

NGINX configuration



- [Nextcloud in the webroot of NGINX](#)
 - [Nextcloud in a subdir of the NGINX webroot](#)
 - [Tips and tricks](#)
 - [PHP-Handler Configuration / Avoiding "502 Bad Gateway"](#)
 - [Suppressing log messages](#)
 - [JavaScript \(.js\) or CSS \(.css\) files not served properly](#)
 - [Upload of files greater than 10 MiB fails](#)
 - [Login loop without any clue in access.log, error.log, nor nextcloud.log](#)
 - [Trusted proxy not detected when using a Unix domain socket](#)
 - ["Access through untrusted domain" error with HTTP/3](#)
-

This page covers how to run a Nextcloud server using NGINX backed by PHP-FPM, which is also an officially supported setup.

- You need to insert the following code into **your Nginx configuration file**. Choose the appropriate example based on whether you are deploying [Nextcloud in the webroot of NGINX](#) (i.e. `https://cloud.example.com/`) or [Nextcloud in a subdir of the NGINX webroot](#) (i.e. `https://cloud.example.com/nextcloud`).
- Adjust the server directive under `upstream php-handler` to match your PHP installation's configured FPM listener (a misconfiguration here will result in a `502 Bad Gateway` - see [PHP-Handler Configuration / Avoiding "502 Bad Gateway"](#) for details)
- Adjust the existing `server_name` directives found under *both* `server` sections to your real hostname
- Adjust `root` to the webroot of your Nextcloud installation
- Adjust the `ssl_certificate` and `ssl_certificate_key` directives to the real paths for your signed certificate and private key. Make sure your SSL certificates are readable by the nginx server process (see [nginx HTTPS SSL Module documentation](#)).
- If using Let's Encrypt as TLS certificate and nginx as webserver, set `ssl_stapling` and `ssl_stapling_verify` to off in main nginx config (see [Let's Encrypt blog post](<https://letsencrypt.org/2024/12/05/ending-ocsp>)).

- Be careful about line breaks if you copy the examples, as long lines may be broken for page display and result in an invalid configuration files.
- Some environments might need a `cgi.fix_pathinfo` set to `1` in their `php.ini`.

Nextcloud in the webroot of NGINX

The following configuration should be used when Nextcloud is placed in the webroot of your nginx installation. In this example it is `/var/www/nextcloud` and it is accessed via `http(s)://cloud.example.com/`

```
# Nextcloud nginx configuration – root installation
# Version 2026-06-09

# PHP-FPM backend.
upstream php-handler {
    # Use one of the options below, not both:
    server 127.0.0.1:9000;
    #server unix:/run/php/php8.2-fpm.sock;
}

# Set the `immutable` cache control options only for assets with a cache busting `v` argument
map $arg_v $asset_immutable {
    "" "",
    default "", immutable;
}

server {
    listen 80;
    listen [::]:80;
    server_name cloud.example.com;

    # Prevent nginx HTTP Server Detection
    server_tokens off;

    # Enforce HTTPS
    return 301 https://$server_name$request_uri;
}
```

```
server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    # With NGinx >= 1.25.1 you should use this instead:
    # listen 443      ssl;
    # listen [::]:443 ssl;
    # http2 on;
    server_name cloud.example.com;

    # Path to the root of your installation
    root /var/www/nextcloud;

    # Use Mozilla's guidelines for SSL/TLS settings
    # https://mozilla.github.io/server-side-tls/ssl-config-generator/
    ssl_certificate      /etc/ssl/nginx/cloud.example.com.crt;
    ssl_certificate_key  /etc/ssl/nginx/cloud.example.com.key;

    # Prevent nginx HTTP Server Detection
    server_tokens off;

    # HSTS settings
    # WARNING: Only add the preload option once you read about
    # the consequences in https://hstspreload.org/. This option
    # will add the domain to a hardcoded list that is shipped
    # in all major browsers and getting removed from this list
    # could take several months.
    #add_header Strict-Transport-Security "max-age=31536000; includeSubDomains; preload"
always;

    # set max upload size and increase upload timeout:
    client_max_body_size 512M;
    client_body_timeout 300s;
    fastcgi_buffers 64 4K;

    # Proxy and client response timeouts
    # Uncomment an increase these if facing timeout errors during large file uploads
    #keepalive_timeout 60s;
    #proxy_connect_timeout 60s;
    #proxy_send_timeout 60s;
```

```

#proxy_read_timeout 60s;
#send_timeout 60s;

# Enable gzip but do not remove ETag headers
gzip on;
gzip_vary on;
gzip_comp_level 4;
gzip_min_length 256;
gzip_proxied expired no-cache no-store private no_last_modified no_etag auth;
gzip_types application/atom+xml text/javascript application/javascript application/json
application/ld+json application/manifest+json application/rss+xml application/vnd.geo+json
application/vnd.ms-fontobject application/wasm application/x-font-ttf application/x-web-app-
manifest+json application/xhtml+xml application/xml font/opentype image/bmp image/svg+xml
image/x-icon text/cache-manifest text/css text/plain text/vcard text/vnd.rim.location.xloc
text/vtt text/x-component text/x-cross-domain-policy;

# Pagespeed is not supported by Nextcloud, so if your server is built
# with the `ngx_pagespeed` module, uncomment this line to disable it.
#pagespeed off;

# The settings allows you to optimize the HTTP2 bandwidth.
# See https://blog.cloudflare.com/delivering-http-2-upload-speed-improvements/
# for tuning hints
client_body_buffer_size 512k;

# HTTP response headers borrowed from Nextcloud `.htaccess`
add_header Referrer-Policy "no-referrer" always;
add_header X-Content-Type-Options "nosniff" always;
add_header X-Frame-Options "SAMEORIGIN" always;
add_header X-Permitted-Cross-Domain-Policies "none" always;
add_header X-Robots-Tag "noindex, nofollow" always;

# Remove X-Powered-By, which is an information leak
fastcgi_hide_header X-Powered-By;

# Set .mjs and .wasm MIME types
# Either include it in the default mime.types list
# and include that list explicitly or add the file extension
# only for Nextcloud like below:

```

```

include mime.types;
types {
    text/javascript mjs;
application/wasm wasm;
}

# Specify how to handle directories -- specifying `/index.php$request_uri`
# here as the fallback means that Nginx always exhibits the desired behaviour
# when a client requests a path that corresponds to a directory that exists
# on the server. In particular, if that directory contains an index.php file,
# that file is correctly served; if it doesn't, then the request is passed to
# the front-end controller. This consistent behaviour means that we don't need
# to specify custom rules for certain paths (e.g. images and other assets,
# `/updater`, `/ocs-provider`), and thus
# `try_files $uri $uri/ /index.php$request_uri`
# always provides the desired behaviour.
index index.php index.html /index.php$request_uri;

# Rule borrowed from `.htaccess` to handle Microsoft DAV clients
location = / {
    if ( $http_user_agent ~ ^DavClnt ) {
        return 302 /remote.php/webdav/$is_args$args;
    }
}

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

# Make a regex exception for `/.well-known` so that clients can still
# access it despite the existence of the regex rule
# `location ~ /\.(|autotest|...)` which would otherwise handle requests
# for `/.well-known`.
location ^~ /.well-known {
    # The rules in this block are an adaptation of the rules
    # in `.htaccess` that concern `/.well-known`.

```

```

location = /.well-known/carddav { return 301 /remote.php/dav/; }
location = /.well-known/caldav { return 301 /remote.php/dav/; }

location /.well-known/acme-challenge { try_files $uri $uri/ =404; }
location /.well-known/pki-validation { try_files $uri $uri/ =404; }

# Let Nextcloud's API for `/.well-known` URIs handle all other
# requests by passing them to the front-end controller.
return 301 /index.php$request_uri;
}

# Rules borrowed from `.htaccess` to hide certain paths from clients
location ~ ^/(?:(?:build|tests|config|lib|3rdparty|templates|data)(?=$|/)) { return 404; }
location ~ ^/(?!(?:\.|autotest|occ|issue|indie|db_|console)) { return 404; }

# Hide metadata files which would otherwise be served as plain files and
# leak dependency information (composer.json, package.json, core/shipped.json).
location ~ ^/(?!(?:composer\.?(?:json|lock)|package(?:-lock)?\.json|core/shipped\.json))$ {
return 404; }

# Pass PHP requests to PHP-FPM.
#
# Important: this block must appear above the static asset locations
# below. Those locations fall back to `/index.php$request_uri`; if
# they appear first, nginx can repeatedly rewrite to `/index.php`,
# causing an internal redirection loop.
location ~ \.php(?:$|/) {
    # Rewrite most PHP requests to Nextcloud's front controller (`/index.php`).
    #
    # (Mirrors the rewrite exceptions in Nextcloud's Apache .htaccess.)
    #
    # Exceptions (not rewritten; must remain directly reachable):
    #   index.php, remote.php, public.php, cron.php, status.php
    #   ocs/v1.php, ocs/v2.php, ocs-provider/*
    #   core/ajax/update.php, updater/*
    #   */richdocumentscode(_arm64)?/proxy
    #
    # Other exceptions (e.g. /.well-known) are handled by dedicated
    # location blocks elsewhere in this config.

```

```

#
# Caution: small edits to this regex can break routing or introduce
# rewrite loops.
rewrite ^/(?!(index|remote|public|cron|status|ocs/v[12]|ocs-
provider\/.+|core\ajax\update|updater\/.+|.+\/richdocumentscode(_arm64)?\/proxy)
/index.php$request_uri;

# Split `/file.php/path/info` into:
# - $fastcgi_script_name: `/file.php`
# - $fastcgi_path_info:  `/path/info`
#
# This is required for entry-points such as `remote.php` and `public.php`,
# which route requests based on PATH_INFO.
fastcgi_split_path_info ^(.+?\.php)(/.*)$;
set $path_info $fastcgi_path_info;    # Save before try_files resets it

# Return 404 for nonexistent PHP scripts (avoids passing arbitrary
# paths to PHP-FPM, which is a known security risk).
try_files $fastcgi_script_name =404;

include fastcgi_params;
fastcgi_pass php-handler;

fastcgi_param SCRIPT_FILENAME          $document_root$fastcgi_script_name;
fastcgi_param PATH_INFO                $path_info;
fastcgi_param HTTPS                    on;          # Assumes TLS terminates here
fastcgi_param modHeadersAvailable      true;        # Avoid duplicate security headers
fastcgi_param front_controller_active  true;        # Enable pretty URLs

# Let nginx handle HTTP error responses from PHP-FPM (e.g. custom
# error pages). Disable for debugging if PHP errors are being hidden.
fastcgi_intercept_errors on;

# Required for uploads: PHP-FPM does not support chunked
# transfer encoding and needs a Content-Length header.
fastcgi_request_buffering on;

# Optional PHP-FPM timeout tuning (e.g. for 504 response timeouts).
# Increase these only if uploads or long-running PHP requests are

```

```

# timing out in your environment.
#fastcgi_read_timeout 60s;
#fastcgi_send_timeout 60s;
#fastcgi_connect_timeout 60s;

# Disable on-disk buffering of FastCGI responses (reduces disk I/O at
# the cost of holding responses in memory).
fastcgi_max_temp_file_size 0;
}

# Serve static files
location ~ \.(?:css|js|mjs|svg|gif|ico|jpg|png|webp|wasm|tflite|map|ogg|flac|mp4|webm)$ {
    try_files $uri /index.php$request_uri;

# HSTS settings
# WARNING: Only add the preload option once you read about
# the consequences in https://hstspreload.org/. This option
# will add the domain to a hardcoded list that is shipped
# in all major browsers and getting removed from this list
# could take several months.
#add_header Strict-Transport-Security "max-age=31536000; includeSubDomains; preload"
always;

# HTTP response headers borrowed from Nextcloud `.htaccess`
add_header Cache-Control                "public, max-
age=15778463$asset_immutable";
add_header Referrer-Policy              "no-referrer"          always;
add_header X-Content-Type-Options       "nosniff"              always;
add_header X-Frame-Options               "SAMEORIGIN"          always;
add_header X-Permitted-Cross-Domain-Policies "none"            always;
add_header X-Robots-Tag                  "noindex, nofollow"    always;
access_log off;      # Optional: Don't log access to assets
}

location ~ \.(otf|woff2?)$ {
    try_files $uri /index.php$request_uri;
    expires 7d;      # Cache-Control policy borrowed from `.htaccess`
    access_log off;  # Optional: Don't log access to assets
}

```

```

# Rule borrowed from `.htaccess`
location /remote {
    return 301 /remote.php$request_uri;
}

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

```

Nextcloud in a subdir of the NGINX webroot

The following config should be used when Nextcloud is placed within a subdir of the webroot of your nginx installation. In this example the Nextcloud files are located at `/var/www/nextcloud` and the Nextcloud instance is accessed via `http(s)://cloud.example.com/nextcloud/`. The configuration differs from the "Nextcloud in webroot" configuration above in the following ways:

- All requests for `/nextcloud` are encapsulated within a single `location` block, namely `location ^~ /nextcloud`.
- The string `/nextcloud` is prepended to all prefix paths.
- The root of the domain is mapped to `/var/www` rather than `/var/www/nextcloud`, so that the URI `/nextcloud` is mapped to the server directory `/var/www/nextcloud`.
- The blocks that handle requests for paths outside of `/nextcloud` (i.e. `/robots.txt` and `/.well-known`) are pulled out of the `location ^~ /nextcloud` block.
- The block which handles `/.well-known` doesn't need a regex exception, since the rule which prevents users from accessing hidden folders at the root of the Nextcloud installation no longer matches that path.

```

# Nextcloud nginx configuration – subdirectory installation (/nextcloud)
# Version 2026-06-09

# PHP-FPM backend.
upstream php-handler {
    # Use one of the options below, not both:
    server 127.0.0.1:9000;
    #server unix:/run/php/php8.2-fpm.sock;
}

```

```

}

# Set the `immutable` cache control options only for assets with a cache busting `v` argument
map $arg_v $asset_immutable {
    "" "";
    default ", immutable";
}

server {
    listen 80;
    listen [::]:80;
    server_name cloud.example.com;

    # Prevent nginx HTTP Server Detection
    server_tokens off;

    # Enforce HTTPS just for `/nextcloud`
    location /nextcloud {
        return 301 https://$server_name$request_uri;
    }
}

server {
    listen 443      ssl http2;
    listen [::]:443 ssl http2;
    # With NGinx >= 1.25.1 you should use this instead:
    # listen 443      ssl;
    # listen [::]:443 ssl;
    # http2 on;
    server_name cloud.example.com;

    # Path to the root of the domain
    root /var/www;

    # Use Mozilla's guidelines for SSL/TLS settings
    # https://mozilla.github.io/server-side-tls/ssl-config-generator/
    ssl_certificate      /etc/ssl/nginx/cloud.example.com.crt;
    ssl_certificate_key  /etc/ssl/nginx/cloud.example.com.key;

```

```

# Prevent nginx HTTP Server Detection
server_tokens off;

# Set .mjs and .wasm MIME types
# Either include it in the default mime.types list
# and include that list explicitly or add the file extension
# only for Nextcloud like below:
include mime.types;
types {
    text/javascript mjs;
    # uncomment below for Nginx <= 1.21.0
    # application/wasm wasm;
}

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

location ^~ /.well-known {
    # The rules in this block are an adaptation of the rules
    # in the Nextcloud `.htaccess` that concern `/.well-known`.

    location = /.well-known/carddav { return 301 /nextcloud/remote.php/dav/; }
    location = /.well-known/caldav  { return 301 /nextcloud/remote.php/dav/; }

    location /.well-known/acme-challenge    { try_files $uri $uri/ =404; }
    location /.well-known/pki-validation    { try_files $uri $uri/ =404; }

    # Let Nextcloud's API for `/.well-known` URIs handle all other
    # requests by passing them to the front-end controller.
    return 301 /nextcloud/index.php$request_uri;
}

location ^~ /nextcloud {
    # set max upload size and increase upload timeout:
    client_max_body_size 512M;
    client_body_timeout 300s;

```

```
fastcgi_buffers 64 4K;

# Proxy and client response timeouts
# Uncomment an increase these if facing timeout errors during large file uploads
#keepalive_timeout 60s;
#proxy_connect_timeout 60s;
#proxy_send_timeout 60s;
#proxy_read_timeout 60s;
#send_timeout 60s;

# Enable gzip but do not remove ETag headers
gzip on;
gzip_vary on;
gzip_comp_level 4;
gzip_min_length 256;
gzip_proxied expired no-cache no-store private no_last_modified no_etag auth;
gzip_types application/atom+xml text/javascript application/javascript
application/json application/ld+json application/manifest+json application/rss+xml
application/vnd.geo+json application/vnd.ms-fontobject application/wasm application/x-font-ttf
application/x-web-app-manifest+json application/xhtml+xml application/xml font/opentype
image/bmp image/svg+xml image/x-icon text/cache-manifest text/css text/plain text/vcard
text/vnd.rim.location.xloc text/vtt text/x-component text/x-cross-domain-policy;

# Pagespeed is not supported by Nextcloud, so if your server is built
# with the `ngx_pagespeed` module, uncomment this line to disable it.
#pagespeed off;

# The settings allows you to optimize the HTTP2 bandwidth.
# See https://blog.cloudflare.com/delivering-http-2-upload-speed-improvements/
# for tuning hints
client_body_buffer_size 512k;

# HSTS settings
# WARNING: Only add the preload option once you read about
# the consequences in https://hstspreload.org/. This option
# will add the domain to a hardcoded list that is shipped
# in all major browsers and getting removed from this list
# could take several months.
#add_header Strict-Transport-Security "max-age=31536000; includeSubDomains; preload"
```

```
always;
```

```
# HTTP response headers borrowed from Nextcloud `.htaccess`
```

```
add_header Referrer-Policy "no-referrer" always;
```

```
add_header X-Content-Type-Options "nosniff" always;
```

```
add_header X-Frame-Options "SAMEORIGIN" always;
```

```
add_header X-Permitted-Cross-Domain-Policies "none" always;
```

```
add_header X-Robots-Tag "noindex, nofollow" always;
```

```
# Remove X-Powered-By, which is an information leak
```

```
fastcgi_hide_header X-Powered-By;
```

```
# Specify how to handle directories -- specifying `/nextcloud/index.php$request_uri`
```

```
# here as the fallback means that Nginx always exhibits the desired behaviour
```

```
# when a client requests a path that corresponds to a directory that exists
```

```
# on the server. In particular, if that directory contains an index.php file,
```

```
# that file is correctly served; if it doesn't, then the request is passed to
```

```
# the front-end controller. This consistent behaviour means that we don't need
```

```
# to specify custom rules for certain paths (e.g. images and other assets,
```

```
# `/updater`, `/ocs-provider`), and thus
```

```
# `try_files $uri $uri/ /nextcloud/index.php$request_uri`
```

```
# always provides the desired behaviour.
```

```
index index.php index.html /nextcloud/index.php$request_uri;
```

```
# Rule borrowed from `.htaccess` to handle Microsoft DAV clients
```

```
location = /nextcloud {
```

```
    if ( $http_user_agent ~ ^DavClnt ) {
```

```
        return 302 /nextcloud/remote.php/webdav/$is_args$args;
```

```
    }
```

```
}
```

```
# Rules borrowed from `.htaccess` to hide certain paths from clients
```

```
location ~ ^/nextcloud/(?::build|tests|config|lib|3rdparty|templates|data)(?:$|/) {
```

```
return 404; }
```

```
location ~ ^/nextcloud/(?::\.|autotest|occ|issue|indie|db_|console) {
```

```
return 404; }
```

```
# Hide metadata files which would otherwise be served as plain files and
```

```
# leak dependency information (composer.json, package.json, core/shipped.json).
```

```

location ~ ^/nextcloud/(?::composer\.(?:json|lock)|package(?:-
lock)?\.json|core/shipped\.json)$ { return 404; }

# Pass PHP requests to PHP-FPM.
#
# Important: this block must appear above the static asset locations
# below. Those locations fall back to `/nextcloud/index.php$request_uri`;
# if they appear first, nginx can repeatedly rewrite to
# `/nextcloud/index.php`, causing an internal redirection loop.
location ~ /\.php(?:$|/) {
    # Rewrite most PHP requests to Nextcloud's front controller
    # (`/nextcloud/index.php`).
    #
    # (Mirrors the rewrite exceptions in Nextcloud's Apache .htaccess.)
    #
    # Exceptions (not rewritten; must remain directly reachable):
    #   index.php, remote.php, public.php, cron.php, status.php
    #   ocs/v1.php, ocs/v2.php, ocs-provider/*
    #   core/ajax/update.php, updater/*
    #   */richdocumentscode(_arm64)?/proxy
    #
    # Other exceptions (e.g. /.well-known) are handled by dedicated
    # location blocks elsewhere in this config.
    #
    # Caution: small edits to this regex can break routing or introduce
    # rewrite loops.
    rewrite ^/nextcloud/(?!(index|remote|public|cron|status|ocs/v[12]|ocs-
provider/\.+|core/ajax/update|updater/\.+|\.+\/richdocumentscode(_arm64)?\/proxy)
/nextcloud/index.php$request_uri;

    # Split `/file.php/path/info` into:
    # - $fastcgi_script_name: `/file.php`
    # - $fastcgi_path_info:  `/path/info`
    #
    # This is required for entry-points such as `remote.php` and `public.php`,
    # which route requests based on PATH_INFO.
    fastcgi_split_path_info ^(.+?\.php)(/.*)$;
    set $path_info $fastcgi_path_info;    # Save before try_files resets it

```

```

# Return 404 for nonexistent PHP scripts (avoids passing arbitrary
# paths to PHP-FPM, which is a known security risk).
try_files $fastcgi_script_name =404;

include fastcgi_params;
fastcgi_pass php-handler;

fastcgi_param SCRIPT_FILENAME      $document_root$fastcgi_script_name;
fastcgi_param PATH_INFO            $path_info;
fastcgi_param HTTPS                on;          # Assumes TLS terminates here
fastcgi_param modHeadersAvailable true;        # Avoid duplicate security
headers
fastcgi_param front_controller_active true;      # Enable pretty URLs

# Let nginx handle HTTP error responses from PHP-FPM (e.g. custom
# error pages). Disable for debugging if PHP errors are being hidden.
fastcgi_intercept_errors on;

# Required for uploads: PHP-FPM does not support chunked
# transfer encoding and needs a Content-Length header.
fastcgi_request_buffering on;

# Optional PHP-FPM timeout tuning (e.g. for 504 response timeouts).
# Increase these only if uploads or long-running PHP requests are
# timing out in your environment.
#fastcgi_read_timeout 60s;
#fastcgi_send_timeout 60s;
#fastcgi_connect_timeout 60s;

# Disable on-disk buffering of FastCGI responses (reduces disk I/O at
# the cost of holding responses in memory).
fastcgi_max_temp_file_size 0;
}

# Serve static files
location ~
\.(?:css|js|mjs|svg|gif|ico|jpg|png|webp|wasm|tflite|map|ogg|flac|mp4|webm)$ {
    try_files $uri /nextcloud/index.php$request_uri;
    # HTTP response headers borrowed from Nextcloud `.htaccess`

```

```

        add_header Cache-Control                "public, max-
age=15778463$asset_immutable";

        add_header Referrer-Policy              "no-referrer"      always;
        add_header X-Content-Type-Options      "nosniff"          always;
        add_header X-Frame-Options              "SAMEORIGIN"      always;
        add_header X-Permitted-Cross-Domain-Policies "none"        always;
        add_header X-Robots-Tag                  "noindex, nofollow" always;
        access_log off;      # Optional: Don't log access to assets
    }

    location ~ \.(otf|woff2?)$ {
        try_files $uri /nextcloud/index.php$request_uri;
        expires 7d;          # Cache-Control policy borrowed from `.htaccess`
        access_log off;      # Optional: Don't log access to assets
    }

    # Rule borrowed from `.htaccess`
    location /nextcloud/remote {
        return 301 /nextcloud/remote.php$request_uri;
    }

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

```

Tips and tricks

PHP-Handler Configuration / Avoiding "502 Bad Gateway"

The `server` line within the `upstream php-handler` above needs to be adjusted to reflect your local PHP FPM configuration. It must match whatever is configured for the `listen` directive within the PHP FPM pool you'll be using for NC.

Many Linux distributions define a listener for a default PHP-FPM pool called `www` in a file called `www.conf` located somewhere like `/etc/php/8.1/pool.d`.

Look for the line that is set to something like:

```
listen = /var/run/php/php-fpm.sock or listen = 127.0.0.1:9000
```

If PHP FPM will be running on the same host as NGINX (it's probably a safe assumption it will be if you're unsure), it is recommended you use the UNIX socket (i.e. `/var/run/php/php-fpm.sock`) rather than TCP (`127.0.0.1:9000`) for maximum performance (though either will work as long as your NGINX and PHP FPM configurations match).

After deciding how you'd prefer to connect NGINX with PHP FPM (and, if necessary, updating your local PHP FPM configuration and restarting FPM), set your NGINX configuration's `upstream php-handler` `server` to match your preference (Note: If using UNIX sockets, prepend `unix:` in the NGINX configuration, but *not* in your PHP FPM `www.conf`).

Suppressing log messages

If you're seeing meaningless messages in your logfile, for example `client denied by server configuration: /var/www/data/htaccess.test.txt`, add this section to your nginx configuration to suppress them:

```
location = /data/htaccess.test.txt {
    allow all;
    log_not_found off;
    access_log off;
}
```

JavaScript (.js) or CSS (.css) files not served properly

A common issue with custom nginx configs is that JavaScript (.js) or CSS (.css) files are not served properly leading to a 404 (File not found) error on those files and a broken webinterface.

This could be caused by the:

```
location ~* \.(?:css|js)$ {
```

block shown above not located **below** the:

```
location ~ /\.php(?:$|\/) {
```

block. Other custom configurations like caching JavaScript (.js) or CSS (.css) files via gzip could also cause such issues.

Another cause of this issue could be not properly including mimetypes in the http block, as shown [here](#).

Upload of files greater than 10 MiB fails

If you configure nginx (globally) to block all requests to (hidden) dot files, it may be not possible to upload files greater than 10 MiB using the webpage due to Nextclouds requirement to upload the file to a URL ending with `/.file`.

You may require to change:

```
location ~ /\. {
```

to the following to re-allow file uploads:

```
location ~ /\.(!file).* {
```

See [issue #8802 on nextcloud/server](#) for more information.

Other parameters besides the above are relevant to uploading large files (see [Uploading big files > 512MB](#)).

Login loop without any clue in access.log, error.log, nor nextcloud.log

If you after fresh installation (Centos 7 with nginx) have problem with first login, you should as first check these files:

```
tail /var/www/nextcloud/data/nextcloud.log
tail /var/log/nginx/access.log
tail /var/log/nginx/error.log
```

If you just see some correct requests in access log, but no login happens, you check access rights for php session and wsdllcache directory. Try to check permissions and execute change if needed:

```
chown nginx:nginx /var/lib/php/session/  
chown root:nginx /var/lib/php/wsdllcache/  
chown root:nginx /var/lib/php/opcache/
```

Trusted proxy not detected when using a Unix domain socket

When an upstream proxy (another nginx instance, Caddy, HAProxy, etc.) passes requests to Nextcloud's nginx via a Unix domain socket, nginx sets `REMOTE_ADDR` to the literal string `unix:` instead of an IP address. Nextcloud cannot parse this as a trusted proxy, causing trusted proxy detection to fail and resulting in errors such as:

```
Unsupported operand types: bool & string in IPAddress.php
```

To fix this, add the following directives to the nginx `server` block that listens on the Unix socket:

```
set_real_ip_from    unix::  
real_ip_header      X-Forwarded-For;
```

This tells nginx to treat the Unix socket peer as a trusted source and extract the real client IP from the `X-Forwarded-For` header passed by the upstream proxy. You must also ensure the upstream proxy sets that header correctly, for example with `proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;`.

"Access through untrusted domain" error with HTTP/3

When HTTP/3 (QUIC) is enabled in nginx, the `HTTP_HOST` FastCGI parameter may not be forwarded correctly to PHP-FPM, causing Nextcloud to show the "Access through untrusted domain" error page even though the domain is listed in `trusted_domains`.

Add the following line to the `fastcgi_param` block in your nginx configuration to explicitly pass the host:

```
fastcgi_param HTTP_HOST $host;
```

Place it alongside the other `fastcgi_param` directives (after `include fastcgi_params;`). This overrides whatever value (or lack thereof) nginx would otherwise derive from the HTTP/3 request headers.

📁 #3

📁 cocobook-sync 📁 2026-10-28 16:26:20 UTC

📁 cocobook-sync 📁 2026-10-29 01:09:14 UTC