How to Install LAMP Stack on Ubuntu 20.04 Server

This tutorial is going to show you how to install LAMP stack on Ubuntu 20.04 LTS. A software stack is a set of software tools bundled together. LAMP stands for Linux, Apache, MariaDB/MySQL and PHP, 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.

Prerequisites

To follow this tutorial, you need an Ubuntu 20.04 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 DigitalOcean via this special link to get $50 free credit. (For new users only). If you are already a DigitalOcean user, then you can register an account on 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 free for life.

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 commands on your Ubuntu 20.04 OS.

```
sudo apt update


sudo apt upgrade
```

Step 2: Install Apache Web Server

Enter the following command to install Apache Web server. The `apache2-utils` package will install some useful utilities like Apache HTTP server benchmarking tool (ab).

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

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 Sat 2020-04-11 11:31:31 CST; 2s ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 53003 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 53011 (apache2)
      Tasks: 55 (limit: 19072)
     Memory: 6.4M
     CGroup: /system.slice/apache2.service
             ├─53011 /usr/sbin/apache2 -k start
             ├─53012 /usr/sbin/apache2 -k start
             └─53013 /usr/sbin/apache2 -k start
```

Hint: If the above command doesn't quit immediately, you can press Q key to gain back control of the terminal.

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 system boot time.

```
sudo systemctl enable apache2
```

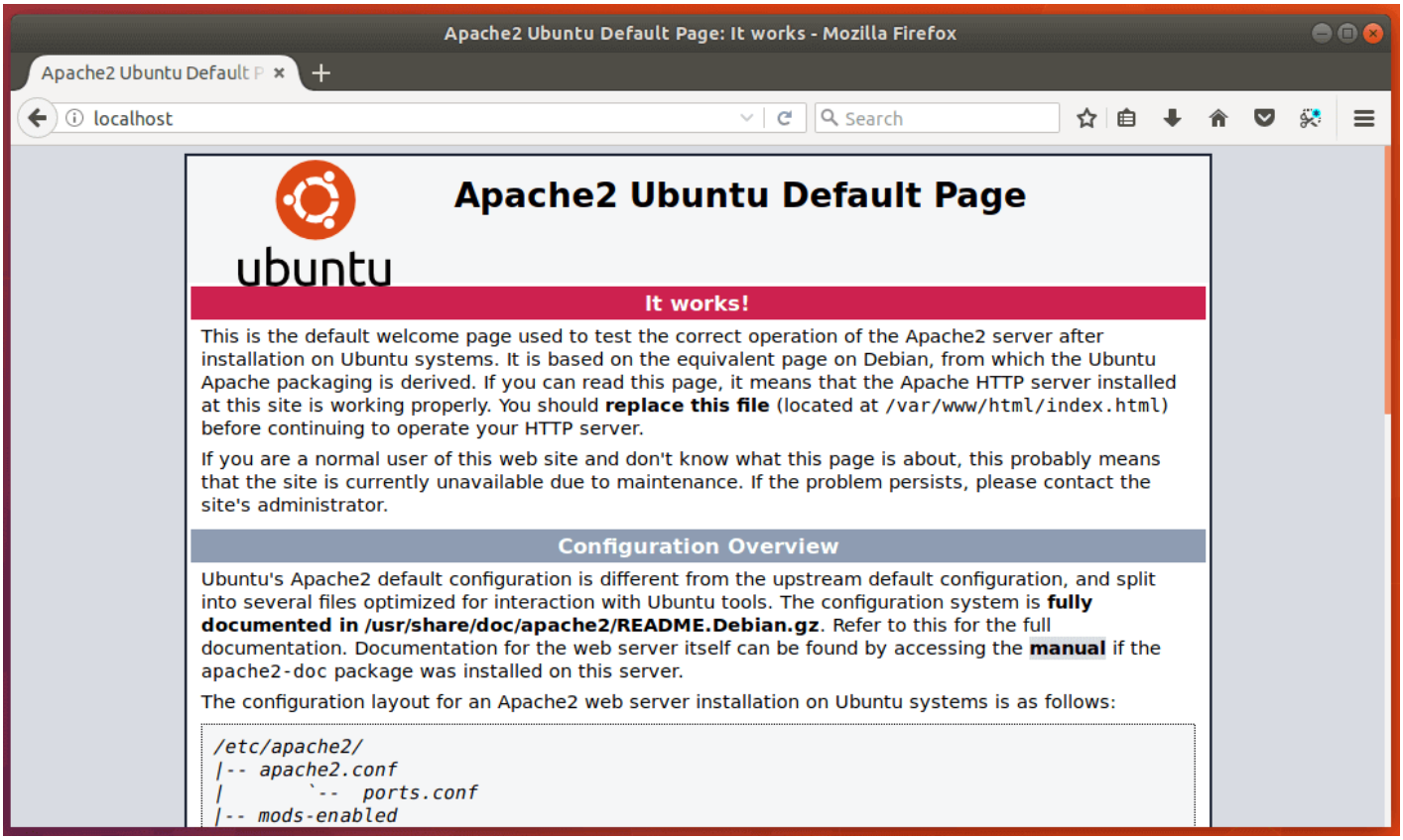
Check Apache version:

```
apache2 -v
```

Output:

```
Server version: Apache/2.4.41 (Ubuntu)
Server built:   2020-03-05T18:51:00
```

Now type in the public IP address of your Ubuntu 20.04 server in the browser address bar. You should see the “It works!” Web page, which means Apache Web server is running properly. If you are installing LAMP on your local Ubuntu 20.04 computer, then 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

MariaDB is a drop-in replacement for MySQL. It is developed by former members of MySQL team who are concerned that Oracle might turn MySQL into a closed-source product. Enter the following command to install MariaDB on Ubuntu 20.04.

```
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 10.3.22 database server
     Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; vendor preset:
enabled)
     Active: active (running) since Fri 2020-04-10 14:19:16 UTC; 18s ago
       Docs: man:mysqld(8)
             https://mariadb.com/kb/en/library/systemd/
   Main PID: 9161 (mysqld)
     Status: "Taking your SQL requests now..."
      Tasks: 31 (limit: 9451)
     Memory: 64.7M
     CGroup: /system.slice/mariadb.service
             └─9161 /usr/sbin/mysqld
```

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

```
sudo systemctl start mariadb
```

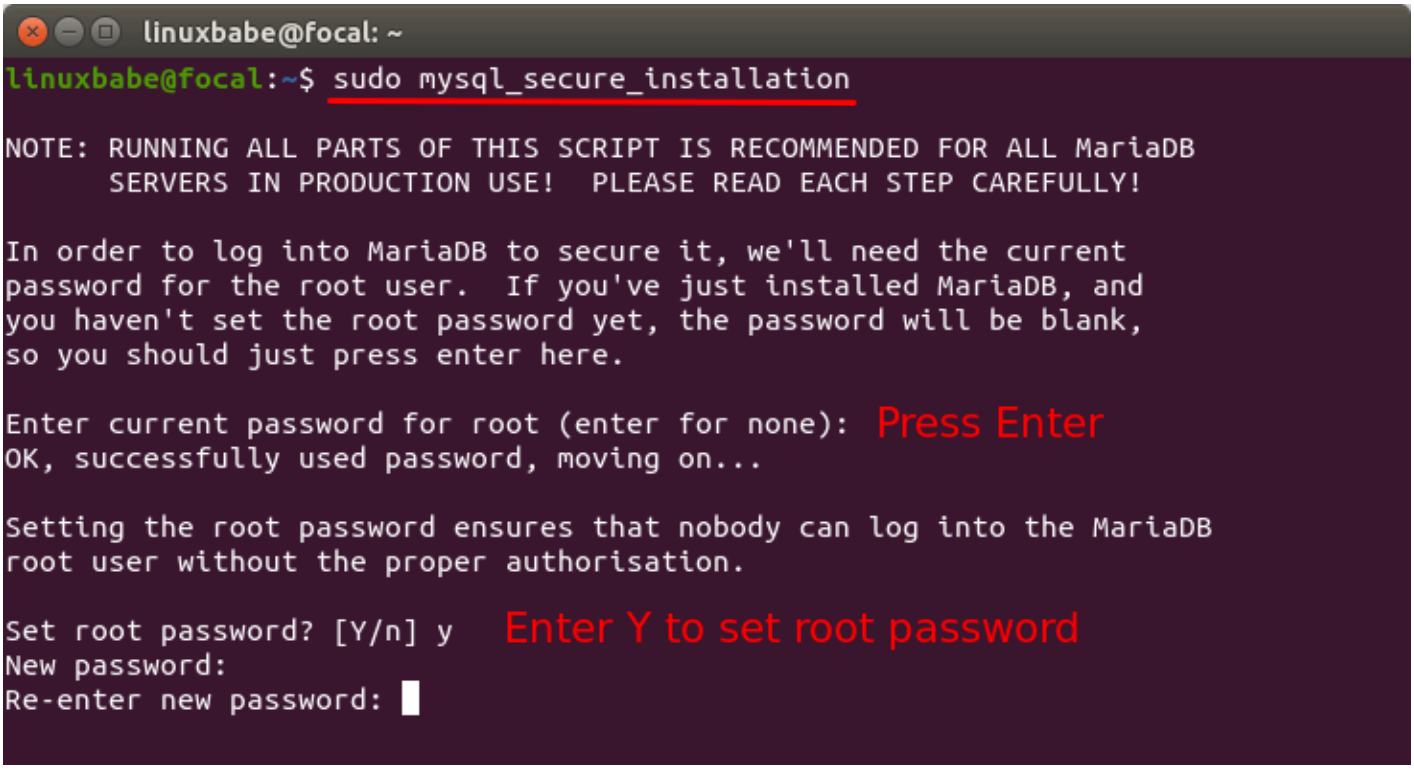
To enable MariaDB to automatically start at boot time, run

```
sudo systemctl enable mariadb
```

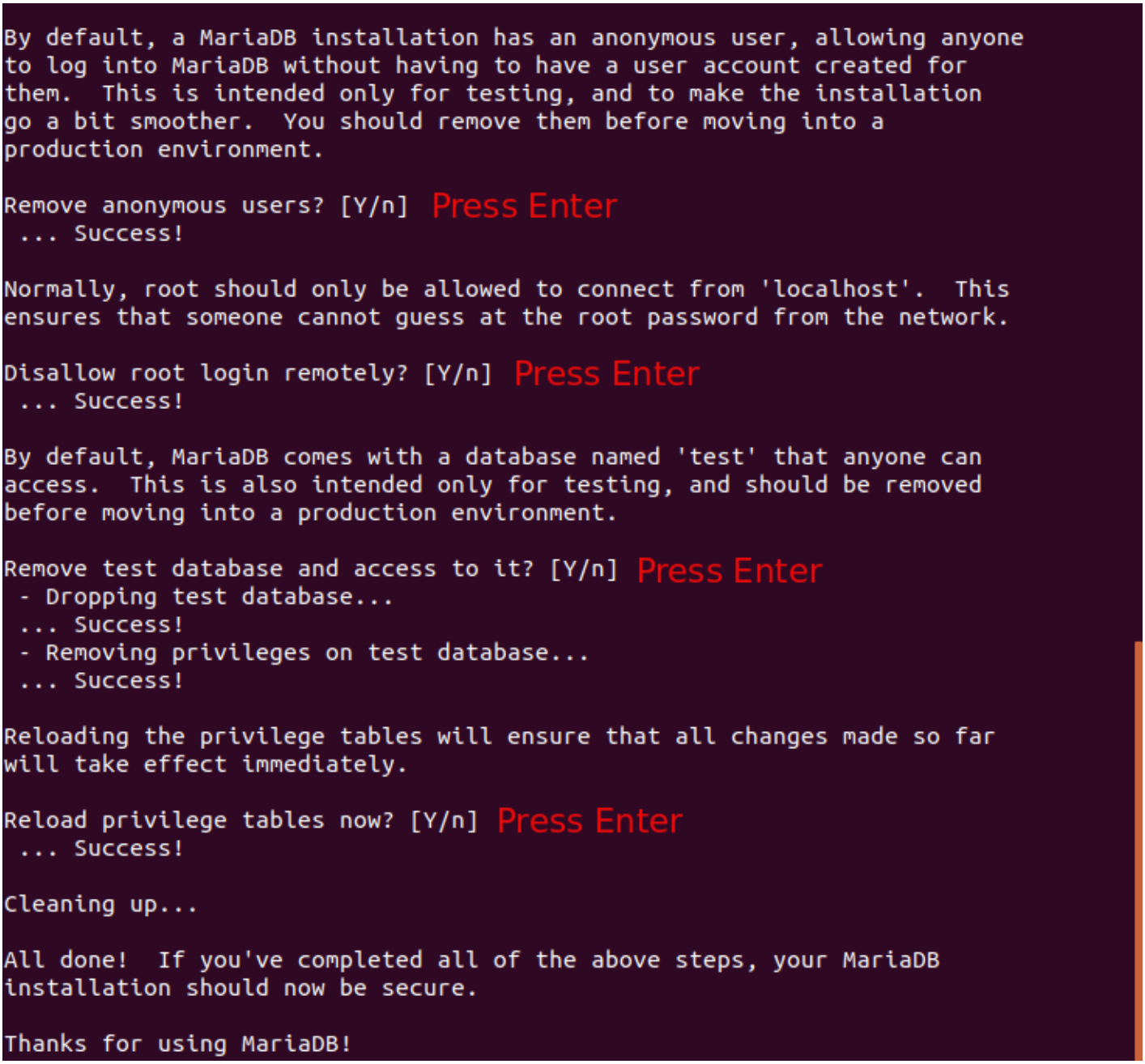
Now run the post-installation security script.

```
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.



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



By default, the MariaDB package on Ubuntu 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
```

To exit, run

```
exit;
```

Check MariaDB server version information.

```
mariadb --version
```

As you can see, we have installed MariaDB 10.3.22.

```
mariadb  Ver 15.1 Distrib 10.3.22-MariaDB, for debian-linux-gnu (x86_64) using readline 5.2
```

Step 4: Install PHP7.4

At the the time of this writing, PHP7.4 is the latest stable version of PHP and has a minor performance edge over PHP7.3. Enter the following command to install PHP7.4 and some common PHP modules.

```
sudo apt install php7.4 libapache2-mod-php7.4 php7.4-mysql php-common php7.4-cli php7.4-common php7.4-json php7.4-opcache php7.4-readline
```

Enable the Apache php7.4 module then restart Apache Web server.

```
sudo a2enmod php7.4


sudo systemctl restart apache2
```

Check PHP version information.

```
php --version
```

Output:

```
PHP 7.4.3 (cli) (built: Mar 26 2020 20:24:23) ( NTS )
Copyright (c) The PHP Group
Zend Engine v3.4.0, Copyright (c) Zend Technologies
    with Zend OPcache v7.4.3, Copyright (c), by Zend Technologies
```

To test PHP scripts with Apache server, we need to create a `info.php` file in the document root directory.

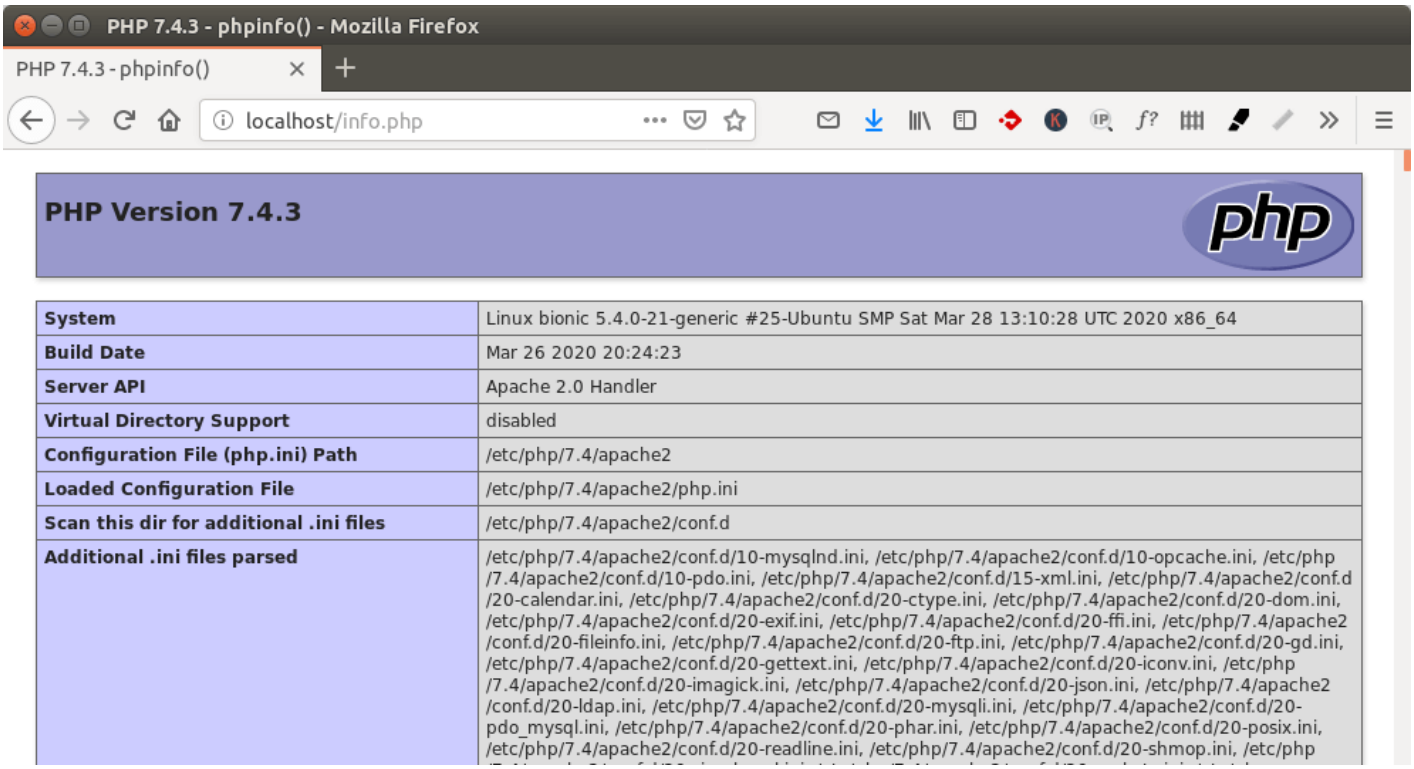
```
sudo nano /var/www/html/info.php
```

Paste the following PHP code into the file.

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

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.



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 Apache PHP7.4 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 PHP7.4 module.

```
sudo a2dismod php7.4
```

Install PHP-FPM.

```
sudo apt install php7.4-fpm
```

Enable `proxy_fcgi` and `setenvif` module.

```
sudo a2enmod proxy_fcgi setenvif
```

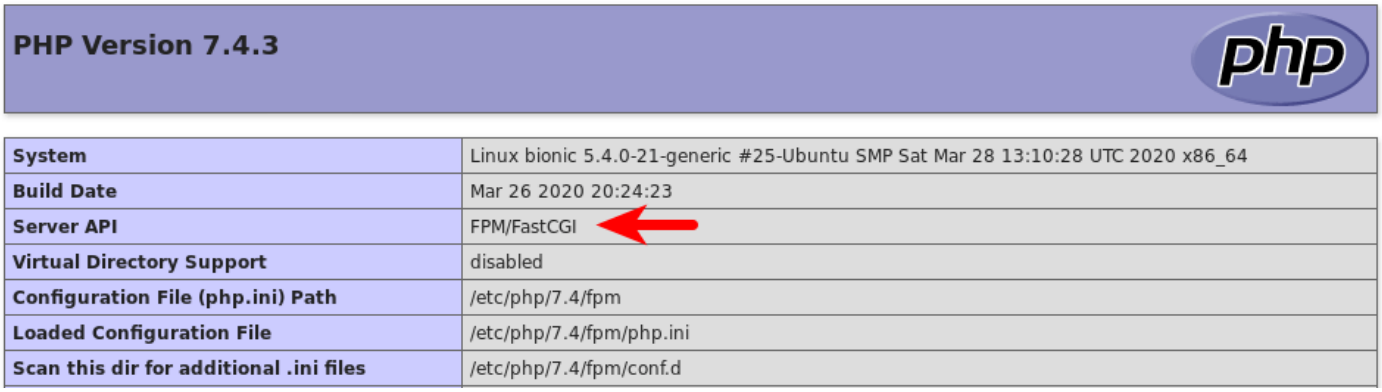
Enable the `/etc/apache2/conf-available/php7.4-fpm.conf` configuration file.

```
sudo a2enconf php7.4-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 from `Apache 2.0 Handler` to `FPM/FastCGI`, which means Apache web server will pass PHP requests to PHP-FPM.



Congrats! You have successfully installed LAMP stack (Apache, MariaDB and PHP7.4) on Ubuntu 20.04. For your server's security, you should delete `info.php` file now to prevent prying eyes.

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

Revision #1
Created 11 July 2021 19:02:09 by Admin
Updated 11 July 2021 19:02:37 by Admin