This tutorial shows how you can install an Apache webserver on a CentOS 7 server with PHP support and MySQL support. LAMP is short for Linux, Apache, MySQL, PHP.

In this tutorial, I use the hostname server1.example.com with the IP: 192.168.0.78. These settings might differ for you, so you have to replace them where appropriate.

I will add the EPEL (Extra Packages for Enterprise Linux) repository here to install latest phpMyAdmin, type the following into the command line as follows:

Note: ↵ = Press the enter key.

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

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. Run this command to install MariaDB with yum:

```
yum -y install mariadb-server mariadb ↵
```

Then we create the system start-up 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 ~]#
```

Installing Apache

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

```
yum -y install httpd
```

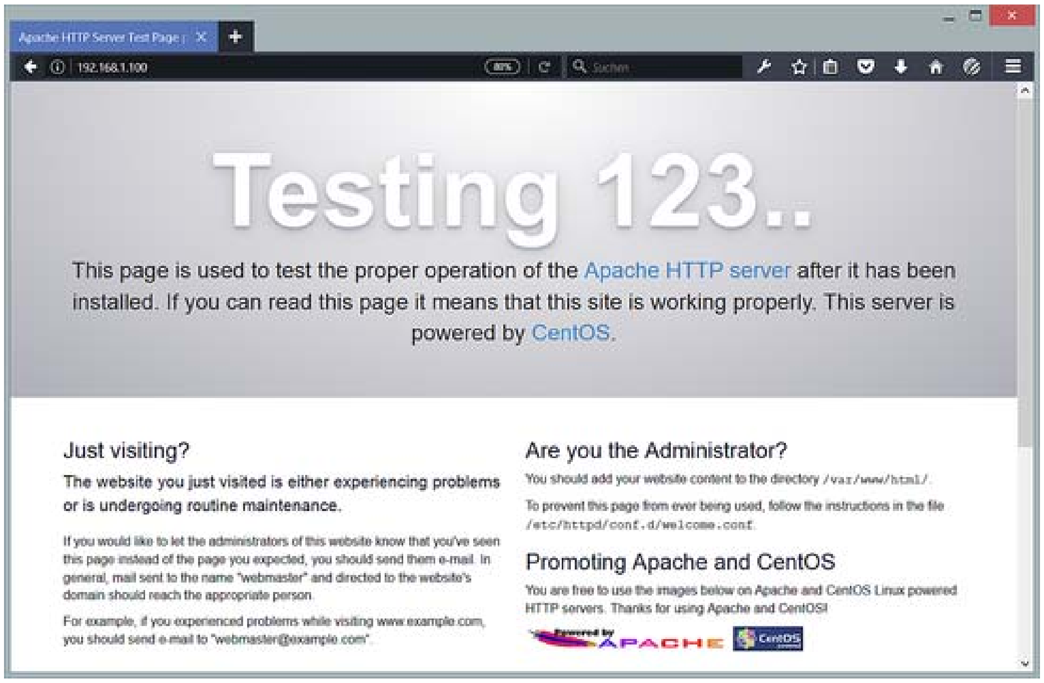
Now configure your system to start Apache at boot time... :

```
systemctl start httpd.service
systemctl enable httpd.service
```

To be able to access the web server from outside, we have to disable the default firewall. As this is only being used for learning purposes and not for a live production environment, we can issue the following command:

```
iptables -F
```

Now direct your browser to the IP address of your server, in my case http://192.168.0.78, and you should see the Apache placeholder page:



Testing 123..

This page is used to test the proper operation of the Apache HTTP server after it has been installed. If you can read this page it means that this site is working properly. This server is powered by CentOS.

Just visiting?

The website you just visited is either experiencing problems or is undergoing routine maintenance.

If you would like to let the administrators of this website know that you've seen this page instead of the page you expected, you should send them e-mail. In general, mail sent to the name "webmaster" and directed to the website's domain should reach the appropriate person.

For example, if you experienced problems while visiting www.example.com, you should send e-mail to "webmaster@example.com".

Are you the Administrator?

You should add your website content to the directory /var/www/html/.

To prevent this page from ever being used, follow the instructions in the file /etc/httpd/conf.d/welcome.conf.

Promoting Apache and CentOS

You are free to use the images below on Apache and CentOS Linux powered HTTP servers. Thanks for using Apache and CentOS!

Installing PHP

The PHP version that ships with CentOS is quite old (PHP 5.4), therefore I will show you in this step how to install the newer PHP versions like PHP 7.1 from Remi repository.

Add the Remi CentOS repository.

```
rpm -Uvh http://rpms.remirepo.net/enterprise/remi-release-7.rpm ↵
```

Install yum-utils as we need the yum-config-manager utility.

```
yum -y install yum-utils ↵
```

and run yum update.

```
yum update ↵
```

To install PHP 7.1, follow the below commands:

```
yum-config-manager --enable remi-php71 ↵
```

```
yum -y install php php-opcache ↵
```

We must restart Apache to apply the changes:

```
systemctl restart httpd.service ↵
```

Testing PHP / Getting Details About Your PHP Installation

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

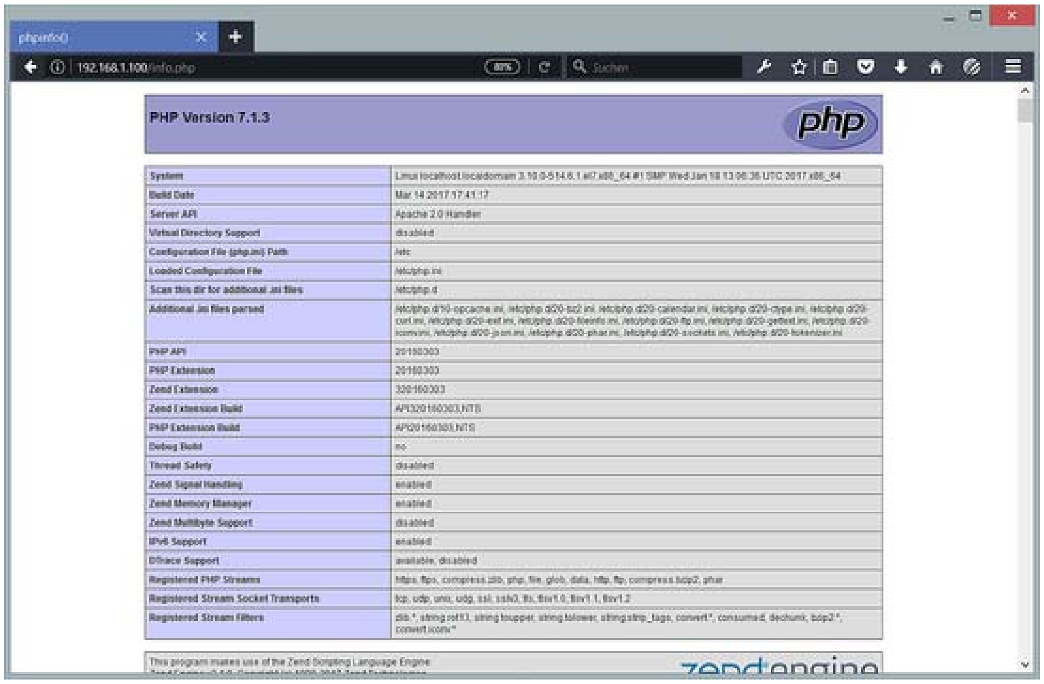
```
nano /var/www/html/info.php ↵
```

Type the following into the editor:

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

Save the file: `CTRL+X`, press the `enter` key to write file then exit the editor.

Now we call that file in a browser (e.g. http://192.168.0.78/info.php):



192.168.1.100/info.php

PHP Version 7.1.3

System	Linux localhost.localdomain 3.10.0-514.6.1.el7.x86_64 #1 SMP Wed Jan 18 13:06:36 UTC 2017 x86_64
Build Date	Mar 14 2017 17:41:17
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/10-opcache.ini, /etc/php.d/20-bz2.ini, /etc/php.d/20-calendar.ini, /etc/php.d/20-ctype.ini, /etc/php.d/20-curl.ini, /etc/php.d/20-exif.ini, /etc/php.d/20-fileinfo.ini, /etc/php.d/20-ftp.ini, /etc/php.d/20-gettext.ini, /etc/php.d/20-iconv.ini, /etc/php.d/20-json.ini, /etc/php.d/20-phar.ini, /etc/php.d/20-sockets.ini, /etc/php.d/20-tokenizer.ini
PHP API	20160303
PHP Extension	20160303
Zend Extension	320160303
Zend Extension Build	API320160303,NTS
PHP Extension Build	API20160303,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
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, compress.bzip2, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, bzip2.*, convert.iconv.*

This program makes use of the Zend Scripting Language Engine:

As you see, PHP 7.1 is working, and it's working through the Apache 2.0 Handler, as shown in the Server API line. If you scroll further down, you will see all modules that are already enabled in PHP. MySQL is not listed there which means we don't have MySQL support in PHP yet.

Setup MySQL Support In PHP

To setup MySQL support in PHP, we can install the php71w-mysql package. It's a good idea to install some other PHP modules as well as you might need them for your applications.

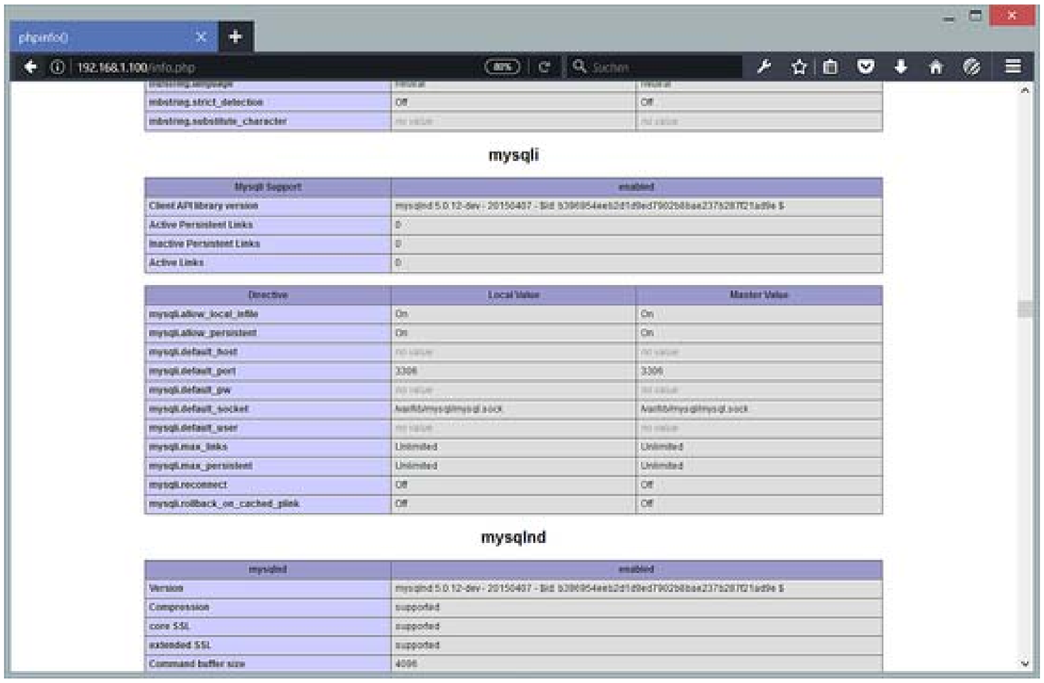
```
yum -y install php-mysql
```

```
yum -y install php-gd php-ldap php-odbc php-pear php-xml
php-xmlrpc php-mbstring php-soap curl curl-devel
```

Now restart Apache web server:

```
systemctl restart httpd.service
```

Now reload http://192.168.0.78/info.php in your browser and scroll down to the modules section again. You should now find lots of new modules like curl etc there.:



mbstring.strict_detection	Off	Off
mbstring.substitute_character	no value	no value

mysqli

MysqlI Support	enabled
Client API library version	mysqlnd 5.0.12-dev - 20150407 - $Id: b396954eeb2d1d9ed7902b8bae237b287f21ad9e $
Active Persistent Links	0
Inactive Persistent Links	0
Active Links	0

Directive	Local Value	Master Value
mysqli.allow_local_infile	On	On
mysqli.allow_persistent	On	On
mysqli.default_host	no value	no value
mysqli.default_port	3306	3306
mysqli.default_pw	no value	no value
mysqli.default_socket	/var/lib/mysql/mysql.sock	/var/lib/mysql/mysql.sock
mysqli.default_user	no value	no value
mysqli.max_links	Unlimited	Unlimited
mysqli.max_persistent	Unlimited	Unlimited
mysqli.reconnect	Off	Off
mysqli.rollback_on_cached_plink	Off	Off

mysqlnd

mysqlnd	enabled
Version	mysqlnd 5.0.12-dev - 20150407 - $Id: b396954eeb2d1d9ed7902b8bae237b287f21ad9e $
Compression	supported
core SSL	supported
extended SSL	supported
Command buffer size	4096

Install phpMyAdmin

phpMyAdmin is a web interface through which you can manage your MySQL databases.

phpMyAdmin can now be installed as follows:

```
yum -y install phpMyAdmin
```

Now we configure phpMyAdmin. We change the Apache configuration so that phpMyAdmin allows connections not just from localhost (by commenting out the <RequireAny> stanza and adding the 'Require all granted' line):

```
nano /etc/httpd/conf.d/phpMyAdmin.conf
```

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

<Directory /usr/share/phpMyAdmin/>
 AddDefaultCharset UTF-8

<IfModule mod_authz_core.c>
# Apache 2.4
# <RequireAny>
# Require ip 127.0.0.1
# Require ip ::1
# </RequireAny>
 Require all granted
 </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:

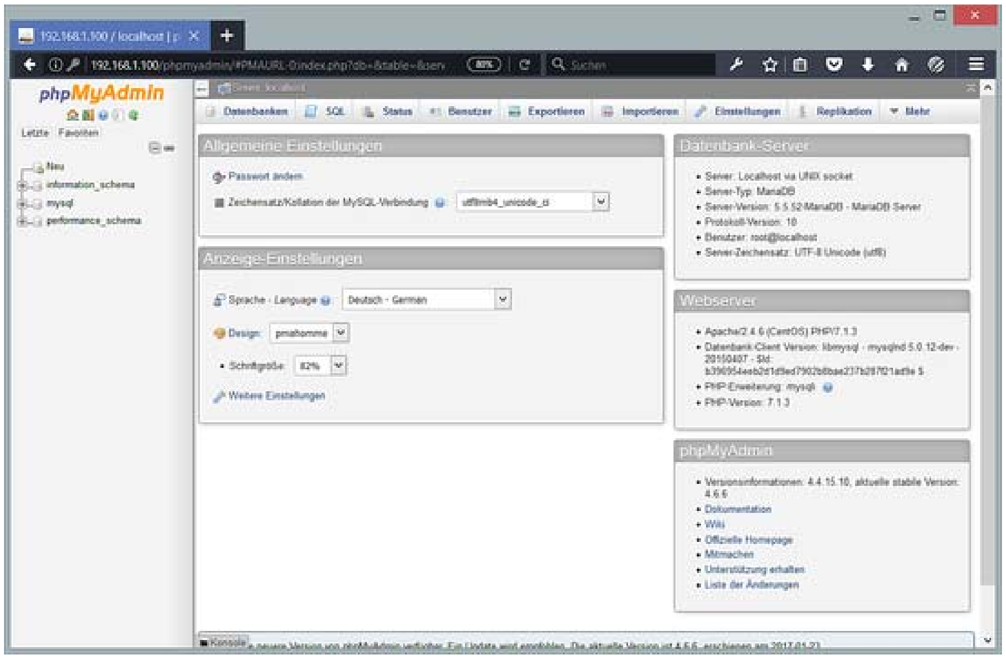
```
nano /etc/phpMyAdmin/config.inc.php
```

```
[...]
$cfg['Servers'][$i]['auth_type']     = 'http';    // Authentication
method (config, http or cookie based)?
[...]
```

Restart Apache:

```
systemctl restart  httpd.service
```

Afterwards, you can access phpMyAdmin under http://192.168.0.78/phpmyadmin/:



Allgemeine Einstellungen

- Passwort ändern
- Zeichensatz/Kollation der MySQL-Verbindung: utf8mb4_unicode_ci

Anzeige-Einstellungen

- Sprache - Language: Deutsch - German
- Design: pmahomme
- Schriftgröße: 82%
- Weitere Einstellungen

Datenbank-Server

- Server: Localhost via UNIX socket
- Server-Typ: MariaDB
- Server-Version: 5.5.52-MariaDB - MariaDB Server
- Protokoll-Version: 10
- Benutzer: root@localhost
- Server-Zeichensatz: UTF-8 Unicode (utf8)

Webserver

- Apache/2.4.6 (CentOS) PHP/7.1.3
- Datenbank-Client Version: libmysql - mysqlnd 5.0.12-dev - 20150407 - $Id: b390954eeb2d1d9ed7902b8bae237b287f21ad9e $
- PHP-Erweiterung: mysqli
- PHP-Version: 7.1.3

phpMyAdmin

- Versionsinformationen: 4.4.15.10, aktuelle stabile Version: 4.6.6
- Dokumentation
- Wiki
- Offizielle Homepage
- Mitmachen
- Unterstützung erhalten
- Liste der Änderungen

Konsole ...e neuere Version von phpMyAdmin verfügbar. Ein Update wird empfohlen. Die aktuelle Version ist 4.6.6, erschienen am 2017-01-23.