How to Install Nginx on Debian 10

PostedJuly 10, 2019

Introduction

Nginx is one of the most popular web servers in the world and responsible for hosting some of the largest and highest-traffic sites on the internet. It is more resource-friendly than Apache in most cases and can be used as a web server or reverse proxy.

In this guide, we'll discuss how to install Nginx on your Debian 10 server.

Prerequisites

Before you begin this guide, you should have a regular, non-root user with sudo privileges configured on your server and an active firewall. You can learn how to set these up by following our initial server setup guide for Debian 10.

When you have an account available, log in as your non-root user to begin.

Step 1 – Installing Nginx

Because Nginx is available in Debian's default repositories, it is possible to install it from these repositories using the `apt` packaging system.

Since this is our first interaction with the `apt` packaging system in this session, let's first update our local package index so that we have access to the most recent package listings:

```
sudo apt update
```

We can now install `nginx`:

```
sudo apt install nginx
```

When prompted to confirm the installation, hit `Enter` to proceed. After that, `apt` will install Nginx and any required dependencies to your server.

Step 2 – Adjusting the Firewall

Before testing Nginx, the firewall software needs to be adjusted to allow access to the service.

List the application configurations that `ufw` knows how to work with by typing:

```
sudo ufw app list
```

You should get a listing of the application profiles:

```
Output
Available applications:
...
  Nginx Full
  Nginx HTTP
  Nginx HTTPS
```

```
...
```

As you can see, there are three profiles available for Nginx:

- **Nginx Full**: This profile opens both port `80` (normal, unencrypted web traffic) and port `443` (TLS/SSL encrypted traffic)
- **Nginx HTTP**: This profile opens only port `80` (normal, unencrypted web traffic)
- **Nginx HTTPS**: This profile opens only port `443` (TLS/SSL encrypted traffic)

It is recommended that you enable the most restrictive profile that will still allow the traffic you've configured. Since we haven't configured SSL for our server yet in this guide, we will only need to allow traffic for HTTP on port `80`.

You can enable this by typing:

```
sudo ufw allow 'Nginx HTTP'
```

You can verify the change by typing:

```
sudo ufw status
```

You should see HTTP traffic allowed in the displayed output:

```
Output
Status: active

To                           Action      From
--                           ------      ----
OpenSSH                      ALLOW       Anywhere
Nginx HTTP                   ALLOW       Anywhere
OpenSSH (v6)                 ALLOW       Anywhere (v6)
Nginx HTTP (v6)              ALLOW       Anywhere (v6)
```

Step 3 – Checking your Web Server

At the end of the installation process, Debian 10 starts Nginx. The web server should already be up and running.

We can check with the `systemd` init system to make sure the service is running by typing:

```
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 Wed 2019-07-03 12:52:54 UTC; 4min 23s ago
     Docs: man:nginx(8)
 Main PID: 3942 (nginx)
    Tasks: 3 (limit: 4719)
   Memory: 6.1M
   CGroup: /system.slice/nginx.service
           ├─3942 nginx: master process /usr/sbin/nginx -g daemon on;
master_process on;
           ├─3943 nginx: worker process
           └─3944 nginx: worker process
```

As you can see above, the service appears to have started successfully. However, the best way to test this is to actually request a page from Nginx.

You can access the default Nginx landing page to confirm that the software is running properly by navigating to your server's IP address. If you do not know your server's IP address, try typing this at your server's command prompt:

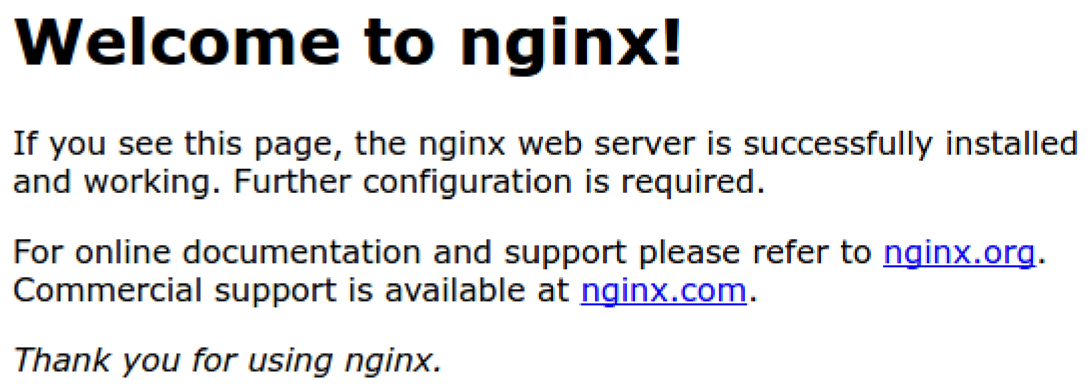
```
ip addr show eth0 | grep inet | awk '{ print $2; }' | sed 's/\/.*$//'
```

You will get back a few lines. You can try each in your web browser to see if they work.

When you have your server's IP address, enter it into your browser's address bar:

```
http://your_server_ip
```

You should see the default Nginx landing page:



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

This page is included with Nginx to show you that the server is running correctly.

Step 4 – Managing the Nginx Process

Now that you have your web server up and running, let's review some basic management commands.

To stop your web server, type:

```
sudo systemctl stop nginx
```

To start the web server when it is stopped, type:

```
sudo systemctl start nginx
```

To stop and then start the service again, type:

```
sudo systemctl restart nginx
```

If you are simply making configuration changes, Nginx can often reload without dropping connections. To do this, type:

```
sudo systemctl reload nginx
```

By default, Nginx is configured to start automatically when the server boots. If this is not what you want, you can disable this behavior by typing:

```
sudo systemctl disable nginx
```

To re-enable the service to start up at boot, you can type:

```
sudo systemctl enable nginx
```

Step 5 – Setting Up Server Blocks

When using the Nginx web server, *server blocks* (similar to virtual hosts in Apache) can be used to encapsulate configuration details and host more than one domain on a single server. We will set up a domain called **your_domain**. To learn more about setting up a domain name with DigitalOcean, see our introduction to DigitalOcean DNS.

Nginx on Debian 10 has one server block enabled by default that is configured to serve documents out of a directory at `/var/www/html`. While this works well for a single site, it can become unmanageable if you are hosting multiple sites. Instead of modifying `/var/www/html`, let's create a directory structure within `/var/www` for the **your_domain** website, leaving `/var/www/html` in place as the default directory to be served if a client request doesn't match any other sites.

Create the directory for **your_domain** as follows, using the `-p` flag to create any necessary parent directories:

```
sudo mkdir -p /var/www/your_domain/html
```

Next, assign ownership of the directory with the `$USER` environment variable, which should reference your current system user:

```
sudo chown -R $USER:$USER /var/www/your_domain/html
```

The permissions of your web root should be correct if you haven't modified your `umask` value, but you can make sure by typing:

```
sudo chmod -R 755 /var/www/your_domain
```

Next, create a sample `index.html` page using `nano` or your favorite editor:

```
nano /var/www/your_domain/html/index.html
```

Inside, add the following sample HTML:

/var/www/your_domain/html/index.html

```
<html>
    <head>
        <title>Welcome to your_domain</title>
    </head>
    <body>
        <h1>Success! Your Nginx server is successfully configured for
<em>your_domain</em>. </h1>
<p>This is a sample page.</p>
    </body>
</html>
```

Save and close the file when you are finished.

In order for Nginx to serve this content, we need to create a server block with the correct directives that point to our custom web root. Instead of modifying the default configuration file directly, let's make a new one at `/etc/nginx/sites-available/your_domain`:

```
sudo nano /etc/nginx/sites-available/your_domain
```

Paste in the following configuration block, which is similar to the default, but updated for our new directory and domain name:

/etc/nginx/sites-available/your_domain

```
server {
        listen 80;
        listen [::]:80;

        root /var/www/your_domain/html;
        index index.html index.htm index.nginx-debian.html;

        server_name your_domain www.your_domain;

        location / {
                try_files $uri $uri/ =404;
        }
}
```

Notice that we've updated the `root` configuration to our new directory, and the `server_name` to our domain name.

Next, let's enable this server block by creating a symbolic link to our custom configuration file inside the `sites-enabled` directory, which Nginx reads from during startup:

```
sudo ln -s /etc/nginx/sites-available/your_domain /etc/nginx/sites-enabled/
```

Two server blocks are now enabled and configured to respond to requests based on their `listen` and `server_name` directives (you can read more about how Nginx processes these directives here):

- `your_domain`: Will respond to requests for `your_domain` and `www.your_domain`.
- `default`: Will respond to any requests on port `80` that do not match the other two blocks.

To avoid a possible hash bucket memory problem that can arise from adding additional server names to your configuration, it is necessary to adjust a single value in the `/etc/nginx/nginx.conf` file. Open the file:

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

Find the `server_names_hash_bucket_size` directive and remove the # symbol to uncomment the line:

/etc/nginx/nginx.conf

```
...
http {
    ...
    server_names_hash_bucket_size 64;
    ...
}
```

```
...
```

Save and close the file when you are finished.

Next, test to make sure that there are no syntax errors in any of your Nginx files:

```
sudo nginx -t
```

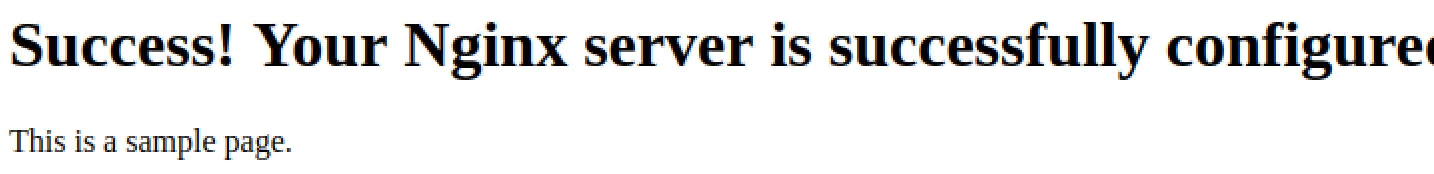
If there aren't any problems, you will see the following output:

```
Output
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

Once your configuration test passes, restart Nginx to enable your changes:

```
sudo systemctl restart nginx
```

Nginx should now be serving your domain name. You can test this by navigating to `http://your_domain`, where you should see something like this:


Success! Your Nginx server is successfully configured

This is a sample page.

Step 6 – Getting Familiar with Important Nginx Files and Directories

Now that you know how to manage the Nginx service itself, you should take a few minutes to familiarize yourself with a few important directories and files.

Content

- `/var/www/html`: The actual web content, which by default only consists of the default Nginx page you saw earlier, is served out of the `/var/www/html` directory. This can be changed by altering Nginx configuration files.

Server Configuration

- `/etc/nginx`: The Nginx configuration directory. All of the Nginx configuration files reside here.
- `/etc/nginx/nginx.conf`: The main Nginx configuration file. This can be modified to make changes to the Nginx global configuration.
- `/etc/nginx/sites-available/`: The directory where per-site server blocks can be stored. Nginx will not use the configuration files found in this directory unless they are linked to the `sites-enabled` directory. Typically, all server block configuration is done in this directory, and then enabled by linking to the other directory.

- `/etc/nginx/sites-enabled/`: The directory where enabled per-site server blocks are stored. Typically, these are created by linking to configuration files found in the `sites-available` directory.
- `/etc/nginx/snippets`: This directory contains configuration fragments that can be included elsewhere in the Nginx configuration. Potentially repeatable configuration segments are good candidates for refactoring into snippets.

Server Logs

- `/var/log/nginx/access.log`: Every request to your web server is recorded in this log file unless Nginx is configured to do otherwise.
- `/var/log/nginx/error.log`: Any Nginx errors will be recorded in this log.

Conclusion

Now that you have your web server installed, you have many options for the type of content you can serve and the technologies you can use to create a richer experience for your users.

https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-debian-10

Install Nginx on Debian 10 Buster

By
koromicha
-
This guide will step you through how to install Nginx on Debian 10 Buster. Nginx is an opensource web server, reverse proxy server for HTTP, HTTPS, SMTP, POP3, and IMAP protocols, load balancer and an HTTP cache.

Install Nginx on Debian 10 Buster

Nginx is available on the default Debian 10 Buster repos and can be installed using the APT package manager.

Run system update

To resynchronize your system packages to their latest versions, run the commands below

```
apt update
apt upgrade
```

Install Nginx on Debian 10 Buster

```
apt install nginx
```

Running Nginx

Upon installation, Nginx is started and enabled to run on system boot.

```
systemctl status nginx

● 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 Thu 2019-07-18 20:48:37 EAT; 17s ago
     Docs: man:nginx(8)
 Main PID: 1497 (nginx)
    Tasks: 2 (limit: 1150)
   Memory: 3.1M
   CGroup: /system.slice/nginx.service
           ├─1497 nginx: master process /usr/sbin/nginx -g daemon on;
master_process on;
           └─1498 nginx: worker process

systemctl is-enabled nginx
enabled
```

To stop or restart Nginx;

```
systemctl stop nginx

systemctl restart nginx
```

To reload Nginx;

```
systemctl reload nginx
```

To verify Nginx for any configuration errors;

```
nginx -t
```

Allow Nginx on Firewall

If UFW is running, allow access to Nginx.

```
ufw allow 'Nginx Full'
```

The above command open ports 80 (HTTP) and 443 (HTTPS) on firewall. If you need to open just port 80 (allow HTTP traffic) or port 443 (allow HTTPS traffic), you can run either of the commands below respectively.

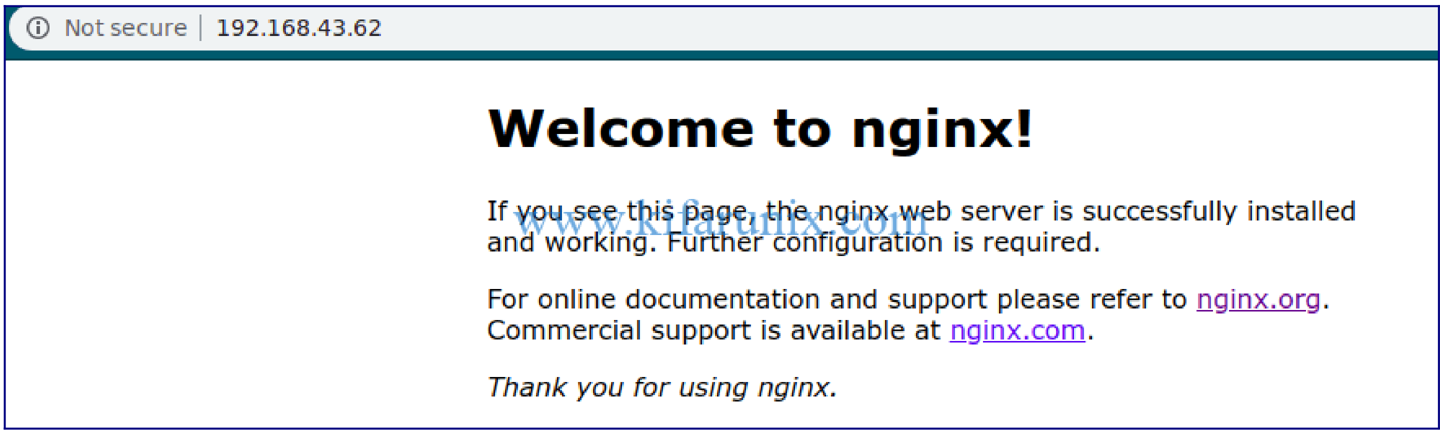
```
ufw allow 'Nginx HTTP'
```

```
ufw allow 'Nginx HTTPS'
```

Accessing Nginx

To verify that Nginx is ready to server your web pages, navigate to the browser and enter the server IP address, **http://<server-IP>**.

If all is well, you should land on Nginx welcome page.



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Nginx is installed successfully on Debian 10 Buster

How to Install Nginx and PHP 7.3 on Debian 10

David Berndtsson
6 months ago
2,002 views
3 min read

Just recently on July 6th, 2019, Debian 10 buster released, and with it came a lot of features. This tutorial will show you how you can install Nginx mainline version on Debian 10 for optimal performance.

Step 1: Install Nginx

Firstly we need to install the prerequisites.

```
sudo apt install curl gnupg2 ca-certificates lsb-release
```

Then we need to add the Nginx mainline package to our repository so that when we run apt install nginx, we will download the mainline version instead of the old stable version.

```
echo "deb http://nginx.org/packages/mainline/debian `lsb_release -cs` nginx" \
    | sudo tee /etc/apt/sources.list.d/nginx.list
```

Optionally, if you would rather want to use the older and slower stable version of Nginx you can do so by running: (Remember that you should only run one of the following)

```
echo "deb http://nginx.org/packages/mainline/debian `lsb_release -cs` nginx" \
    | sudo tee /etc/apt/sources.list.d/nginx.list
```

Next we need to download the signing key so that we can verify its authenticity

```
curl -fsSL https://nginx.org/keys/nginx_signing.key | sudo apt-key add -
```

If it prints out **OK!** then you are good to go!

Now that we have downloaded & verified its authnicity lets install it!

```
sudo apt update
```

```
sudo apt install nginx
```

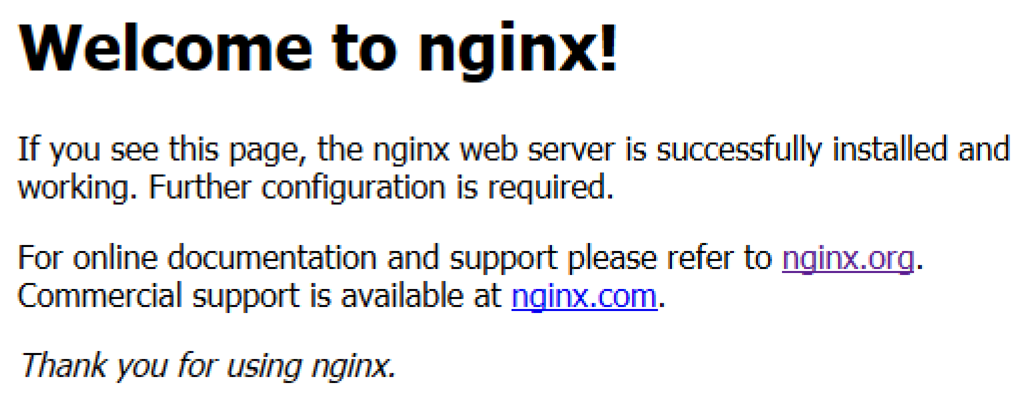
That’s it! You have now installed the latest release of Nginx on Debian 10. You should now start it!

```
sudo systemctl start nginx.service
```

Don’t forget to make it automaticly start on system boot as well.

```
sudo systemctl enable nginx.service
```

Visit your server’s IP address in your webbrowser. You should now see something along these lines if it is working correctly.


Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Step 2: Install PHP 7.3

To add the repositories for PHP:

```
sudo apt-get install software-properties-common
```

Run this to download your added repositories:

```
sudo apt update
```

We can now install PHP now that we have all of the required repositories.

```
sudo
 apt install php7.3-fpm php7.3-common php7.3-mysql php7.3-gmp
php7.3-curl php7.3-intl php7.3-mbstring php7.3-xmlrpc php7.3-gd
php7.3-xml php7.3-cli php7.3-zip php7.3-soap php7.3-imap
```

It is recommended to raise the **memory limit** to improve the overall performance. Your PHP configuration is located in**`/etc/php/7.3/fpm/php.ini`**.

```
sudo nano /etc/php/7.3/fpm/php.ini
```

Press + and search for **memory_limit.**

Replace it with **memory_limit = 256**

Save & exit by pressing + followed by

Step 3: Configure Nginx

Add nginx to www-data group

```
sudo usermod -a -G www-data nginx
```

Change owner of directory to www-data

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

Go into your **default.conf** file

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

Replace your existing configuration file with the one below

```
server {
    listen       80;
    server_name  localhost;

    root   /usr/share/nginx/html;
    index  index.php index.html index.htm;

   location / {
    if ($request_uri ~ ^/(.*)\.html$) {
        return 302 /$1;
        }
    }

    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }
        location ~ \.php$ {
        fastcgi_split_path_info ^(.+\.php)(/.+)$;
        fastcgi_pass   unix:/var/run/php/php7.3-fpm.sock;
        fastcgi_index  index.php;
        fastcgi_param  SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include        fastcgi_params;
    }

}
```

Save & exit by pressing + followed by

Now restart your Nginx:

```
 service nginx restart
```

That’s it! However, you may now test to see if php works

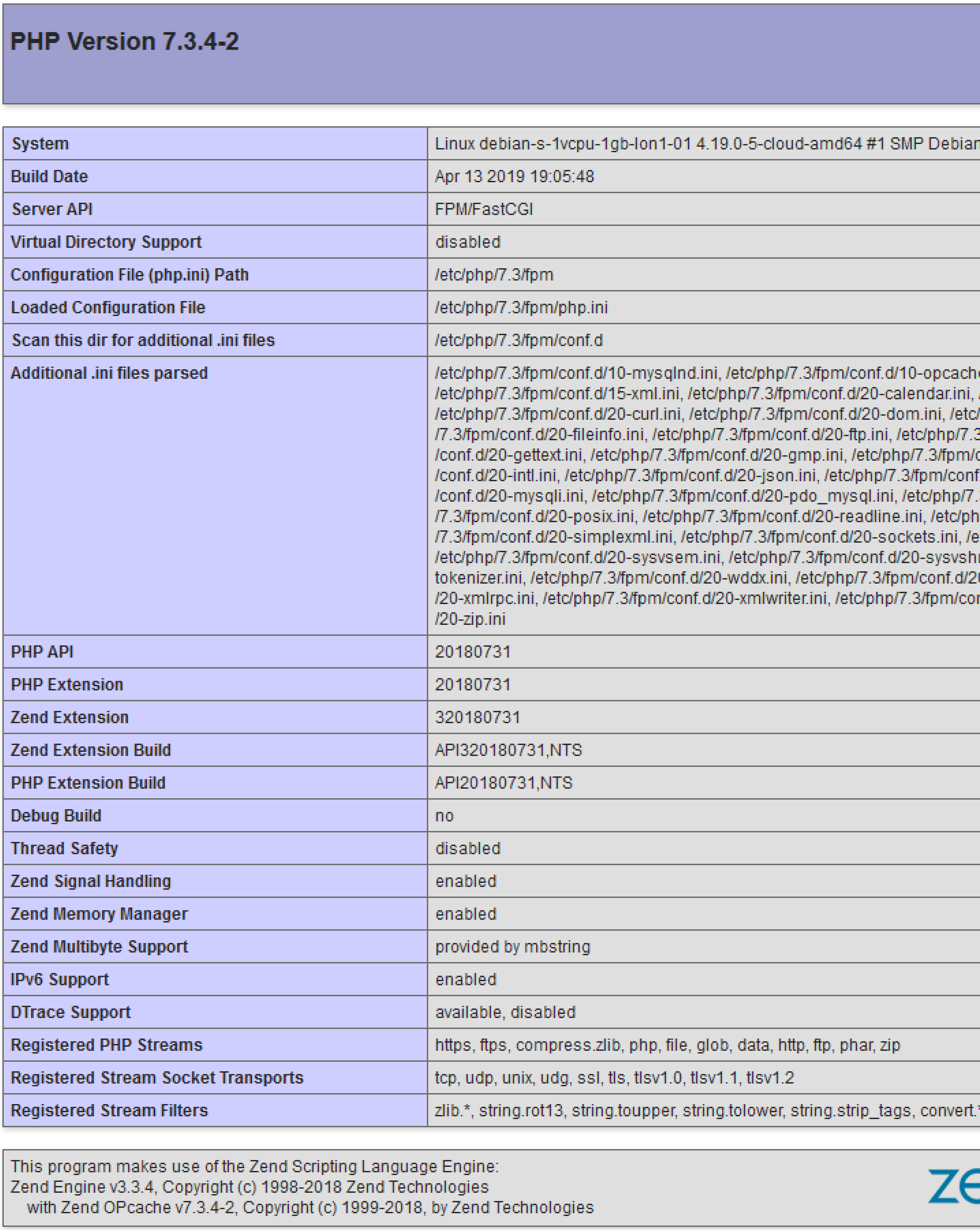
```
nano /usr/share/nginx/html/phpinfo.php
```

Add the following lines

```
<?php

phpinfo();

?>
```

Save & exit by pressing + followed by

Now go to **YOURSERVERIP/phpinfo.php** in your web browser. You should see something along these lines



PHP Version 7.3.4-2

System	Linux debian-s-1vcpu-1gb-lon1-01 4.19.0-5-cloud-amd64 #1 SMP Debian
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
Additional .ini files parsed	/etc/php/7.3/fpm/conf.d/10-mysqlnd.ini, /etc/php/7.3/fpm/conf.d/10-opcach /etc/php/7.3/fpm/conf.d/15-xml.ini, /etc/php/7.3/fpm/conf.d/20-calendar.ini, /etc/php/7.3/fpm/conf.d/20-curl.ini, /etc/php/7.3/fpm/conf.d/20-dom.ini, /etc/ /7.3/fpm/conf.d/20-fileinfo.ini, /etc/php/7.3/fpm/conf.d/20-ftp.ini, /etc/php/7.3 /conf.d/20-gettext.ini, /etc/php/7.3/fpm/conf.d/20-gmp.ini, /etc/php/7.3/fpm/c /conf.d/20-intl.ini, /etc/php/7.3/fpm/conf.d/20-json.ini, /etc/php/7.3/fpm/conf /conf.d/20-mysqli.ini, /etc/php/7.3/fpm/conf.d/20-pdo_mysql.ini, /etc/php/7. /7.3/fpm/conf.d/20-posix.ini, /etc/php/7.3/fpm/conf.d/20-readline.ini, /etc/ph /7.3/fpm/conf.d/20-simplexml.ini, /etc/php/7.3/fpm/conf.d/20-sockets.ini, /e /etc/php/7.3/fpm/conf.d/20-sysvsem.ini, /etc/php/7.3/fpm/conf.d/20-sysvsh tokenizer.ini, /etc/php/7.3/fpm/conf.d/20-wddx.ini, /etc/php/7.3/fpm/conf.d/2 /20-xmlrpc.ini, /etc/php/7.3/fpm/conf.d/20-xmlwriter.ini, /etc/php/7.3/fpm/co /20-zip.ini
PHP API	20180731
PHP Extension	20180731
Zend Extension	320180731
Zend Extension Build	API320180731,NTS
PHP Extension Build	API20180731,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar, zip
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.

This program makes use of the Zend Scripting Language Engine:
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

That's it! Now you know how to install Nginx and PHP 7.3 on Debian 10.

Debian 10 Nginx

David Berndtsson

Member at HostUp and loves to write tutorials that actually work.

https://hostup.org/blog/how-to-install-nginx-and-php-7-3-on-debian-10/

How To Install Linux, Nginx, MariaDB, PHP (LEMP stack) on Debian 10

PostedJuly 10, 2019 16.1k views Nginx LEMP Debian 10

- By Brian Boucheron and Erika Heidi

Introduction

The LEMP software stack is a group of software that can be used to serve dynamic web pages and web applications. The name "LEMP" is an acronym that describes a **L**inux operating system, with an (**E**)Nginx web server. The backend data is stored in a **M**ariaDB database and the dynamic processing is handled by **P**HP.

Although this software stack typically includes **MySQL** as the database management system, some Linux distributions — including Debian — use MariaDB as a drop-in replacement for MySQL.

In this guide, you'll install a LEMP stack on a Debian 10 server using MariaDB as the database management system.

Prerequisites

To complete this guide, you will need access to a Debian 10 server. This server should have a regular user configured with `sudo` privileges and a firewall enabled with `ufw`. To set this up, you can follow our Initial Server Setup with Debian 10 guide.

Step 1 — Installing the Nginx Web Server

In order to serve web pages to your site visitors, we are going to employ Nginx, a popular web server which is well known for its overall performance and stability.

All of the software you will be using for this procedure will come directly from Debian's default package repositories. This means you can use the `apt` package management suite to complete the installation.

Since this is the first time you'll be using `apt` for this session, you should start off by updating your local package index. You can then install the server:

```
sudo apt update
sudo apt install nginx
```

On Debian 10, Nginx is configured to start running upon installation.

If you have the `ufw` firewall running, you will need to allow connections to Nginx. You should enable the most restrictive profile that will still allow the traffic you want. Since you haven't configured SSL for your server yet, for now you only need to allow HTTP traffic on port `80`.

You can enable this by typing:

```
sudo ufw allow 'Nginx HTTP'
```

You can verify the change by typing:

```
sudo ufw status
```

You should see HTTP traffic allowed in the displayed output:

```
Output
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
Nginx HTTP                 ALLOW       Anywhere
OpenSSH (v6)               ALLOW       Anywhere (v6)
Nginx HTTP (v6)            ALLOW       Anywhere (v6)
```

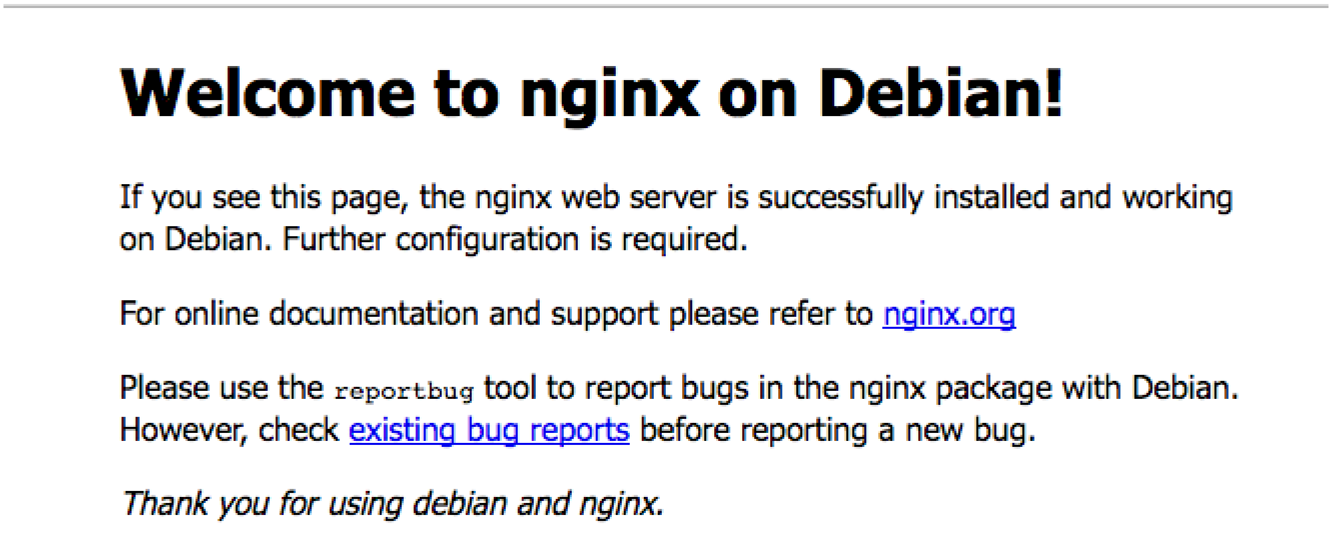
Now, test if the server is up and running by accessing your server's domain name or public IP address in your web browser. If you do not have a domain name pointed at your server and you do not know your server's public IP address, you can find it by typing one of the following into your terminal:

```
ip addr show eth0 | grep inet | awk '{ print $2; }' | sed 's/\/.*$//'
```

This will print out a few IP addresses. You can try each of them in turn in your web browser.

Type one of the addresses that you receive in your web browser. It should take you to Nginx's default landing page:

```
http://your_domain_or_IP
```



Welcome to nginx on Debian!

If you see this page, the nginx web server is successfully installed and working on Debian. Further configuration is required.

For online documentation and support please refer to nginx.org

Please use the `reportbug` tool to report bugs in the nginx package with Debian. However, check existing bug reports before reporting a new bug.

Thank you for using debian and nginx.

If you see the above page, you have successfully installed Nginx.

Step 2 — Installing MariaDB

Now that you have a web server up and running, you need to install the database system to be able to store and manage data for your site.

In Debian 10, the metapackage `mysql-server`, which was traditionally used to install the MySQL server, was replaced by `default-mysql-server`. This metapackage references MariaDB, a community fork of the original MySQL server by Oracle, and it's currently the default MySQL-compatible database server available on debian-based package manager repositories.

For longer term compatibility, however, it's recommended that instead of using the metapackage you install MariaDB using the program's actual package, `mariadb-server`.

To install this software, run:

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

When the installation is finished, it's recommended that you run a security script that comes pre-installed with MariaDB. This script will remove some insecure default settings and lock down access to your database system. Start the interactive script by running:

```
sudo mysql_secure_installation
```

This script will take you through a series of prompts where you can make some changes to your MariaDB setup. The first prompt will ask you to enter the current **database root** password. This is not to be confused with the **system root**. The **database root** user is an administrative user with full privileges over the database system. Because you just installed MariaDB and haven't made any configuration changes yet, this password will be blank, so just press `ENTER` at the prompt.

The next prompt asks you whether you'd like to set up a **database root** password. Because MariaDB uses a special authentication method for the **root** user that is typically safer than using a password, you don't need to set this now. Type `N` and then press `ENTER`.

From there, you can press `Y` and then `ENTER` to accept the defaults for all the subsequent questions. This will remove anonymous users and the test database, disable remote **root** login, and load these new rules so that MariaDB immediately respects the changes you have made.
When you're finished, log in to the MariaDB console by typing:

```
sudo mariadb
```

This will connect to the MariaDB server as the administrative database user **root**, which is inferred by the use of `sudo` when running this command. You should see output like this:

```
Output
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 74
Server version: 10.3.15-MariaDB-1 Debian 10

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)]>
```

Notice that you didn't need to provide a password to connect as the **root** user. That works because the default authentication method for the administrative MariaDB user is `unix_socket` instead of `password`. Even though this might look like a security concern at first, it makes the database server more secure because the only users allowed to log in as the **root** MariaDB user are the system users with sudo privileges connecting from the console or through an application running with the same privileges. In practical terms, that means you won't be able to use the administrative database **root** user to connect from your PHP application.

For increased security, it's best to have dedicated user accounts with less expansive privileges set up for every database, especially if you plan on having multiple databases hosted on your server. To demonstrate such a setup, we'll create a database named **example_database** and a user named **example_user**, but you can replace these names with different values.
To create a new database, run the following command from your MariaDB console:

```
CREATE DATABASE example_database;
```

Now you can create a new user and grant them full privileges on the custom database you've just created. The following command defines this user's password as `password`, but you should replace this value with a secure password of your own choosing.

```
GRANT ALL ON example_database.* TO 'example_user'@'localhost' IDENTIFIED BY
'password' WITH GRANT OPTION;
```

This will give the **example_user** user full privileges over the **example_database** database, while preventing this user from creating or modifying other databases on your server.

Flush the privileges to ensure that they are saved and available in the current session:

```
FLUSH PRIVILEGES;
```

Following this, exit the MariaDB shell:

```
exit
```

You can test if the new user has the proper permissions by logging in to the MariaDB console again, this time using the custom user credentials:

```
mariadb -u example_user -p
```

Note the `-p` flag in this command, which will prompt you for the password used when creating the **example_user** user. After logging in to the MariaDB console, confirm that you have access to the **example_database** database:

```
SHOW DATABASES;
```

This will give you the following output:

```
Output
+--------------------+
| Database           |
+--------------------+
| example_database   |
| information_schema |
+--------------------+
2 rows in set (0.000 sec)
```

To exit the MariaDB shell, type:

```
exit
```

At this point, your database system is set up and you can move on to installing PHP, the final component of the LEMP stack.

Step 3 — Installing PHP for Processing

You have Nginx installed to serve your content and MySQL installed to store and manage your data. Now you can install PHP to process code and generate dynamic content for the web server.

While Apache embeds the PHP interpreter in each request, Nginx requires an external program to handle PHP processing and act as *bridge* between the PHP interpreter itself and the web server. This allows for a better overall performance in most PHP-based websites, but it requires additional configuration. You'll need to install `php-fpm`, which stands for "PHP fastCGI process manager", and tell Nginx to pass PHP requests to this software for processing. Additionally, you'll need `php-mysql`, a PHP module that allows PHP to communicate with MySQL-based databases. Core PHP packages will automatically be installed as dependencies.

To install the `php-fpm` and `php-mysql` packages, run:

```
sudo apt install php-fpm php-mysql
```

You now have your PHP components installed. Next, you'll configure Nginx to use them.

Step 4 — Configuring Nginx to Use the PHP Processor

When using the Nginx web server, *server blocks* (similar to virtual hosts in Apache) can be used to encapsulate configuration details and host more than one domain on a single server. In this guide, we'll use **your_domain** as example domain name. To learn more about setting up a domain name with DigitalOcean, see our introduction to DigitalOcean DNS.

On Debian 10, Nginx has one server block enabled by default and is configured to serve documents out of a directory at `/var/www/html`. While this works well for a single site, it can become difficult to manage if you are hosting multiple sites. Instead of modifying `/var/www/html`, let's create a directory structure within `/var/www` for the **your_domain** website, leaving `/var/www/html` in place as the default directory to be served if a client request doesn't match any other sites.

Create the root web directory for **your_domain** as follows:

```
sudo mkdir /var/www/your_domain
```

Next, assign ownership of the directory with the $USER environment variable, which should reference your current system user:

```
sudo chown -R $USER:$USER /var/www/your_domain
```

Then, open a new configuration file in Nginx's `sites-available` directory using your preferred command-line editor. Here, we'll use `nano`:

```
sudo nano /etc/nginx/sites-available/your_domain
```

This will create a new blank file. Paste in the following bare-bones configuration:

/etc/nginx/sites-available/your_domain

```
server {
    listen 80;
    listen [::]:80;

    root /var/www/your_domain;
    index index.php index.html index.htm;

    server_name your_domain;

    location / {
        try_files $uri $uri/ =404;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/var/run/php/php7.3-fpm.sock;
    }
}
```

This is a basic configuration that listens on port `80` and serves files from the web root you just created. It will only respond to requests to the host or IP address provided after `server_name`, and any files ending in `.php` will be processed by `php-fpm` before Nginx sends the results to the user.

When you're done editing, save and close the file. If you used `nano` to create the file, do so by typing `CTRL`+`X` and then `y` and `ENTER` to confirm.

Activate your configuration by linking to the config file from Nginx's `sites-enabled` directory:

```
sudo ln -s /etc/nginx/sites-available/your_domain /etc/nginx/sites-enabled/
```

This will tell Nginx to use the configuration next time it is reloaded. You can test your configuration for syntax errors by typing:

```
sudo nginx -t
```

If any errors are reported, go back to your configuration file to review its contents before continuing.

When you are ready, reload Nginx to make the changes:

```
sudo systemctl reload nginx
```

Next, you'll create a file in your new web root directory to test out PHP processing.

Step 5 — Creating a PHP File to Test Configuration

Your LEMP stack should now be completely set up. You can test it to validate that Nginx can correctly hand `.php` files off to your PHP processor.

You can do this by creating a test PHP file in your document root. Open a new file called `info.php` within your document root in your text editor:

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

Type or paste the following lines into the new file. This is valid PHP code that will return information about your server:

/var/www/your_domain/info.php

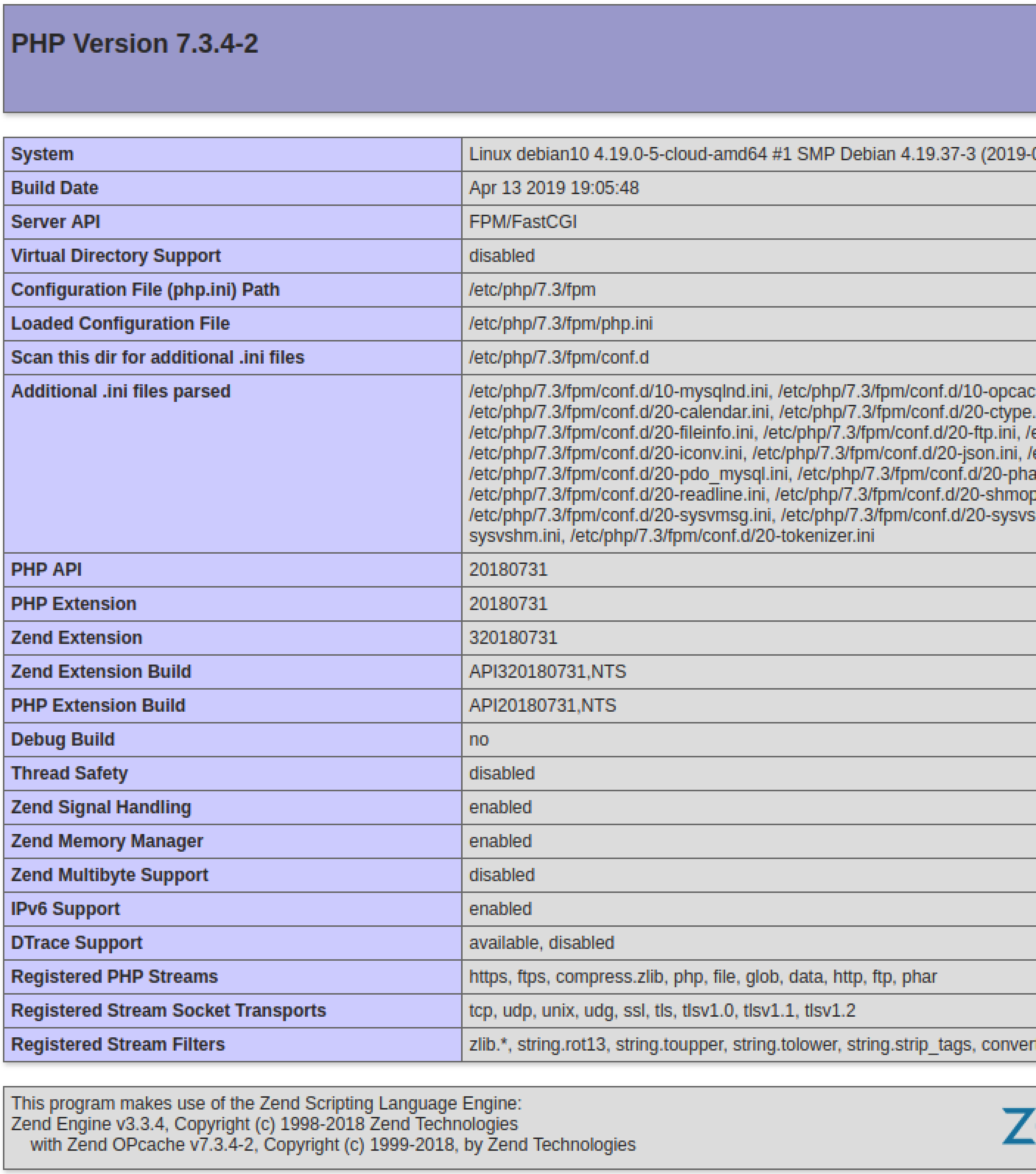
```
<?php
phpinfo();
```

When you are finished, save and close the file by typing `CTRL`+`X` and then `y` and `ENTER` to confirm.

You can now access this page in your web browser by visiting the domain name or public IP address you’ve set up in your Nginx configuration file, followed by `/info.php`:

```
http://your_domain/info.php
```

You will see a web page containing detailed information about your server:



PHP Version 7.3.4-2

System	Linux debian10 4.19.0-5-cloud-amd64 #1 SMP Debian 4.19.37-3 (2019-(
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
Additional .ini files parsed	/etc/php/7.3/fpm/conf.d/10-mysqlnd.ini, /etc/php/7.3/fpm/conf.d/10-opcac /etc/php/7.3/fpm/conf.d/20-calendar.ini, /etc/php/7.3/fpm/conf.d/20-ctype. /etc/php/7.3/fpm/conf.d/20-fileinfo.ini, /etc/php/7.3/fpm/conf.d/20-ftp.ini, /e /etc/php/7.3/fpm/conf.d/20-iconv.ini, /etc/php/7.3/fpm/conf.d/20-json.ini, /e /etc/php/7.3/fpm/conf.d/20-pdo_mysql.ini, /etc/php/7.3/fpm/conf.d/20-pha /etc/php/7.3/fpm/conf.d/20-readline.ini, /etc/php/7.3/fpm/conf.d/20-shmop /etc/php/7.3/fpm/conf.d/20-sysvmsg.ini, /etc/php/7.3/fpm/conf.d/20-sysvs sysvshm.ini, /etc/php/7.3/fpm/conf.d/20-tokenizer.ini
PHP API	20180731
PHP Extension	20180731
Zend Extension	320180731
Zend Extension Build	API320180731,NTS
PHP Extension Build	API20180731,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, conver

This program makes use of the Zend Scripting Language Engine:
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

After checking the relevant information about your PHP server through that page, it's best to remove the file you created as it contains sensitive information about your PHP environment and your Debian server. You can use `rm` to remove that file:

```
rm /var/www/your_domain/info.php
```

You can always regenerate this file if you need it later. Next, we’ll test the database connection from the PHP side.

Step 6 — Testing Database Connection from PHP (Optional)

If you want to test if PHP is able to connect to MariaDB and execute database queries, you can create a test table with dummy data and query for its contents from a PHP script.

First, connect to the MariaDB console with the database user you created in Step 2 of this guide:

```
mariadb -u example_user -p
```

Create a table named **todo_list**. From the MariaDB console, run the following statement:

```
CREATE TABLE example_database.todo_list (
    item_id INT AUTO_INCREMENT,
    content VARCHAR(255),
    PRIMARY KEY(item_id)
);
```

Now, insert a few rows of content in the test table. You might want to repeat the next command a few times, using different values:

```
INSERT INTO example_database.todo_list (content) VALUES ("My first important
item");
```

To confirm that the data was successfully saved to your table, run:

```
SELECT * FROM example_database.todo_list;
```

You will see the following output:

```
Output
+---------+--------------------------+
| item_id | content                  |
+---------+--------------------------+
|       1 | My first important item  |
|       2 | My second important item |
|       3 | My third important item  |
|       4 | and this one more thing  |
+---------+--------------------------+
4 rows in set (0.000 sec)
```

After confirming that you have valid data in your test table, you can exit the MariaDB console:

```
exit
```

Now you can create the PHP script that will connect to MariaDB and query for your content. Create a new PHP file in your custom web root directory using your preferred editor. We’ll use `nano` for that:

```
nano /var/www/your_domain/todo_list.php
```

Add the following content to your PHP script:

```
<?php
$user = "example_user";
$password = "password";
```

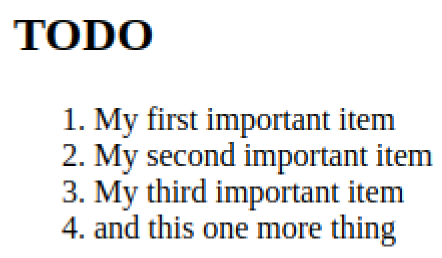
```
$database = "example_database";
$table = "todo_list";

try {
  $db = new PDO("mysql:host=localhost;dbname=$database", $user, $password);
  echo "<h2>TODO</h2><ol>";
  foreach($db->query("SELECT content FROM $table") as $row) {
    echo "<li>" . $row['content'] . "</li>";
  }
  echo "</ol>";
} catch (PDOException $e) {
    print "Error!: " . $e->getMessage() . "<br/>";
    die();
}
```

Save and close the file when you’re done editing.

You can now access this page in your web browser by visiting the domain name or public IP address you’ve set up in your Nginx configuration file, followed by `/todo_list.php`:

```
http://your_domain/todo_list.php
```

You should see a page like this, showing the content you’ve inserted in your test table:



TODO

1. My first important item
2. My second important item
3. My third important item
4. and this one more thing

That means your PHP environment is ready to connect and interact with your MariaDB server.

Conclusion

In this guide, you’ve built a flexible foundation for serving PHP websites and applications to your visitors, using Nginx as web server. You’ve set up Nginx to handle PHP requests through `php-fpm`, and you also set up a MariaDB database to store your website’s data.

To further improve your current setup, you can install Composer for dependency and package management in PHP, and you can also install an OpenSSL certificate for your website using Let’s Encrypt.

https://www.digitalocean.com/community/tutorials/how-to-install-linux-nginx-mariadb-php-lemp-stack-on-debian-10

How to Install LEMP Stack on Debian 10 Buster Server/Desktop

Last Updated: August 10, 2019 Xiao Guoan (Admin)
3 Comments
Debian

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.

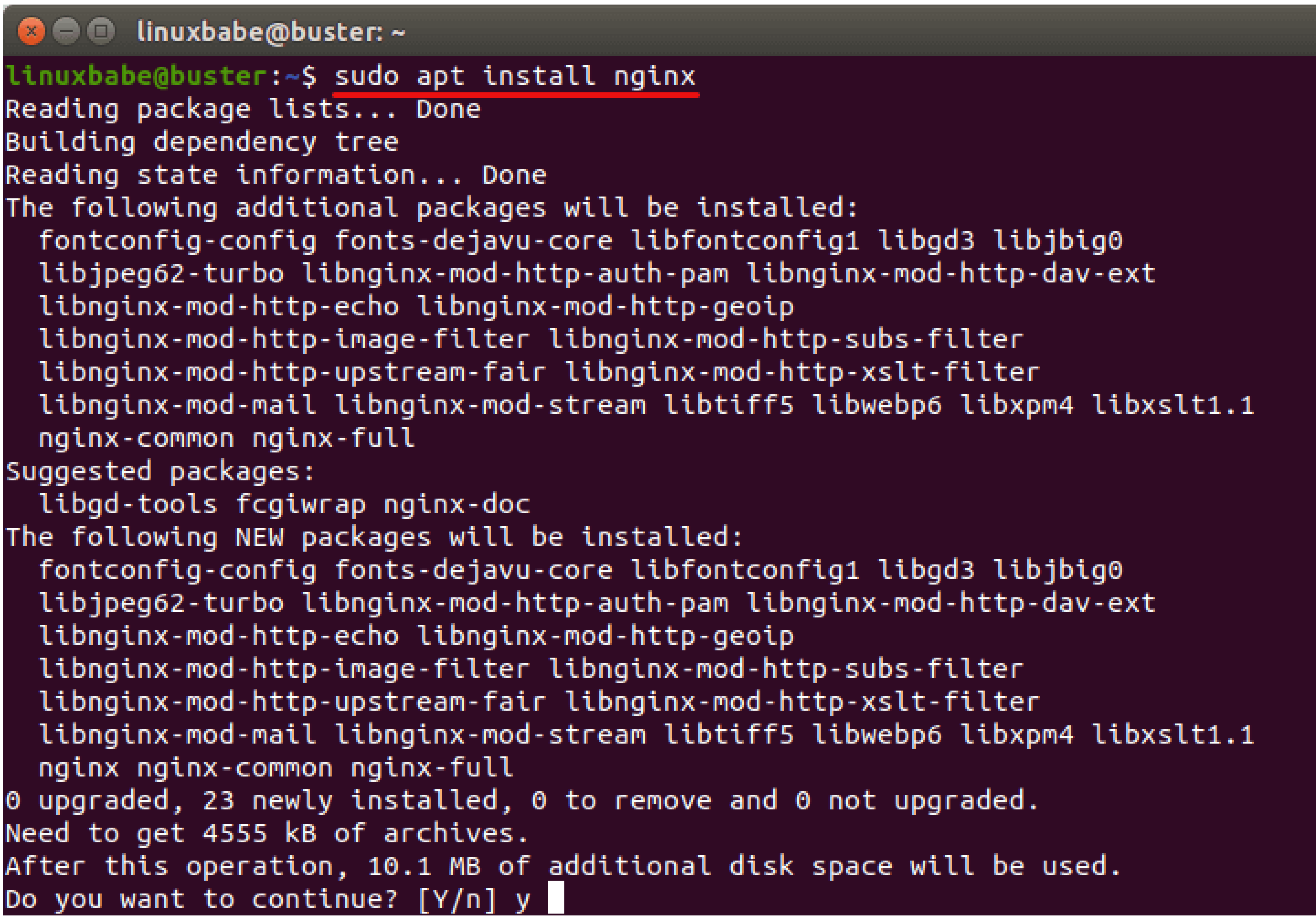
```
sudo apt update
```

```
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
```



```
linuxbabe@buster:~$ sudo apt install nginx
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  fontconfig-config fonts-dejavu-core libfontconfig1 libgd3 libjbig0
  libjpeg62-turbo libnginx-mod-http-auth-pam libnginx-mod-http-dav-ext
  libnginx-mod-http-echo libnginx-mod-http-geoip
  libnginx-mod-http-image-filter libnginx-mod-http-subs-filter
  libnginx-mod-http-upstream-fair libnginx-mod-http-xslt-filter
  libnginx-mod-mail libnginx-mod-stream libtiff5 libwebp6 libxpm4 libxslt1.1
  nginx-common nginx-full
Suggested packages:
  libgd-tools fcgiwrap nginx-doc
The following NEW packages will be installed:
  fontconfig-config fonts-dejavu-core libfontconfig1 libgd3 libjbig0
  libjpeg62-turbo libnginx-mod-http-auth-pam libnginx-mod-http-dav-ext
  libnginx-mod-http-echo libnginx-mod-http-geoip
  libnginx-mod-http-image-filter libnginx-mod-http-subs-filter
  libnginx-mod-http-upstream-fair libnginx-mod-http-xslt-filter
  libnginx-mod-mail libnginx-mod-stream libtiff5 libwebp6 libxpm4 libxslt1.1
  nginx nginx-common nginx-full
0 upgraded, 23 newly installed, 0 to remove and 0 not upgraded.
Need to get 4555 kB of archives.
After this operation, 10.1 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

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
```

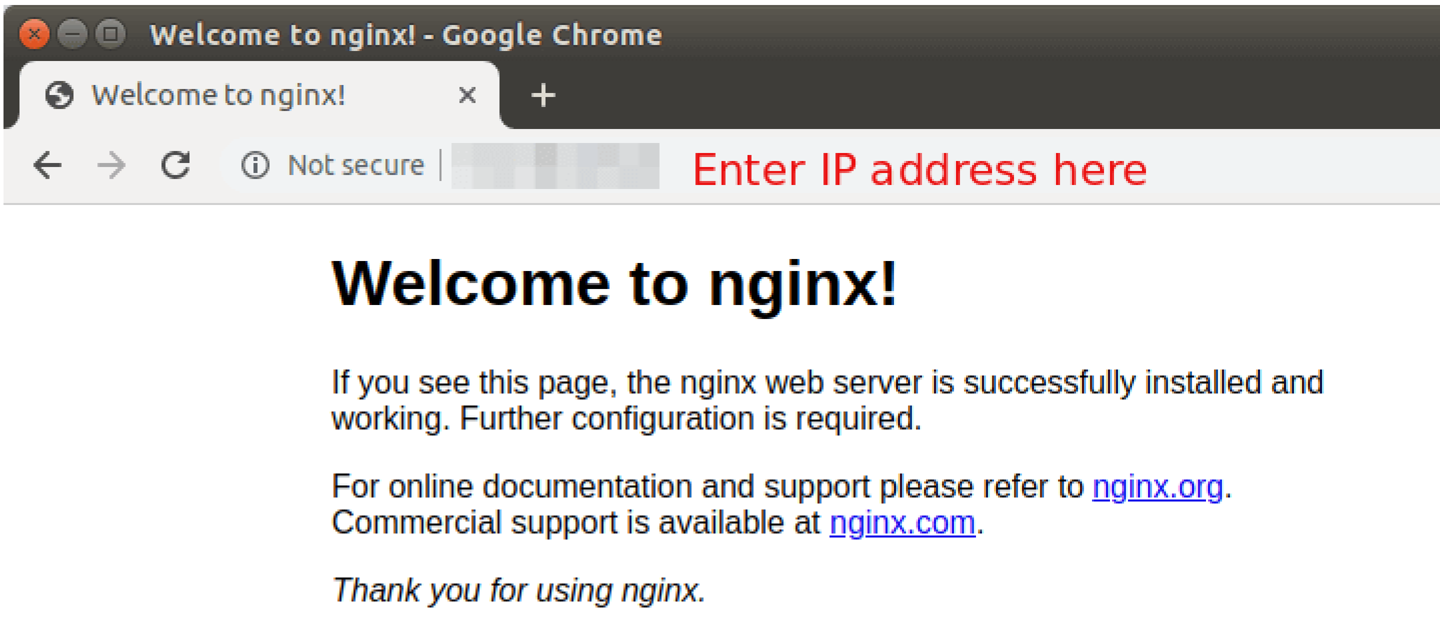
Check Nginx version:

```
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.



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

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
```

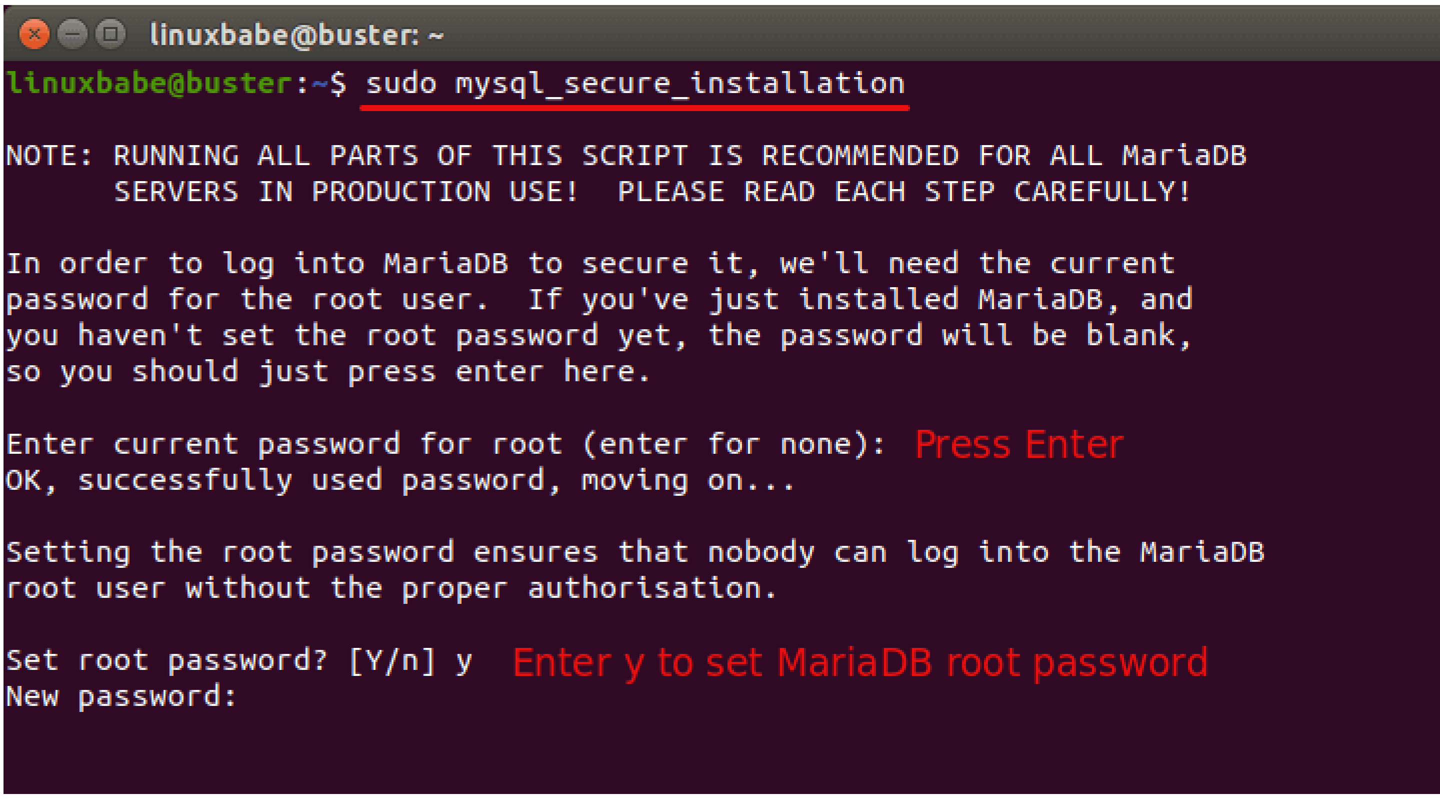
To enable MariaDB to automatically start at boot time, run

```
sudo systemctl enable mariadb
```

Now run the post installation security script.

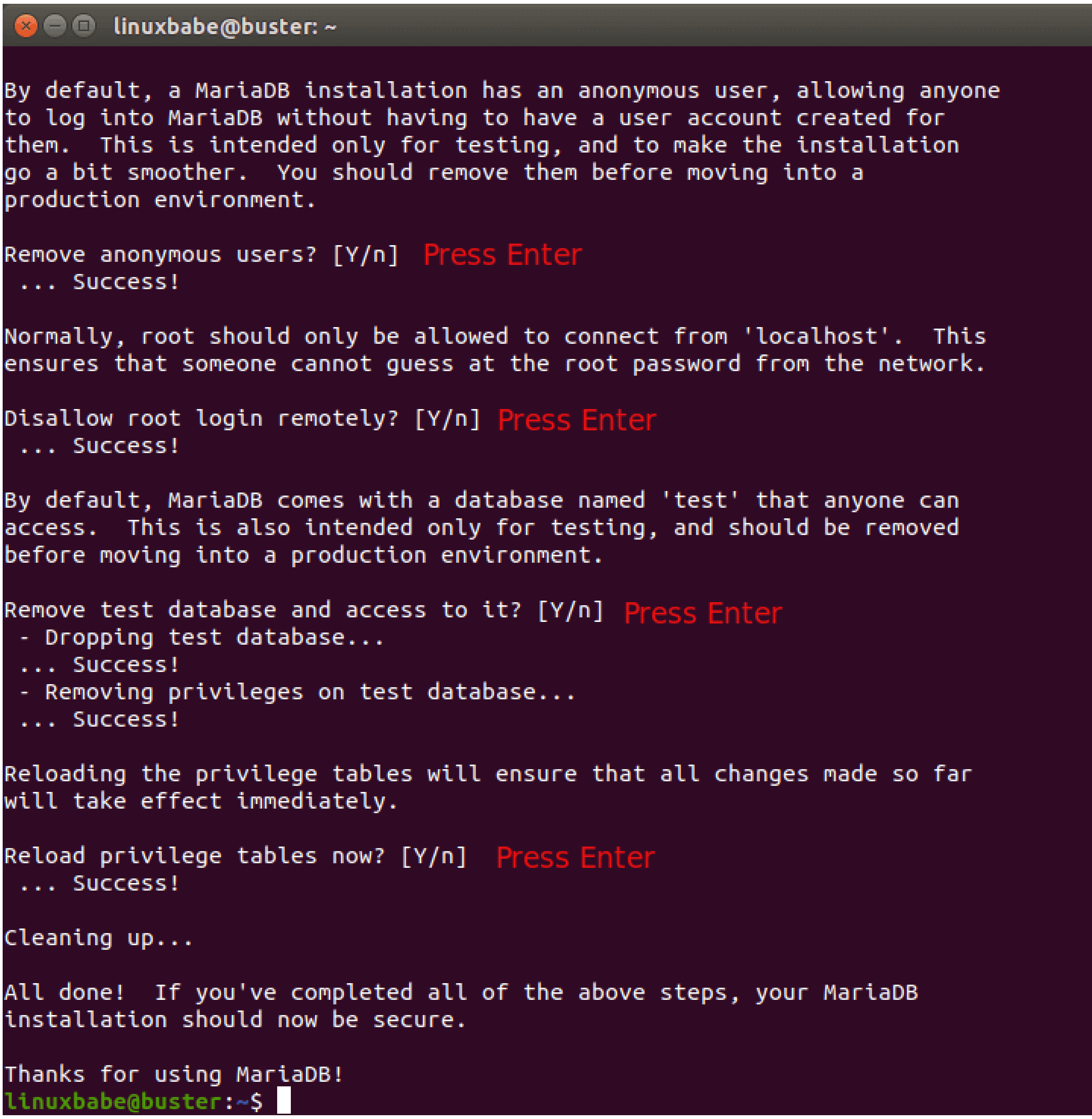
```
sudo mysql_secure_installation
```

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.



```
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):
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
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.)



```
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 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
```

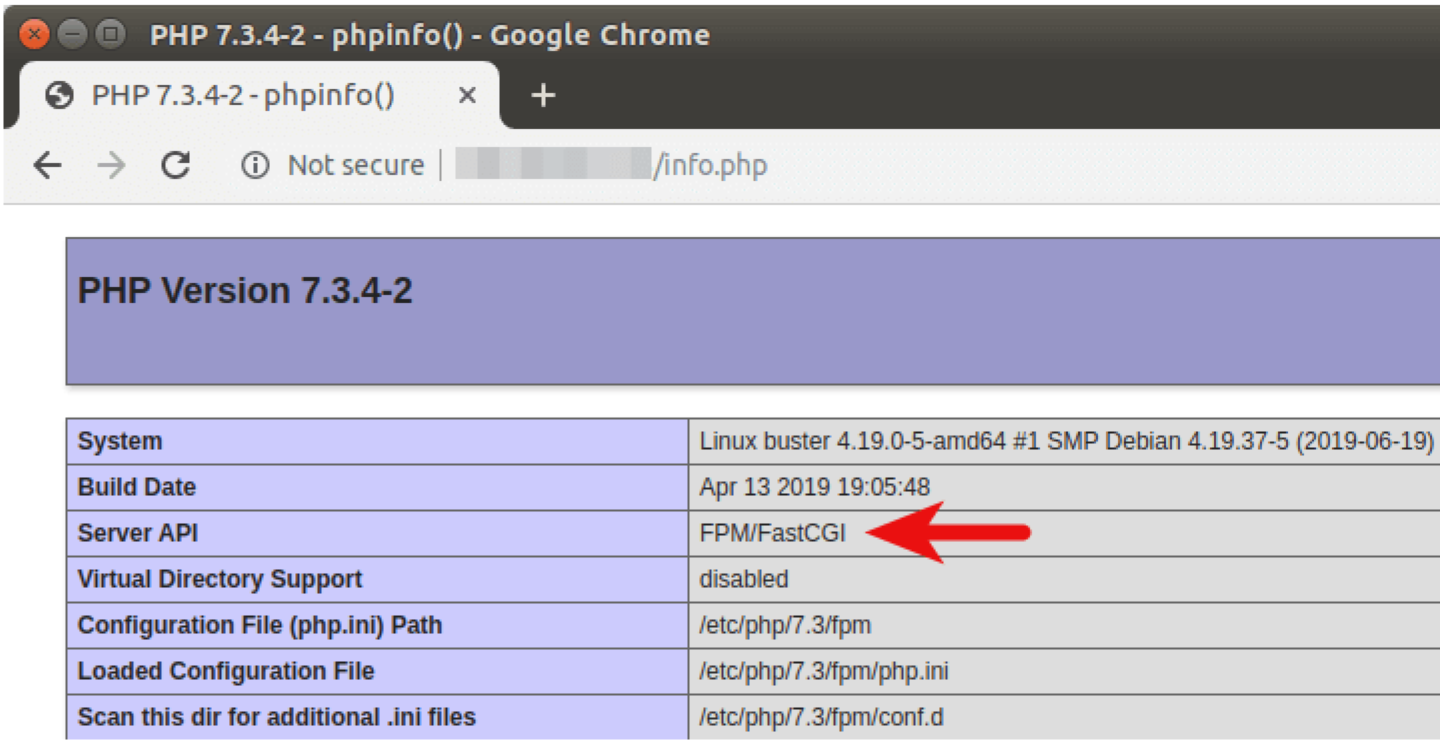
Paste the following PHP code into the file.

```
<?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.



PHP Version 7.3.4-2

System	Linux buster 4.19.0-5-amd64 #1 SMP Debian 4.19.37-5 (2019-06-19)
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

Wrapping Up

Congrats! You have successfully installed Nginx, MariaDB and PHP7.3 on Debian 10 Buster. For your server's security, you should delete info.php file now to prevent prying eyes.

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

I hope this tutorial helped you **install LEMP stack on Debian 10 Buster**. As always, if you found this post useful, then subscribe to our free newsletter to get new tutorials. Take care

https://www.linuxbabe.com/debian/install-lemp-stack-debian-10-buster