How To Install Linux, Nginx, MySQL, PHP (LEMP stack) on Debian 9

5

Posted September 13, 2018 18.7k LEMP NGINX MYSQL DEBIAN DEBIAN 9

By: Brian Hogan By: Brian Boucheron

Not using **Debian 9**? Choose a different version:

CentOS 7

Debian 8

Ubuntu 18.04

Introduction

The LEMP software stack is a group of software that can be used to serve dynamic web pages and web applications. This is an acronym that describes a Linux operating system, with an Nginx web server. The backend data is stored in the MySQL database and the dynamic processing is handled by PHP.

In this guide, you'll install a LEMP stack on a Debian server using the packages provided by the operating system.

Prerequisites

To complete this guide, you will need a Debian 9 server with a non-root user with `sudo` privileges. You can set up a user with these privileges in our Initial Server Setup with Debian 9 guide.

Step 1 — Installing the Nginx Web Server

In order to display web pages to our site visitors, we are going to employ Nginx, a modern, efficient web server.

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

Since this is our first time using `apt` for this session, we should start off by updating our local package index. We can then install the server:

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

On Debian 9, 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 we haven't configured SSL for our server yet, in this guide, we will only need to allow 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)
```

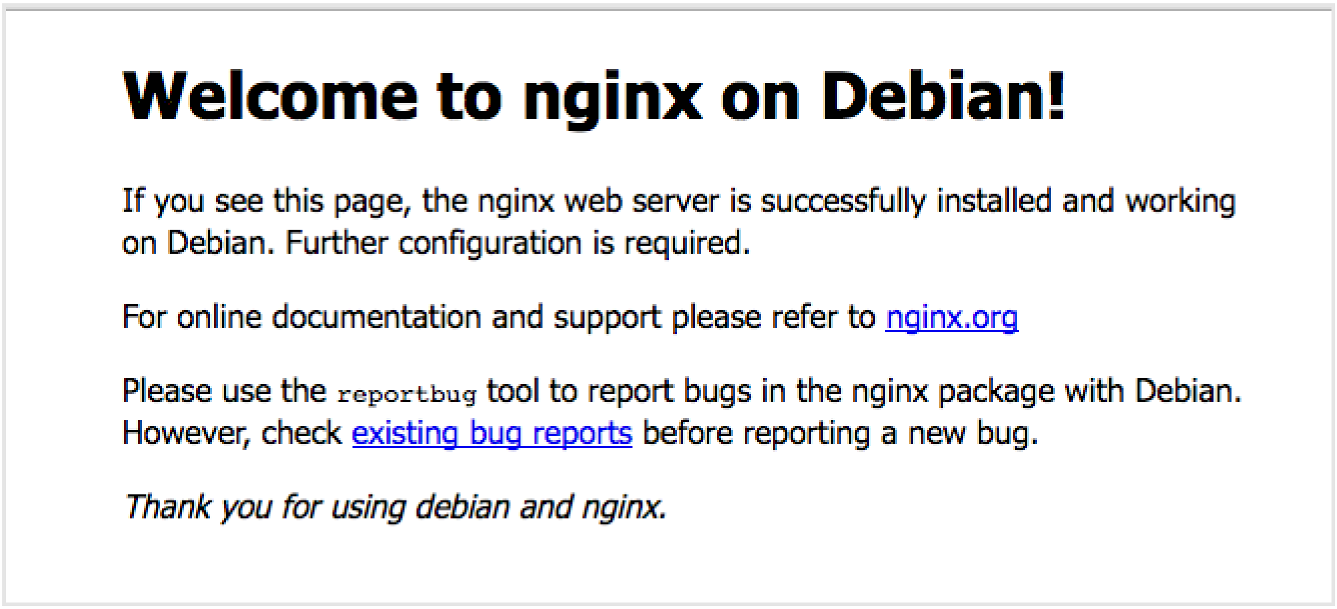
Now, test if the server is up and running by accessing your server's domain name or public IP address in your web browser. If you do not have a domain name pointed at your server and you do not know your server's public IP address, you can find it by typing one of the following into your terminal:

```
$ 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 MySQL to Manage Site Data

Now that we have a web server, we need to install MySQL, a database management system, to store and manage the data for our site.

You can install this easily by typing:

```
$ sudo apt install mysql-server
```

Note: In Debian 9 a community fork of the MySQL project – MariaDB – is packaged as the default MySQL variant. While, MariaDB works well in most cases, if you need features found only in Oracle's MySQL, you can install and use packages from a repository maintained by the MySQL developers. To install the official MySQL server, use our tutorial *How To Install the Latest MySQL on Debian 9*.

The MySQL database software is now installed, but its configuration is not complete.

To secure the installation, we can run a security script that will ask whether we want to modify some insecure defaults. Begin the script by typing:

```
$ sudo mysql_secure_installation
```

You will be asked to enter the password for the MySQL **root** account. We haven't set this yet, so just hit `ENTER`. Then you'll be asked you if you want to set that password. You should type `y` then set a **root** password.

For the rest of the questions the script asks, you should press `y`, followed by the `ENTER` key at each prompt. This will remove some anonymous users and the test database, disable remote root logins, and load these new rules so that MySQL immediately respects the changes you have made.

At this point, your database system is now set up and secured. Let's set up PHP.

Step 3 — Installing PHP for Processing

We now have Nginx installed to serve our pages and MySQL installed to store and manage our data. However, we still don't have anything that can generate dynamic content. That's where PHP comes in.

Since Nginx does not contain native PHP processing like some other web servers, we will need to install `fpm`, which stands for "fastCGI process manager". We will tell Nginx to pass PHP requests to this software for processing. We'll also install an additional helper package that will allow PHP to communicate with our MySQL database backend. The installation will pull in the necessary PHP core files to make that work.

Then install the `php-fpm` and `php-mysql` packages:

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

We now have our PHP components installed. Next we'll configure Nginx to use them.

Step 4 — Configuring Nginx to Use the PHP Processor

Now we have all of the required components installed. The only configuration change we still need is to tell Nginx to use our PHP processor for dynamic content.

We do this on the server block level (server blocks are similar to Apache's virtual hosts). We're going to leave the default Nginx configuration alone and instead create a new configuration file and new web root directory to hold our PHP files. We'll name the configuration file and the directory after the domain name or hostname that the server should respond to.

First, create a new directory in `/var/www` to hold the PHP site:

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

Then, open a new configuration file in Nginx's `sites-available` directory:

```
$ 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/php7.0-fpm.sock;
    }
}
```

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

Save and close the file when you're done customizing it.

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

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

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

```
$ sudo nginx -t
```

If any errors are reported, go back and recheck your file before continuing.

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

```
$ sudo systemctl reload nginx
```

Next we'll create a file in our new web root directory to test out PHP processing.

Step 5 — Create a PHP File to Test Configuration

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

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

```
$ sudo 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 our server:

/var/www/your_domain/info.php

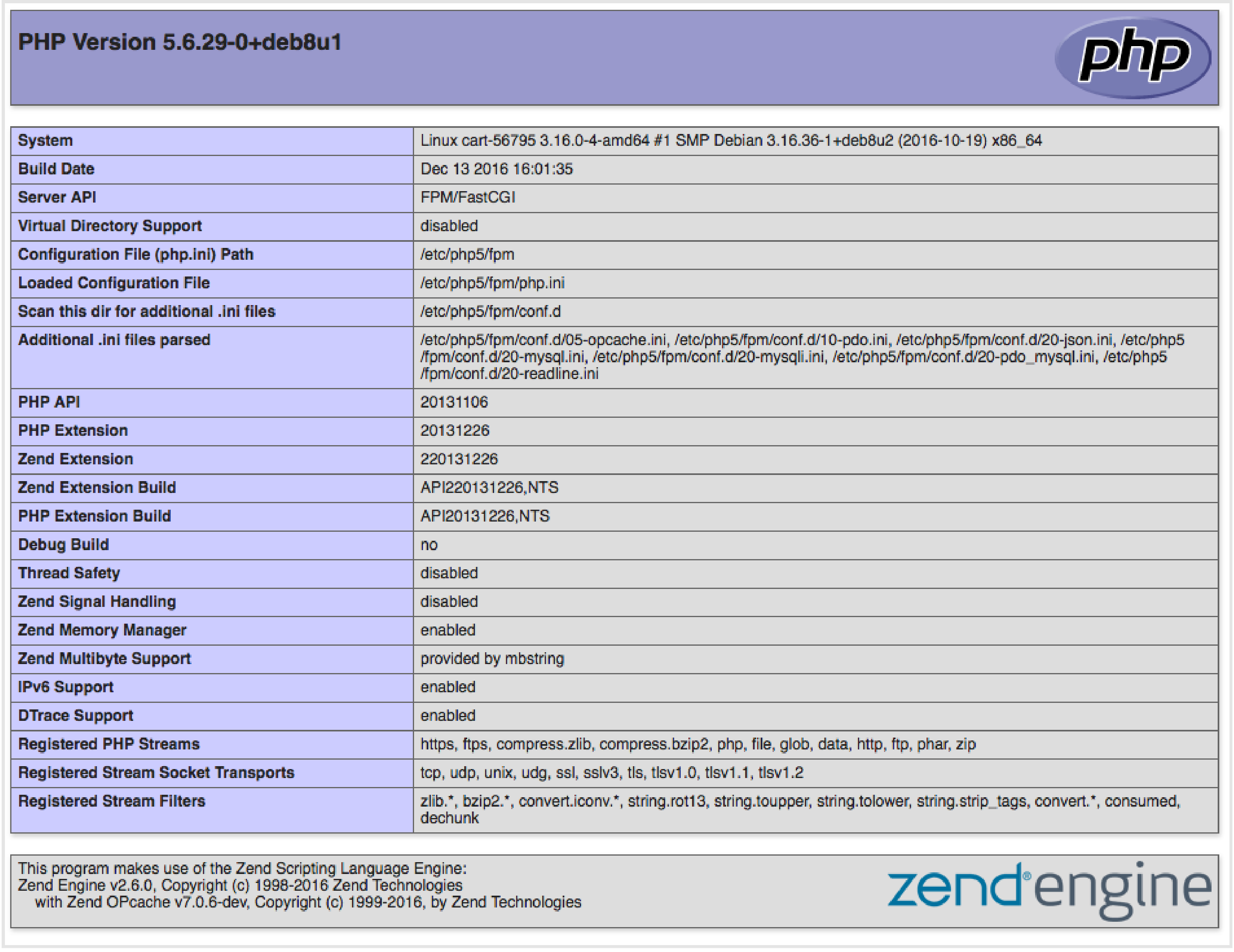
```
<?php
  phpinfo();
?>
```

When you are finished, save and close the file.

Now, you can visit this page in your web browser by visiting your server's domain name or public IP address followed by `/info.php`:

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

You should see a web page that has been generated by PHP with information about your server:



PHP Version 5.6.29-0+deb8u1

System	Linux cart-56795 3.16.0-4-amd64 #1 SMP Debian 3.16.36-1+deb8u2 (2016-10-19) x86_64
Build Date	Dec 13 2016 16:01:35
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/fpm
Loaded Configuration File	/etc/php5/fpm/php.ini
Scan this dir for additional .ini files	/etc/php5/fpm/conf.d
Additional .ini files parsed	/etc/php5/fpm/conf.d/05-opcache.ini, /etc/php5/fpm/conf.d/10-pdo.ini, /etc/php5/fpm/conf.d/20-json.ini, /etc/php5/fpm/conf.d/20-mysql.ini, /etc/php5/fpm/conf.d/20-mysqli.ini, /etc/php5/fpm/conf.d/20-pdo_mysql.ini, /etc/php5/fpm/conf.d/20-readline.ini
PHP API	20131106
PHP Extension	20131226
Zend Extension	220131226
Zend Extension Build	API220131226,NTS
PHP Extension Build	API20131226,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	enabled
Registered PHP Streams	https, ftps, compress.zlib, compress.bzip2, php, file, glob, data, http, ftp, phar, zip
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, sslv3, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	zlib.*, bzip2.*, convert.iconv.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk

This program makes use of the Zend Scripting Language Engine:
Zend Engine v2.6.0, Copyright (c) 1998-2016 Zend Technologies
with Zend OPcache v7.0.6-dev, Copyright (c) 1999-2016, by Zend Technologies

If you see a page that looks like this, you've set up PHP processing with Nginx successfully.

After verifying that Nginx renders the page correctly, it's best to remove the file you created as it can actually give unauthorized users some hints about your configuration that may help them try to break in.

For now, remove the file by typing:

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

You can always regenerate this file if you need it later.

Conclusion

You should now have a LEMP stack configured on your Debian server. This gives you a very flexible foundation for serving web content to your visitors.

By: Brian Hogan   By: Brian Boucheron   Upvote (5)   Subscribe   Share

Related Tutorials

How To Deploy a PHP Application with Kubernetes on Ubuntu 16.04

How To Ensure Code Quality with SonarQube on Ubuntu 18.04

How To Set Up a Private Docker Registry on Ubuntu 18.04

How To Sync and Share Your Files with Seafile on Ubuntu 18.04

How to Set Up an Nginx Ingress with Cert-Manager on DigitalOcean Kubernetes

8 Comments

Leave a comment...

Log In to Comment

NancyMGonzales93 *September 18, 2018*

0 I don't think I understand it correctly. Trying to set thing up for a gaming system at 7sultans. Almost done, but having some troubles with Node.js. Actually, I did get it, thank to your guide but still... have no clue what that

line means.

sazzbot *September 22, 2018*

0 "This is a very basic configuration that listens on port 80 and serves files from the web root we just created. It will only respond to requests to the name provided after server_name, and any files ending in .php will be processed by the php-fpm process before Nginx sends the results to the user.

Save and close the file when you're done customizing it."

Everything worked great until I got to this part.
"Save and close the file..."

I was not able to do that. Save it or close it. I am looking through nginx info and even things like :x and :wq which may save it, but the console shows the same thing with a list of commands using up arrow and a letter. None of them said save and exit and when I tried any of them, they did nothing anyway..
Save and close! It sounds SO easy!

felipespamspam *October 24, 2018*

0 The "up arrow" (caret) is a symbol for Ctrl key, so for example "^X" means Ctrl+X in the context of the nano text editor.

Marioqueirozjr *October 6, 2018*

0 404 Not Found
nginx/1.10.3

felipespamspam *October 24, 2018*

0 I had a similar but different problem when trying to access my website from the local network using my no-ip domain name after setting up my router port forwarding.

Outside my network using domain name: works.
Outside my network using external ip: works.
Inside my network using local ip: works.
Inside my network using domain name: does not work.
Outside my network using external ip: does not work.

I think in my case the HTTP server of my router is overriding some local address. The router has an internal HTTP server used for its admin web interface.

felipespamspam *October 24, 2018*

0 Also another problem I had was mistaking the `info.php` file (used in the tutorial) with the very commonly used `index.php`. In my case this file didn't exist because it simply wasn't created. When trying to load `mydomain/index.php` I got exactly the same 404 message as you:

```
404 Not Found
```

```
404 Not Found
nginx/1.10.3
```

But when loading `mydomain/info.php` it worked fine just like in this tutorial.

---

brocroft *December 13, 2018*

2 Good tutorial, thanks.

There's a typo in the symlink part. ".conf" shouldn't be in the file name there, otherwise it will create a symlink to nowhere, as we didn't use ".conf" in the original sites-available one.

---

leonardorh *January 17, 2019*

0 Someone else mentioned it here, I am pretty sure you don't need the .conf format in sites-enabled. I followed the tutorial and I could not get nginx to read the config file, so I just linked it with no .conf and it worked.