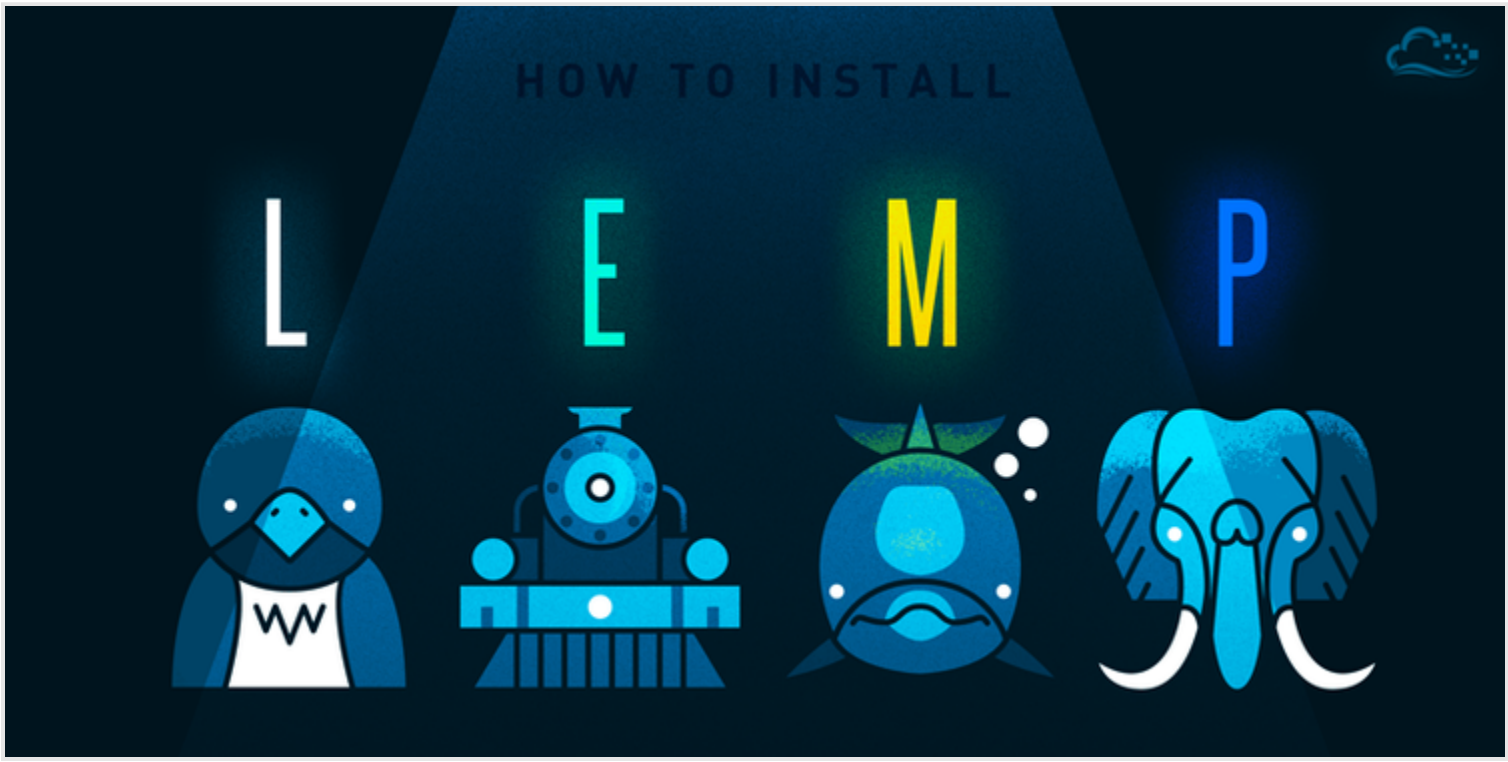


How To Install Linux, Nginx, MySQL, PHP (LEMP stack) in Ubuntu 16.04

Posted April 21, 2016 896.7k LEMP MYSQL PHP NGINX UBUNTU UBUNTU 16.04

By: Justin Ellingwood

Not using **Ubuntu 16.04**? Choose a different version:

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, we will demonstrate how to install a LEMP stack on an Ubuntu 16.04 server. The Ubuntu operating system takes care of the first requirement. We will describe how to get the rest of the components up and running.

Prerequisites

Before you complete this tutorial, you should have a regular, non-root user account on your server with `sudo` privileges. You can learn how to set up this type of account by completing our Ubuntu 16.04 initial server setup.

Once you have your user available, sign into your server with that username. You are now ready to begin the steps outlined in this guide.

Step 1: Install 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 Ubuntu'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-get update
$ sudo apt-get install nginx
```

On Ubuntu 16.04, Nginx is configured to start running upon installation.

If you have the `ufw` firewall running, as outlined in our initial setup guide, you will need to allow connections to Nginx. Nginx registers itself with `ufw` upon installation, so the procedure is rather straight forward.

It is recommended that you 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)
```

With the new firewall rule added, you can 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.

As an alternative, you can check which IP address is accessible as viewed from other locations on the internet:

```
$ curl -4 icanhazip.com
```

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

```
http://server_domain_or_IP
```

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

Step 2: Install 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-get install mysql-server
```

You will be asked to supply a root (administrative) password for use within the MySQL system.

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

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

```
$ mysql_secure_installation
```

You will be asked to enter the password you set for the MySQL root account. Next, you will be asked if you want to configure the `VALIDATE PASSWORD PLUGIN`.

> **Warning:** Enabling this feature is something of a judgment call. If enabled, passwords which don't match the specified criteria will be rejected by MySQL with an error. This will cause issues if you use a weak password in conjunction with software which automatically configures MySQL user credentials, such as the Ubuntu packages for phpMyAdmin. It is safe to leave validation disabled, but you should always use strong, unique passwords for database credentials.

Answer **y** for yes, or anything else to continue without enabling.

```
VALIDATE PASSWORD PLUGIN can be used to test passwords
and improve security. It checks the strength of password
and allows the users to set only those passwords which are
secure enough. Would you like to setup VALIDATE PASSWORD plugin?

Press y|Y for Yes, any other key for No:
```

If you've enabled validation, you'll be asked to select a level of password validation. Keep in mind that if you enter **2**, for the strongest level, you will receive errors when attempting to set any password which does not contain numbers, upper and lowercase letters, and special characters, or which is based on common dictionary words.

```
There are three levels of password validation policy:

LOW    Length >= 8
MEDIUM Length >= 8, numeric, mixed case, and special characters
STRONG Length >= 8, numeric, mixed case, special characters and dictionary                  file
```

```
STRONG Length >= 8, numeric, mixed case, special characters and dictionary                  file

Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG: 1
```

If you enabled password validation, you'll be shown a password strength for the existing root password, and asked you if you want to change that password. If you are happy with your current password, enter **n** for "no" at the prompt:

```
Using existing password for root.

Estimated strength of the password: 100
Change the password for root ? ((Press y|Y for Yes, any other key for No) : n
```

For the rest of the questions, you should press **Y** and hit 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 we have made.

At this point, your database system is now set up and we can move on.

Step 3: Install 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. We can use PHP for this.

Since Nginx does not contain native PHP processing like some other web servers, we will need to install `php-fpm`, which stands for "fastCGI process manager". We will tell Nginx to pass PHP requests to this software for processing.

We can install this module and will also grab an additional helper package that will allow PHP to communicate with our database backend. The installation will pull in the necessary PHP core files. Do this by typing:

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

Configure the PHP Processor

We now have our PHP components installed, but we need to make a slight configuration change to make our setup more secure.

Open the main `php-fpm` configuration file with root privileges:

```
$ sudo nano /etc/php/7.0/fpm/php.ini
```

What we are looking for in this file is the parameter that sets `cgi.fix_pathinfo`. This w
out with a semi-colon (;) and set to "1" by default.

This is an extremely insecure setting because it tells PHP to attempt to execute the closest file it can find if the requested PHP file cannot be found. This basically would allow users to craft PHP requests in a way that would allow them to execute scripts that they shouldn't be allowed to execute.

We will change both of these conditions by uncommenting the line and setting it to "0" like this:

/etc/php/7.0/fpm/php.ini

```
cgi.fix_pathinfo=0
```

Save and close the file when you are finished.

Now, we just need to restart our PHP processor by typing:

```
$ sudo systemctl restart php7.0-fpm
```

This will implement the change that we made.

Step 4: Configure 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). Open the default Nginx server block configuration file by typing:

```
$ sudo nano /etc/nginx/sites-available/default
```

Currently, with the comments removed, the Nginx default server block file looks like this:

/etc/nginx/sites-available/default

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

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

    server_name _;

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

We need to make some changes to this file for our site.

- First, we need to add `index.php` as the first value of our `index` directive so that files named `index.php` are served, if available, when a directory is requested.
- We can modify the `server_name` directive to point to our server's domain name or public IP address.
- For the actual PHP processing, we just need to uncomment a segment of the file that handles PHP requests by removing the pound symbols (#) from in front of each line. This will be the `location ~\.php$` location block, the included `fastcgi-php.conf` snippet, and the socket associated with `php-fpm`.
- We will also uncomment the location block dealing with `.htaccess` files using the same method. Nginx doesn't process these files. If any of these files happen to find their way into the document root, they should not be served to visitors.

The changes that you need to make are in red in the text below:

/etc/nginx/sites-available/default

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

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

    server_name server_domain_or_IP;

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

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/run/php/php7.0-fpm.sock;
    }

    location ~ /\.ht {
        deny all;
    }
}
```

When you've made the above changes, you can save and close the file.

Test your configuration file 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 necessary changes:

```
$ sudo systemctl reload nginx
```

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/html/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/html/info.php

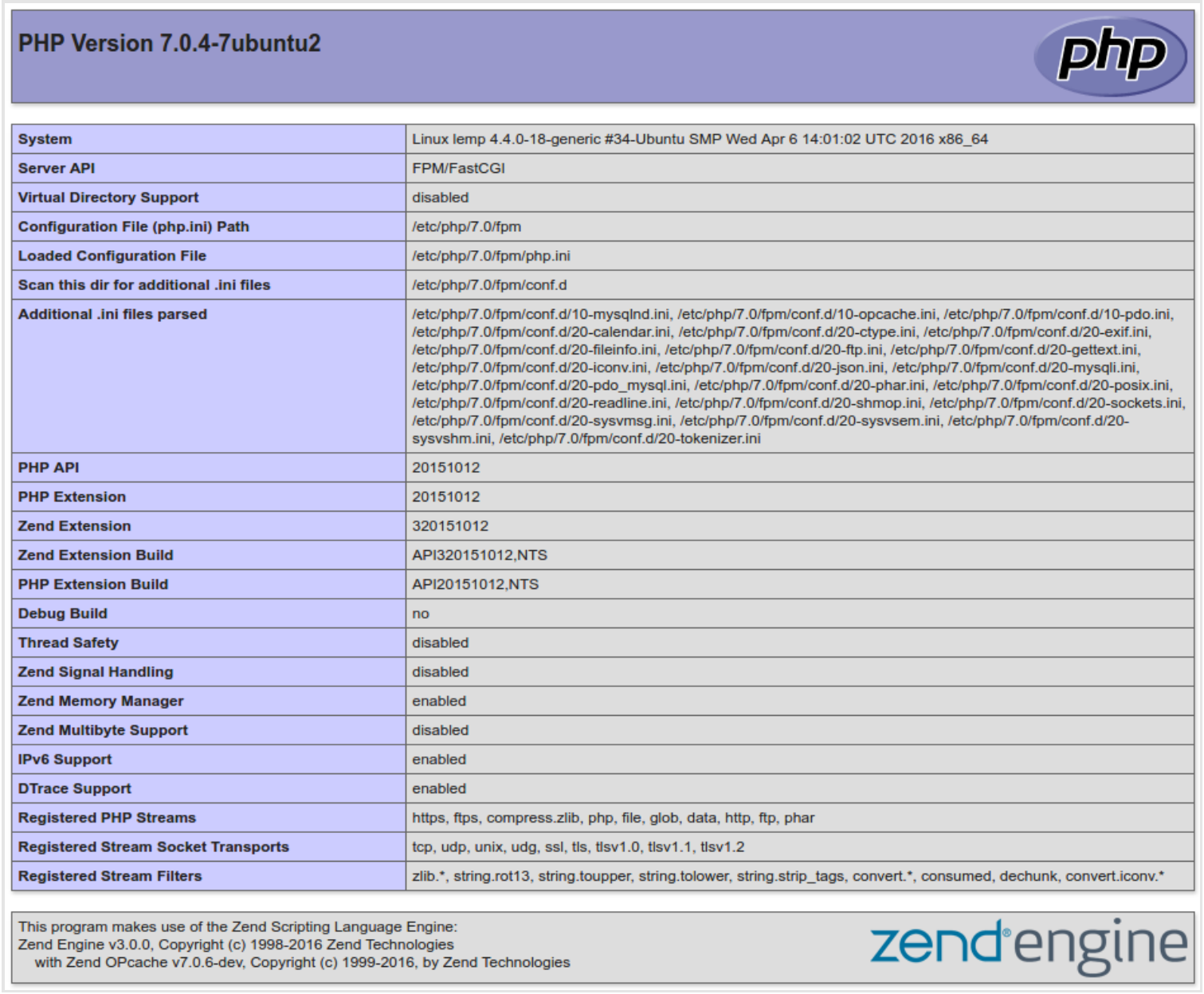
```
<?php
phpinfo();
```

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://server_domain_or_IP/info.php
```

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



PHP Version 7.0.4-7ubuntu2

System	Linux lemp 4.4.0-18-generic #34-Ubuntu SMP Wed Apr 6 14:01:02 UTC 2016 x86_64
Server API	FPM/FastCGI
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.0/fpm
Loaded Configuration File	/etc/php/7.0/fpm/php.ini
Scan this dir for additional .ini files	/etc/php/7.0/fpm/conf.d
Additional .ini files parsed	/etc/php/7.0/fpm/conf.d/10-mysqlnd.ini, /etc/php/7.0/fpm/conf.d/10-opcache.ini, /etc/php/7.0/fpm/conf.d/10-pdo.ini, /etc/php/7.0/fpm/conf.d/20-calendar.ini, /etc/php/7.0/fpm/conf.d/20-ctype.ini, /etc/php/7.0/fpm/conf.d/20-exif.ini, /etc/php/7.0/fpm/conf.d/20-fileinfo.ini, /etc/php/7.0/fpm/conf.d/20-ftp.ini, /etc/php/7.0/fpm/conf.d/20-gettext.ini, /etc/php/7.0/fpm/conf.d/20-iconv.ini, /etc/php/7.0/fpm/conf.d/20-json.ini, /etc/php/7.0/fpm/conf.d/20-mysqli.ini, /etc/php/7.0/fpm/conf.d/20-pdo_mysql.ini, /etc/php/7.0/fpm/conf.d/20-phar.ini, /etc/php/7.0/fpm/conf.d/20-posix.ini, /etc/php/7.0/fpm/conf.d/20-readline.ini, /etc/php/7.0/fpm/conf.d/20-shmop.ini, /etc/php/7.0/fpm/conf.d/20-sockets.ini, /etc/php/7.0/fpm/conf.d/20-sysvmsg.ini, /etc/php/7.0/fpm/conf.d/20-sysvsem.ini, /etc/php/7.0/fpm/conf.d/20-sysvshm.ini, /etc/php/7.0/fpm/conf.d/20-tokenizer.ini
PHP API	20151012
PHP Extension	20151012
Zend Extension	320151012
Zend Extension Build	API320151012,NTS
PHP Extension Build	API20151012,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	disabled
IPv6 Support	enabled
DTrace Support	enabled
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, convert.*, consumed, dechunk, convert.iconv.*

This program makes use of the Zend Scripting Language Engine:
Zend Engine v3.0.0, 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. You can always regenerate this file if you need it later.

For now, remove the file by typing:

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

Conclusion

You should now have a LEMP stack configured on your Ubuntu 16.04 server. This gives you a very flexible foundation for serving web content to your visitors.

Related Tutorials

How to Deploy a Symfony 4 Application to Production with LEMP on Ubuntu 18.04

How To Install WordPress with LEMP on Debian 9

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

How To Install WordPress with LEMP on Ubuntu 18.04

How To Install Linux, Nginx, MySQL, PHP (LEMP stack) on Ubuntu 18.04

97 Comments

Leave a comment...

Log In to Comment

chrispappas *April 24, 2016*

1

SCROLL TO TOP

Can someone suggest a cloud-config script for the above? If not, I will work on creating one as I think this would be useful and more tweakable than the DO-provided One-Click LEMP (not available yet as I'm commenting)

mirzazeyrek *June 19, 2016*

2 Use this one for one click install:

https://github.com/mirzazeyrek/lemp-wordpress-stack

chodhary *October 5, 2016*

0 I love this script. All you need is sit back and relax.

marlonlamancio *December 18, 2016*

0 Search for Easy Engine

sandagreens *November 29, 2017*

0 Check: https://github.com/santoshbaggam/stacker

iamkingsleyf *April 25, 2016*

0 Nice guide.... waiting for wordpress + LEMP

jellingwood MOD *April 25, 2016*

1 @iamkingsleyf: Good timing! I just published it here.

> **How To Install WordPress with LEMP on Ubuntu 16.04**
> by Justin Ellingwood
>
> WordPress is the most popular CMS (content management system) on the internet. It allows you to easily set up flexible blogs and websites on top of a MySQL backend with PHP processing. WordPress has seen incredible adoption and is a great choice for getting a website up and...

Coxxs *April 26, 2016*

0 Great guide!

iparr *April 27, 2016*

0 I'd be interested to hear how to use PHP 5.6 on 16.04.

Guides exist elsewhere online, but I'm unsure how using php-fpm with Nginx afterwards fits in. Will it work?

Maybe you'd recommend sticking 14.04 if PHP 5.x support is needed with Nginx?

---

redzwan81 *April 28, 2016*

0 I did upgrade from ubuntu 15, I run virtual host where previously I keep all site configuration at /etc/php5/fpm/pool.d/ , after upgrade I moved those configuration to /etc/php/7.0/fpm/pool.d/
But when I restart php7.0-fpm service, It loaded but inactive...

---

redzwan81 *April 28, 2016*

0 Result from service php7.0-fpm status
php7.0-fpm.service - The PHP 7.0 FastCGI Process Manager
Loaded: loaded (/lib/systemd/system/php7.0-fpm.service; enabled; vendor preset: enabled)
Active: inactive (dead) since Thu 2016-04-28 10:17:16 AEST; 4s ago
Process: 8774 ExecStart=/usr/sbin/php-fpm7.0 --nodaemonize --fpm-config /etc/php/7.0/fpm/php-fpm.conf (code
Process: 8764 ExecStartPre=/usr/lib/php/php7.0-fpm-checkconf (code=exited, status=0/SUCCESS)
Main PID: 8774 (code=exited, status=0/SUCCESS)

---

redzwan81 *April 28, 2016*

0 Problem solved, apparently not enough resource. Now I'm wondering if we can run php7.0 for certain apps, and php5.6 for other apps..

---

luismuzquiz *April 30, 2016*

2 Whats the difference between using:

```
fastcgi_pass unix:/run/php/php7.0-fpm.sock;
```

and

```
fastcgi_pass 127.0.0.1:9000;
```

on Nginx server blocks

Is one better than the other?

---

jellingwood MOD *May 12, 2016*

2 @luismuzquiz: Those lines specify how Nginx should try to connect to the backend pr
PHP-FPM).

The first line that you mention uses a Unix socket. This is basically a specialized file that let's applications on the same host pass data between one another. Since Unix sockets can only be used for communication within a single host, they're often considered to be superior to network communication from a security standpoint and also usually has a better performance profile. By default, Ubuntu's PHP-FPM installation is configured to use a Unix socket, so that is why we select that line.

The second line is a connecting over a network or internet socket. It is bound to the local loopback device, so it should only be accessible to other applications on the same host as well. Different dynamic processors are configured to listen on different sockets, ports, etc. out of the box, so this second line is configured in order to accommodate the default listening port of a different backend processor (PHP-CGI I think).

If you wanted to, you could configure these connections to be whatever ports or sockets you want, but the default server block file is set up to match the defaults of some of its more commonly used backend processors. Since we're using PHP-FPM in this guide, we use the Unix socket. Hope that helps!

---

luismuzquiz *May 13, 2016*

0 thanks 4 your explanation @jellingwood =)

---

marlonlamancio *December 19, 2016*

0 great and clear explanation and tutorial @jellingwood !! thank you! <3

---

luismuzquiz *May 1, 2016*

2 If you are trying to install JOOMLA or WP you will need to:

```
sudo apt-get install php7.0-xml
```

---

arijuki *May 6, 2016*

0 Good point! Toom me some time to figure out this, because at least Jetpack not working without. Also GD or Imagick needed for pics. Other tutorial proposes 'apt-get install php7.0-fpm php7.0-mysql php7.0-curl php7.0-gd php7.0-json php7.0-mcrypt php7.0-opcache php7.0-xml'. Not sure if all of these are needed or not?

---

luismuzquiz *May 6, 2016*

0 and if you a are using K2 2.7 you may need to install php7.0-gd so you can upload images through K2 component

---

mimrahe *May 13, 2016*

1

nice guide
when i want to see info.php nginx shows 502 Bad Gateway!
what is this and why there is?
how can i fix it?
thank for the help!

kashminder *May 29, 2016*

0 [deleted]

kashminder *May 29, 2016*

0 Please make sure that you have uncommented the line
fastcgi_pass unix:/run/php/php7.0-fpm.sock;
So the section should look like:

```
location ~ \.php$ {
        include snippets/fastcgi-php.conf;

    #   # With php7.0-cgi alone:
    #   fastcgi_pass 127.0.0.1:9000;
    #   # With php7.0-fpm:
      fastcgi_pass unix:/run/php/php7.0-fpm.sock;
```

gurabli *June 5, 2016*

2 I had this problem too, everything was configured properly. Well, not exactly. After reinstalling my Ubuntu Server 16.04 few times, and running several test on vbox, i finally discovered that for some strange and odd reason the php-fm listen address is different (it was perfect on vbox installs, doing the same on server it was not working). Check /etc/php/7.0/fpm/pool.d/www.conf file and look for the line "listen". In my case it is: listen = /run/php/php7.0-fpm.sock

Now if you look at the default nginx config for php, it is probably /var/run/php7.0-fpm.sock. Adjust this line to /run/php/php7.0-fpm.sock in nginx, restart nginx and there you go, you have a working php.

The default nginx config might be different since I was using the ppa stable nginx to install nginx, not the ubuntu repo.

behappyhappybee *June 12, 2016*

0 Thank you:)

Darmoktalk *May 16, 2016*

SCROLL TO TOP

0 Everything is working as it should after your tutorial. Thanks for the write-up!

---

tmacka *May 31, 2016*

0 Hi,

So I am having issues getting NGINX to work after installing it, when NOT using a domain name e.g. just use IP to access it. This issue is occurring both on a local Raspberry Pi and on a DO droplet.

First off, after I installed NGINX, i COULD see the "Welcome to NGINX" test page, but as soon as I made changes to sites-available by deleting default from sites-enabled I can get my own page to work.

Basically, I added my own root dir: /var/www/test - 755 for directories, 644 files test:test for user/group

In /etc/nginx/sites-available/test:

```
server {
        listen 80 default_server;

        root /var/www/test;

        # Add index.php to the list if you are using PHP
        index.html;

        server_name _;

        location / {
                # First attempt to serve request as file, then
                # as directory, then fall back to displaying a 404.
                try_files $uri $uri/ =404;
        }
}
```

I added this to sites-enabled.

But nothing shows up when I enter the droplets IP in. The same IP that gave me the "Welcome to NGINX" test page.

I have setup countless droplets successfully when using a domain name, but in this case I wish to not use a domain name at all and nothing is working.

Any ideas to get the web server to work without a using a domain?

Also I tried using `curl -I http://111.111.111.111`
I get:

```
curl: (7) Failed to connect to 111.111.111.111 port 80: Connection refused
```

tmacka *May 31, 2016*

0 figured it out:

Hey,

I just came accross an issue. Doesn't matter if its a local setup or one with a domain name.

When you create a symbolic frpom sites-available to sites-enabled you have to use the whole path to each location.

e.g. you can't

```
cd /etc/nginx/sites-available/
ln -s monitor ../sites-enabled/
```

It has to be:

```
ln -s  /etc/nginx/sites-available/monitor  /etc/nginx/sites-enabled/
```

jhhemal *June 1, 2016*

0 Thank you. It helps me a lot :)

Radic *June 25, 2016*

0 Stuck on http://mydomain.com/info.php thing.

My /etc/nginx/sites-available/default is:
server {
listen 80 default*server;*
*listen [::]:80 default*server;
root /var/www/html;
index index.php index.html index.htm index.nginx-debian.html;
server*name mydomain.com;*
*return 301 https://$server*name$request*uri;*
location / {
*try*files $uri $uri/ =404;
}
location ~ .php$ {
include snippets/fastcgi-php.conf;
fastcgi_pass unix:/run/php/php7.0-fpm.sock;
}
location ~ /.ht {
deny all;
}

location ~ /.well-known {
allow all;
}
}

server {
listen 443 ssl http2 default*server;*
*listen [::]:443 ssl http2 default*server;
include snippets/ssl-mydomain.com.conf;
include snippets/ssl-params.conf;
}

When trying to connect https://mydomain.com/info.php get "404 Not Found", but log show me:
[error]: *4 open() "/usr/share/nginx/html/info.php" failed (2: No such file or directory)

Seems like nginx treats /usr/share/nginx/html as documents root directory instead of /var/www/html :(

---

dariobratic *September 16, 2016*

0 Same problem. Did you find solution?

---

samuellavoie *July 10, 2016*

2 Wondering why you do not suggest modifying the `Unix user/group of processes` and changing the default *www-data* to the user credentials created on Ubuntu?

```
user = www-data
group = www-data
```

---

georgetour *August 8, 2016*

0 Great article.Seeing your first page in a VPS it's so NICE!

Jervisbay *August 9, 2016*

0 Hi Justin,

What an excellent guide. Thanks for putting the right level of details in it.
In following the steps I was able to get my LEMP stack up and running.

Keep up the good work.

---

chris769852 *August 30, 2016*

0 Hats off to you guys for providing great tutorials like this! Thanks for it!

paulg *September 13, 2016*

0 Has anyone been able to come up with a caching strategy for Ubunt 16.04, Nginx 1.10 & PHP 7? Seems like every guide online is for Ubuntu 14.04, Nginx 1.8 and PHP 5.X.

Have gone through about 10 different guides, but none give full object caching, full page caching, and the ability to clear the cache, ideally via the Wordpress Nginx Helper plugin (which is maintained by rtCamp, who, as far as I'm aware, has all of his scripts setup for old ubuntu, old nginx and old php).

Trying to get all of this working with HTTP/2, which means I can't run Ubuntu 14.04 (due to some requirements from Chrome), or Nginx 1.8, as it does not support HTTP/2 (far as I'm aware).

So, some guidance on getting something in place for these newer versions with HTTP/2 would be extraordinarily helpful, and much appreciated.

Lilap *September 20, 2016*

0 Hello,

I would like to know if you have ever considered MariaDB over Mysql? Or would you stick with Mysql?

Just wanting to know as I am considering changing to MariaDB as I heard it was a lot faster than Mysql?

Thanks

Load More Comments