Nginx

- How to Install LEMP Stack on Ubuntu 20.04 Server
- How to Install LEMP Stack on Debian 10 Buster Server
- Install LEMP Stack (Nginx, MariaDB, PHP7.2) on RHEL 8/CentOS 8

How to Install LEMP Stack on Ubuntu 20.04 Server

This tutorial is going to show you how to install LEMP stack (Nginx, MariaDB, and PHP7.4) on Ubuntu 20.04. A software stack is a set of software tools bundled together. LEMP stands for Linux, Nginx (Engine-X), 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; Nginx 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 LEMP 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 LEMP stack, it’s a good practice to update repository and software packages by running the following commands on your Ubuntu 20.04 OS.

```
sudo apt update


sudo apt upgrade
```

Step 2: Install Nginx Web Server

Nginx is a high-performance web server and very popular these days. It also can be used as a reverse proxy and caching server. Enter the following command to install Nginx Web server.

```
sudo apt install nginx
```

After it’s installed, we can enable Nginx to auto-start at boot time by running the following command.

```
sudo systemctl enable nginx
```

Then start Nginx with this command:

```
sudo systemctl start nginx
```

Now check out its status.

```
sudo systemctl status nginx
```

Output:

```
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2020-04-10 14:11:43 UTC; 3s ago
       Docs: man:nginx(8)
    Process: 8533 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on;
(code=exited, status=0/SUCCESS)
    Process: 8545 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited,
status=0/SUCCESS)
   Main PID: 8549 (nginx)
      Tasks: 3 (limit: 9451)
     Memory: 3.9M
     CGroup: /system.slice/nginx.service
             ├─8549 nginx: master process /usr/sbin/nginx -g daemon on; master_process
on;
             ├─8550 nginx: worker process
             └─8551 nginx: worker process
```

“**Enabled**” indicates that auto-start at boot time is enabled and we can see that Nginx is running. You can also see how much RAM Nginx is using from the output. If the above command doesn’t immediately quit after running. You need to press “**q**” to make it quit.

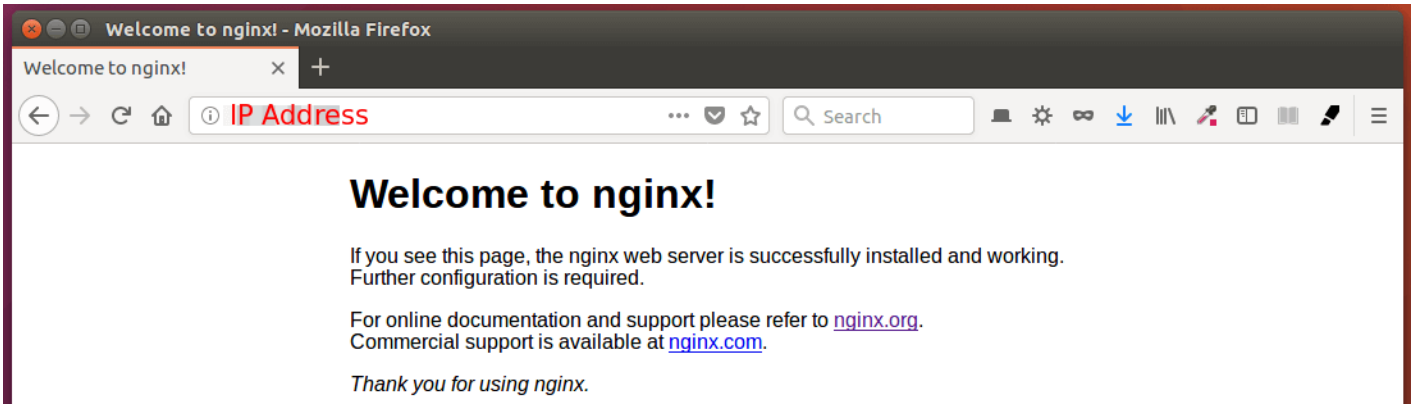
Check Nginx version.

```
nginx -v
```

Output:

```
nginx version: nginx/1.17.9 (Ubuntu)
```

Now type in the public IP address of your Ubuntu 20.04 server in the browser address bar. You should see the “Welcome to Nginx” Web page, which means Nginx Web server is running properly. If you are installing LEMP 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
```

Finally, we need to make `www-data` (Nginx user) as the owner of web directory. By default, it’s owned by the root user.

```
sudo chown www-data: www-data /usr/share/nginx/html -R
```

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 stared. 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
```

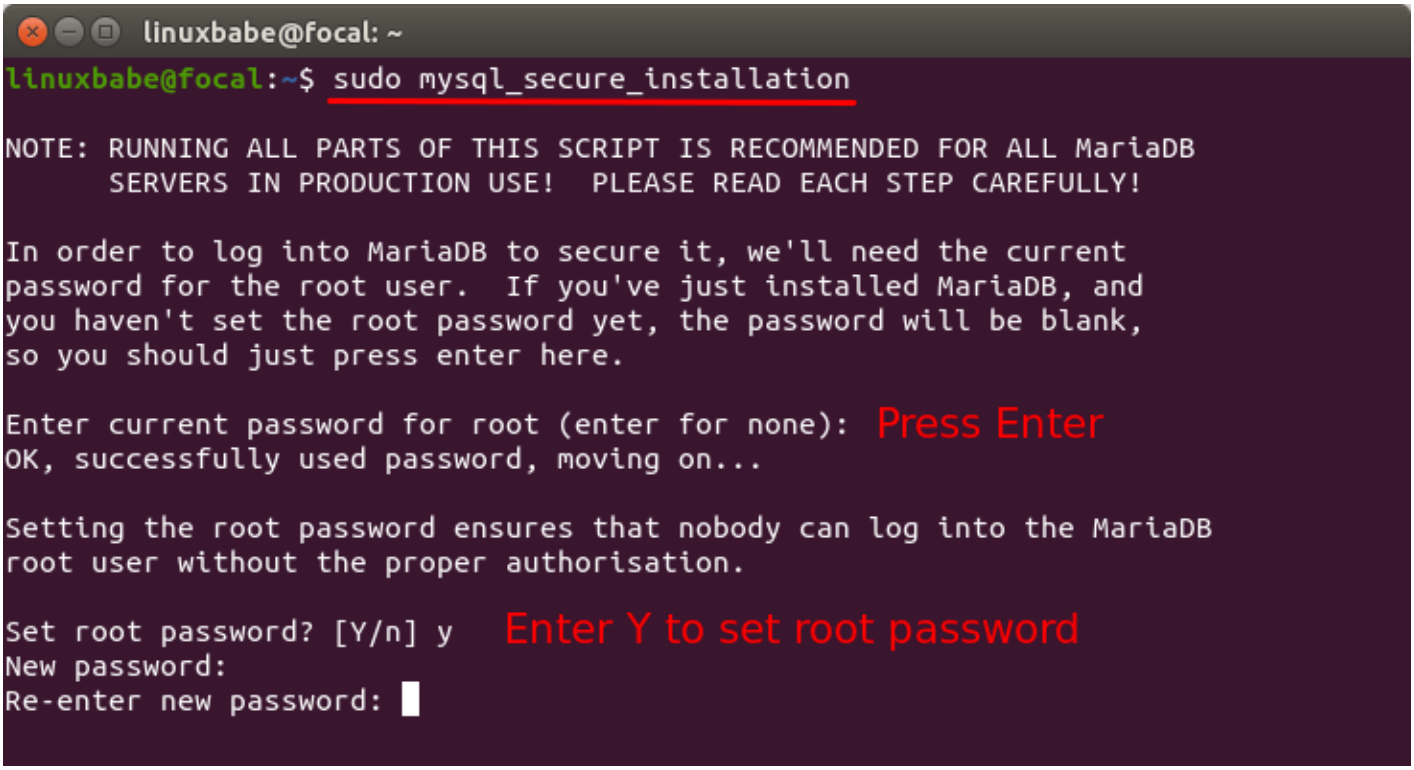
To enable MariaDB to automatically start at boot time, run

```
sudo systemctl enable mariadb
```

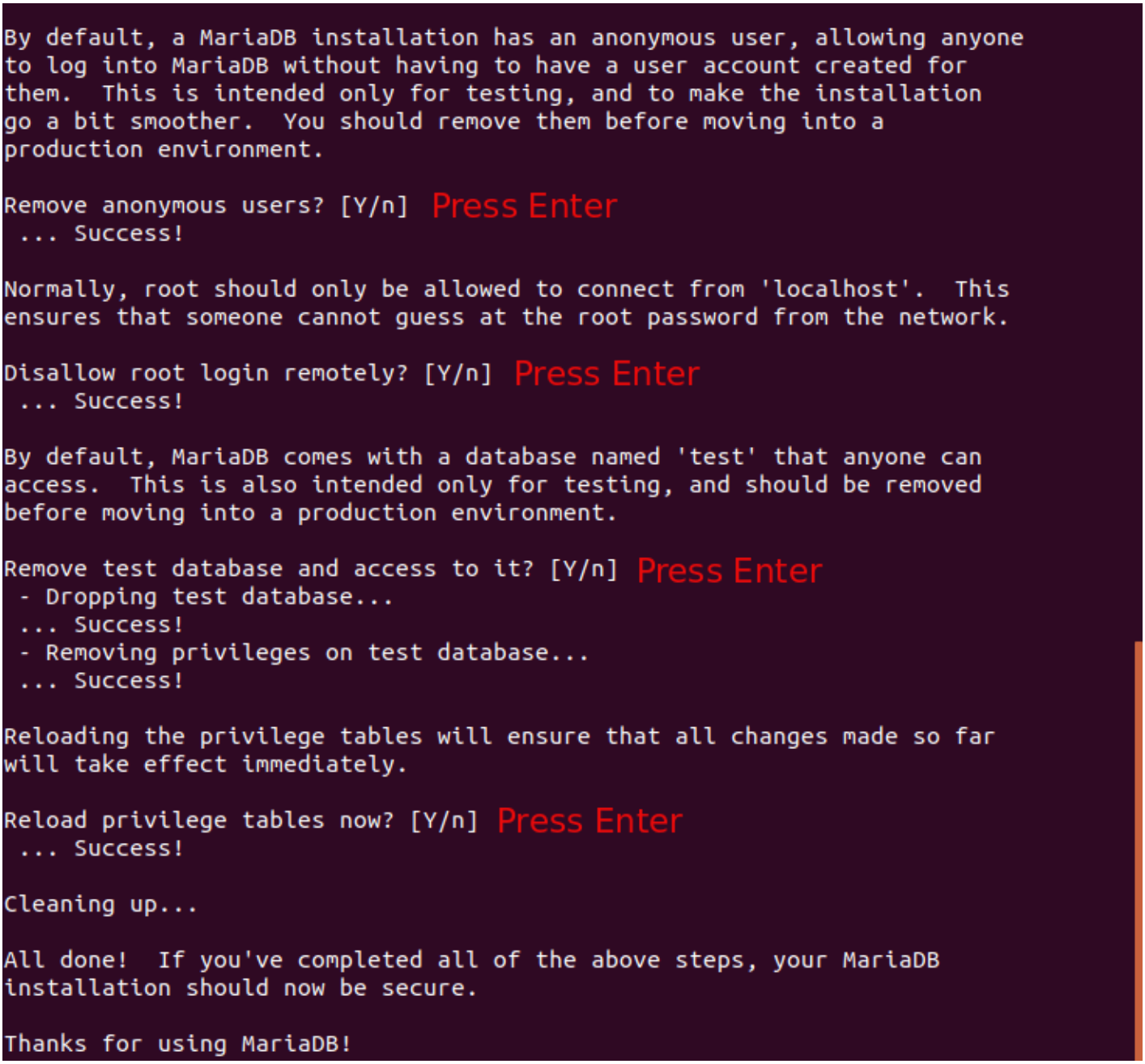
Now run the post installation security script.

```
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 MaraiDB 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

PHP7.4 is included in Ubuntu 20.04 repository and has a minor performance improvement over PHP7.3. Enter the following command to install PHP7.4 and some common extensions.

```
sudo apt install php7.4 php7.4-fpm php7.4-mysql php-common php7.4-cli php7.4-common php7.4-json php7.4-opcache php7.4-readline php7.4-mbstring php7.4-xml php7.4-gd php7.4-curl
```

PHP extensions are commonly needed for content management systems (CMS) like WordPress. For example, if your installation lacks `php7.4-xml`, then some of your WordPress site pages may be blank and you can find an error in Nginx error log like:

```
PHP message: PHP Fatal error:  Uncaught Error: Call to undefined function xml_parser_create()
```

Installing these PHP extensions ensures that your CMS runs smoothly. Now start php7.4-fpm.

```
sudo systemctl start php7.4-fpm
```

Enable auto-start at boot time.

```
sudo systemctl enable php7.4-fpm
```

Check status:

```
systemctl status php7.4-fpm
```

Sample output:

```
● php7.4-fpm.service - The PHP 7.4 FastCGI Process Manager
     Loaded: loaded (/lib/systemd/system/php7.4-fpm.service; enabled; vendor pr>
     Active: active (running) since Fri 2020-04-10 14:40:26 UTC; 12s ago
       Docs: man:php-fpm7.4(8)
    Process: 21019 ExecStartPost=/usr/lib/php/php-fpm-socket-helper install /ru>
   Main PID: 21012 (php-fpm7.4)
```

```
     Status: "Processes active: 0, idle: 2, Requests: 0, slow: 0, Traffic: 0req>
      Tasks: 3 (limit: 9451)
     Memory: 9.4M
     CGroup: /system.slice/php7.4-fpm.service
             ├─21012 php-fpm: master process (/etc/php/7.4/fpm/php-fpm.conf)
             ├─21017 php-fpm: pool www
             └─21018 php-fpm: pool www
```

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

Step 5: Create an Nginx Server Block

An Nginx server block is like a virtual host in Apache. We will not use the default server block because it's inadequate to run PHP code and if we modify it, it becomes a mess. So remove the `default` symlink in `sites-enabled` directory by running the following command. (It's still available as `/etc/nginx/sites-available/default`.)

```
sudo rm /etc/nginx/sites-enabled/default
```

Then use a command-line text editor like Nano to create a brand new server block file under **/etc/nginx/conf.d/** directory.

```
sudo nano /etc/nginx/conf.d/default.conf
```

Paste the following text into the file. The following snippet will make Nginx listen on IPv4 port 80 and IPv6 port 80 with a catch-all server name.

```
server {
  listen 80;
  listen [::]:80;
  server_name _;
  root /usr/share/nginx/html/;
  index index.php index.html index.htm index.nginx-debian.html;

  location / {
    try_files $uri $uri/ /index.php;
```

```
  }

  location ~ \.php$ {
    fastcgi_pass unix:/run/php/php7.4-fpm.sock;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    include fastcgi_params;
    include snippets/fastcgi-php.conf;
  }

 # A long browser cache lifetime can speed up repeat visits to your page
  location ~* \.(jpg|jpeg|gif|png|webp|svg|woff|woff2|ttf|css|js|ico|xml)$ {
       access_log        off;
       log_not_found     off;
       expires           360d;
  }

  # disable access to hidden files
  location ~ /\.ht {
      access_log off;
      log_not_found off;
      deny all;
  }
}
```

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 test Nginx configurations.

```
sudo nginx -t
```

If the test is successful, reload Nginx.

```
sudo systemctl reload nginx
```

Step 6: Test PHP

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

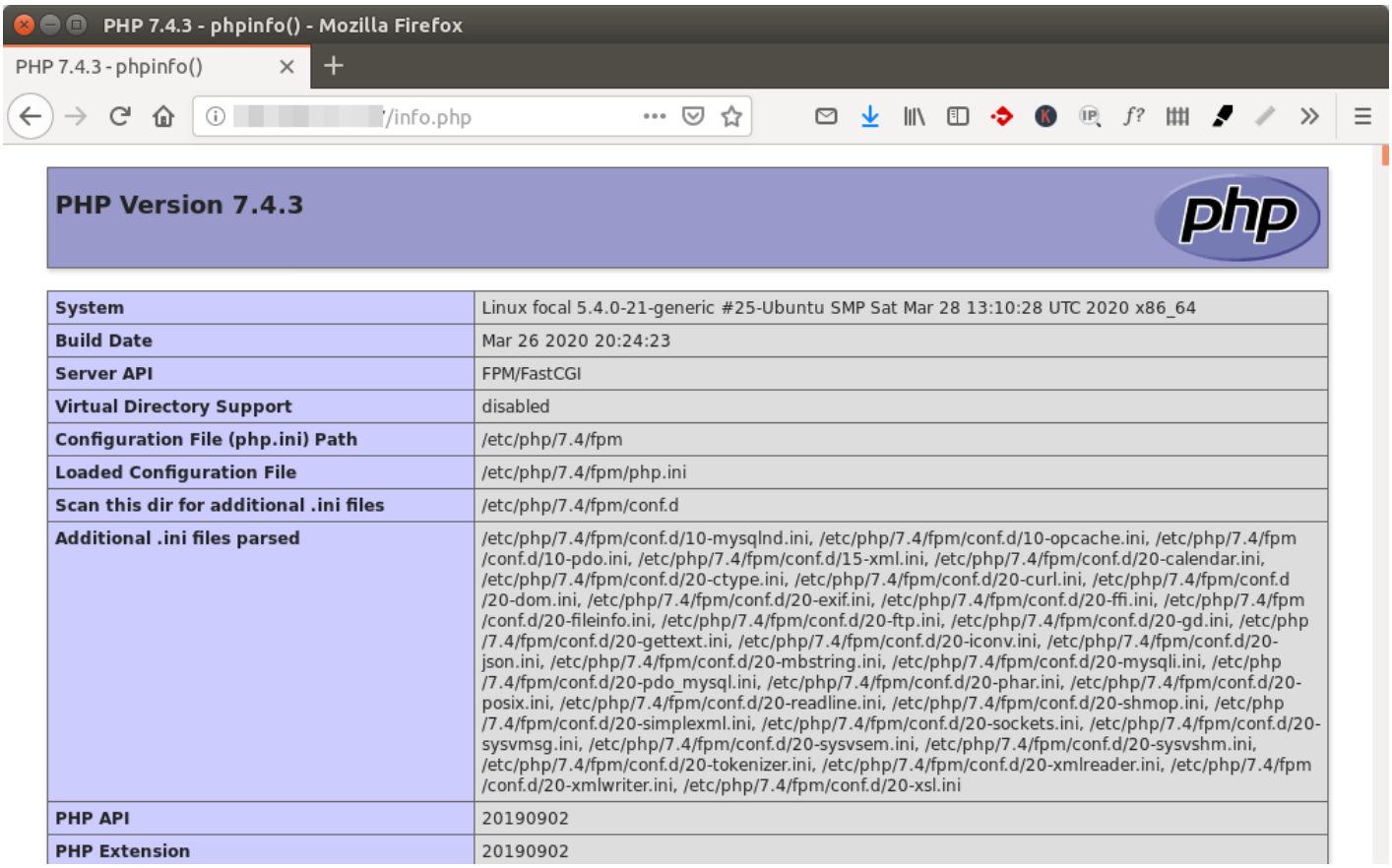
```
sudo nano /usr/share/nginx/html/info.php
```

Paste the following PHP code into the file.

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

Save and close the file. 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 Nginx web server.



System	Linux focal 5.4.0-21-generic #25-Ubuntu SMP Sat Mar 28 13:10:28 UTC 2020 x86_64
Build Date	Mar 26 2020 20:24:23
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.4/fpm
Loaded Configuration File	/etc/php/7.4/fpm/php.ini
Scan this dir for additional .ini files	/etc/php/7.4/fpm/conf.d
Additional .ini files parsed	/etc/php/7.4/fpm/conf.d/10-mysqlnd.ini, /etc/php/7.4/fpm/conf.d/10-opcache.ini, /etc/php/7.4/fpm/conf.d/10-pdo.ini, /etc/php/7.4/fpm/conf.d/15-xml.ini, /etc/php/7.4/fpm/conf.d/20-calendar.ini, /etc/php/7.4/fpm/conf.d/20-ctype.ini, /etc/php/7.4/fpm/conf.d/20-curl.ini, /etc/php/7.4/fpm/conf.d/20-dom.ini, /etc/php/7.4/fpm/conf.d/20-exif.ini, /etc/php/7.4/fpm/conf.d/20-ffi.ini, /etc/php/7.4/fpm/conf.d/20-fileinfo.ini, /etc/php/7.4/fpm/conf.d/20-ftp.ini, /etc/php/7.4/fpm/conf.d/20-gd.ini, /etc/php/7.4/fpm/conf.d/20-gettext.ini, /etc/php/7.4/fpm/conf.d/20-iconv.ini, /etc/php/7.4/fpm/conf.d/20-json.ini, /etc/php/7.4/fpm/conf.d/20-mbstring.ini, /etc/php/7.4/fpm/conf.d/20-mysqli.ini, /etc/php/7.4/fpm/conf.d/20-pdo_mysql.ini, /etc/php/7.4/fpm/conf.d/20-phar.ini, /etc/php/7.4/fpm/conf.d/20-posix.ini, /etc/php/7.4/fpm/conf.d/20-readline.ini, /etc/php/7.4/fpm/conf.d/20-shmop.ini, /etc/php/7.4/fpm/conf.d/20-simplexml.ini, /etc/php/7.4/fpm/conf.d/20-sockets.ini, /etc/php/7.4/fpm/conf.d/20-sysvmsg.ini, /etc/php/7.4/fpm/conf.d/20-sysvsem.ini, /etc/php/7.4/fpm/conf.d/20-sysvshm.ini, /etc/php/7.4/fpm/conf.d/20-tokenizer.ini, /etc/php/7.4/fpm/conf.d/20-xmlreader.ini, /etc/php/7.4/fpm/conf.d/20-xmlwriter.ini, /etc/php/7.4/fpm/conf.d/20-xsl.ini
PHP API	20190902
PHP Extension	20190902

Congrats! You have successfully installed Nginx, MariaDB and PHP7.4 on Ubuntu 20.04. For your server's security, you should delete `info.php` file now to prevent hacker seeing it.

```
sudo rm /usr/share/nginx/html/info.php
```

Troubleshooting Tip

If you encounter errors, you can check the Nginx error log (`/var/log/nginx/error.log`) to find out

what’s wrong.

Nginx Automatic Restart

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

```
sudo systemctl restart nginx
```

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

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

Then create a file under this directory.

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

Add the following lines in the file, which will make Nginx automatically restart **5 seconds** after a failure is detected. The default value of `RetartSec` is **100ms**, which is too small. Nginx may complain that “start request repeated too quickly” if `RestartSec` is not big enough.

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

Save and close the file. Then reload systemd for the changes to take effect.

```
sudo systemctl daemon-reload
```

To check if this would work, kill Nginx with:

```
sudo pkill nginx
```

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

```
systemctl status nginx
```

How to Install LEMP Stack on Debian 10 Buster Server

This tutorial is going to show you how to install Nginx, MariaDB and PHP7.3 (LEMP stack) on Debian 10 Buster. A software stack is a set of software tools bundled together. LEMP stands for **L**inux, **Nginx**, **M**ariaDB/**M**ySQL and **P**HP, all of which are open source and free to use. It is a very common software stack that powers dynamic websites and web applications. Linux is the operating system; Nginx 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 LEMP 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 LEMP 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.

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

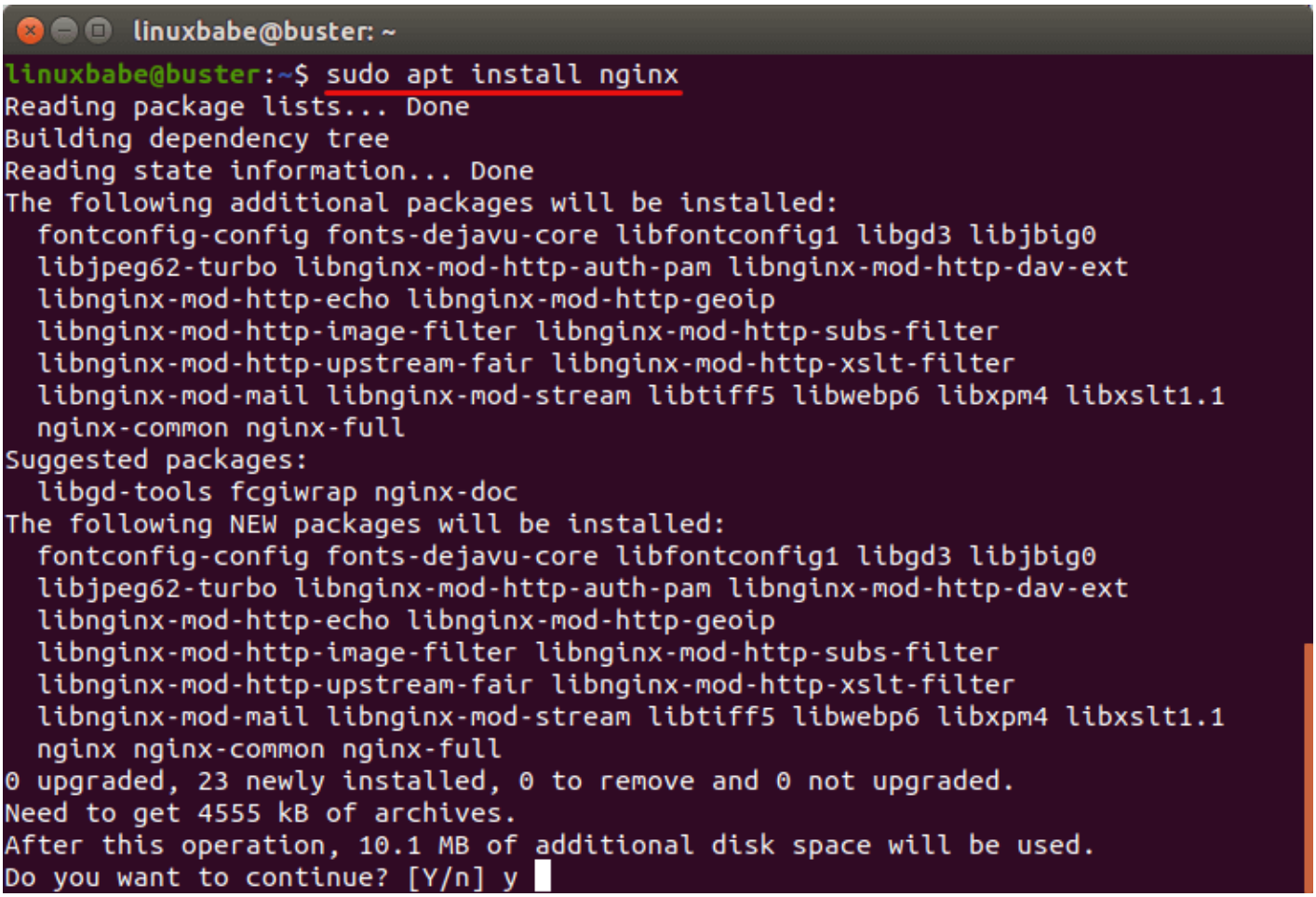
```
sudo apt update


sudo apt upgrade
```

Step 2: Install Nginx Web Server on Debian 10

Nginx is a high performance web server and very popular these days. It also can be used as a reverse proxy and caching server. Enter the following command to install Nginx Web server.

```
sudo apt install nginx
```



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

```
systemctl status nginx
```

Sample output:

```
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: en
   Active: active (running) since Sat 2019-08-10 06:20:26 UTC; 54s ago
     Docs: man:nginx(8)
 Main PID: 19713 (nginx)
    Tasks: 2 (limit: 1149)
   Memory: 4.6M
   CGroup: /system.slice/nginx.service
           ├─19713 nginx: master process /usr/sbin/nginx -g daemon on; master_pr
           └─19714 nginx: worker process
```

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

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

```
sudo systemctl start nginx
```

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

```
sudo systemctl enable nginx
```

Check Nginx version:

```
sudo nginx -v
```

Output:

```
nginx version: nginx/1.14.2
```

Now type in the public IP address of your Debian 10 server in the browser address bar. You should see the default "Welcome to nginx" Web page, which means Nginx Web server is running properly. If you are installing LEMP 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` (Nginx user) as the owner of document root (also known as web root). By default it's owned by the root user. (Note that Nginx by default uses `/usr/share/nginx/html/` as web root, whereas Apache web server uses `/var/www/html/` as we root.)

```
sudo chown www-data: www-data /usr/share/nginx/html/ -R
```

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:
   Active: active (running) since Sat 2019-08-10 06:38:58 UTC; 13s ago
     Docs: man:mysqld(8)
           https://mariadb.com/kb/en/library/systemd/
 Main PID: 20669 (mysqld)
   Status: "Taking your SQL requests now..."
    Tasks: 31 (limit: 1149)
   Memory: 77.7M
   CGroup: /system.slice/mariadb.service
           └─20669 /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

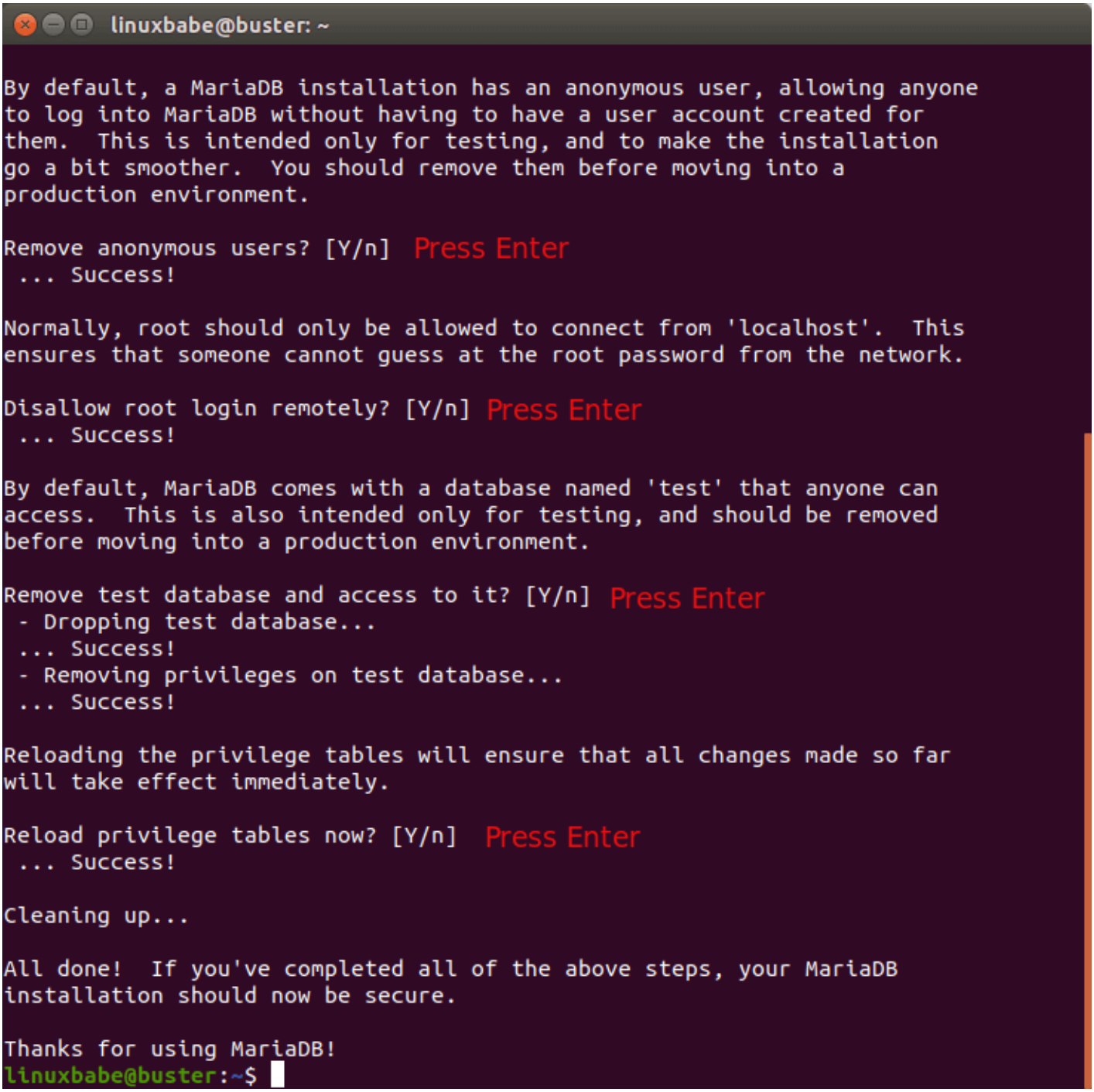
```
sudo systemctl enable mariadb
```

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 and some common PHP extensions from Debian 10 repository.

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

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
```

Now start php7.3-fpm.

```
sudo systemctl start php7.3-fpm
```

Enable auto-start at boot time.

```
sudo systemctl enable php7.3-fpm
```

Check status:

```
systemctl status php7.3-fpm
```

Step 5: Create a Nginx Server Block

A Nginx server block is like a virtual host in Apache. We will not use the default server block because it's inadequate to run PHP code and if we modify it, it becomes a mess. So remove the `default` symlink in `sites-enabled` directory by running the following command. (It's still available as `/etc/nginx/sites-available/default`.)

```
sudo rm /etc/nginx/sites-enabled/default
```

Then create a brand new server block file under **/etc/nginx/conf.d/** directory with a command line text editor, such as Nano.

```
sudo nano /etc/nginx/conf.d/default.conf
```

Paste the following text into the file. The following snippet will make Nginx listen on IPv4 port 80 and IPv6 port 80 with a catch-all server name.

```
server {
  listen 80;
  listen [::]:80;
  server_name _;
  root /usr/share/nginx/html/;
  index index.php index.html index.htm index.nginx-debian.html;

  location / {
    try_files $uri $uri/ /index.php;
  }

  location ~ \.php$ {
    fastcgi_pass unix:/run/php/php7.3-fpm.sock;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
```

```
    include fastcgi_params;
    include snippets/fastcgi-php.conf;
  }

 # A long browser cache lifetime can speed up repeat visits to your page
  location ~* \.(jpg|jpeg|gif|png|webp|svg|woff|woff2|ttf|css|js|ico|xml)$ {
       access_log        off;
       log_not_found     off;
       expires           360d;
  }

  # disable access to hidden files
  location ~ /\.ht {
      access_log off;
      log_not_found off;
      deny all;
  }
}
```

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 test Nginx configurations.

```
sudo nginx -t
```

If the test is successful, reload Nginx.

```
sudo systemctl reload nginx
```

Step 6: Test PHP

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

```
sudo nano /usr/share/nginx/html/info.php
```

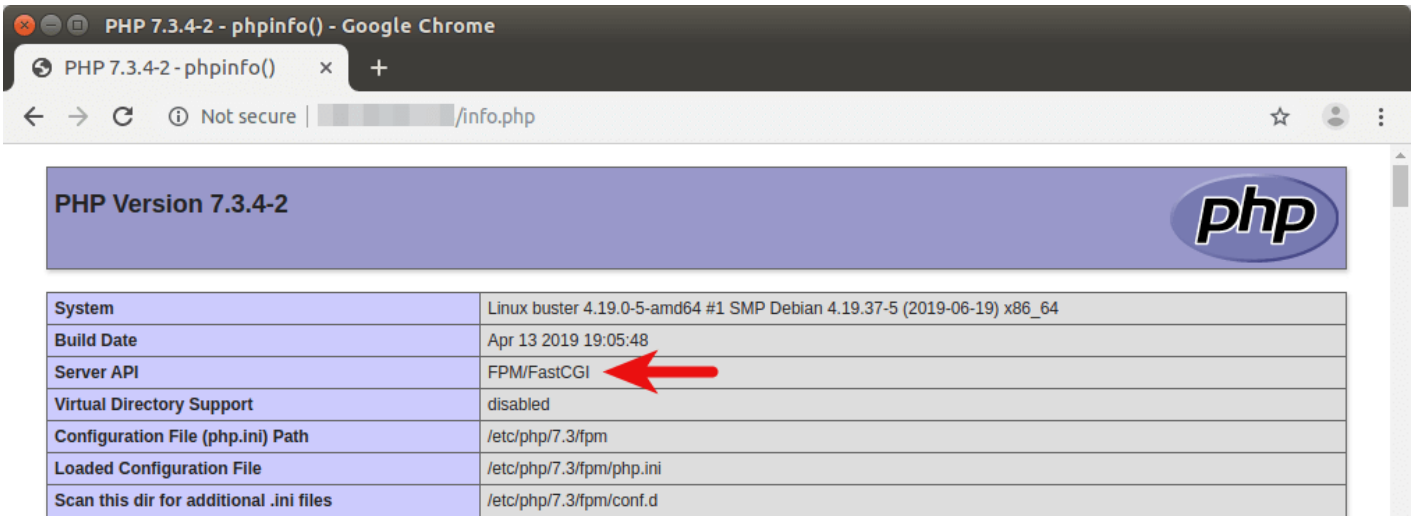
Paste the following PHP code into the file.

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

Save and close the file.

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 Nginx web server. You can find that Zend OPcache is enabled.



Nginx Automatic Restart

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

```
sudo systemctl restart nginx
```

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

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

Then create a file under this directory.

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

Add the following lines in the file, which will make Nginx 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 Nginx with:

```
sudo pkill nginx
```

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

```
systemctl status nginx
```

Install LEMP Stack (Nginx, MariaDB, PHP7.2) on RHEL 8/CentOS 8

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

What's LEMP Stack?

A software stack is a set of software tools bundled together. LEMP stands for Linux, Nginx (pronounced engine X), MariaDB/MySQL and PHP, 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; Nginx 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: Install Nginx Web Server

on RHEL 8/CentOS 8

Nginx is a high performance web server and very popular these days. It also can be used as a reverse proxy and caching server. Enter this command to install Nginx Web server.

```
yum install nginx -y
```

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

```
systemctl start nginx
```

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

```
systemctl enable nginx
```

Now check its status.

```
systemctl status nginx
```

Output:

```
● nginx.service - The nginx HTTP and reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; vendor preset: disabled)
   Active: active (running) since Wed 2018-12-05 02:04:00 EST; 7s ago
 Main PID: 5032 (nginx)
    Tasks: 2 (limit: 11512)
   Memory: 8.4M
   CGroup: /system.slice/nginx.service
           ├─5032 nginx: master process /usr/sbin/nginx
           └─5034 nginx: worker process
```

“**Enabled**” indicates that auto start at boot time is enabled and we can see that Nginx is running. Notice that the above command will not immediately quit after running. You need to press “**q**” to make it quit.

Check Nginx version.

```
nginx -v
```

Output:

```
nginx version: nginx/1.14.1
```

If you are installing LEMP on your local RHEL 8/CentOS 8 computer, then type `127.0.0.1` or `localhost` in the browser address bar. You should see the “Welcome to Nginx” Web page, which means Nginx Web server is running properly.

By default, RHEL 8/CentOS 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 Nginx 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 Nginx web page is accessible publicly.

Finally, we need to make user `nginx` as the owner of web directory. By default it's owned by the root user.

```
chown nginx:nginx /usr/share/nginx/html -R
```

Step 2: Install MariaDB Database Server on RHEL 8/CentOS 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 RHEL 8/CentOS 8.

```
yum 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:

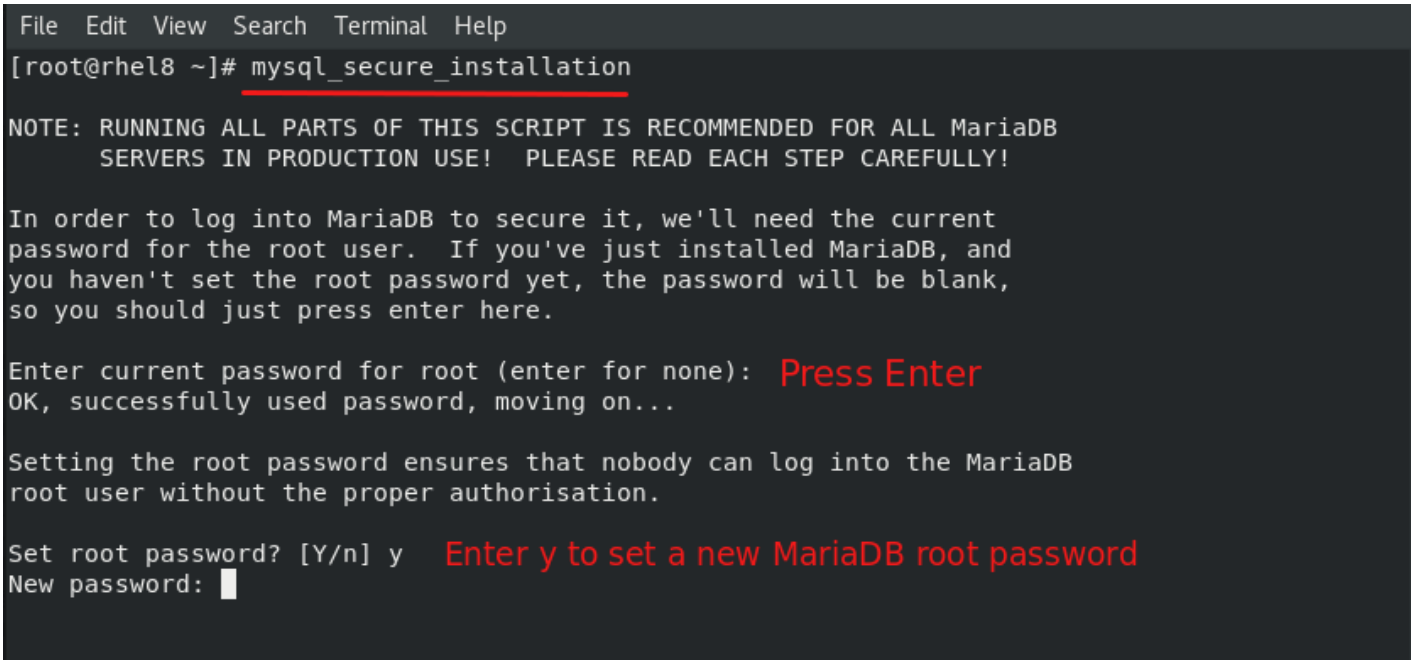
```
● mariadb.service - MariaDB 10.3 database server
   Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; vendor preset:
disable>
   Active: active (running) since Wed 2018-12-05 02:40:44 EST; 8s ago
     Docs: man:mysqld(8)
           https://mariadb.com/kb/en/library/systemd/
 Main PID: 17582 (mysqld)
   Status: "Taking your SQL requests now..."
    Tasks: 30 (limit: 11512)
   Memory: 75.2M
```

```
    CGroup: /system.slice/mariadb.service
            └─17582 /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.)

```
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!
[root@rhel8 ~]#
```

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

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

```
File  Edit  View  Search  Terminal  Help
[root@rhel8 ~]# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 18
Server version: 10.3.10-MariaDB MariaDB Server

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)]> exit;
Bye
[root@rhel8 ~]#
```

To exit, run

```
exit;
```

Step 3: Install PHP-FPM on RHEL 8/CentOS 8

Install PHP and related modules using the following command:

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

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:
disable>
   Active: active (running) since Wed 2018-12-05 03:06:01 EST; 22s ago
 Main PID: 18631 (php-fpm)
   Status: "Processes active: 0, idle: 5, Requests: 0, slow: 0, Traffic: 0req/sec"
    Tasks: 6 (limit: 11512)
   Memory: 29.6M
   CGroup: /system.slice/php-fpm.service
           ├─18631 php-fpm: master process (/etc/php-fpm.conf)
           ├─18632 php-fpm: pool www
```

“**Enabled**” indicates that auto start at boot time is enabled and we can see that PHP-FPM is running. Now edit the PHP-FPM config file:

```
nano /etc/php-fpm.d/www.conf
```

By default, PHP-FPM runs as the `apache` user. Since we are using Nginx web server, we need to change it. Find the following two lines.

```
user = apache
group = apache
```

Change them to

```
user = nginx
group = nginx
```

In this file you can find the following line.

```
listen = /run/php-fpm/www.sock
```

This indicates that PHP-FPM is listening on a Unix socket instead of a TCP/IP socket, which is good. Save and close the file. Reload PHP-FPM for the changes to take effect.

```
systemctl reload php-fpm
```

Step 4: Test PHP

By default, the Nginx package on RHEL 8/CentOS 8 includes configurations for PHP-FPM (`/etc/nginx/conf.d/php-fpm.conf` and `/etc/nginx/default.d/php.conf`). To test PHP-FPM with Nginx Web server, we need to create a `info.php` file in the document root directory.

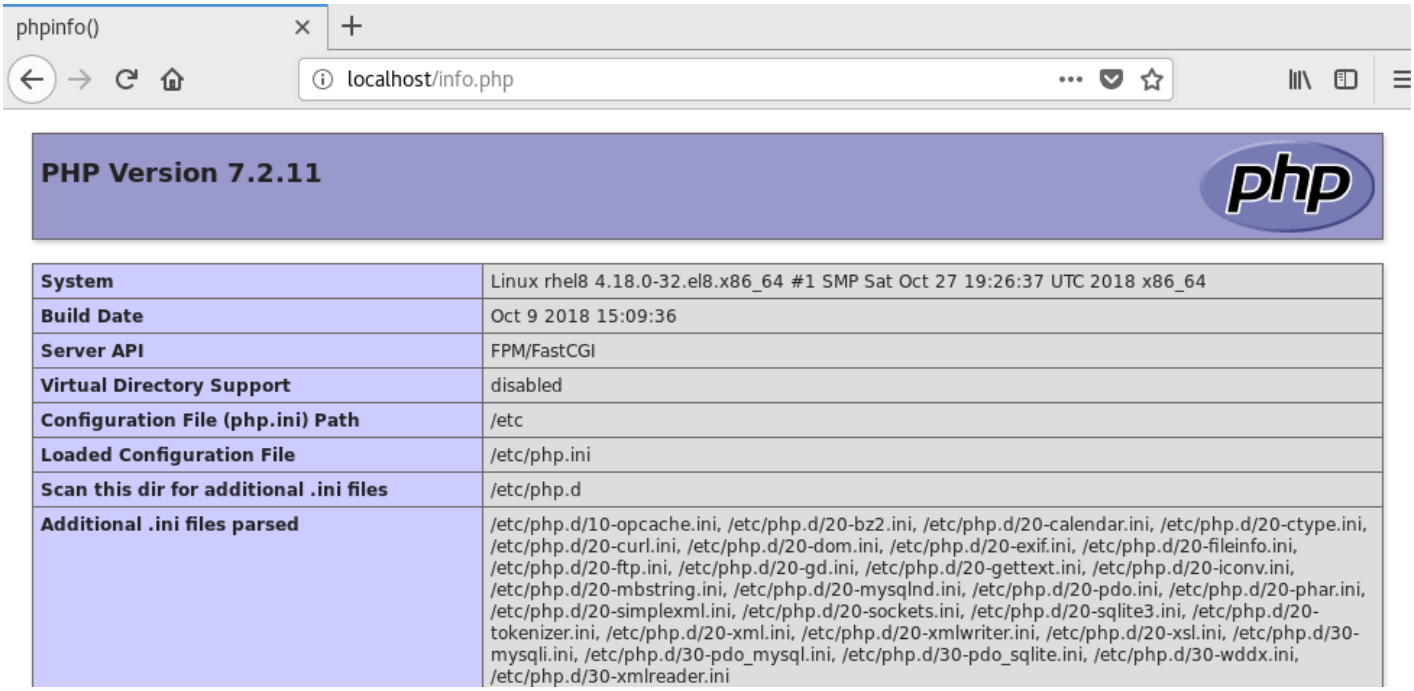
```
nano /usr/share/nginx/html/info.php
```

Paste the following PHP code into the file.

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

Save and close the file. If you installed LEMP stack on a local RHEL 8/CentOS 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 Nginx 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.



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

```
sudo systemctl restart nginx php-fpm
```

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

Nginx Automatic Restart

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

```
sudo systemctl restart nginx
```

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

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

Then create a file under this directory.

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

Add the following lines in the file, which will make Nginx 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 Nginx with:

```
sudo pkill nginx
```

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

```
systemctl status nginx
```