Apache

- How to Install LAMP Stack on Ubuntu 20.04 Server
- How to Install LAMP Stack on CentOS 8/RHEL 8
- How to Install LAMP Stack on Debian 10 Buster Server

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
```

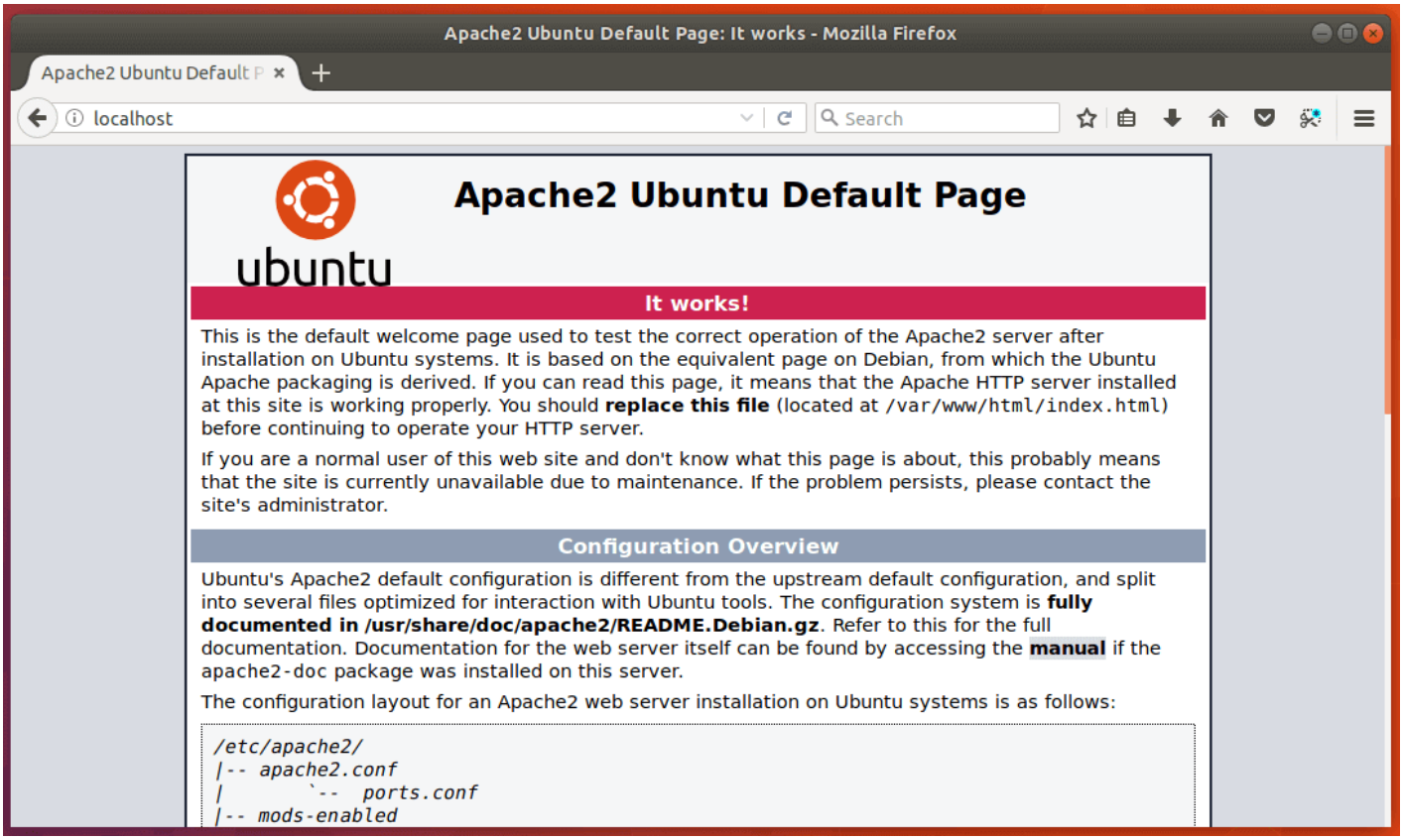
Check Apache version:

```
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
```

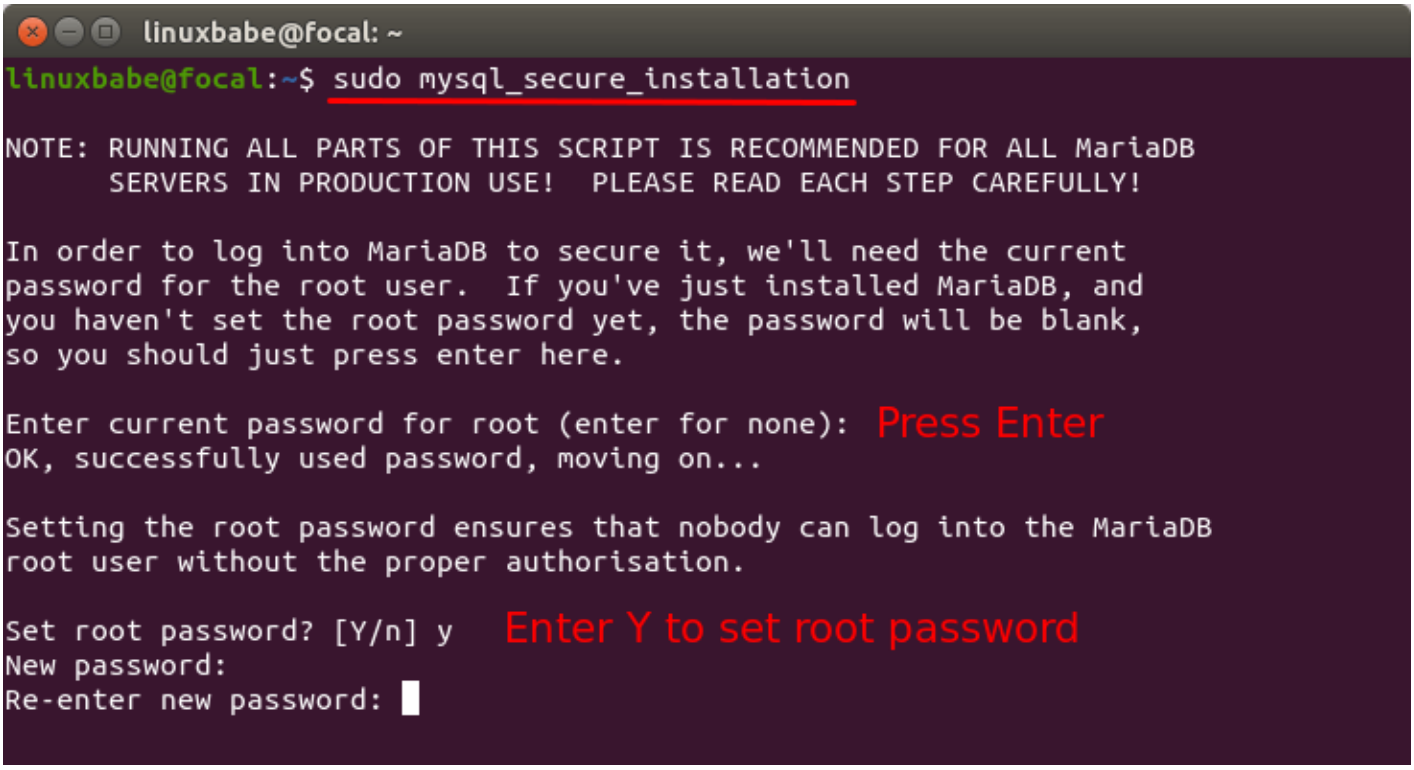
To enable MariaDB to automatically start at boot time, run

```
sudo systemctl enable mariadb
```

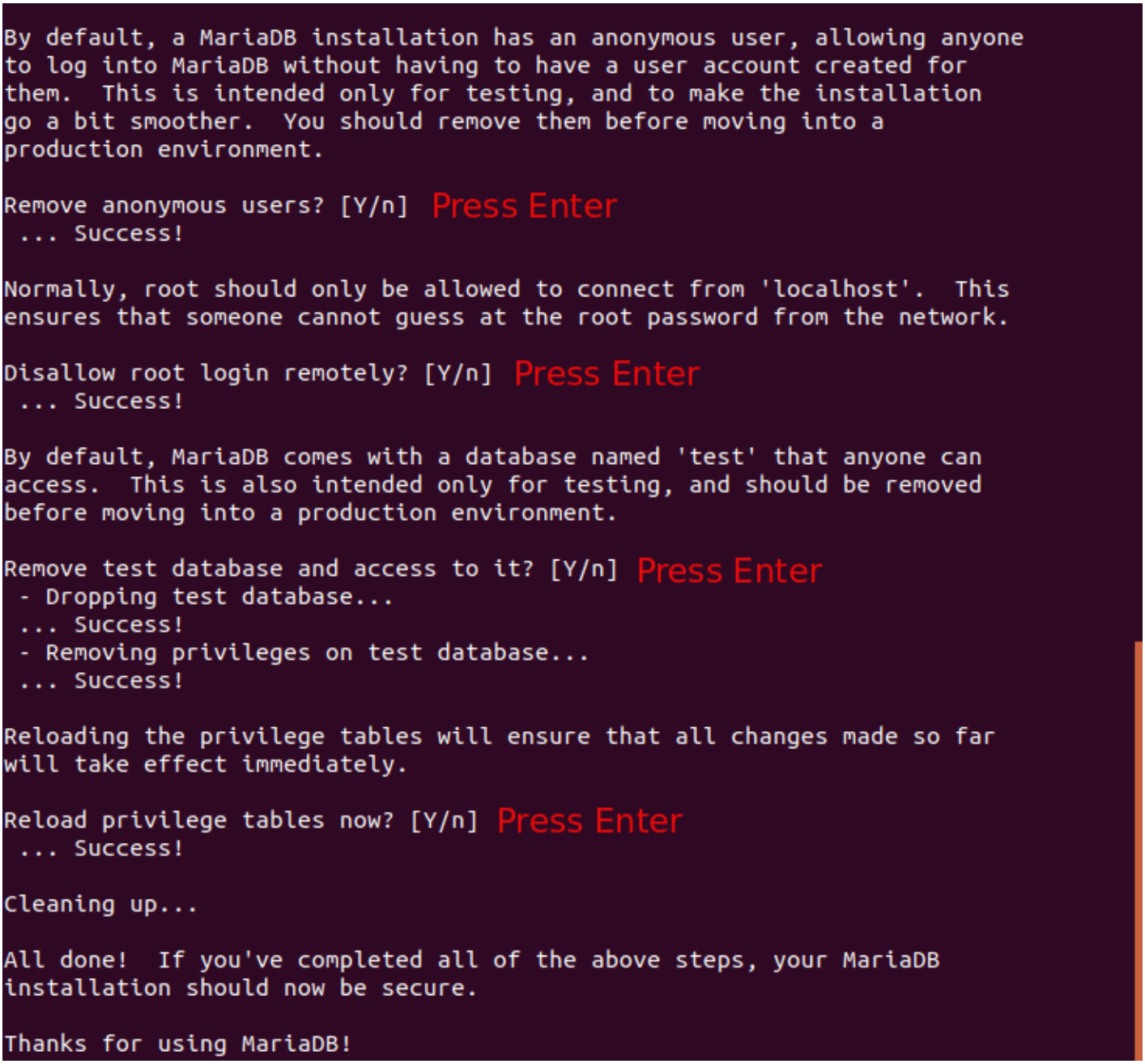
Now run the post-installation security script.

```
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.

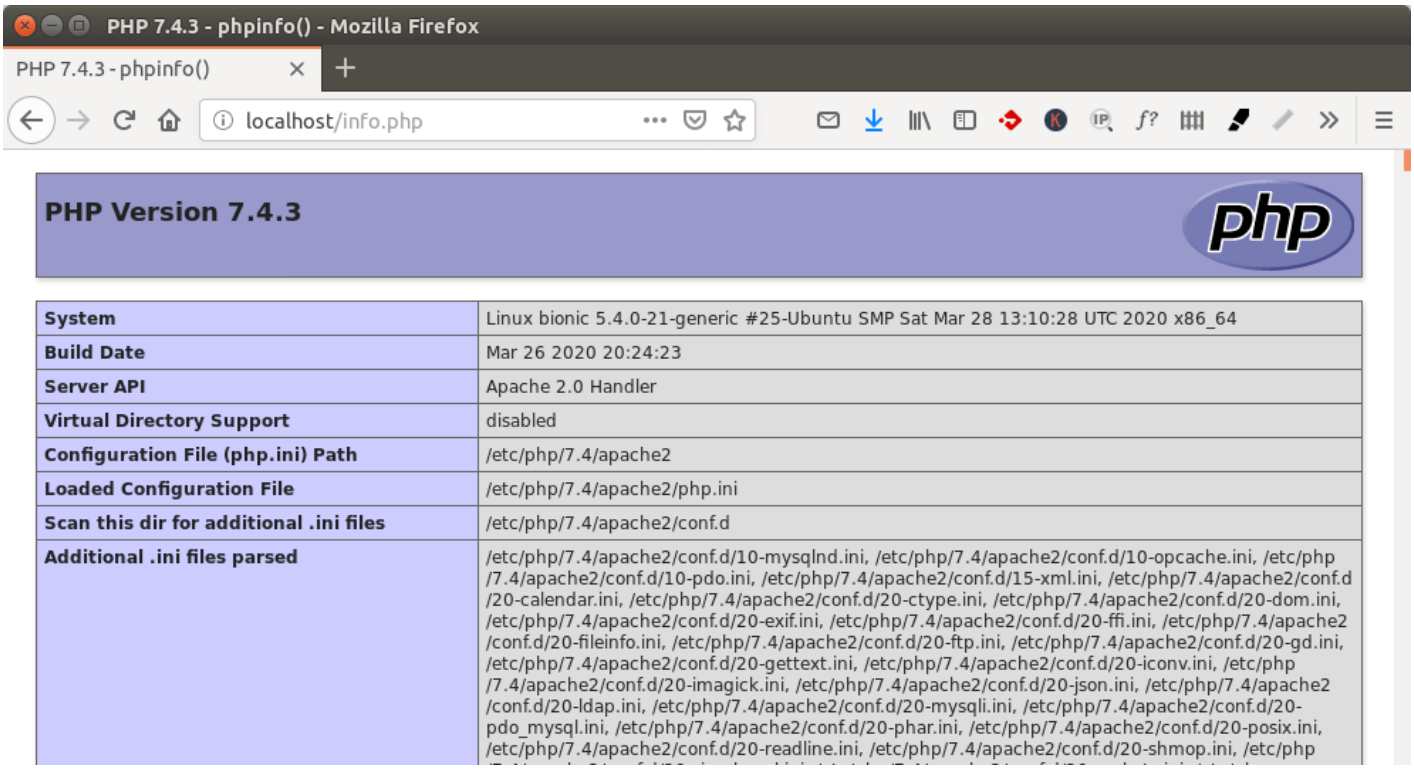
```
sudo nano /var/www/html/info.php
```

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
```

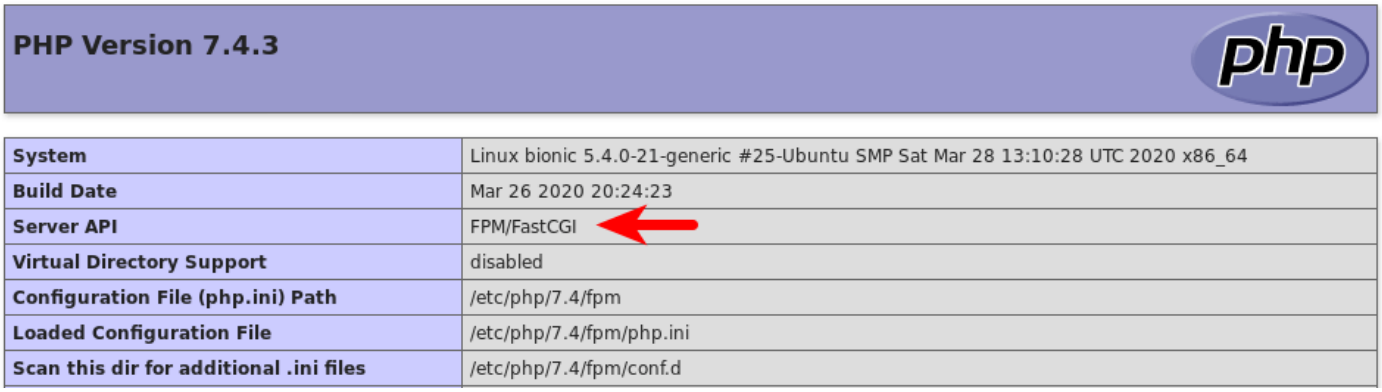
Enable the `/etc/apache2/conf-available/php7.4-fpm.conf` configuration file.

```
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
```

How to Install LAMP Stack on CentOS 8/RHEL 8

This tutorial is going to show you how to install LAMP stack on CentOS 8 and RHEL 8.

What’s LAMP Stack?

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. 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

You can download and install RHEL 8 by following the tutorial below.

- How to Download and Install RHEL 8 for Free

If you are looking for a VPS (Virtual Private Server), then you can register an account at Vultr via my referral link to get $50 free credit for use over 30 days.

This tutorial uses root account to manage administration tasks. To switch to root, run the following command and enter root password.

```
su -
```

Step 1: Update Software Packages

Before we install the LAMP stack, it’s a good idea to run the following command to update repository and software packages.

```
dnf update
```

Step 2: Install Apache Web Server on CentOS 8/RHEL 8

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

```
dnf install httpd httpd-tools
```

After it’s installed, we can start Apache with this command:

```
systemctl start httpd
```

Enable Apache to auto start at system boot time by running the following command.

```
systemctl enable httpd
```

Now check its status.

```
systemctl status httpd
```

Output:

```
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
   Active: active (running) since Sat 2019-10-12 06:43:15 UTC; 14s ago
     Docs: man:httpd.service(8)
 Main PID: 14515 (httpd)
   Status: "Running, listening on: port 80"
    Tasks: 213 (limit: 5092)
   Memory: 24.8M
   CGroup: /system.slice/httpd.service
           ├─14515 /usr/sbin/httpd -DFOREGROUND
           ├─14516 /usr/sbin/httpd -DFOREGROUND
           ├─14517 /usr/sbin/httpd -DFOREGROUND
           ├─14518 /usr/sbin/httpd -DFOREGROUND
```

```
           └─14519 /usr/sbin/httpd -DFOREGROUND
```

"**Enabled**" indicates that auto start at boot time is enabled and we can see that Apache is running.

Hint: If the above command doesn't immediately quit after running. You need to press "**q**" to make it quit.

Check Apache version.

```
httpd -v
```

Output:

```
Server version: Apache/2.4.37 (centos)
Server built: Oct 7 2019 21:42:02
```

To test if Apache web server is running properly, we can create an `index.html` file under the default document root (`/var/www/html/`) with the following command.

```
echo "Welcome to this site!" > /var/www/html/index.html
```

If you are installing LAMP on your local CentOS 8/RHEL 8 computer, then type `127.0.0.1` or `localhost` in the browser address bar. You should see the welcome message, which means Apache Web server is running properly.

By default, CentOS 8/RHEL 8 forbids public access to port 80. To allow other computers to access the web page, we need to open port 80 in firewalld, the dynamic firewall manager on RHEL/CentOS. Run the following command to open port 80.

```
firewall-cmd --permanent --zone=public --add-service=http
```

If you want to enable HTTPS on Apache later, then you also need to open port 443.

```
firewall-cmd --permanent --zone=public --add-service=https
```

The `--permanent` option will make this firewall rule persistent across system reboots. Next, reload

the firewall daemon for the change to take effect.

```
systemctl reload firewalld
```

Now the Apache web page is accessible publicly.

We need to make user `apache` as the owner of web directory. By default it's owned by the root user.

```
chown apache:apache /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 apachectl configtest` 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. Install the Nano command-line text editor and use it to create a new configuration file.

```
sudo dnf install nano


sudo nano /etc/httpd/conf.d/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`. Reload Apache for the change to take effect.

```
sudo systemctl reload httpd
```

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

Step 3: Install MariaDB Database Server on CentOS 8/RHEL 8

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 CentOS 8/RHEL 8.

```
dnf install mariadb-server mariadb -y
```

After it's installed, we need to start it.

```
systemctl start mariadb
```

Enable auto start at system boot time.

```
systemctl enable mariadb
```

Check status:

```
systemctl status mariadb
```

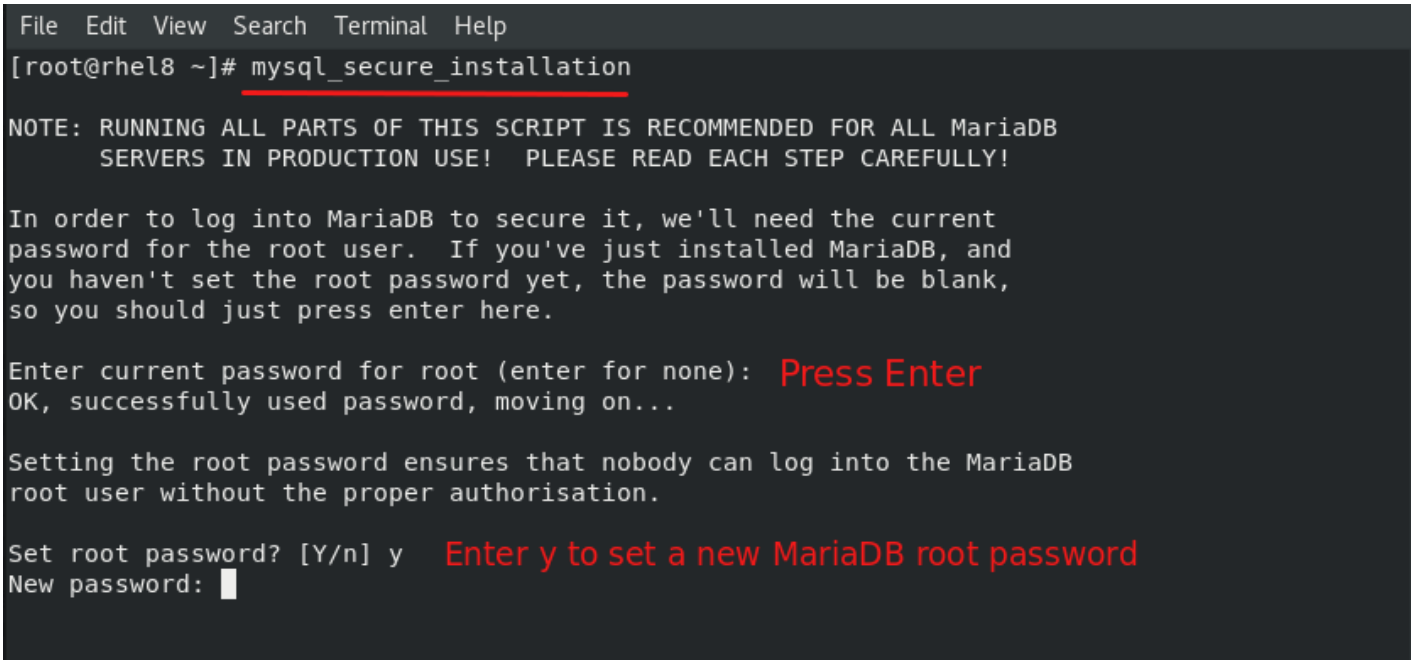
output:

```
● mariadb.service - MariaDB 10.3 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; vendor preset: disabled)
   Active: active (running) since Sat 2019-10-12 09:02:53 UTC; 33s ago
     Docs: man:mysqld(8)
           https://mariadb.com/kb/en/library/systemd/
 Main PID: 18608 (mysqld)
   Status: "Taking your SQL requests now..."
    Tasks: 30 (limit: 5092)
   Memory: 77.0M
   CGroup: /system.slice/mariadb.service
           └─18608 /usr/libexec/mysqld --basedir=/usr
```

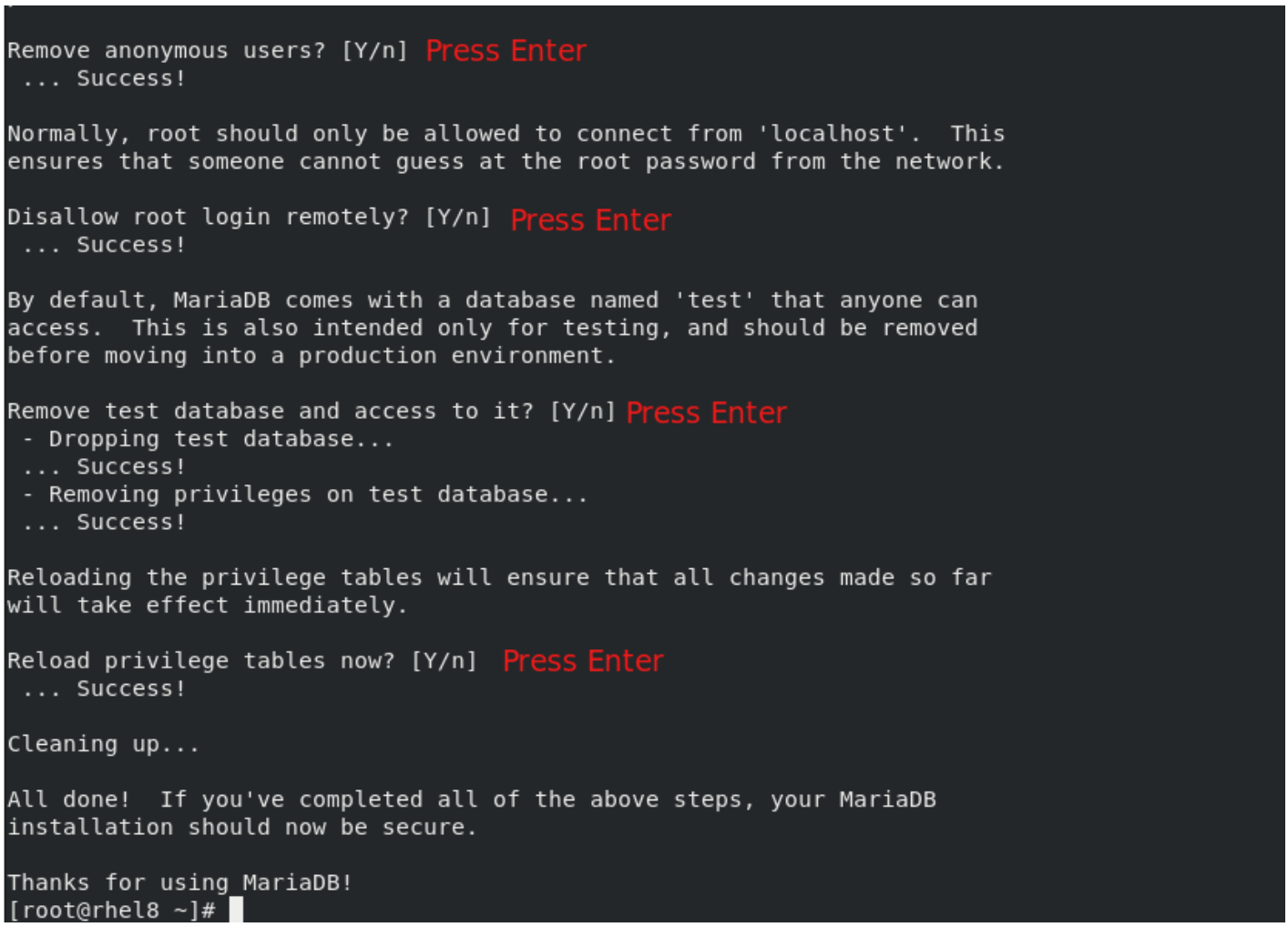
"**Enabled**" indicates that auto start at boot time is enabled and we can see that MariaDB server is running. Now we need to run the security script.

```
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.



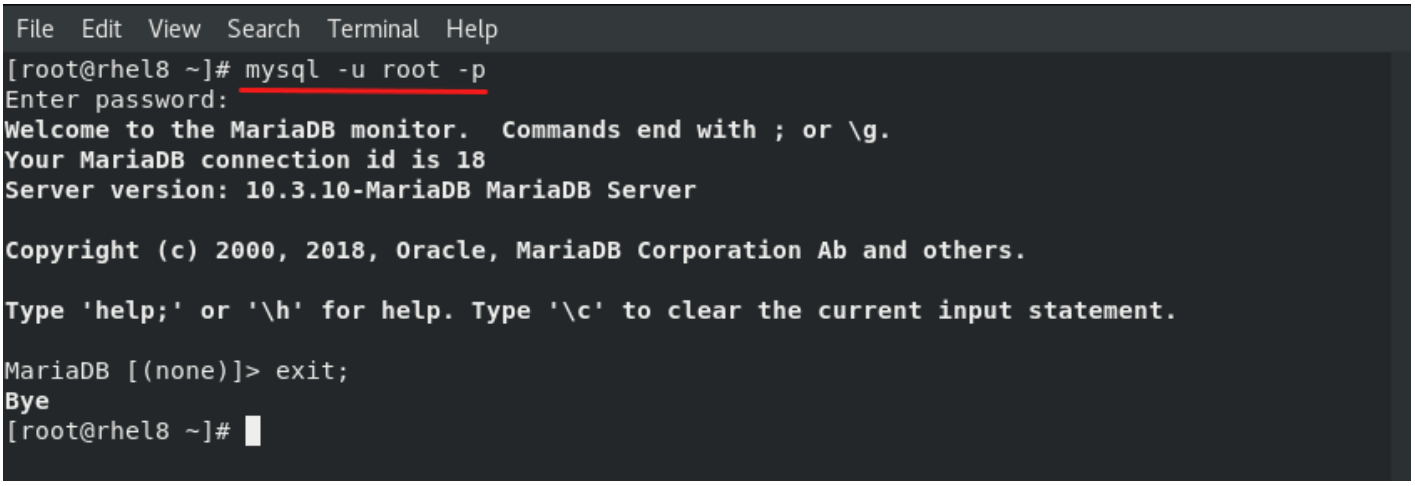
Next, you can press Enter to answer all remaining questions, which will remove anonymous user, disable remote root login and remove test database. This step is a basic requirement for MariaDB database security. (Note that the letter `Y` is capitalized, which means it's the default answer.)



Now you can run the following command and enter MariaDB root password to log into MariaDB

shell.

```
mysql -u root -p
```



To exit, run

```
exit;
```

Step 4: Install PHP on CentOS 8/RHEL 8

Install PHP and some common modules using the following command.

```
dnf install php php-fpm php-mysqlnd php-opcache php-gd php-xml php-mbstring -y
```

Apache web server on CentOS 8/RHEL 8 by default uses PHP-FPM instead of mod_php to run PHP code, so in the above command we also installed `php-fpm`. After it's installed, we need to start it.

```
systemctl start php-fpm
```

Enable auto start at system boot time.

```
systemctl enable php-fpm
```

Check status:

```
systemctl status php-fpm
```

output:

```
● php-fpm.service - The PHP FastCGI Process Manager
   Loaded: loaded (/usr/lib/systemd/system/php-fpm.service; enabled; vendor preset: disabled)
   Active: active (running) since Sat 2019-10-12 09:54:37 UTC; 3s ago
 Main PID: 19755 (php-fpm)
   Status: "Ready to handle connections"
    Tasks: 6 (limit: 5092)
   Memory: 24.5M
   CGroup: /system.slice/php-fpm.service
           ├─19755 php-fpm: master process (/etc/php-fpm.conf)
           ├─19757 php-fpm: pool www
           ├─19758 php-fpm: pool www
           ├─19759 php-fpm: pool www
           ├─19760 php-fpm: pool www
           └─19761 php-fpm: pool www
```

“**Enabled**” indicates that auto start at boot time is enabled and we can see that PHP-FPM is running. The `php-fpm` package installs a `php.conf` file in `/etc/httpd/conf.d/` directory, so we need to restart Apache web server, in order to run PHP code.

```
systemctl restart httpd
```

We also need to run the following command to tell SELinux to allow Apache to execute PHP code via PHP-FPM.

```
setsebool -P httpd_execmem 1
```

Step 5: Test PHP

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

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

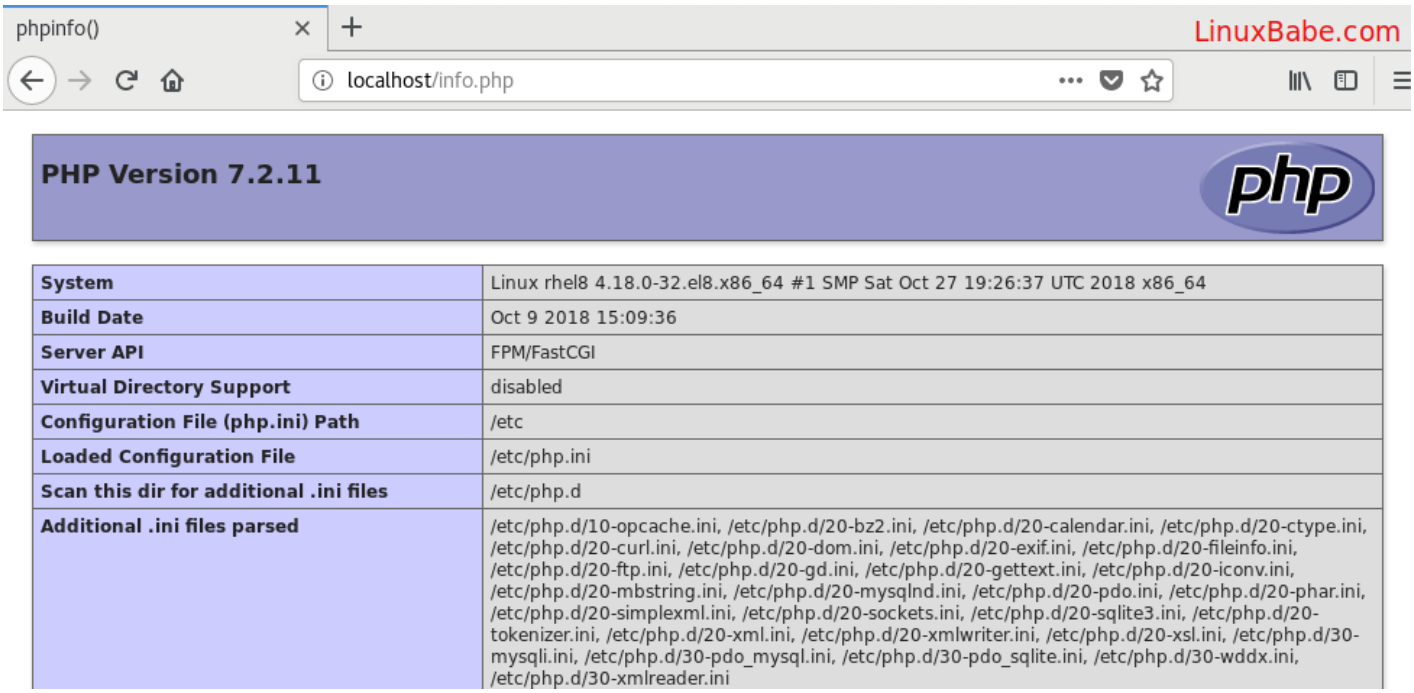
Paste the following PHP code into the file.

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

Save and close the file. If you installed LAMP stack on a local CentOS 8/RHEL 8 server, type in `127.0.0.1/info.php` or `localhost/info.php` in the browser address bar. You should see your

server's PHP information. This means PHP scripts can run properly with Apache web server.

If RHEL 8/CentOS is running on a remote server, then enter `server-ip-address/info.php` in browser address bar. Replace `sever-ip-address` with your actual IP address.



System	Linux rhel8 4.18.0-32.el8.x86_64 #1 SMP Sat Oct 27 19:26:37 UTC 2018 x86_64
Build Date	Oct 9 2018 15:09:36
Server API	FPM/FastCGI
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-dom.ini, /etc/php.d/20-exif.ini, /etc/php.d/20-fileinfo.ini, /etc/php.d/20-ftp.ini, /etc/php.d/20-gd.ini, /etc/php.d/20-gettext.ini, /etc/php.d/20-iconv.ini, /etc/php.d/20-mbstring.ini, /etc/php.d/20-mysqlnd.ini, /etc/php.d/20-pdo.ini, /etc/php.d/20-phar.ini, /etc/php.d/20-simplexml.ini, /etc/php.d/20-sockets.ini, /etc/php.d/20-sqlite3.ini, /etc/php.d/20-tokenizer.ini, /etc/php.d/20-xml.ini, /etc/php.d/20-xmlwriter.ini, /etc/php.d/20-xsl.ini, /etc/php.d/30-mysqli.ini, /etc/php.d/30-pdo_mysql.ini, /etc/php.d/30-pdo_sqlite.ini, /etc/php.d/30-wddx.ini, /etc/php.d/30-xmlreader.ini

If the browser fails to display the PHP info but prompt you to download the **info.php** file, simply restart Apache and PHP-FPM.

```
sudo systemctl restart httpd php-fpm
```

Then you should be able to see the PHP info in the web browser.

Apache Automatic Restart

If for any reason your Apache process is killed, you need to run the following command to restart it.

```
sudo systemctl restart httpd
```

Instead of manually typing this command, we can make Apache automatically restart by editing the `httpd.service` systemd service unit. To override the default systemd service configuration, we create a separate directory.

```
sudo mkdir -p /etc/systemd/system/httpd.service.d/
```

Then create a file under this directory.

```
sudo nano /etc/systemd/system/httpd.service.d/restart.conf
```

Add the following lines in the file, which will make Apache automatically restart 5 seconds after a failure is detected.

```
[Service]
Restart=always
RestartSec=5s
```

Save and close the file. Then reload systemd.

```
sudo systemctl daemon-reload
```

To check if this would work, kill Apache with:

```
sudo pkill httpd
```

Then check Apache status. You will find Apache automatically restarted.

```
systemctl status httpd
```

Allow Apache to Make Outgoing Network Connections

By default, SELinux forbids Apache to make outgoing network connections. If Apache needs to make requests to an outside network service, then run the following command to allow this action.

```
setsebool -P httpd_can_network_connect on
```

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.

```
```

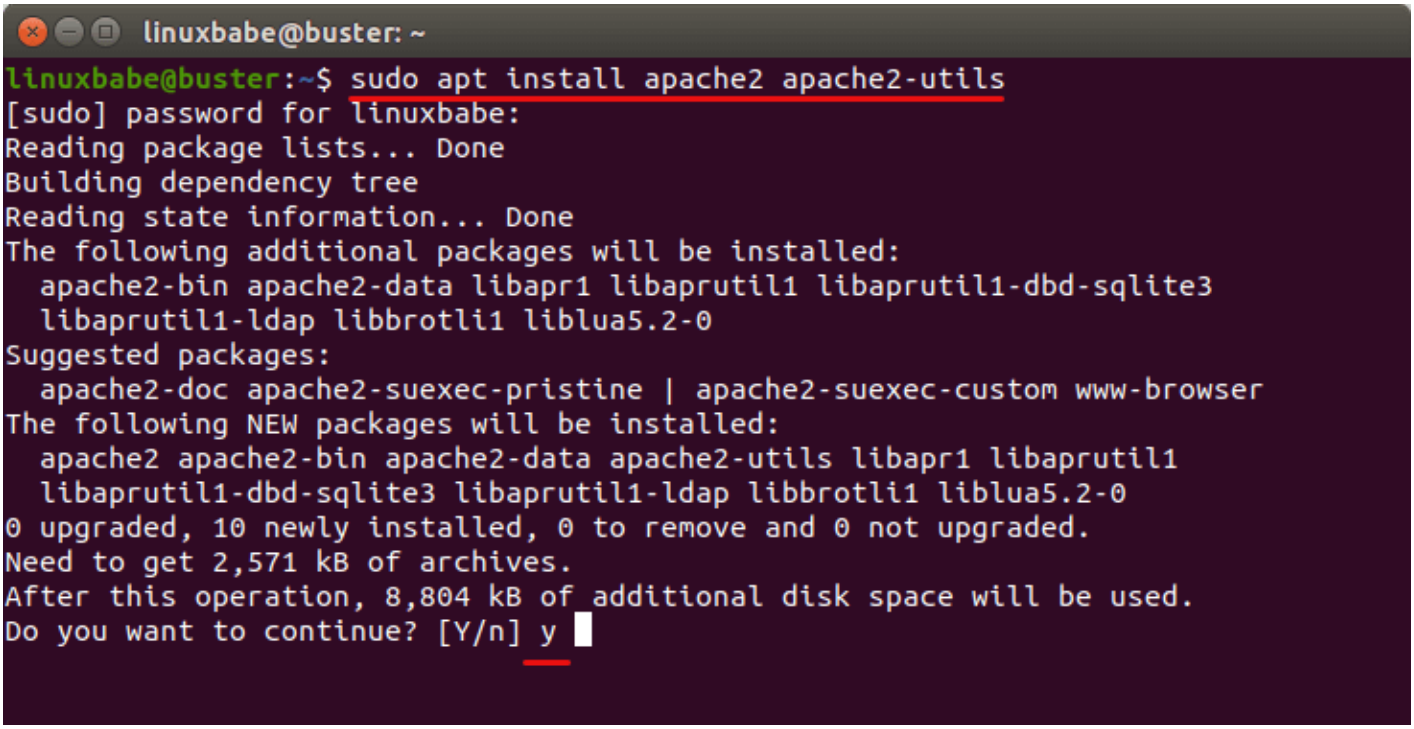
```
sudo apt update


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
```



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
```

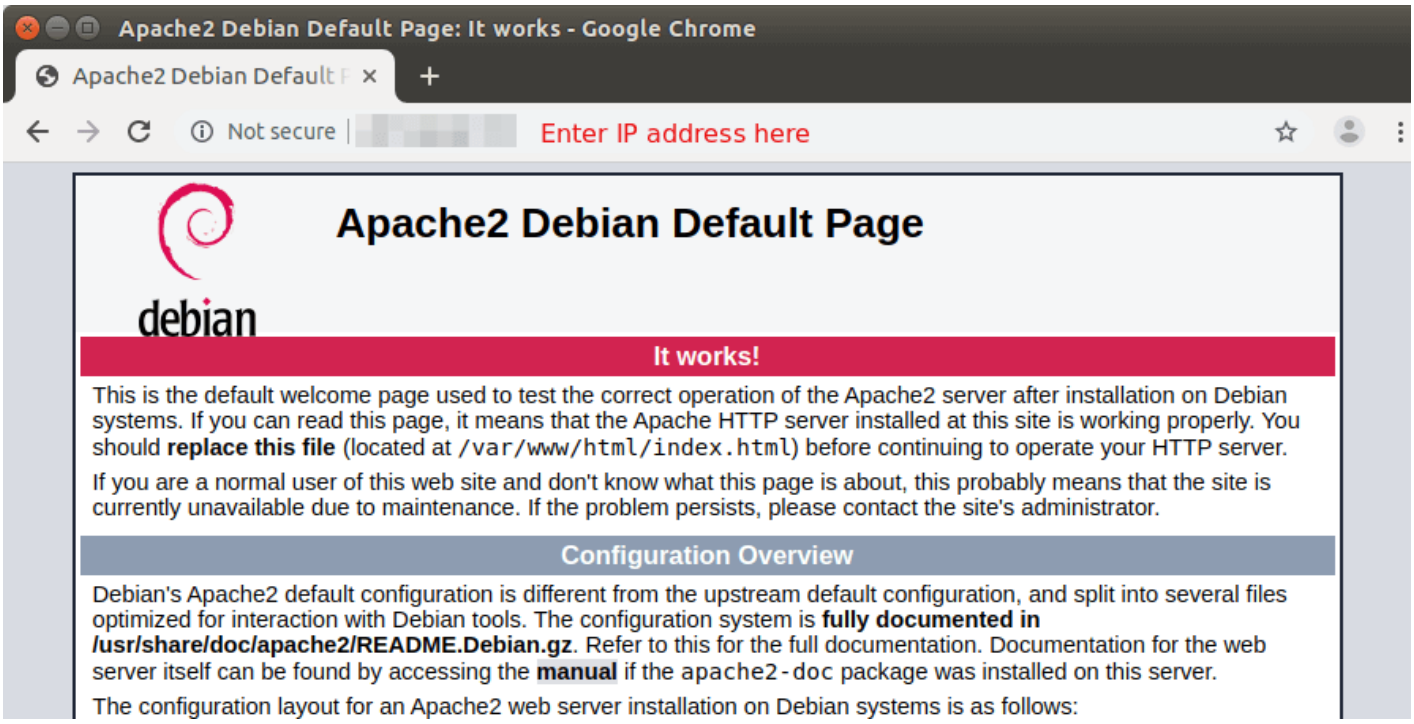
Check Apache version:

```
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 stared. 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
```

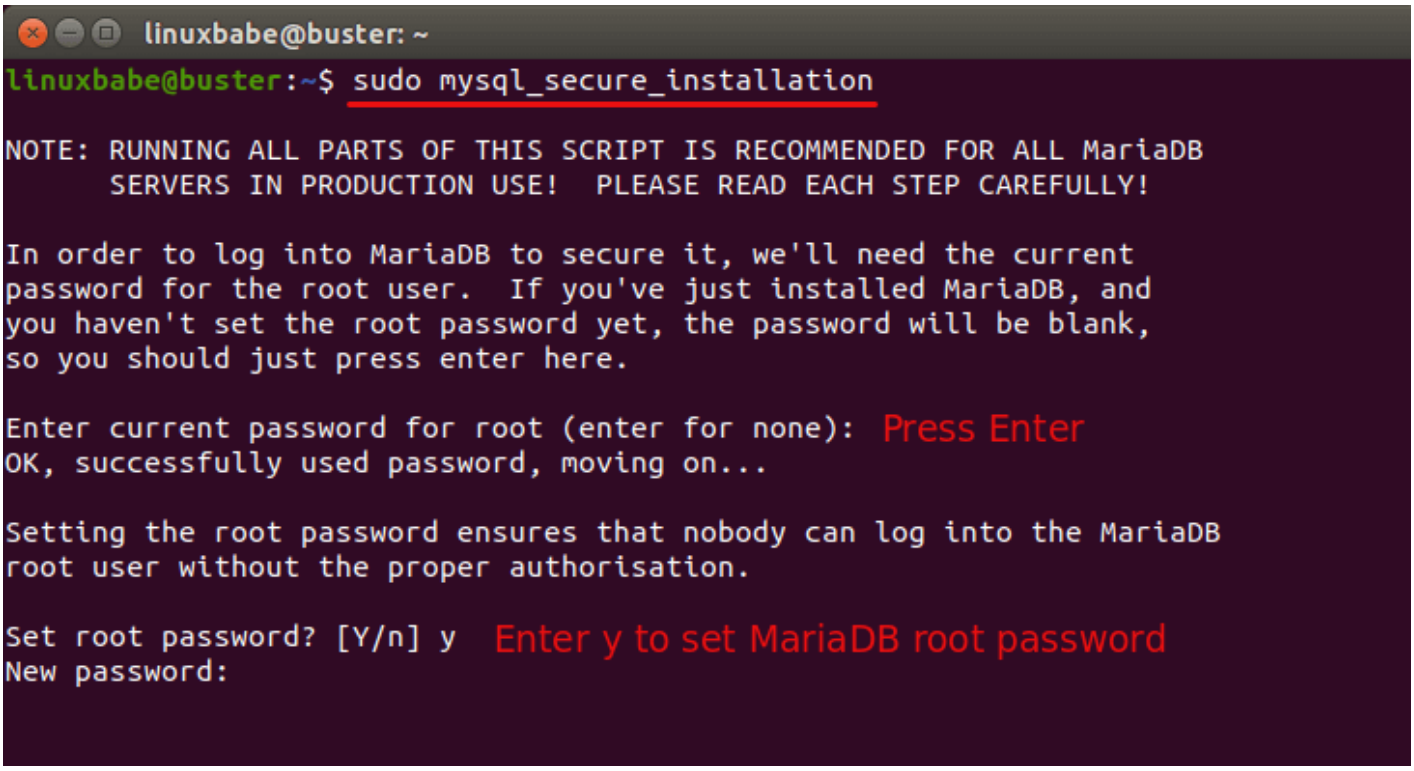
To enable MariaDB to automatically start at boot time, run

```
sudo systemctl enable mariadb
```

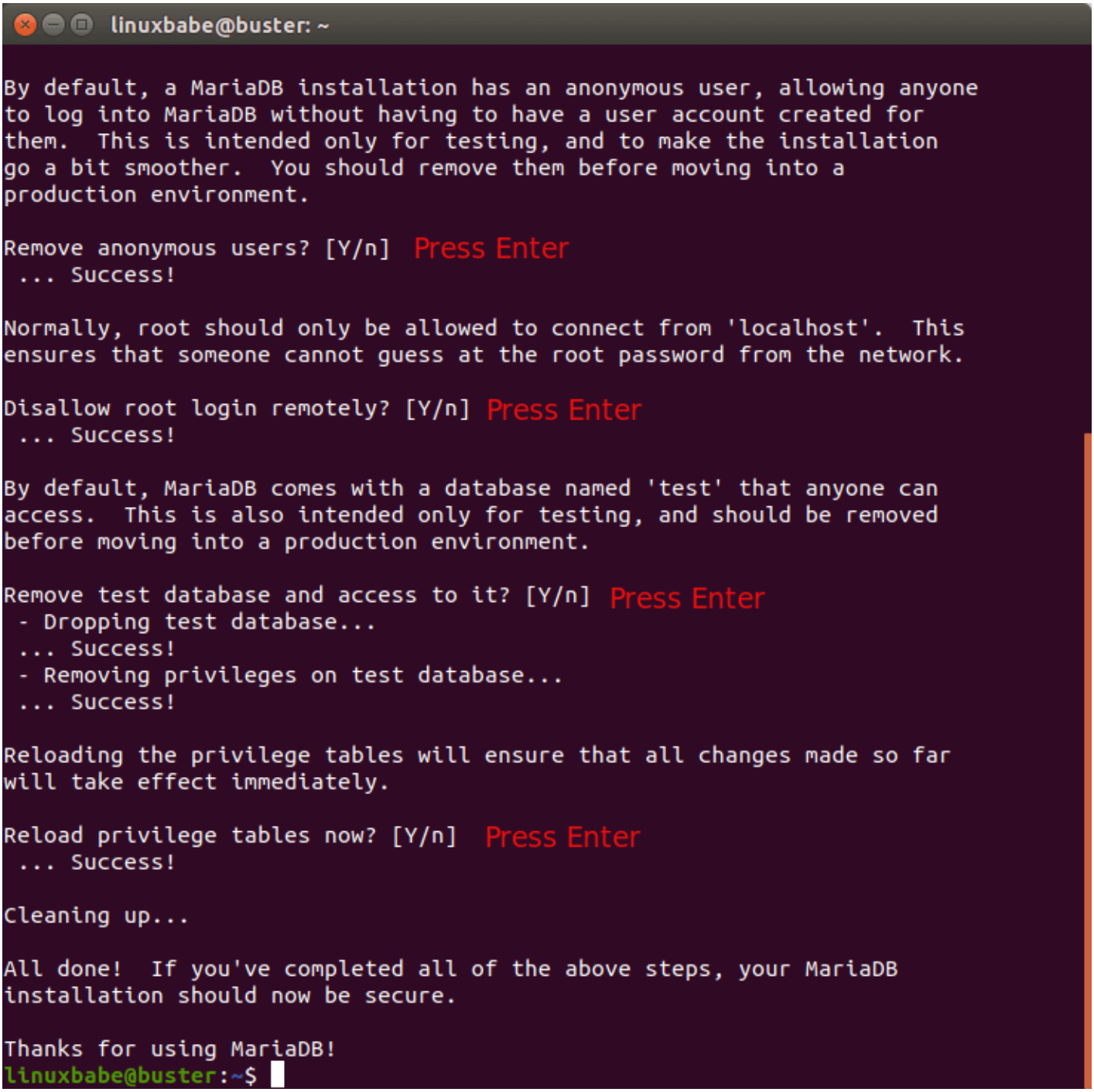
Now run the post installation security script.

```
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.



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.)



By default, the MaraiDB 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
```

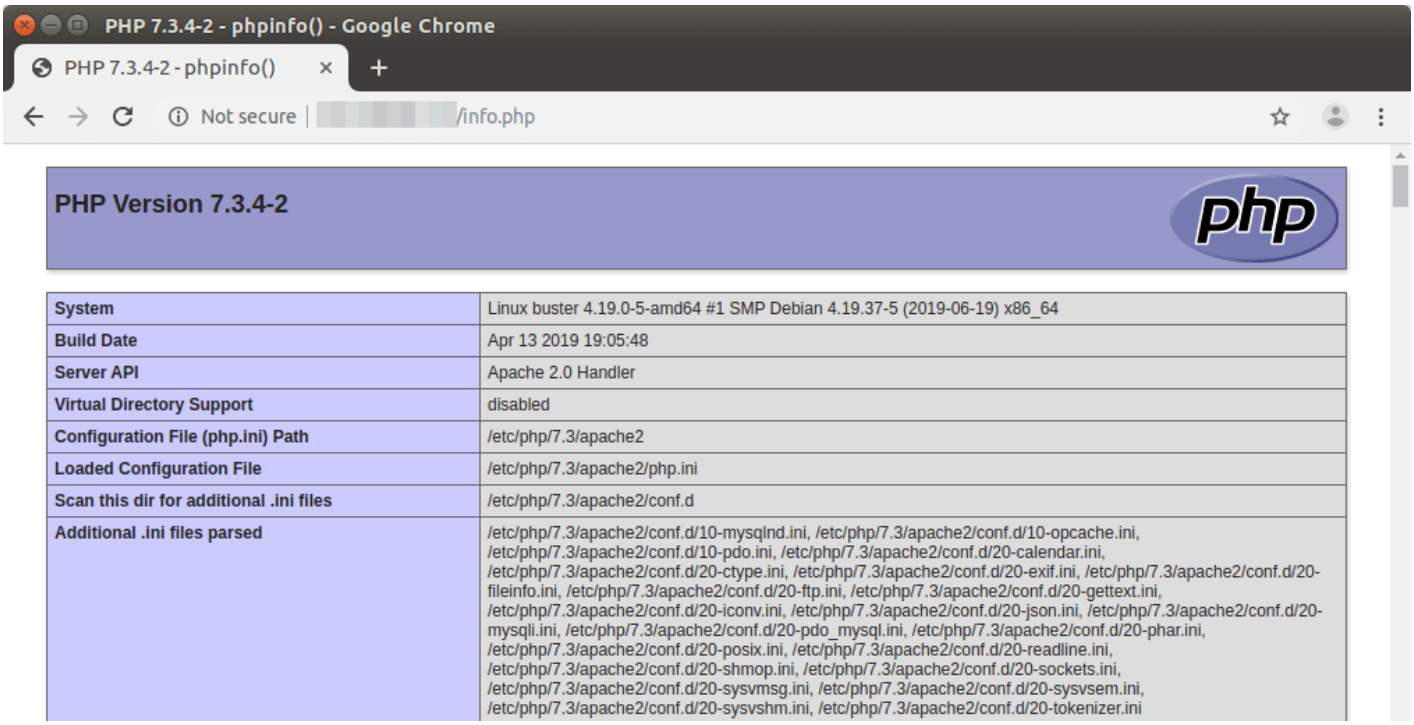
Paste the following PHP code into the file.

```
<?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 `sever-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.



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
```

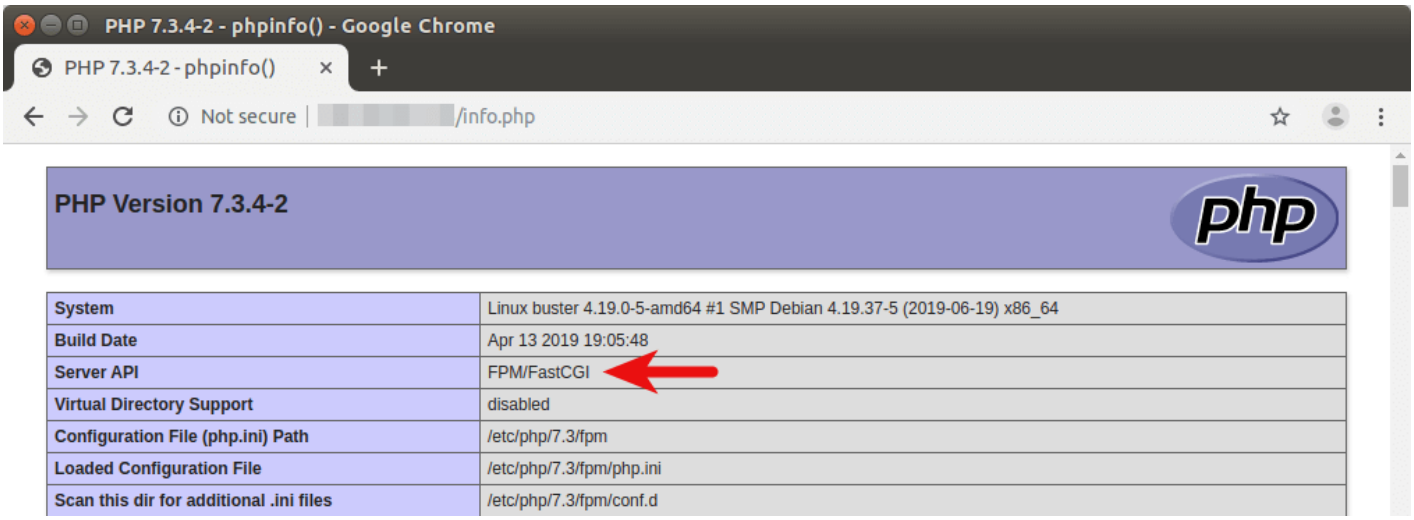
Enable the `/etc/apache2/conf-available/php7.3-fpm.conf` configuration file.

```
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