

Nginx



- [Generic and Multi-Site Support](#)
 - [Main \(generic\) startup file](#)
 - [Per Site configuration](#)
 - [Global restrictions file](#)
 - [General WordPress rules](#)
 - [WordPress Multisite](#)
 - [WordPress 3.5 and up](#)
 - [WordPress 3.5 and up Subdirectory Examples](#)
 - [WordPress 3.5 and up Subdomains Examples](#)
 - [WordPress 3.4 and below](#)
 - [WordPress <=3.4 Subdirectory Examples](#)
 - [WordPress <=3.4 Subdomains Examples](#)
 - [HTTPS in Nginx](#)
 - [WP Super Cache Rules](#)
 - [W3 Total Cache Rules](#)
- [Nginx fastcgi_cache](#)
- [Better Performance for Static Files in Multisite \(WP <= 3.4\)](#)
- [Warning](#)
- [Resources](#)
 - [Reference](#)
 - [External Links](#)
 - [Scripts & Tools](#)
 - [Securing Nginx](#)

While the LAMP stack (Linux + Apache + MySQL + PHP) is very popular for powering WordPress, it is also possible to use Nginx. WordPress supports Nginx, and some large WordPress sites, such as WordPress.com, are powered by Nginx.

When talking about Nginx, it is important to know that there are multiple ways to implement Nginx. It can be setup as a reverse-proxy in front of Apache, which is a very powerful setup that allows

you to use all of the features and power of Apache, while benefiting from the speed of Nginx. Most websites that report using Nginx as the server (based on stats gathered from HTTP response headers), are actually Apache running with Nginx as the reverse proxy. (The HTTP response headers showing “Nginx” are being reported by the reverse-proxy, not the server itself.)

This guide is referring to a standalone Nginx setup, where it is used as the primary server instead of Apache. It should be noted that Nginx is not a completely interchangeable substitute for Apache. There are a few key differences affecting WordPress implementation that you need to be aware of before you proceed:

- With Nginx there is no directory-level configuration file like Apache’s .htaccess or IIS’s web.config files. All configuration has to be done at the server level by an administrator, and WordPress cannot modify the configuration, like it can with Apache or IIS.
- Pretty Permalinks functionality is slightly different when running Nginx.
- Since Nginx does not have .htaccess-type capability and WordPress cannot automatically modify the server configuration for you, it cannot generate the rewrite rules for you.
- Without modifications to your install, “index.php” will be added to your Permalinks. (There are ways to mitigate this with plugins (see below) and/or adding custom code to your child theme’s functions.php.)
- However, if you do want to have some (limited) .htaccess capability, it is technically possible to do add by installing the [htscanner PECL extension for PHP](#). (However, this is not a perfect solution so be sure to test and debug thoroughly before using on a live site.)

This guide is not going to cover how to install and configure Nginx, so this assumes that you have already installed Nginx and have a basic understanding of how to work with and debug it.

Generic and Multi-Site Support

To make WordPress work with Nginx you have to configure the backend php-cgi. The options available are `fastcgi` or `php-fpm`. Here, php-fpm is being used because it is included with PHP 5.3+, so installing it is straight forward.

The Nginx configuration has been broken up into five distinct files and is heavily commented to make each option easier to understand. The [author](#) also made a best-effort attempting to follow “best practices” for nginx configurations.

Main (generic) startup file

This is equivalent to `/etc/nginx/nginx.conf` (or `/etc/nginx/conf/nginx.conf` if you’re using Arch Linux).

```
# Generic startup file.
user {user} {group};

#usually equal to number of CPUs you have. run command "grep processor /proc/cpuinfo | wc -l"
to find it
worker_processes auto;
worker_cpu_affinity auto;

error_log /var/log/nginx/error.log;
pid /var/run/nginx.pid;

# Keeps the logs free of messages about not being able to bind().
#daemon off;

events {
    worker_connections 1024;
}

http {
    #rewrite_log on;

    include mime.types;
    default_type application/octet-stream;
    access_log /var/log/nginx/access.log;
    sendfile on;
    #tcp_nopush on;
    keepalive_timeout 3;
    #tcp_nodelay on;
    #gzip on;
    #php max upload limit cannot be larger than this
    client_max_body_size 13m;
    index index.php index.html index.htm;

    # Upstream to abstract backend connection(s) for PHP.
    upstream php {
        #this should match value of "listen" directive in php-fpm pool
        server unix:/tmp/php-fpm.sock;
        # server 127.0.0.1:9000;
    }
}
```

```
include sites-enabled/*;  
}
```

Per Site configuration

```
# Redirect everything to the main site. We use a separate server statement and NOT an if  
statement - see https://www.nginx.com/resources/wiki/start/topics/depth/ifisevil/  
  
server {  
    server_name _;  
    return 302 $scheme://example.com$request_uri;  
}  
  
server {  
    server_name example.com;  
    root /var/www/example.com;  
  
    index index.php;  
  
    include global/restrictions.conf;  
  
    # Additional rules go here.  
  
    # Only include one of the files below.  
    include global/wordpress.conf;  
    # include global/wordpress-ms-subdir.conf;  
    # include global/wordpress-ms-subdomain.conf;  
}
```

Splitting sections of the configuration into multiple files allows the same logic to be reused over and over. A 'global' subdirectory is used to add extra rules for general purpose use (either `/etc/nginx/conf/global/` or `/etc/nginx/global/` depending on how your nginx install is set up).

Global restrictions file

```
# Global restrictions configuration file.  
# Designed to be included in any server {} block.
```

```

location = /favicon.ico {
    log_not_found off;
    access_log off;
}

location = /robots.txt {
    allow all;
    log_not_found off;
    access_log off;
}

# Deny all attempts to access hidden files such as .htaccess, .htpasswd, .DS_Store (Mac).
# Keep logging the requests to parse later (or to pass to firewall utilities such as fail2ban)
location ~ /\. {
    deny all;
}

# Deny access to any files with a .php extension in the uploads directory
# Works in sub-directory installs and also in multisite network
# Keep logging the requests to parse later (or to pass to firewall utilities such as fail2ban)
location ~* /(?:uploads|files)/.*\.php$ {
    deny all;
}

```

General WordPress rules

For single site installations, here is the `global/wordpress.conf` file:

```

# WordPress single site rules.
# Designed to be included in any server {} block.
# Upstream to abstract backend connection(s) for php
upstream php {
    server unix:/tmp/php-cgi.socket;
    server 127.0.0.1:9000;
}

server {
    ## Your website name goes here.
    server_name domain.tld;
}

```

```

## Your only path reference.
root /var/www/wordpress;

## This should be in your http block and if it is, it's not needed here.
index index.php;

location = /favicon.ico {
    log_not_found off;
    access_log off;
}

location = /robots.txt {
    allow all;
    log_not_found off;
    access_log off;
}

location / {
    # This is cool because no php is touched for static content.
    # include the "?$args" part so non-default permalinks doesn't break when using query
string
    try_files $uri $uri/ /index.php?$args;
}

location ~ \.php$ {
    #NOTE: You should have "cgi.fix_pathinfo = 0;" in php.ini
    include fastcgi.conf;
    fastcgi_intercept_errors on;
    fastcgi_pass php;
}

location ~* \.(js|css|png|jpg|jpeg|gif|ico)$ {
    expires max;
    log_not_found off;
}
}

```

This is more up-to-date example for Nginx:

<https://www.nginx.com/resources/wiki/start/topics/recipes/wordpress/>

WordPress Multisite

For multisite installations, use one of the below sections for the `global/wordpress.conf` file, depending on the version of WordPress that was in use when multisite was *activated*, as well as the domain/subdirectory configuration.

WordPress 3.5 and up

If you activated Multisite on WordPress 3.5 or later, use one of these.

WordPress 3.5 and up Subdirectory Examples

```
# WordPress multisite subdirectory config file for WP 3.5 and up.
server {
    server_name example.com ;

    root /var/www/example.com/htdocs;
    index index.php;

    if (!-e $request_filename) {
        rewrite /wp-admin$ $scheme://$host$request_uri/ permanent;
        rewrite ^(/[^\/]*)?(/wp-.* )$2 last;
        rewrite ^(/[^\/]*)?(/*.*\.(php))$2 last;
    }

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

    location ~ /\.php$ {
        try_files $uri =404;
        include fastcgi_params;
        fastcgi_pass php;
    }

    #add some rules for static content expiry-headers here
}
```

WordPress 3.5 and up Subdomains Examples

```
# WordPress multisite subdomain config file for WP 3.5 and up.
server {
    server_name example.com *.example.com ;

    root /var/www/example.com/htdocs;
    index index.php;

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

    location ~ \.php$ {
        try_files $uri =404;
        include fastcgi_params;
        fastcgi_pass php;
    }

    #add some rules for static content expiry-headers here
}
```

WordPress 3.4 and below

If you originally activated Multisite with WordPress with 3.4 or older, you need to use one of these:

WordPress <=3.4 Subdirectory Examples

```
# WordPress multisite subdirectory config file for WP 3.4 and below.

map $uri $blogname{
    ~^(?P<blogpath>/[^\/]++)files/(.*)      $blogpath ;
}

map $blogname $blogid{
    default -999;

    #Ref: https://wordpress.org/extend/plugins/nginx-helper/
    #include /var/www/wordpress/wp-content/plugins/nginx-helper/map.conf ;
}

server {
```

```

server_name example.com ;

root /var/www/example.com/htdocs;
index index.php;

location ~ ^(/[^\/]++)?files/(.+) {
    try_files /wp-content/blogs.dir/$blogid/files/$2 /wp-includes/ms-files.php?file=$2 ;
    access_log off;      log_not_found off; expires max;
}

#avoid php readfile()
location ^~ /blogs.dir {
    internal;
    alias /var/www/example.com/htdocs/wp-content/blogs.dir ;
    access_log off;      log_not_found off; expires max;
}

if (!-e $request_filename) {
    rewrite /wp-admin$ $scheme://$host$request_uri/ permanent;
    rewrite ^(/[^\/]++)?(/wp-.* ) $2 last;
    rewrite ^(/[^\/]++)?(/*.*\.php) $2 last;
}

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

location ~ \.php$ {
    try_files $uri =404;
    include fastcgi_params;
    fastcgi_pass php;
}

#add some rules for static content expiry-headers here
}

```

NGINX provides 2 special directive: X-Accel-Redirect and map. Using these 2 directives, one can eliminate performance hit for static-file serving on WordPress multisite network.

WordPress <=3.4 Subdomains Examples

```
# WordPress multisite subdomain config file for WP 3.4 and below.
map $http_host $blogid {
    default            -999;

    #Ref: https://wordpress.org/extend/plugins/nginx-helper/
    #include /var/www/wordpress/wp-content/plugins/nginx-helper/map.conf ;

}

server {
    server_name example.com *.example.com ;

    root /var/www/example.com/htdocs;
    index index.php;

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

    location ~ \.php$ {
        try_files $uri =404;
        include fastcgi_params;
        fastcgi_pass php;
    }

    #WPMU Files
    location ~ ^/files/(.*)$ {
        try_files /wp-content/blogs.dir/$blogid/$uri /wp-includes/ms-files.php?file=$1 ;
        access_log off; log_not_found off; expires max;
    }

    #WPMU x-sendfile to avoid php readfile()
    location ^~ /blogs.dir {
        internal;
        alias /var/www/example.com/htdocs/wp-content/blogs.dir;
        access_log off;    log_not_found off;    expires max;
    }

    #add some rules for static content expiry-headers here
}
```

HTTPS in Nginx

Enabling HTTPS in Nginx is relatively simple.

```
server {  
    # listens both on IPv4 and IPv6 on 443 and enables HTTPS and HTTP/2 support.  
    # HTTP/2 is available in nginx 1.9.5 and above.  
    listen *:443 ssl;  
    listen [::]:443 ssl;  
    http2 on;  
  
    # indicate locations of SSL key files.  
    ssl_certificate /srv/www/ssl/ssl.crt;  
    ssl_certificate_key /srv/www/ssl/ssl.key;  
    ssl_dhparam /srv/www/master/ssl/dhparam.pem;  
  
    # indicate the server name  
    server_name example.com *.example.com;  
  
    # Enable HSTS. This forces SSL on clients that respect it, most modern browsers. The  
includeSubDomains flag is optional.  
    add_header Strict-Transport-Security "max-age=31536000; includeSubDomains";  
  
    # Set caches, protocols, and accepted ciphers. This config will merit an A+ SSL Labs score  
as of Sept 2015.  
    ssl_session_cache shared:SSL:20m;  
    ssl_session_timeout 10m;  
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;  
    ssl_prefer_server_ciphers on;  
    ssl_ciphers 'ECDH+AESGCM:ECDH+AES256:ECDH+AES128:DH+3DES:!ADH:!AECDH:!MD5';  
}
```

Mozilla offers an [excellent SSL config generation tool](#) as well.

WP Super Cache Rules

```

# WP Super Cache rules.
# Designed to be included from a 'wordpress-ms-...' configuration file.

set $cache_uri $request_uri;

# POST requests and urls with a query string should always go to PHP
if ($request_method = POST) {
    set $cache_uri 'null cache';
}

if ($query_string != "") {
    set $cache_uri 'null cache';
}

# Don't cache uris containing the following segments
if ($request_uri ~* "(/wp-admin/|/xmlrpc.php|wp-(app|cron|login|register|mail).php|wp-
*.php|/feed/|index.php|wp-comments-popup.php|wp-links-opml.php|wp-
locations.php|sitemap(_index)?.xml|[a-z0-9_-]+-sitemap([0-9]+)?.xml)") {
    set $cache_uri 'null cache';
}

# Don't use the cache for logged in users or recent commenters
if ($http_cookie ~* "comment_author|wordpress_[a-f0-9]+|wp-postpass|wordpress_logged_in") {
    set $cache_uri 'null cache';
}

# START MOBILE
# Mobile browsers section to server them non-cached version. COMMENTED by default as most
modern wordpress themes including twenty-eleven are responsive. Uncomment config lines in this
section if you want to use a plugin like WP-Touch
# if ($http_x_wap_profile) {
#     set $cache_uri 'null cache';
#}

# if ($http_profile) {
#     set $cache_uri 'null cache';
#}

# if ($http_user_agent ~* (2.0\
MMP|240x320|400X240|AvantGo|BlackBerry|Blazer|Cellphone|Danger|DoCoMo|Elaine/3.0|EudoraWeb|Goo

```

```

glebot-Mobile|hiptop|IEMobile|KYOCERA/WX310K|LG/U990|MIDP-2.|MMEF20|MOT-
V|NetFront|Newt|Nintendo\ Wii|Nitro|Nokia|Opera\ Mini|Palm|PlayStation\
Portable|portalmmm|Proxinet|ProxiNet|SHARP-TQ-GX10|SHG-i900|Small|SonyEricsson|Symbian\
OS|SymbianOS|TS21i-10|UP.Browser|UP.Link|webOS|Windows\ CE|WinWAP|YahooSeeker/M1A1-
R2D2|iPhone|iPod|Android|BlackBerry9530|LG-TU915\ Obigo|LGE\ VX|webOS|Nokia5800)) {
    # set $cache_uri 'null cache';
#}

#if ($http_user_agent ~* (w3c\ |w3c-|acs-
|alav|alca|amoi|audi|avan|benq|bird|blac|blaz|brew|cell|cldc|cmd-
|dang|doco|eric|hipt|htc|inno|ipaq|ipod|jigs|kddi|keji|leno|lg-c|lg-d|lg-g|lge-
|lg/u|maui|maxo|midp|mits|mmef|mobi|mot-|moto|mwbp|nec-
|newt|noki|palm|pana|pant|phil|play|port|prox|qwap|sage|sams|sany|sch-|sec-|send|seri|sgh-
|shar|sie-|siem|smal|smar|sony|sph-|symb|t-mo|teli|tim-|tosh|tsm-|upgl|upsi|vk-v|voda|wap-
|wapa|wapi|wapp|wapr|webc|winw|winw|xda\ |xda-)) {
    # set $cache_uri 'null cache';
#}
#END MOBILE

# Use cached or actual file if they exists, otherwise pass request to WordPress
location / {
    try_files /wp-content/cache/supercache/$http_host/$cache_uri/index.html $uri $uri/
/index.php?$args ;
}

```

Experimental modifications:

If you are using HTTPS, the latest development version of WP Super Cache may use a different directory structure to differentiate between HTTP and HTTPS. try_files line may look like below:

```

location / {
    try_files /wp-content/cache/supercache/$http_host/$cache_uri/index-https.html $uri $uri/
/index.php?$args ;
}

```

W3 Total Cache Rules

W3 Total Cache uses different directory structure for disk-based cache storage depending on WordPress configuration.

Cache validation checks will remain common as shown below:

```
#W3 TOTAL CACHE CHECK
set $cache_uri $request_uri;

# POST requests and urls with a query string should always go to PHP
if ($request_method = POST) {
    set $cache_uri 'null cache';
}
if ($query_string != "") {
    set $cache_uri 'null cache';
}

# Don't cache uris containing the following segments
if ($request_uri ~* "(/wp-admin/|/xmlrpc.php|wp-(app|cron|login|register|mail).php|wp-
*.php|/feed/|index.php|wp-comments-popup.php|wp-links-opml.php|wp-
locations.php|sitemap(_index)?.xml|[a-z0-9_-]+-sitemap([0-9]+)?.xml)") {
    set $cache_uri 'null cache';
}

# Don't use the cache for logged in users or recent commenters
if ($http_cookie ~* "comment_author|wordpress_[a-f0-9]+|wp-postpass|wordpress_logged_in") {
    set $cache_uri 'null cache';
}

#ADD mobile rules from WP SUPER CACHE section above

#APPEND A CODE BLOCK FROM BELOW...
```

FOR Normal WordPress (without Multisite)

Use following:

```
# Use cached or actual file if they exists, otherwise pass request to WordPress
location / {
    try_files /wp-content/w3tc/pgcache/$cache_uri/_index.html $uri $uri/ /index.php?$args ;
}
```

FOR Multisite with subdirectories

Use the following:

```

if ( $request_uri ~* "^/([_0-9a-zA-Z-]+)/.*" ){
    set $blog $1;
}

set $blog "${blog}.";

if ( $blog = "blog." ){
    set $blog "";
}

# Use cached or actual file if they exists, otherwise pass request to WordPress
location / {
    try_files /wp-content/w3tc-$blog$host/pgcache$cache_uri/_index.html $uri $uri/
/index.php?$args ;
}

```

FOR Multisite with Subdomains/Domain-mapping

Use following:

```

location / {
    try_files /wp-content/w3tc-$host/pgcache/$cache_uri/_index.html $uri $uri/
/index.php?$args;
}

```

Notes

- Nginx can handle gzip & browser cache automatically so better leave that part to nginx.
- W3 Total Cache Minify rules will work with above config without any issues.

Nginx fastcgi_cache

Nginx can perform caching on its own end to reduce load on your server. When you want to use Nginx's built-in fastcgi_cache, you better compile nginx with [fastcgi_cache_purge](#) module. It will help nginx purge cache for a page when it gets edited. On the WordPress side, you need to install a plugin like [Nginx Helper](#) to utilize fastcgi_cache_purge feature.

Config will look like below:

Define a Nginx cache zone in http{...} block, outside server{...} block

```
#move next 3 lines to /etc/nginx/nginx.conf if you want to use fastcgi_cache across many sites
fastcgi_cache_path /var/run/nginx-cache levels=1:2 keys_zone=WORDPRESS:500m inactive=60m;
fastcgi_cache_key "$scheme$request_method$host$request_uri";
fastcgi_cache_use_stale error timeout invalid_header http_500;
```

For WordPress site config, in server{..} block add a cache check block as follow

```
#fastcgi_cache start
set $no_cache 0;

# POST requests and urls with a query string should always go to PHP
if ($request_method = POST) {
    set $no_cache 1;
}
if ($query_string != "") {
    set $no_cache 1;
}

# Don't cache uris containing the following segments
if ($request_uri ~* "(/wp-admin/|/xmlrpc.php|wp-(app|cron|login|register|mail).php|wp-
*.php|/feed/|index.php|wp-comments-popup.php|wp-links-opml.php|wp-
locations.php|sitemap(_index)?.xml|[a-z0-9_-]+-sitemap([0-9]+)?.xml)") {
    set $no_cache 1;
}

# Don't use the cache for logged in users or recent commenters
if ($http_cookie ~* "comment_author|wordpress_[a-f0-9]+|wp-
postpass|wordpress_no_cache|wordpress_logged_in") {
    set $no_cache 1;
}
```

Then make changes to PHP handling block

Just add this to the following php block. Note the line `fastcgi_cache_valid 200 60m;` which tells nginx only to cache 200 responses(normal pages), which means that redirects are not cached. This is important for multilanguage sites where, if not implemented, nginx would cache the main url in one language instead of redirecting users to their respective content according to their language.

```
fastcgi_cache_bypass $no_cache;
fastcgi_no_cache $no_cache;
```

```
fastcgi_cache WORDPRESS;
fastcgi_cache_valid 200 60m;
```

Such that it becomes something like this

```
location ~ [^/]\.php(/|$) {
    fastcgi_split_path_info ^(.+?\.php)(/.*)$;
    if (!-f $document_root$fastcgi_script_name) {
        return 404;
    }
    # This is a robust solution for path info security issue and works with "cgi.fix_pathinfo
= 1" in /etc/php.ini (default)

    include fastcgi.conf;
    fastcgi_index index.php;
    # fastcgi_intercept_errors on;
    fastcgi_pass php;

    fastcgi_cache_bypass $no_cache;
    fastcgi_no_cache $no_cache;

    fastcgi_cache WORDPRESS;
    fastcgi_cache_valid 200 60m;
}
```

Finally add a location for conditional purge

```
location ~ /purge(/.*) {
    # Uncomment the following two lines to allow purge only from the webserver
    # allow 127.0.0.1;
    # deny all;

    fastcgi_cache_purge WORDPRESS "$scheme$request_method$host$1";
}
```

If you get an ‘unknown directive “fastcgi_cache_purge”’ error check that your Nginx installation has fastcgi_cache_purge module.

Better Performance for Static Files in Multisite (WP <= 3.4)

By default, on multisite networks activated prior to 3.5, a static file request brings php into picture i.e. `ms-files.php` file. You can get much better performance using Nginx `Map{..}` directive.

In Nginx config for your site, above `server{..}` block, add a section as follows:

```
map $http_host $blogid {
    default                0;

    example.com            1;
    site1.example.com      2;
    site1.com               2;
}
```

It is just a list of site-names and blog-ids. You can use [Nginx helper](#) to get such a list of site-name/blog-id pairs. This plugin will also generate a `map.conf` file which you can directly include in the `map{}` section like this:

```
map $http_host $blogid {
    default                0;

    include /path/to/map.conf ;
}
```

After creating a `map{..}` section, you just need to make one more change in your Nginx config so requests for `/files/` will be first processed using nginx `map{..}`:

```
location ~ ^/files/(.*)$ {
    try_files /wp-content/blogs.dir/$blogid/$uri /wp-includes/ms-files.php?file=$1 ;
    access_log off; log_not_found off; expires max;
}
```

Notes

- Whenever a new site is created, deleted or an extra domain is mapped to an existing site, Nginx helper will update `map.conf` file automatically but you will still need to reload Nginx config manually. You can do that anytime later. Till then, only files for new sites will be

served using php-fpm.

- This method does not generate any symbolic links. So, there will be no issues with accidental deletes or backup scripts that follow symbolic links.
- For large networks, this will scale-up nicely as there will be a single map.conf file.

A couple of final but important notes: This whole setup assumes that the root of the site is the blog and that all files that will be referenced reside on the host. If you put the blog in a subdirectory such as /blog, then the rules will have to be modified. Perhaps someone can take these rules and make it possible to, for instance, use a:

```
set $wp_subdir "/blog";
```

directive in the main 'server' block and have it automagically apply to the generic WP rules.

Warning

A typo in [Global restrictions file](#) can create loopholes. To test if your "uploads" directory is really protected, create a PHP file with some content (example:), upload it to "uploads" directory (or one of its sub-directories), then try to access (execute) it from your browser.

Resources

Reference

- [nginx + php-fpm + PHP APC + WordPress multisite \(subdirectory\) + WP Super Cache.](#)

External Links

- [Nginx WordPress wiki page](#)
- [LEMP guides on Linode's Library](#)
- [Various guides about Nginx on Linode's Library](#)
- [Lightning fast WordPress with Php-fpm and Nginx](#)
- [Virtual Hosts Examples](#)
- [List of 20+ WordPress-Nginx Tutorials for common situations](#)
- [An introduction to Nginx configuration](#)

- [A comprehensive blog series on hosting WordPress yourself using Nginx](#)
- [WordPress Installation CentminMod](#)
- [Nginx WordPress Installation Guide](#)

Scripts & Tools

For WordPress Nginx scripted installation [CentminMod](#) can be used for CentOS.

Securing Nginx

- [Securing Nginx and PHP](#)
- [Setting up PHP-FastCGI and nginx? Don't trust the tutorials: check your configuration!](#)

📄 #2

📄 📄 📄 2026-10-31 18:03:49 UTC

📄 📄 📄 2026-10-31 18:03:52 UTC