

Setting up a Debian 10 LAMP Server for PHP

Web Development

In this article, I am going to show you how to setup a LAMP (Linux, Apache, MySQL/MariaDB, PHP) server for PHP web development. I am going to use the newly released Debian 10 Buster GNU/Linux distribution for the demonstration. So, let's get started.

Updating APT Package Repository Cache:

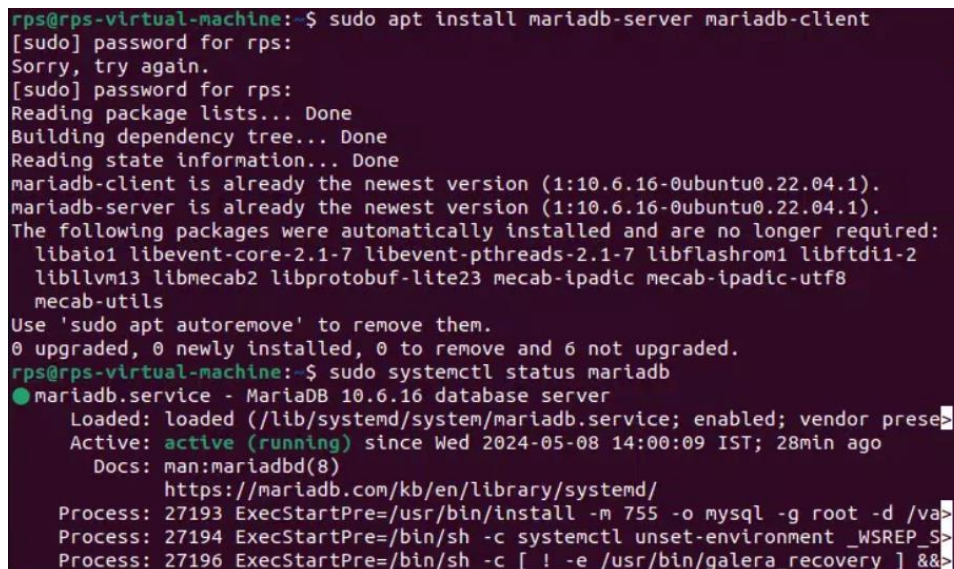
First, update the APT package repository cache with the following command:

```
$ sudo apt update
```

The APT package repository cache should be updated.

Installing and Configuring MySQL/MariaDB:

```
$ sudo apt install mariadb-server mariadb-client
```



```
rps@rps-virtual-machine:~$ sudo apt install mariadb-server mariadb-client
[sudo] password for rps:
Sorry, try again.
[sudo] password for rps:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
mariadb-client is already the newest version (1:10.6.16-0ubuntu0.22.04.1).
mariadb-server is already the newest version (1:10.6.16-0ubuntu0.22.04.1).
The following packages were automatically installed and are no longer required:
  libaio1 libevent-core-2.1-7 libevent-pthreads-2.1-7 libflashrom1 libftdi1-2
  liblvm13 libmecab2 libprotobuf-lite23 mecab-ipadic mecab-ipadic-utf8
  mecab-utils
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 6 not upgraded.
rps@rps-virtual-machine:~$ sudo systemctl status mariadb
● mariadb.service - MariaDB 10.6.16 database server
   Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; vendor prese
   Active: active (running) since Wed 2024-05-08 14:00:09 IST; 28min ago
     Docs: man:mariabdb(8)
           https://mariadb.com/kb/en/library/systemd/
   Process: 27193 ExecStartPre=/usr/bin/install -m 755 -o mysql -g root -d /va
   Process: 27194 ExecStartPre=/bin/sh -c systemctl unset-environment _WSREP_S
   Process: 27196 ExecStartPre=/bin/sh -c [ ! -e /usr/bin/galera_recovery ] &&>
```

To confirm the installation, press **Y** and then press **<Enter>**.

The APT package manager will download and install all the required packages.

At this point, MariaDB server and client packages will be installed.

Now, check whether **mariadb** service is running with the following command:

```
$ sudo systemctl status mariadb
```

As you can see, the **mariadb** service is running. It's also **enabled** to automatically start on system boot.

If in any case, **mariadb** service is not running, then start the service with the following command:

```
$ sudo systemctl start mariadb
```

```
$ sudo mysql_secure_installation
```

Press <Enter>.

```
May 08 14:00:09 rps-virtual-machine systemd[1]: Started MariaDB 10.6.16 database>
May 08 14:00:09 rps-virtual-machine /etc/mysql/debian-start[27240]: Upgrading M>
May 08 14:00:09 rps-virtual-machine /etc/mysql/debian-start[27243]: Looking for>
May 08 14:00:09 rps-virtual-machine /etc/mysql/debian-start[27243]: Looking for>
May 08 14:00:09 rps-virtual-machine /etc/mysql/debian-start[27243]: This instal>
rps@rps-virtual-machine:~$ sudo systemctl start mariadb
rps@rps-virtual-machine:~$ sudo mysql_secure_installation
```

```
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
haven't set the root password yet, you should just press enter here.
```

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

```
Setting the root password or using the unix_socket ensures that nobody
can log into the MariaDB root user without the proper authorisation.
```

```
You already have your root account protected, so you can safely answer 'n'.
```

```
You already have your root account protected, so you can safely answer 'n'.
```

```
Switch to unix_socket authentication [Y/n] Y
```

```
Enabled successfully!
```

```
Reloading privilege tables..
```

```
... Success!
```

```
You already have your root account protected, so you can safely answer 'n'.
```

```
Change the root password? [Y/n] n
```

```
... skipping.
```

```
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] y
```

```
... Success!
```

```
Normally, root should only be allowed to connect from 'localhost'. This
```

Creating New MySQL/MariaDB Users and Databases:

```
$ sudo mysql -u root -p
```

Now, type in the MariaDB **root** password you've already set and press <Enter>.

```
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] n
... skipping.

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] y
- 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] y
... Success!

Cleaning up...

All done! If you've completed all of the above steps, your MariaDB
```

Now, create a database **app1** with the following SQL statement:

```
$ CREATE DATABASE app1;
```

Now, create a new user (let's say, **shovon**), set a password for the user (let's say **123**) and grant the user permission to use the database **app1** with the following SQL statement:

```
installation should now be secure.

Thanks for using MariaDB!
rps@rps-virtual-machine:~$ sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 48
Server version: 10.6.16-MariaDB-0ubuntu0.22.04.1 Ubuntu 22.04

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE app1;
ERROR 1007 (HY000): Can't create database 'app1'; database exists
MariaDB [(none)]> GRANT ALL ON app1.* TO 'shovon'@'localhost' IDENTIFIED BY '123';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> FLUSH PREVILIGES;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MariaDB server version for the right syntax to use near 'PRE
VILIGES' at line 1
MariaDB [(none)]> FLUSH PRIVILEGES;
```



```
$ GRANT ALL ON app1.* TO 'rps'@'localhost' IDENTIFIED BY '123';
```

And then,

```
$ FLUSH PRIVILEGES;
```

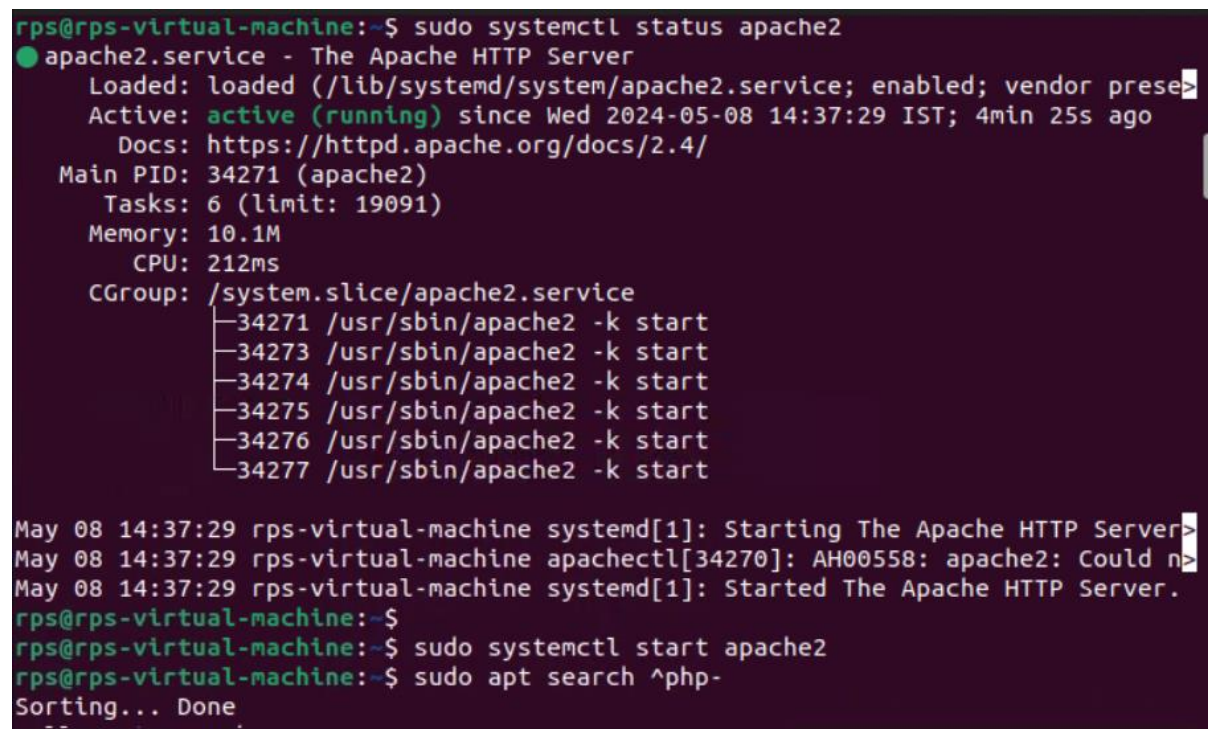
Now, exit out of the MariaDB shell as follows:

```
$ \q
```

Installing Apache Web Server and PHP:

Now, install Apache 2 web server and PHP with the following command:

```
$ sudo apt install apache2 php
```



```
rps@rps-virtual-machine:~$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor prese
   Active: active (running) since Wed 2024-05-08 14:37:29 IST; 4min 25s ago
     Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 34271 (apache2)
     Tasks: 6 (limit: 19091)
    Memory: 10.1M
       CPU: 212ms
    CGroup: /system.slice/apache2.service
           └─34271 /usr/sbin/apache2 -k start
             └─34273 /usr/sbin/apache2 -k start
               └─34274 /usr/sbin/apache2 -k start
                 └─34275 /usr/sbin/apache2 -k start
                   └─34276 /usr/sbin/apache2 -k start
                     └─34277 /usr/sbin/apache2 -k start

May 08 14:37:29 rps-virtual-machine systemd[1]: Starting The Apache HTTP Server
May 08 14:37:29 rps-virtual-machine apache2ctl[34270]: AH00558: apache2: Could n
May 08 14:37:29 rps-virtual-machine systemd[1]: Started The Apache HTTP Server.
rps@rps-virtual-machine:~$
rps@rps-virtual-machine:~$ sudo systemctl start apache2
rps@rps-virtual-machine:~$ sudo apt search ^php-
Sorting... Done
```

Installing PHP Extensions:

Debian 10 official package repository has a lot of PHP extensions pre-packaged.

You can list all the available PHP extensions/libraries with the following command:

```
$ sudo apt search ^php-
```

The package name of all the PHP extensions including their version number and short description should be listed. It's a very long list. So, it may take a while to find what you're looking for this way.

To install the most common PHP extensions/libraries, run the following command:

```
rps@rps-virtual-machine:~$ sudo apt install php-curl php-gd php-mbstring php-mysql
php-zip php-json php-xml
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libaio1 libevent-core-2.1-7 libevent-pthreads-2.1-7 libflashrom1 libftdi1-2
  libllvm13 libmecab2 libprotobuf-lite23 mecab-ipadic mecab-ipadic-utf8
  mecab-utils
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libonig5 libzip4 php8.1-curl php8.1-gd php8.1-mbstring php8.1-mysql
  php8.1-xml php8.1-zip
The following NEW packages will be installed:
  libonig5 libzip4 php-curl php-gd php-json php-mbstring php-mysql php-xml
  php-zip php8.1-curl php8.1-gd php8.1-mbstring php8.1-mysql php8.1-xml
  php8.1-zip
0 upgraded, 15 newly installed, 0 to remove and 6 not upgraded.
Need to get 1,073 kB of archives.
After this operation, 3,318 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://in.archive.ubuntu.com/ubuntu jammy/main amd64 libonig5 amd64 6.9.7.
```

Once the PHP extensions are installed, restart the Apache web server as follows:

```
$ sudo systemctl restart apache2
```

Enabling/Disabling Apache Modules:

Apache web server comes with a lot of modules. You can enable or disable them as you need.

To list all the available Apache 2 modules, run the following command

```
$ ls /etc/apache2/mods-available
```

As you can see, all the available Apache 2 modules are listed.

```

rps@rps-virtual-machine:~$ sudo systemctl restart apache2
rps@rps-virtual-machine:~$ ls /etc/apache2/mods-available
access_compat.load    dir.load              proxy_express.load
actions.conf          dump_io.load          proxy_fcgi.load
actions.load          echo.load             proxy_fdpass.load
alias.conf            env.load              proxy_ftp.conf
alias.load            expires.load          proxy_ftp.load
allowmethods.load     ext_filter.load       proxy_hcheck.load
asis.load             file_cache.load       proxy_html.conf
auth_basic.load        filter.load            proxy_html.load
auth_digest.load       headers.load          proxy_http2.load
auth_form.load         heartbeat.load         proxy_http.load
authn_anon.load        heartmonitor.load     proxy.load
authn_core.load        http2.conf            proxy_scgi.load
authn_dbd.load         http2.load            proxy_uwsgi.load
authn_dbm.load         ident.load            proxy_wstunnel.load
authn_file.load        imagemap.load         ratelimit.load
authn_socache.load     include.load          reflector.load
authnz_fcgi.load       info.conf             remoteip.load
authnz_ldap.load       info.load             reqtimeout.conf
authz_core.load         lbmethod_bybusyness.load reqtimeout.load

```

To enable a module (let's say, **rewrite**), run the following command:

```
$ sudo a2enmod rewrite
```

Don't forget to restart the Apache 2 web server if you enable/disable Apache 2 modules. To restart the Apache 2 web server, run the following command:

```
$ sudo systemctl restart apache2
```

To list all the enabled/active Apache 2 modules, run the following command:

```
$ sudo a2query -m
```

```

rps@rps-virtual-machine:~$ sudo a2enmod rewrite
Enabling module rewrite.
To activate the new configuration, you need to run:
    systemctl restart apache2
rps@rps-virtual-machine:~$ sudo systemctl restart apache2
rps@rps-virtual-machine:~$ sudo a2query -m
access_compat (enabled by maintainer script)
autoindex (enabled by maintainer script)
env (enabled by maintainer script)
mpm_prefork (enabled by maintainer script)
php8.1 (enabled by maintainer script)
authn_core (enabled by maintainer script)
authz_user (enabled by maintainer script)
rewrite (enabled by site administrator)
authz_host (enabled by maintainer script)
mime (enabled by maintainer script)
status (enabled by maintainer script)
setenvif (enabled by maintainer script)
auth_basic (enabled by maintainer script)
reqtimeout (enabled by maintainer script)
dir (enabled by maintainer script)
alias (enabled by maintainer script)

```


Changing Apache Run User:

The default Apache run user on Debian 10 is **www-data** and the default web root directory is **/var/www/html**. So, as an ordinary user, you won't be able to create files/directories, or modify existing files/directories in the web root directory. As you're setting up a development LAMP server, this is not what you want. To solve this problem, you should change the Apache run user to your login user and change the owner and group of the webroot **/var/www/html** to your login user.

To change the Apache run user, edit **/etc/apache2/envvars** configuration file with the following command:

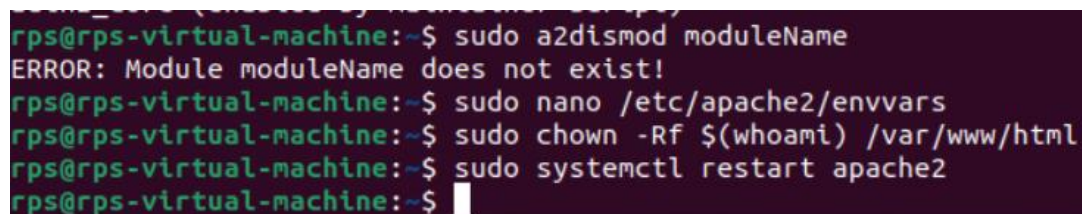
```
$ sudo nano /etc/apache2/envvars
```

Now, change the owner and group of the **/var/www/html** directory to the username of your login user with the following command:

```
$ sudo chown -Rf $(whoami):$(whoami) /var/www/html
```

Now, restart the Apache 2 web server with the following command:

```
$ sudo systemctl restart apache2
```



```
rps@rps-virtual-machine:~$ sudo a2dismod moduleName
ERROR: Module moduleName does not exist!
rps@rps-virtual-machine:~$ sudo nano /etc/apache2/envvars
rps@rps-virtual-machine:~$ sudo chown -Rf $(whoami) /var/www/html
rps@rps-virtual-machine:~$ sudo systemctl restart apache2
rps@rps-virtual-machine:~$
```

Contents of **index.php**:

```
<?php
$host = "localhost";
$user = "shovon";
$pass = "123";
$db = "app1";
```

```

try {
$conn = new PDO("mysql:host=$host;dbname=$db", $user, $pass);
$conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

echo "Connected successfully";
} catch(PDOException $e) {
echo "Connection failed: " . $e->getMessage();
}
?>

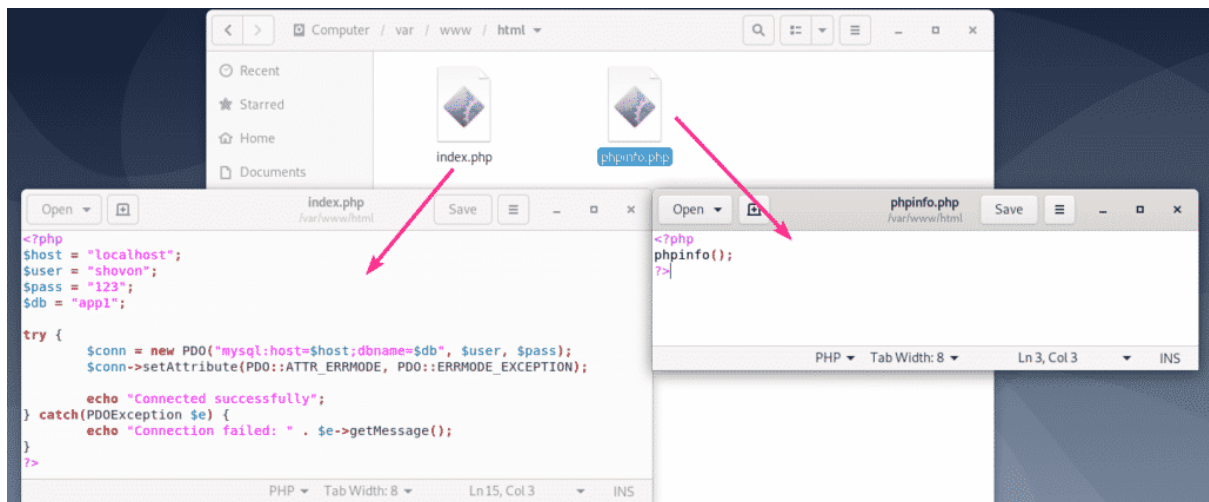
```

Contents of phpinfo.php:

```

<?php
phpinfo();
?>

```



```

rps@rps-virtual-machine:/var/www/html$ vi index.php
rps@rps-virtual-machine:/var/www/html$ vi phpinfo.php

```

Now, you should be able to access the PHP scripts from your browser as you can see in the screenshot below.

http://localhost



<http://localhost/phpinfo.php>

PHP Version 8.1.2-1ubuntu2.17	
System	Linux rps-virtual-machine 6.5.0-28-generic #29~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Thu Apr 4 14:39:20 UTC 2 x86_64
Build Date	May 1 2024 10:10:07
Build System	Linux
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/8.1/apache2
Loaded Configuration File	/etc/php/8.1/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/8.1/apache2/conf.d
Additional .ini files parsed	/etc/php/8.1/apache2/conf.d/10-mysqld.ini, /etc/php/8.1/apache2/conf.d/10-opcache.ini, /etc/php/8.1/apache2/conf.d/10-pdo.ini, /etc/php/8.1/apache2/conf.d/15-xml.ini, /etc/php/8.1/apache2/conf.d/20-calendar.ini, /etc/php/8.1/apache2/conf.d/20-ctype.ini, /etc/php/8.1/apache2/conf.d/20-curl.ini, /etc/php/8.1/apache2/conf.d/20-dom.ini, /etc/php/8.1/apache2/conf.d/20-exif.ini, /etc/php/8.1/apache2/conf.d/20-ffi.ini, /etc/php/8.1/apache2/conf.d/20-fileinfo.ini, /etc/php/8.1/apache2/conf.d/20-ftp.ini, /etc/php/8.1/apache2/conf.d/20-gd.ini, /etc/php/8.1/apache2/conf.d/20-gettext.ini, /etc/php/8.1/apache2/conf.d/20-iconv.ini, /etc/php/8.1/apache2/conf.d/20-mbstring.ini, /etc/php/8.1/apache2/conf.d/20-mysqli.ini, /etc/php/8.1/apache2/conf.d/20-pdo_mysql.ini, /etc/php/8.1/apache2/conf.d/20-phar.ini, /etc/php/8.1/apache2/conf.d/20-posix.ini, /etc/php/8.1/apache2/conf.d/20-readline.ini, /etc/php/8.1/apache2/conf.d/20-shmop.ini, /etc/php/8.1/apache2/conf.d/20-simplexml.ini, /etc/php/8.1/apache2/conf.d/20-sockets.ini, /etc/php/8.1/apache2/conf.d/20-sysvmsg.ini, /etc/php/8.1/apache2/conf.d/20-sysvsem.ini, /etc/php/8.1/apache2/conf.d/20-sysvshm.ini, /etc/php/8.1/apache2/conf.d/20-tokenizer.ini, /etc/php/8.1/apache2/conf.d/20-xmlreader.ini, /etc/php/8.1/apache2/conf.d/20-xmlwriter.ini, /etc/php/8.1/apache2/conf.d/20-xsl.ini, /etc/php/8.1/apache2/conf.d/20-zip.ini
PHP API	20210902
PHP Extension	20210902
Zend Extension	420210902
Zend Extension Build	API420210902.NTS
PHP Extension Build	API20210902.NTS

So, that's how I setup a Debian 10 LAMP server for PHP web development. Thanks for reading this article.

THANK YOU