

# Nextcloud

- [Instalar NextCloud](#)
- [Resolución de problemas Nextcloud](#)
- [Borrar versiones en Nextcloud](#)
- [No puede acceder a los archivos porque están bloqueados](#)
- [Entrar con el usuario para ejecutar occ](#)
- [Limpieza de papelera](#)

# Instalar NextCloud

## Ejecutar un update y upgrade

```
apt update -y  
apt upgrade -y
```

Instalar curl

```
apt install curl -y
```

Podemos usar php8.1 o bien php8.2

## PHP 8.1

## Preparar la instalación de PHP8.1

```
apt install -y gnupg2 ca-certificates apt-transport-https software-properties-common  
wget -q0 - https://packages.sury.org/php/apt.gpg | apt-key add  
echo "deb https://packages.sury.org/php/ buster main" | tee /etc/apt/sources.list.d/php.list  
apt update -y
```

## Instalar PHP 8.1

```
apt install php8.1
```

## Instalar módulos de PHP necesarios

```
apt install php8.1-mysql php8.1-imap php8.1-ldap php8.1-xml php8.1-gd php8.1-curl php8.1-  
mbstring php8.1-zip php8.1-simplexml php8.1-dom php8.1-intl php8.1-fpm php8.1-bcmath php8.1-  
gmp php8.1-imagick -y
```

# PHP 8.2

## Instalar PHP 8.2

```
apt install php8.2
```

## Instalar módulos de PHP necesarios

```
apt install php8.2-mysql php8.2-imap php8.2-ldap php8.2-xml php8.2-gd php8.2-curl php8.2-mbstring php8.2-zip php8.2-simplexml php8.2-dom php8.2-intl php8.2-fpm php8.2-bcmath php8.2-gmp php8.2-imagick -y
```

# Soporte SVG

## Instalar soporte para SVG

```
apt-get install libmagickcore-6.q16-6-extra
```

# Instalar Mariadb y soporte para PHP

```
apt -y install mariadb-server php-mysql
```

Instalar soporte para php 8 en apache2 (luego se quitará), ya que se instala por defecto con el php

```
apt install apache2 libapache2-mod-php8.1
```

# Configurar Mariadb

```
mysql -u root -p
```

Generar la BBDD de Nextcloud así como las credenciales

```
CREATE DATABASE nextcloud;
GRANT ALL ON nextcloud.* TO 'nextcloud_user'@'localhost' IDENTIFIED BY 'mipassword';
FLUSH PRIVILEGES;
EXIT;
```

# Descargar e instalar Nextcloud

```
cd /tmp
apt install zip -y
wget https://download.nextcloud.com/server/releases/latest.zip
unzip latest-22.zip
mv nextcloud /var/www/
cd /var/www/
chown -R www-data:www-data nextcloud
chmod -R 755 nextcloud
```

## Instalar Nginx

Pararemos el servicio Apache2 y lo desactivaremos

```
service apache2 stop
systemctl disable apache2
```

## Instalaremos nginx

```
apt-get install nginx -y
```

## Arrancaremos y activaremos Nginx y php-fpm 8.1

```
service nginx start
systemctl enable nginx
service php8.1-fpm start
systemctl enable php8.1-fpm
```

## Arrancaremos y activaremos Nginx y php-fpm 8.2

```
service nginx start
systemctl enable nginx
service php8.2-fpm start
systemctl enable php8.2-fpm
```

## Crear el site nextcloud en nginx

Crearemos el sitio en /etc/nginx/sites-available/

```
nano /etc/nginx/sites-available/my-nextcloud.conf
```

La configuración por defecto que nos aparece en la documentación de Nextcloud es la siguiente:

```
upstream php-handler {
    server 127.0.0.1:9000;
    #server unix: /var/run/php/php8.2-fpm.sock;
}

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

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

server {
    listen 443      ssl http2;
    listen [::]:443 ssl http2;
    server_name cloud.example.com;

    # 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;

    # 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=15768000; includeSubDomains; preload;"
    always;

    # set max upload size
    client_max_body_size 512M;
    fastcgi_buffers 64 4K;
```

```

# 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 application/javascript application/json
application/ld+json application/manifest+json application/rss+xml application/vnd.geo+json
application/vnd.ms-fontobject 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;

# 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-Download-Options "noopen" always;
add_header X-Frame-Options "SAMEORIGIN" always;
add_header X-Permitted-Cross-Domain-Policies "none" always;
add_header X-Robots-Tag "none" always;
add_header X-XSS-Protection "1; mode=block" always;

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

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

# 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`, `/ocm-provider`, `/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; }

```

```
# Ensure this block, which passes PHP files to the PHP process, is above the blocks
# which handle static assets (as seen below). If this block is not declared first,
# then Nginx will encounter an infinite rewriting loop when it prepends `/index.php`
# to the URI, resulting in a HTTP 500 error response.
```

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

```
    fastcgi_split_path_info ^(.+?\.php)(/.*)$;
```

```
    set $path_info $fastcgi_path_info;
```

```
    try_files $fastcgi_script_name =404;
```

```
    include fastcgi_params;
```

```
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
```

```
    fastcgi_param PATH_INFO $path_info;
```

```
    fastcgi_param HTTPS on;
```

```
    fastcgi_param modHeadersAvailable true;         # Avoid sending the security headers
```

```
twice
```

```
    fastcgi_param front_controller_active true;     # Enable pretty urls
```

```
    fastcgi_pass php-handler;
```

```
    fastcgi_intercept_errors on;
```

```
    fastcgi_request_buffering off;
```

```
}
```

```
location ~ \.(?:css|js|svg|gif|png|jpg|ico)$ {
```

```
    try_files $uri /index.php$request_uri;
```

```
    expires 6M;                # Cache-Control policy borrowed from `.htaccess`
```

```
    access_log off;            # Optional: Don't log access to assets
```

```
}
```

```
location ~ \.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;
    }
}

```

Tendremos que cambiar lo siguiente:

php-handler. Comentar la linea de 127.0.0.1 y descomentar la del socket fpm

Para ello comprobaremos si existe el socket

```

root@demo: /etc/nginx/sites-available# ls -la /var/run/php/php8.1-fpm.sock
srw-rw-r-- 1 www-data www-data 0 Aug  8 23:56 /var/run/php/php8.1-fpm.sock
root@demo: /etc/nginx/sites-available#

```

Si existe usaremos el socket de php-fpm. Si no tendremos que iniciar el php-fpm

```

upstream php-handler {
    #server 127.0.0.1:9000;
    server unix:/var/run/php/php8.2-fpm.sock;
}

```

Si no tenemos certificado de pago, comentaremos las líneas de los certificados

```

# 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;

```

Cambiaremos el nombre del server por el nuestro, en lugar de cloud.example.com, pondremos el nuestro. Tanto el apartado de server:80 como en el de server:443 Cambiaremos la ruta a la que apunta

```

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

```

En la parte de add\_header, agregaremos esto al final

```

add_header Strict-Transport-Security "max-age=15768000" always;

```

Agregaremos el sitio a los sitios disponibles

```
cd /etc/nginx/sites-enabled
ln -s /etc/nginx/sites-available/my-nextcloud.conf my-nextcloud.conf
```

# Certificado LetsEncrypt

Instalaremos las librerías para el certificado de LetsEncrypt

```
apt install python3-acme python3-certbot python3-mock python3-openssl python3-pkg-resources pyth
```

Instalaremos el certbot

```
apt install python3-certbot-nginx
```

Generamos el certificado

```
certbot --nginx -d my-nextcloud.com
```

## Caché

Podemos usar memcached o redis para la caché.

## Instalar Memcached

para instalar memcached, primero lo instalaremos

```
apt install memcached
apt install php-memcached
```

A continuación lo habilitaremos al inicio y lo arrancaremos

```
systemctl enable memcached
systemctl start memcached
```

Y por último agregaremos la configuración en el archivo config.php

```
'memcache.local' => '\OC\Memcache\Memcached',
'memcache.distributed' => '\OC\Memcache\Memcached',
'memcached_servers' => array(
    array('127.0.0.1', 11211),
),
```

# Instalar redis

Para instalar redis habilitaremos redis y apcu

```
apt install php8.1-apcu php8.1-redis redis-server
```

```
apt install php8.2-apcu php8.2-redis redis-server
```

Habilitamos redis

```
systemctl enable redis-server
```

Ahora configuramos redis

Buscaremos la siguiente línea **unixsocket /var/run/redis/redis.sock** y la descomentaremos. Debajo de esa línea nos encontraremos con **unixsocketperm 700** el cual también descomentaremos y cambiaremos por **unixsocketperm 770**. Guardamos y salimos.

Añadimos el usuario redis al grupo de Apache

```
usermod -a -G redis www-data
```

Finalmente, reiniciamos los servicios de Apache y Redis:

```
# systemctl restart nginx
# systemctl enable redis-server
# systemctl restart redis-server
```

Tenemos que configurar NextCloud para que use Redis, para ello tendremos que modificar el fichero de configuración de NextCloud agregaremos la configuración en el archivo config.php y añadiremos lo siguiente:

```
'memcache.local' => '\\0C\\Memcache\\APCu',
'memcache.locking' => '\\0C\\Memcache\\Redis',
'filelocking.enabled' => 'true',
'redis' =>
    array (
        'host' => '127.0.0.1',
        'port' => 6379,
        'timeout' => 0.0,
    ),
```

Desde **Nextcloud 12**, se requiere una configuración adicional para configurar correctamente **Opcache de PHP**. Se muestra el error «**La Opcache de PHP no está bien configurada. Para mejorar el rendimiento se recomienda usar las siguientes configuraciones en el** `php.ini` **»**. Para ello tendremos que editar el fichero `/etc/php/8.1/fpm/php.ini`.

Al final del fichero añadimos lo siguiente:

```
; Nextcloud Opcache settings
opcache.enable=1
opcache.enable_cli=1
opcache.interned_strings_buffer=8
opcache.max_accelerated_files=10000
opcache.memory_consumption=128
opcache.save_comments=1
opcache.revalidate_freq=1
```

Y reiniciamos PHP-FPM:

```
systemctl restart php8.1-fpm
```

# Ajustes finales

## Tamaño de subida de archivos

Modificaremos el fichero `/etc/php/8.1/fpm/php.ini` y modificaremos los parámetros `upload_max_filesize` y `post_max_size` y les pondremos el valor de `2048M`. Finalmente, reiniciaremos PHP-FPM para que entre en efecto la modificación

## Enlaces amigables

Otra cosa que es bueno configurar son los enlaces amigables para que las **URL's** sean visualmente más fácil de recordar de **NextCloud** para ello modificamos el archivo `config.php` y añadimos lo siguiente:

```
'htaccess.RewriteBase' => '/',
```

Y si reiniciamos el servicio del Nginx veremos que de las URL

`nube.tecnocratica.net/index.php/apps/` pasaremos a `nube.tecnocratica.net/apps/`.



# Resolución de problemas

## Nextcloud

### Comprobar la versión de Nextcloud

```
sudo -u www-data php occ -V
```

### Agregar archivos añadidos a mano a Nextcloud

El comando scan busca nuevos archivos y actualiza la caché de archivos. Puede volver a escanear todos los archivos, por usuario, una lista de usuarios delimitada por espacios y limitar la ruta de búsqueda. Si no usas --quiet, las estadísticas se mostrarán al final del escaneo:

```
cd /var/www/html/nextcloud
root@nextcloud: /var/www/html/nextcloud# sudo -u www-data php occ files:scan usuario.a.escanear
```

El comando puede ejecutarse con el parámetro --help para proporcionarnos ayuda

```
sudo -u www-data php occ files:scan --help
Usage:
files:scan [-p|--path="..."] [-q|--quiet] [-v|vv|vvv --verbose] [--all]
[user_id1] ... [user_idN]
```

#### Arguments:

user_id	will rescan all files of the given user(s)
---------	--

#### Options:

--path	limit rescan to the user/path given
--all	will rescan all files of all known users
--quiet	suppress any output
--verbose	files and directories being processed are shown additionally during scanning
--unscanned	scan only previously unscanned file

### Configurar la zona horaria

```
sudo -u www-data php occ config:system:set logtimezone  
--value="Europe/Madrid"
```

## Deshabilitar el modo mantenimiento

```
sudo -u www-data php occ config:system:set maintenance --value=false  
--type=boolean
```

## Comprobar y añadir índices perdidos

Puede suceder que agreguemos de vez en cuando nuevos índices a las tablas de bases de datos ya existentes, por ejemplo, para mejorar el rendimiento. Para verificar la base de datos en busca de índices que faltan, ejecuta el siguiente comando:

```
sudo -u www-data php occ db:add-missing-indices
```

## Trashbin (papelera)

La configuración de retención de los ficheros en la papelera se realiza mediante parámetros de configuración para permitir al administrador del servidor Nextcloud la capacidad de adaptar este comportamiento, estos son los siguientes:

```
auto
```

Comportamiento estándar.

```
D, auto
```

Permite definir los días mínimos que se mantiene un archivo con el comportamiento estándar.

```
auto, D
```

Eliminar después de varios días, pero antes si se requiere espacio

```
D1, D2
```

No eliminar antes de D1, pero eliminar definitivamente después de una cierta cantidad de días (D2)

```
disabled
```

Deshabilita la eliminación automática

Entonces, para eliminar archivos automáticamente después de 30 días y darle a Nextcloud la capacidad de eliminar antes si el espacio finalmente se agota, puedes agregar esto a tu config.php:

```
'trashbin_retention_obligation' => 'auto, 30',
```

Para garantizar que todos los archivos se conserven durante 30 días, pero que se eliminen definitivamente después de 35 días, puedes agregar lo siguiente a tu config.php:

```
'trashbin_retention_obligation' => '30, 35',
```

Este comando solo está disponible cuando la aplicación "Archivos eliminados" (files\_trashbin) está habilitada. Para eliminar la papelera de un usuario o varios

```
sudo -u www-data php occ trashbin:cleanup usuario1 usuario2 usuario3
```

O bien de todos los usuarios

```
sudo -u www-data php occ trashbin:cleanup --all-users
```

```
sudo -u www-data php occ db:convert-filecache-bigint
```

<https://nube.ateinco.net/remote.php/dav>



# Borrar versiones en Nextcloud

Este comando solo está disponible cuando la aplicación "Versions" (files\_versions) está habilitada.

Para eliminar versiones de todos los usuarios cuando no se especifica ninguno

```
sudo -u www-data php occ versions:cleanup
```

Para eliminar versiones de archivos para usuarios específicos

```
sudo -u www-data php occ versions:cleanup eduardo javier tomas
```

## Ajuste del control de versiones

Las versiones se pueden gestionar con el ajuste *versions\_retention\_obligation* del fichero de configuración

```
'versions_retention_obligation' => 'auto',
```

### Los valores pueden ser:

#### **auto**

Configuración predeterminada. Las versiones caducan automáticamente de acuerdo con las reglas de caducidad. Consulte Control de versiones de archivos y antigüedad para obtener más información.

#### **D, auto**

Mantener las versiones al menos durante D días, aplicar reglas de caducidad a todas las versiones anteriores a D días

#### **auto, D**

Eliminar todas las versiones que tengan más de D días automáticamente, eliminar otras versiones de acuerdo con las reglas de caducidad

#### **D1, D2**

Mantener las versiones durante al menos D1 días y eliminarlas cuando exceda D2 días

#### **disabled**

limpieza de versiones automática deshabilitada, las versiones se mantendrán para siempre

# No puede acceder a los archivos porque están bloqueados

Pon Nextcloud en modo de mantenimiento: edita config/config.php y cambia esta línea:

```
'maintenance' => true,
```

Vacía la tabla

```
oc_file_locks
```

Se puede llamar así o simplemente file\_locks

```
DELETE FROM oc_file_locks WHERE 1;
```

Vuelve a desactivar el modo mantenimiento

```
'maintenance' => false,
```

Asegúrate de que los trabajos cron se ejecutan correctamente (en la página de administración te indica cuándo se ejecutó cron por última vez)

# Entrar con el usuario para ejecutar occ

En la carpeta del Nextcloud tenemos que ejecutar

```
su - usuario-del-alojamiento -s /bin/bash
```

Y luego ya con este usuario podremos ejecutar el occ

Nota: hay que ejecutar con una versión php 7.4 o superior para las nuevas versiones de NextCloud

Por ejemplo:

```
/opt/php-7.4/bin/php
```

# Limpieza de papelera

Hay un parámetro en el Nextcloud que controla la retención de la papelera

```
'trashbin_retention_obligation' => 'auto',
```

Hay varios modificadores

- **auto.** Configuración predeterminada. mantiene los archivos y las carpetas en la papelera durante 30 días y los elimina automáticamente en cualquier momento si se necesita espacio (nota: es posible que los archivos no se eliminen si no se necesita espacio).
- **D, auto.** Mantiene los archivos y las carpetas en la papelera durante D+ días; elimínalos en cualquier momento si necesitas espacio (nota: es posible que los archivos no se eliminen si no se necesita espacio)
- **auto, D.** Elimina todos los archivos en la papelera que tengan más de D días automáticamente.
- **D1, D2.** Mantiene los archivos y las carpetas en la papelera durante al menos D1 días y los elimina cuando superen los D2 días
- **disabled.** La limpieza automática de la papelera está deshabilitada, los archivos y las carpetas se conservarán para siempre