

How to Install LAMP Stack on Debian 10 Buster Server

This tutorial is going to show you how to install Apache, MariaDB and PHP7.3 (LAMP stack) on [Debian 10 Buster](#). A software stack is a set of software tools bundled together. LAMP stands for **L**inux, **A**pache, **M**ariaDB/**M**ySQL and **P**HP, all of which are open source and free to use. It is the most common software stack that powers dynamic websites and web applications. Linux is the operating system; Apache is the web server; MariaDB/MySQL is the database server and PHP is the server-side scripting language responsible for generating dynamic web pages.

All of the four components are free and open-source. However, since MySQL is now owned by Oracle and there's a chance that Oracle turns it to a closed-source product, we will choose MariaDB instead of MySQL.

Prerequisites of Installing LAMP Stack on Debian 10 Buster

To follow this tutorial, you need a Debian 10 OS running on your local computer or on a remote server.

If you are looking for a VPS (Virtual Private Server), then you can register an account at Vultr via [this special link](#) to get \$50 free credit (for new users only). And if you need to set up LAMP stack with a domain name, I recommend buying domain names from [NameCheap](#) because the price is low and they give whois privacy protection for free.

Please note that you need to have root privilege when installing software on Debian. You can add **sudo** at the beginning of a command, or use `|su -|` command to switch to root user.

Step 1: Update Software Packages

Before we install the LAMP stack, it's a good idea to update repository and software packages. Run the following command on your Debian 10 OS.

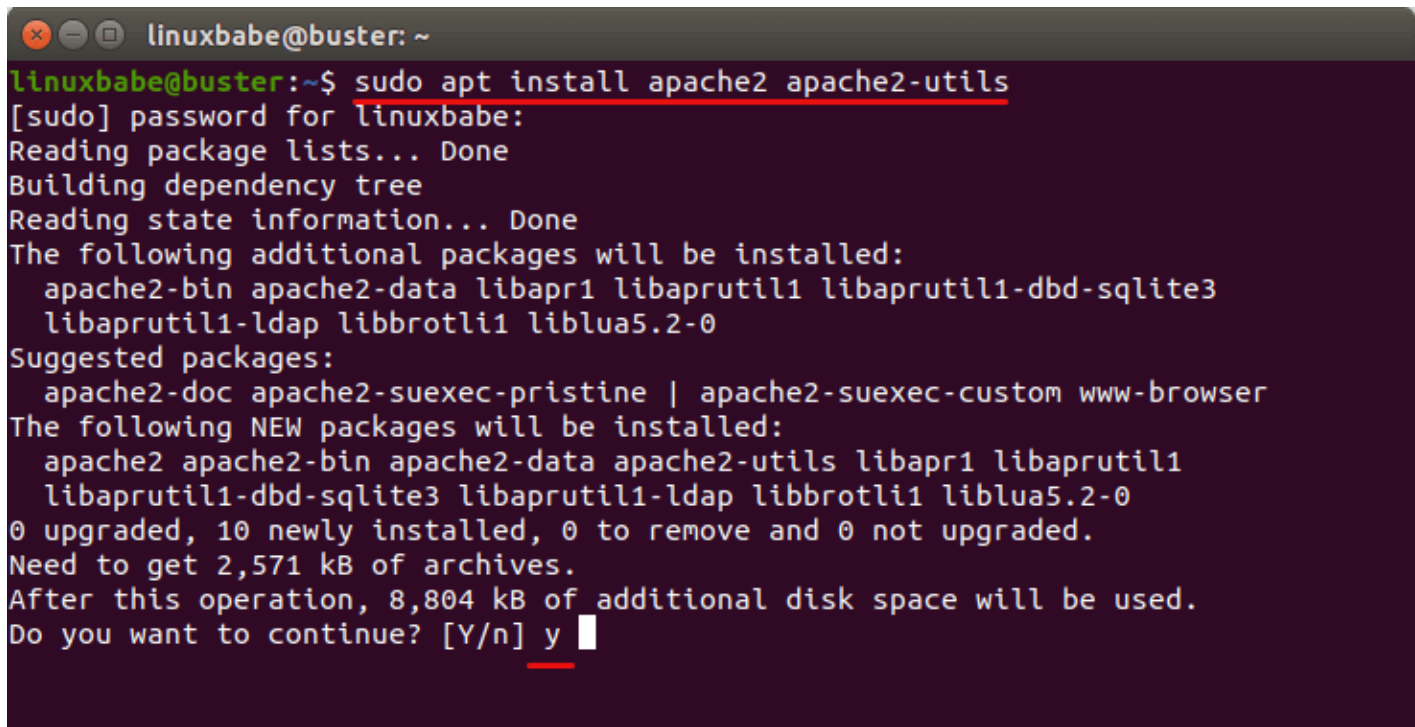
```
sudo apt update
```

```
sudo apt upgrade
```

Step 2: Install Apache Web Server on Debian 10

Enter the following command to install Apache Web server. The `apache2-utils` package will install some useful utilities, such as the Apache HTTP server benchmarking tool `ab` and the user authentication management tool `htpasswd`.

```
sudo apt install apache2 apache2-utils
```



```
linuxbabe@buster: ~  
linuxbabe@buster:~$ sudo apt install apache2 apache2-utils  
[sudo] password for linuxbabe:  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following additional packages will be installed:  
  apache2-bin apache2-data libapr1 libaprutil1 libaprutil1-dbd-sqlite3  
  libaprutil1-ldap libbrotli1 liblua5.2-0  
Suggested packages:  
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser  
The following NEW packages will be installed:  
  apache2 apache2-bin apache2-data apache2-utils libapr1 libaprutil1  
  libaprutil1-dbd-sqlite3 libaprutil1-ldap libbrotli1 liblua5.2-0  
0 upgraded, 10 newly installed, 0 to remove and 0 not upgraded.  
Need to get 2,571 kB of archives.  
After this operation, 8,804 kB of additional disk space will be used.  
Do you want to continue? [Y/n] y
```

After it's installed, Apache should be automatically started. Check its status with `systemctl`.

```
systemctl status apache2
```

Sample output:

- `apache2.service` - The Apache HTTP Server
Loaded: loaded (`/lib/systemd/system/apache2.service`; enabled; vendor preset: enabled)
Active: active (running) since Thu 2019-07-11 13:30:35 UTC; 4min 31s ago

```
Docs: https://httpd.apache.org/docs/2.4/
Process: 17962 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
Main PID: 17966 (apache2)
Tasks: 55 (limit: 545)
Memory: 4.8M
CGroup: /system.slice/apache2.service
├─17966 /usr/sbin/apache2 -k start
├─17967 /usr/sbin/apache2 -k start
└─17968 /usr/sbin/apache2 -k start
```

If it's not running, use `systemctl` to start it.

```
sudo systemctl start apache2
```

It's also a good idea to enable Apache to automatically start at boot time.

```
sudo systemctl enable apache2
```

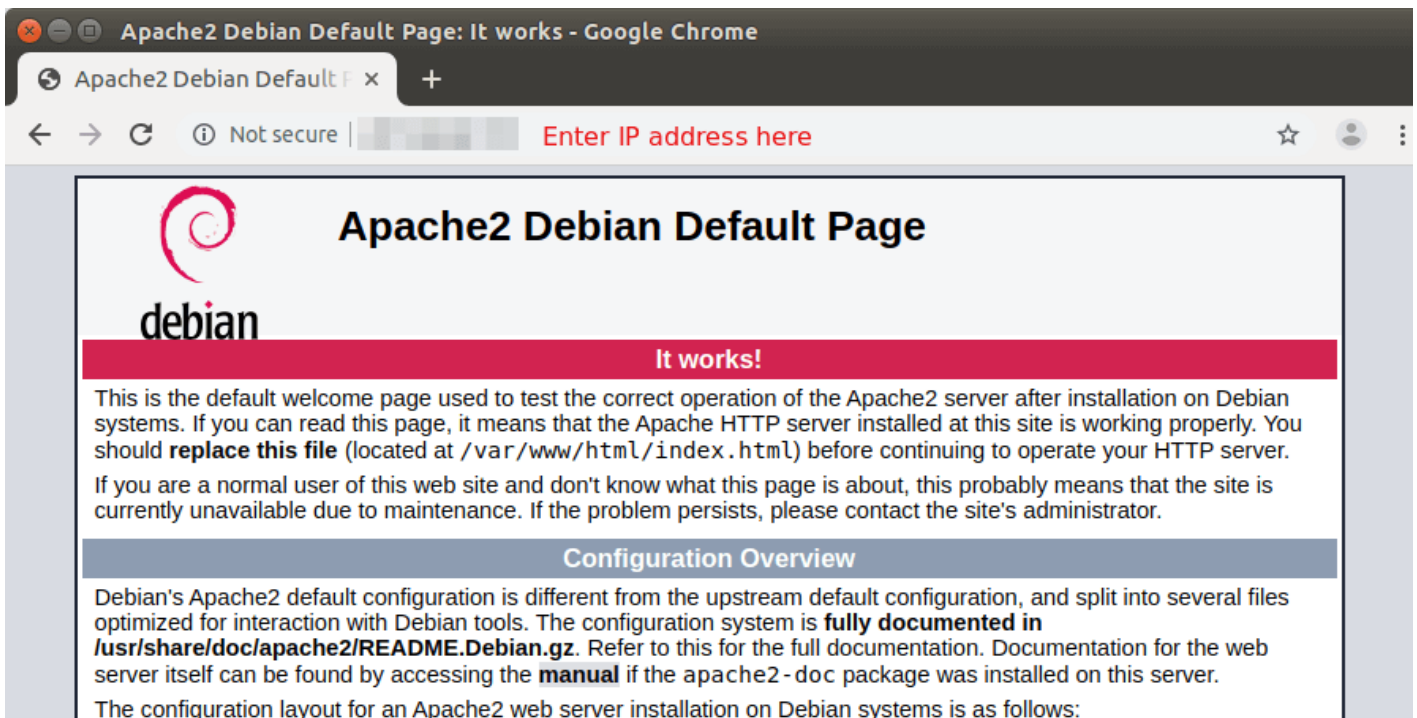
Check Apache version:

```
sudo apache2 -v
```

Output:

```
Server version: Apache/2.4.38 (Debian)
Server built: 2019-04-07T18:15:40
```

Now type in the public IP address of your Debian 10 server in the browser address bar. You should see "It works!" Web page, which means Apache Web server is running properly. If you are installing LAMP on your local Debian 10 computer, then you should type `127.0.0.1` or `localhost` in the browser address bar.



If the connection is refused or failed to complete, there might be a firewall preventing incoming requests to TCP port 80. If you are using iptables firewall, then you need to run the following command to open TCP port 80.

```
sudo iptables -I INPUT -p tcp --dport 80 -j ACCEPT
```

If you are using [UFW firewall](#), then run this command to open TCP port 80.

```
sudo ufw allow http
```

Now we need to set `www-data` (Apache user) as the owner of document root (otherwise known as web root). By default it's owned by the root user.

```
sudo chown www-data: www-data /var/www/html/ -R
```

By default, Apache uses the system hostname as its global `ServerName`. If the system hostname can't be resolved in DNS, then you will probably see the following error after running `sudo apache2ctl -t` command.

```
AH00558: apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.0.1. Set the 'ServerName' directive globally to suppress this message
```

To solve this problem, we can set a global `ServerName` in Apache. Use the Nano command-line text editor to create a new configuration file.

```
sudo nano /etc/apache2/conf-available/servername.conf
```

Add the following line in this file.

```
ServerName localhost
```

Save and close the file. To save a file in Nano text editor, press `Ctrl+O`, then press Enter to confirm. To exit, press `Ctrl+X`. Then enable this config file.

```
sudo a2enconf servername.conf
```

Reload Apache for the change to take effect.

```
sudo systemctl reload apache2
```

Now if you run the `sudo apache2ctl -t` command again, you won't see the above error message.

Step 3: Install MariaDB Database Server on Debian 10

MariaDB is a drop-in replacement for MySQL. Enter the following command to install it on Debian 10.

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

After it's installed, MariaDB server should be automatically started. Use **systemctl** to check its status.

```
systemctl status mariadb
```

Output:

```
● mariadb.service - MariaDB 10.3.15 database server
   Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2019-07-11 13:57:03 UTC; 16s ago
     Docs: man:mysqld(8)
           https://mariadb.com/kb/en/library/systemd/
  Main PID: 18566 (mysqld)
    Status: "Taking your SQL requests now..."
```

```
Tasks: 31 (limit: 545)
Memory: 73.9M
CGroup: /system.slice/mariadb.service
└─18566 /usr/sbin/mysqld
```

If it's not running, start it with this command:

```
sudo systemctl start mariadb
```


To enable MariaDB to automatically start at boot time, run

```
sudo systemctl enable mariadb
```

Now run the post installation security script.

```
sudo mysql_secure_installation
```

When it asks you to enter MariaDB root password, press Enter key as the root password isn't set yet. Then enter **y** to set the root password for MariaDB server.



```
linuxbabe@buster: ~
linuxbabe@buster:~$ 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
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): Press 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] y Enter y to set MariaDB root password
New password:
```

Next, you can just press Enter to answer all remaining questions. This will remove anonymous user, disable remote root login and remove test database. This step is a basic requirement for MariaDB database security. (Note that the letter **Y** is capitalized, which means it's the default answer.)

```
linuxbabe@buster: ~  
  
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] Press 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] Press 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] Press 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] Press 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!  
linuxbabe@buster:~$
```

By default, the MariaDB package on Debian uses `unix_socket` to authenticate user login, which basically means you can use username and password of the OS to log into MariaDB console. So you can run the following command to login without providing MariaDB root password.

```
sudo mariadb -u root
```

or

```
sudo mysql -u root
```

To exit, run

```
exit;
```

Check MariaDB server version information.

```
mariadb --version
```

Output:

```
mariadb Ver 15.1 Distrib 10.3.15-MariaDB, for debian-linux-gnu (x86_64) using readline 5.2
```

Step 4: Install PHP7.3 on Debian 10

At the the time of this writing, PHP7.3 is the latest stable version of PHP and has minor performance improvement over previous versions. Enter the following command to install PHP7.3 from Debian 10 repository.

```
sudo apt install php7.3 libapache2-mod-php7.3 php7.3-mysql php-common php7.3-cli php7.3-common php7.3-json php7.3-opcache php7.3-readline
```

Enable the Apache php7.3 module then restart Apache Web server.

```
sudo a2enmod php7.3
```

```
sudo systemctl restart apache2
```

Check PHP version information.

```
php --version
```

Output:

```
PHP 7.3.4-2 (cli) (built: Apr 13 2019 19:05:48) ( NTS )
Copyright (c) 1997-2018 The PHP Group
Zend Engine v3.3.4, Copyright (c) 1998-2018 Zend Technologies
with Zend OPcache v7.3.4-2, Copyright (c) 1999-2018, by Zend Technologies
```

To test PHP scripts with Apache server, we need to create a `info.php` file in the Web root directory with a command line text editor, such as Nano.

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

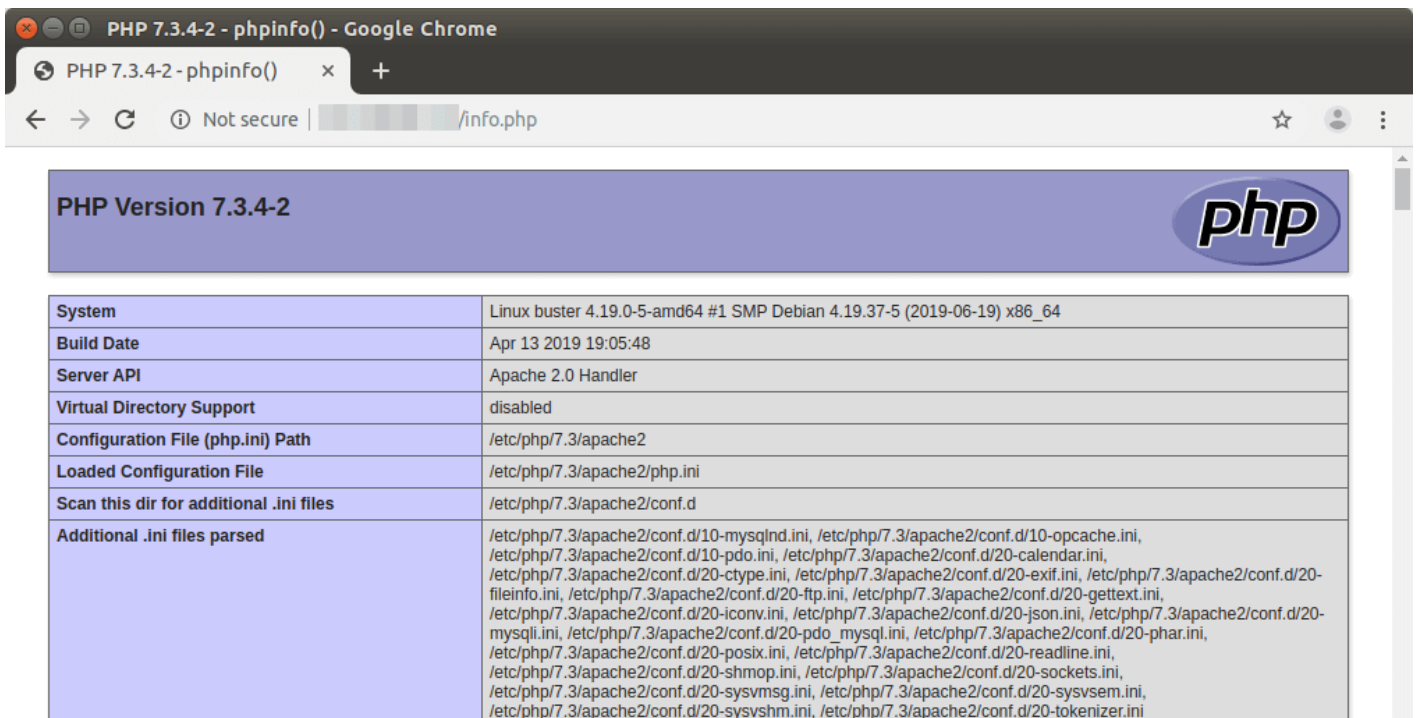
Paste the following PHP code into the file.

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

Save and close the file. (To save a file in Nano text editor, press `Ctrl+O`, then press Enter to confirm. To exit, press `Ctrl+X`.)

Now in the browser address bar, enter `server-ip-address/info.php`. Replace `server-ip-address` with your actual IP. If you follow this tutorial on your local computer, then type `127.0.0.1/info.php` or `localhost/info.php`.

You should see your server's PHP information. This means PHP scripts can run properly with Apache web server. You can find that Zend OPcache is enabled.



System	Linux buster 4.19.0-5-amd64 #1 SMP Debian 4.19.37-5 (2019-06-19) x86_64
Build Date	Apr 13 2019 19:05:48
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.3/apache2
Loaded Configuration File	/etc/php/7.3/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.3/apache2/conf.d
Additional .ini files parsed	/etc/php/7.3/apache2/conf.d/10-mysqlnd.ini, /etc/php/7.3/apache2/conf.d/10-opcache.ini, /etc/php/7.3/apache2/conf.d/10-pdo.ini, /etc/php/7.3/apache2/conf.d/20-calendar.ini, /etc/php/7.3/apache2/conf.d/20-ctype.ini, /etc/php/7.3/apache2/conf.d/20-exif.ini, /etc/php/7.3/apache2/conf.d/20-fileinfo.ini, /etc/php/7.3/apache2/conf.d/20-ftp.ini, /etc/php/7.3/apache2/conf.d/20-gettext.ini, /etc/php/7.3/apache2/conf.d/20-iconv.ini, /etc/php/7.3/apache2/conf.d/20-json.ini, /etc/php/7.3/apache2/conf.d/20-mysqli.ini, /etc/php/7.3/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.3/apache2/conf.d/20-phar.ini, /etc/php/7.3/apache2/conf.d/20-posix.ini, /etc/php/7.3/apache2/conf.d/20-readline.ini, /etc/php/7.3/apache2/conf.d/20-shmop.ini, /etc/php/7.3/apache2/conf.d/20-sockets.ini, /etc/php/7.3/apache2/conf.d/20-sysmsg.ini, /etc/php/7.3/apache2/conf.d/20-syssem.ini, /etc/php/7.3/apache2/conf.d/20-sysvshm.ini, /etc/php/7.3/apache2/conf.d/20-tokenizer.ini

How to Run PHP-FPM with Apache

There are basically two ways to run PHP code with Apache web server:

- Apache PHP module
- PHP-FPM.

In the above steps, the PHP7.3 module is used to handle PHP code, which is usually fine. But in some cases, you need to run PHP code with PHP-FPM instead. Here's how.

Disable the Apache PHP7.3 module.

```
sudo a2dismod php7.3
```

Install PHP-FPM.

```
sudo apt install php7.3-fpm
```

Enable `proxy_fcgi` and `setenvif` module.

```
sudo a2enmod proxy_fcgi setenvif
```

Enable the `/etc/apache2/conf-available/php7.3-fpm.conf` configuration file.

```
sudo a2enconf php7.3-fpm
```


Restart Apache for the changes to take effect.

```
sudo systemctl restart apache2
```

Now if you refresh the `info.php` page in your browser, you will find that Server API is changed to `FPM/FastCGI`, which means Apache web server will run PHP code with PHP-FPM.

PHP Version 7.3.4-2



System	Linux buster 4.19.0-5-amd64 #1 SMP Debian 4.19.37-5 (2019-06-19) x86_64
Build Date	Apr 13 2019 19:05:48
Server API	FPM/FastCGI 
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.3/fpm
Loaded Configuration File	/etc/php/7.3/fpm/php.ini
Scan this dir for additional .ini files	/etc/php/7.3/fpm/conf.d

Revision #1

Created 11 July 2021 19:03:27 by Admin

Updated 11 July 2021 19:03:56 by Admin