's default repositories using yum:

sudo yum install php-gd

Now we need to restart Apache so that it recognizes the new module:

sudo service httpd restart

We are now ready to download and install WordPress from the project's website. Luckily, the WordPress team always links the most recent stable version of their software to the same URL, so we can get the most up-to-date version of WordPress by typing this:

cd ~

yum install wget

wget http://wordpress.org/latest.tar.gz

This will download a compressed archive file that contains all of the WordPress files that we need. We can extract the archived files to rebuild the WordPress directory with tar:

tar -xzvf latest.tar.gz

cp -r ~/wordpress/\* /var/www/html/

mkdir /var/www/html/wp-content/uploads

Now we need to assign the correct ownership and permissions to our WordPress files and folders. This will increase security while still allowing WordPress to function as intended. To do this, we'll use chown to grant ownership to Apache's user and group:

sudo chown -R apache:apache /var/www/html/\*

With this change, the web server will be able to create and modify WordPress files, and will also allow us to upload content to the server.

# **3.** Configure WordPress

Most of the configuration required to use WordPress will be completed through a web interface later on. However, we need to do some work from the command line to ensure that WordPress can connect to the MySQL database that we created for it.

Begin by moving into the Apache root directory where you installed WordPress:

cd /var/www/html

The main configuration file that WordPress relies on is called wp-config.php. A sample configuration file that mostly matches the settings we need is included by default. All we have to do is copy it to the default configuration file location, so that WordPress can recognize and use the file:

cp wp-config-sample.php wp-config.php

Now that we have a configuration file to work with, let's open it in a text editor:

nano wp-config.php

The only modifications we need to make to this file are to the parameters that hold our database information. We will need to find the section titled MySQL settings and change the DB\_NAME, DB\_USER, and DB\_PASSWORD variables in order for WordPress to correctly connect and authenticate to the database that we created.

Fill in the values of these parameters with the information for the database that you created. It should look like this:

// \*\* MySQL settings - You can get this info from your web host \*\* //

/\*\* The name of the database for WordPress \*/

define('DB\_NAME', 'wordpress');

/\*\* MySQL database username \*/

define('DB\_USER', 'wordpressuser');

/\*\* MySQL database password \*/

define('DB\_PASSWORD', 'password');

These are the only values that you need to change, so save and close the file when you are finished.

# **4.** Complete Installation Through the Web Interface

Now that you have your files in place and your software is configured, you can complete the WordPress installation through the web interface. In your web browser, navigate to your server's domain name or public IP address:

http://10.0.10.10

First, you will need to select the language that you would like to install WordPress with. After selecting a language and clicking on **Continue**, you will be presented with the WordPress initial configuration page, where you will create an initial administrator account:

Code for virtual host:

<VirtualHost \*:80>

ServerName www.example2.com

DocumentRoot /var/www/example2.com/public\_html

ServerAlias example2.com

ErrorLog /var/www/example2.com/error.log

CustomLog /var/www/example2.com/requests.log combined

</VirtualHost>

<VirtualHost \*:80>

ServerName site01.vn

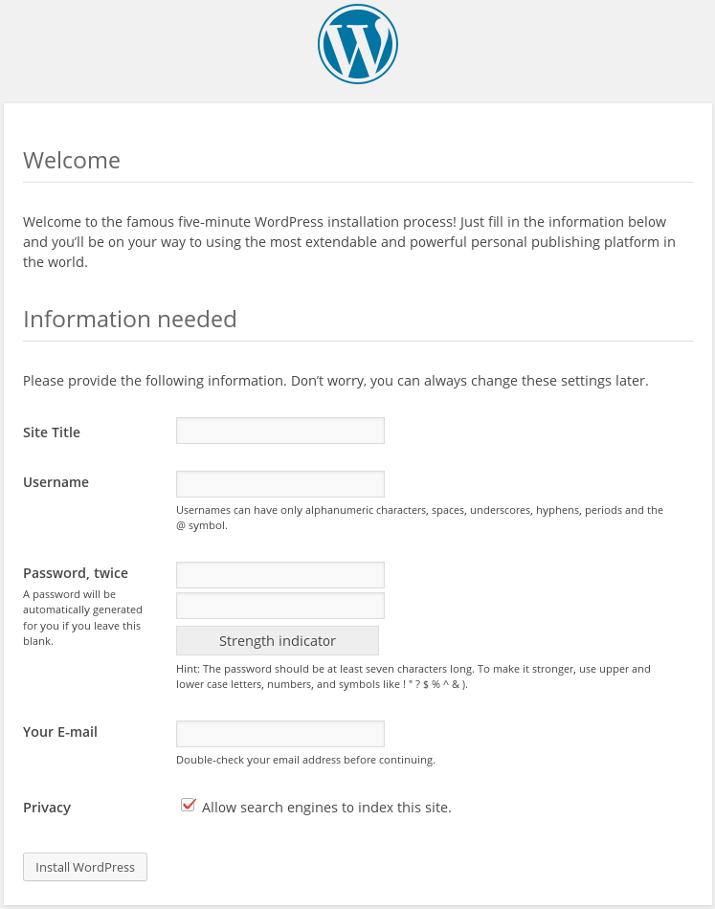
DocumentRoot /var/www/site01

ServerAlias www/site01.vn

# ErrorLog /var/www/example2.com/error.log

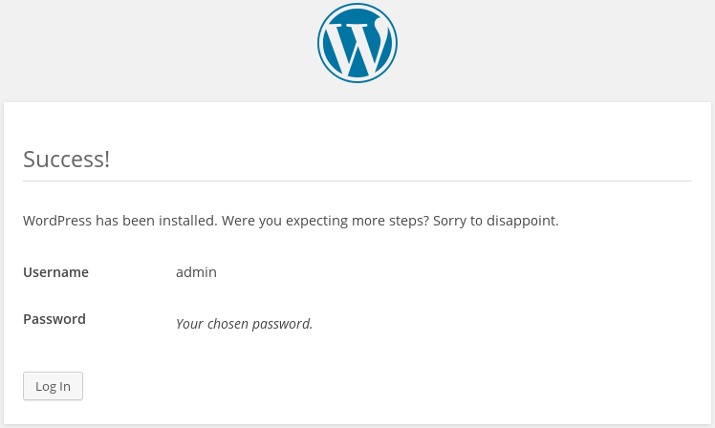
# CustomLog /var/www/example2.com/requests.log combined

</VirtualHost>

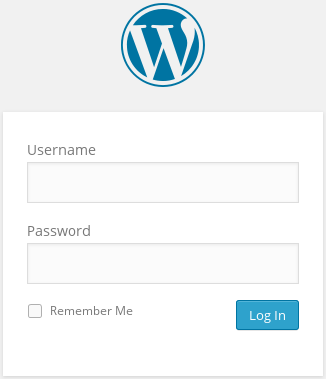


Fill out the information for the site and administrative account that you wish to make. When you are finished, click on the **Install WordPress** button at the bottom to continue.

WordPress will confirm the installation, and then ask you to log in with the account that you just created:



To continue, hit the **Log in** button at the bottom, then fill out your administrator account information:



After hitting **Log in**, you will be presented with your new WordPress dashboard:

