Nextcloud Server Administration Manual

version latest

The Nextcloud developers

June 16, 2025

Contents

Table of contents

Introduction

Welcome to the Nextcloud Server Administration Guide. This guide describes administration tasks for Nextcloud, the flexible open source file synchronization and sharing solution. Nextcloud includes the Nextcloud server, which runs on Linux, client applications for Microsoft Windows, macOS and Linux, and mobile clients for the Android and iOS operating systems.

Current editions of Nextcloud manuals are always available online at docs.nextcloud.com.

Nextcloud server is available:

- As a free, full featured community-supported server, with all enterprise features.
- Or with full enterprise support, including phone and email access to Nextcloud developers.

Videos and blogs

See the official Nextcloud channel on YouTube for tutorials, overviews, and conference videos.

Visit our blog for latest news and to learn more about what is going on in and around Nextcloud.

Target audience

This guide is for users who want to install, administer, and optimize their Nextcloud servers. To learn more about the Nextcloud Web user interface, and desktop and mobile clients, please refer to their respective manuals:

- Nextcloud User Manual
- Nextcloud Desktop Client

Release notes

See the official changelog for release notes.

Maintenance and release schedule

You can find the detailed schedule for major and maintenance releases at: detailed schedule.

Major releases

Major releases are typically scheduled once every 4 months with the first 10 weeks being the development phase followed by freeze phase with four beta release, two RCs and one final each one with an interval of 1 week. Specific dates for each release can be found on detailed schedule.

Major releases are planned to be actively maintained for at least 8 months after their release. For long term support options check out the Nextcloud Subscription offered by Nextcloud GmbH.

Maintenance releases

Maintenance releases are scheduled in a 4 week cycle with one week before the release date having the freeze and RC 1.

Critical changes

You can find important documentation for app developers here: https://docs.nextcloud.com/server/latest/developer_manual/app_publishing_maintenance/app_upgrade_guide/index.html Each document lists a link to the breaking changes of the corresponding release.

Installation and server configuration

System requirements

Server

For best performance, stability and functionality we have documented some recommendations for running a Nextcloud server.

> **Note**
>
> If you plan a setup for your organization and you rely on professional deployment consulting (e.g. efficient and reliable scaling) and support, we strongly recommend you to check out our enterprise support.

Platform	Options
Operating System (64bit)	• **Ubuntu 20.04 LTS** (recommended) • **Red Hat Enterprise Linux 8** (recommended) • Debian 10 (Buster) • SUSE Linux Enterprise Server 15 • openSUSE Leap 42.1+ • CentOS 8
Database	• **MySQL 8.0+ or MariaDB 10.2/10.3/10.4/10.5/10.6** (recommended) • Oracle Database 11g (*only as part of an enterprise subscription*) • PostgreSQL 9.6/10/11/12/13 • SQLite (*only recommended for testing and minimal-instances*)
Webserver	• **Apache 2.4 with** `mod_php` **or** `php-fpm` (recommended) • nginx with `php-fpm`
PHP Runtime	• 7.3 • 7.4 • **8.0** (*recommended*)

See Installation on Linux for minimum PHP-modules and additional software for installing Nextcloud.

CPU Architecture and OS

A 64bit CPU and 64bit OS is required for Nextcloud to run well.

Memory

Memory requirements for running a Nextcloud server are greatly variable, depending on the numbers of users, apps, files and volume of server activity.

Nextcloud needs a minimum of **128MB** RAM per process, and we recommend a minimum of **512MB** RAM per process.

Database requirements for MySQL / MariaDB

The following is currently required if you're running Nextcloud together with a MySQL / MariaDB database:

- InnoDB storage engine (MyISAM is not supported)
- “READ COMMITED” transaction isolation level (See: Database “READ COMMITTED” transaction isolation level)
- Disabled or BINLOG_FORMAT = ROW configured Binary Logging (See: https://dev.mysql.com/doc/refman/5.7/en/binary-log-formats.html)
- For **Emoji (UTF8 4-byte) support** see Enabling MySQL 4-byte support

Desktop client

We strongly recommend using the latest version of your operating system to get the full and most stable experience out of our clients.

- **Windows** 8.1+
- **macOS** Lion (10.7)+ (64-bit only)
- **Linux** (CentOS 6.5+, Ubuntu 14.04+, Fedora 21+, openSUSE 13, SUSE Linux Enterprise 11 SP3+, Debian 8 (Jessie)+, Red Hat Enterprise Linux 7)

Mobile apps

We strongly recommend using the latest version of your mobile operating system to get the full and most stable experience out of our mobile apps.

- **iOS** 11.x+
- **Android** 4.x+

Note

The separate Nextcloud Talk app requires iOS 10.x+ or Android 5.x+.

Web browser

For the best experience with the Nextcloud web interface, we recommend that you use the latest and supported version of a browser from this list, or one based on those:

- Microsoft **Edge**
- Mozilla **Firefox**
- Google **Chrome**/Chromium
- Apple **Safari**

Note

If you want to use Nextcloud Talk you should use Mozilla **Firefox** 52+ or Google **Chrome**/Chromium 49+ to have the full experience with video calls and screensharing. Google Chrome/Chromium requires an additional plugin for screensharing.

Deployment recommendations

Find up-to-date deployment recommendations for enterprises in our customer portal.

Installation on Linux

In case you prefer installing from the source tarball, you can setup Nextcloud from scratch using a classic LAMP stack (Linux, Apache, MySQL/MariaDB, PHP). This document provides a complete walk-through for installing Nextcloud on Ubuntu 18.04 LTS Server with Apache and MariaDB, using the Nextcloud .tar archive. This method is recommended to install Nextcloud.

Note

Admins of SELinux-enabled distributions such as CentOS, Fedora, and Red Hat Enterprise Linux may need to set new rules to enable installing Nextcloud. See SELinux configuration tips for a suggested configuration.

If you prefer a more automated installation of Nextcloud and there are no packages for your Linux distribution, you have the option to install the community Snap Package. This includes a full production-ready stack, will maintain your HTTPS certificates for you, and will automatically update as needed to stay secure. You can also use the Nextcloud VM scripts to install directly on a clean Ubuntu Server. It will setup everything for you and include scripts for automated installation of apps like; Collabora, OnlyOffice, Talk and so on. Please note that those two options are not officially supported by Nextcloud GmbH.

This installation guide is giving a general overview of required dependencies and their configuration. For a distribution specific setup guide have a look at the Example installation on Ubuntu 20.04 LTS and Example installation on CentOS 8.

Prerequisites for manual installation

The Nextcloud .tar archive contains all of the required PHP modules. This section lists all required and optional PHP modules. Consult the PHP manual for more information on modules. Your Linux distribution should have packages for all required modules. You can check the presence of a module by typing `php -m | grep -i <module_name>`. If you get a result, the module is present.

Required:

- PHP (7.3 or 7.4)
- PHP module ctype
- PHP module curl
- PHP module dom
- PHP module fileinfo (included with PHP)
- PHP module filter (only on Mageia and FreeBSD)
- PHP module GD
- PHP module hash (only on FreeBSD)
- PHP module JSON
- PHP module libxml (Linux package libxml2 must be >=2.7.0)
- PHP module mbstring
- PHP module openssl
- PHP module posix
- PHP module session
- PHP module SimpleXML
- PHP module XMLReader
- PHP module XMLWriter
- PHP module zip
- PHP module zlib

Database connectors (pick the one for your database:)

- PHP module pdo_sqlite (>= 3, usually not recommended for performance reasons)
- PHP module pdo_mysql (MySQL/MariaDB)
- PHP module pdo_pgsql (PostgreSQL)

Recommended packages:

- PHP module bz2 (recommended, required for extraction of apps)
- PHP module intl (increases language translation performance and fixes sorting of non-ASCII characters)

Required for specific apps:

- PHP module ldap (for LDAP integration)
- PHP module smbclient (SMB/CIFS integration, see SMB/CIFS)
- PHP module ftp (for FTP storage / external user authentication)
- PHP module imap (for external user authentication)
- PHP module bcmath (for passwordless login)
- PHP module gmp (for passwordless login)

Recommended for specific apps (*optional*):

- PHP module gmp (for SFTP storage)
- PHP module exif (for image rotation in pictures app)

For enhanced server performance (*optional*) select one of the following memcaches:

- PHP module apcu (>= 4.0.6)
- PHP module memcached
- PHP module redis (>= 2.2.6, required for Transactional File Locking)

See Memory caching to learn how to select and configure a memcache.

For preview generation (*optional*):

- PHP module imagick
- avconv or ffmpeg
- OpenOffice or LibreOffice

For command line processing (*optional*):

- PHP module pcntl (enables command interruption by pressing `ctrl-c`)

For command line updater (*optional*):

- PHP module phar (upgrades Nextcloud by running `sudo -u www-data php /var/www/nextcloud/updater/updater.phar`)

You don't need the WebDAV module for your Web server (i.e. Apache's `mod_webdav`), as Nextcloud has a built-in WebDAV server of its own, SabreDAV. If `mod_webdav` is enabled you must disable it for Nextcloud. (See Apache Web server configuration for an example configuration.)

Apache Web server configuration

Configuring Apache requires the creation of a single configuration file. On Debian, Ubuntu, and their derivatives, this file will be /etc/apache2/sites-available/nextcloud.conf. On Fedora, CentOS, RHEL, and similar systems, the configuration file will be /etc/httpd/conf.d/nextcloud.conf.

You can choose to install Nextcloud in a directory on an existing webserver, for example *https://www.example.com/nextcloud/*, or in a virtual host if you want Nextcloud to be accessible from its own subdomain such as *https://cloud.example.com/*.

To use the directory-based installation, put the following in your nextcloud.conf replacing the **Directory** and **Alias** filepaths with the filepaths appropriate for your system:

```
Alias /nextcloud "/var/www/nextcloud/"

<Directory /var/www/nextcloud/>
  Require all granted
  AllowOverride All
  Options FollowSymLinks MultiViews

  <IfModule mod_dav.c>
    Dav off
  </IfModule>
</Directory>
```

To use the virtual host installation, put the following in your nextcloud.conf replacing **ServerName**, as well as the **DocumentRoot** and **Directory** filepaths with values appropriate for your system:

```
<VirtualHost *:80>
  DocumentRoot /var/www/nextcloud/
  ServerName  your.server.com

  <Directory /var/www/nextcloud/>
    Require all granted
    AllowOverride All
    Options FollowSymLinks MultiViews

    <IfModule mod_dav.c>
      Dav off
    </IfModule>
  </Directory>
</VirtualHost>
```

On Debian, Ubuntu, and their derivatives, you should run the following command to enable the configuration:

```
a2ensite nextcloud.conf
```

Additional Apache configurations

- For Nextcloud to work correctly, we need the module `mod_rewrite`. Enable it by running:

  ```
  a2enmod rewrite
  ```

  Additional recommended modules are `mod_headers`, `mod_env`, `mod_dir` and `mod_mime`:

  ```
  a2enmod headers
  a2enmod env
  a2enmod dir
  a2enmod mime
  ```

  If you're running `mod_fcgi` instead of the standard `mod_php` also enable:

  ```
  a2enmod setenvif
  ```

- You must disable any server-configured authentication for Nextcloud, as it uses Basic authentication internally for DAV services. If you have turned on authentication on a parent folder (via e.g. an `AuthType Basic` directive), you can turn off the authentication specifically for the Nextcloud entry. Following the above example configuration file, add the following line in the `<Directory>` section:

  ```
  Satisfy Any
  ```

- When using SSL, take special note of the ServerName. You should specify one in the server configuration, as well as in the CommonName field of the certificate. If you want your Nextcloud to be reachable via the internet, then set both of these to the domain you want to reach your Nextcloud server.

- Now restart Apache:

```
service apache2 restart
```

- If you're running Nextcloud in a subdirectory and want to use CalDAV or CardDAV clients make sure you have configured the correct Service discovery URLs.

Pretty URLs

Pretty URLs remove the `index.php`-part in all Nextcloud URLs, for example in sharing links like `https://example.org/nextcloud/index.php/s/Sv1b7krAUqmF8QQ`, making URLs shorter and thus prettier.

`mod_env` and `mod_rewrite` must be installed on your webserver and the .htaccess must be writable by the HTTP user. To enable `mod_env` and `mod_rewrite`, run `sudo a2enmod env` and `sudo a2enmod rewrite`. Then you can set in the config.php two variables:

```
'overwrite.cli.url' => 'https://example.org/nextcloud',
'htaccess.RewriteBase' => '/nextcloud',
```

if your setup is available on `https://example.org/nextcloud` or:

```
'overwrite.cli.url' => 'https://example.org/',
'htaccess.RewriteBase' => '/',
```

if it isn't installed in a subfolder. Finally run this occ-command to update your .htaccess file:

```
sudo -u www-data php /var/www/nextcloud/occ maintenance:update:htaccess
```

After each update, these changes are automatically applied to the `.htaccess`-file.

Enabling SSL

Note

You can use Nextcloud over plain HTTP, but we strongly encourage you to use SSL/TLS to encrypt all of your server traffic, and to protect user's logins and data in transit.

Apache installed under Ubuntu comes already set-up with a simple self-signed certificate. All you have to do is to enable the ssl module and the default site. Open a terminal and run:

```
a2enmod ssl
a2ensite default-ssl
service apache2 reload
```

Note

Self-signed certificates have their drawbacks - especially when you plan to make your Nextcloud server publicly accessible. You might want to consider getting a certificate signed by a commercial signing authority. Check with your domain name registrar or hosting service for good deals on commercial certificates.

Installation wizard

After restarting Apache you must complete your installation by running either the graphical Installation Wizard, or on the command line with the `occ` command. To enable this, change the ownership on your Nextcloud directories to your HTTP user:

```
chown -R www-data:www-data /var/www/nextcloud/
```

> **Note**
>
> Admins of SELinux-enabled distributions may need to write new SELinux rules to complete their Nextcloud installation; see SELinux configuration tips.

To use `occ` see Installing from command line.

To use the graphical Installation Wizard see Installation wizard.

SELinux configuration tips

See SELinux configuration for a suggested configuration for SELinux-enabled distributions such as Fedora and CentOS.

php.ini configuration notes

Keep in mind that changes to `php.ini` may have to be configured on more than one ini file. This can be the case, for example, for the `date.timezone` setting.

php.ini - used by the Web server:

```
  /etc/php/7.4/apache2/php.ini
or
  /etc/php/7.4/fpm/php.ini
or ...
```

php.ini - used by the php-cli and so by Nextcloud CRON jobs:

```
/etc/php/7.4/cli/php.ini
```

> **Note**
>
> Path names have to be set in respect of the installed PHP (>= 7.3 or 7.4) as applicable.

php-fpm configuration notes

System environment variables

When you are using `php-fpm`, system environment variables like PATH, TMP or others are not automatically populated in the same way as when using `php-cli`. A PHP call like `getenv('PATH');` can therefore return an empty result. So you may need to manually configure environment variables in the appropropriate `php-fpm` ini/config file.

Here are some example root paths for these ini/config files:

Debian/Ubuntu/Mint	CentOS/Red Hat/Fedora
`/etc/php/7.4/fpm/`	`/etc/php-fpm.d/`

In both examples, the ini/config file is called `www.conf`, and depending on the distro version or customizations you have made, it may be in a subdirectory such as `pool.d`.

Usually, you will find some or all of the environment variables already in the file, but commented out like this:

```
;env[HOSTNAME] = $HOSTNAME
;env[PATH] = /usr/local/bin:/usr/bin:/bin
;env[TMP] = /tmp
;env[TMPDIR] = /tmp
;env[TEMP] = /tmp
```

Uncomment the appropriate existing entries. Then run `printenv PATH` to confirm your paths, for example:

```
$ printenv PATH
/home/user/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:
/sbin:/bin:/
```

If any of your system environment variables are not present in the file then you must add them.

Alternatively it is possible to use the environment variables of your system by modifying:

```
/etc/php/7.4/fpm/pool.d/www.conf
```

and uncommenting the line:

```
clear_env = no
```

When you are using shared hosting or a control panel to manage your Nextcloud VM or server, the configuration files are almost certain to be located somewhere else, for security and flexibility reasons, so check your documentation for the correct locations.

Please keep in mind that it is possible to create different settings for `php-cli` and `php-fpm`, and for different domains and Web sites. The best way to check your settings is with PHP version and information.

Maximum upload size

If you want to increase the maximum upload size, you will also have to modify your `php-fpm` configuration and increase the `upload_max_filesize` and `post_max_size` values. You will need to restart `php-fpm` and your HTTP server in order for these changes to be applied.

.htaccess notes for Apache

Nextcloud comes with its own `nextcloud/.htaccess` file. Because `php-fpm` can't read PHP settings in `.htaccess` these settings and permissions must be set in the `nextcloud/.user.ini` file.

Other Web servers

- NGINX configuration

Installing on Windows (virtual machine)

If you are using Windows, the easiest way to get Nextcloud up and running is using a virtual machine (VM). There are two options:

- **Enterprise/SME appliance**

Nextcloud GmbH maintains a free appliance built on the Univention Corporate Server (UCS) with easy graphical setup and web-based administration. It includes user management via LDAP, can replace an existing Active Directory setup and has optional ONLYOFFICE and Collabora Online integration, with many more applications available for easy and quick install.

It can be installed on hardware or run in a virtual machine using VirtualBox, VMWare (ESX) and KVM images.

Download the the Appliance here:

- Univention Corporate Server (UCS)

- **Home User/SME appliance**

The Nextcloud VM is maintained by T&M Hansson IT and several different versions are offered. Collabora, OnlyOffice, Full Text Search and other apps can easily be installed with the included scripts which you can choose to run during the first setup, or download them later and run it afterwards. You can find all the currently available automated app installations on GitHub.

The VM comes in different sizes and versions.

You can find all the available versions here.

For complete instructions and downloads see:

- Nextcloud VM (GitHub)
- Nextcloud VM (T&M Hansson IT)

Note

You can install the VM on several different operating systems as long as you can mount OVA, VMDK, or VHD/VHDX VM in your hypervisor. If you are using KVM then you need to install the VM from the scripts on GitHub. You can follow the instructions in the README.

Installing via Snap packages

A snap is a zip file containing an application together with its dependencies, and a description of how it should safely be run on your system, especially the different ways it should talk to other software. Most importantly snaps are designed to be secure, sandboxed, containerized applications isolated from the underlying system and from other applications.

To install the Nextcloud Snap Package, run the following command in a terminal:

```
sudo snap install nextcloud
```

Note

The snapd technology is the core that powers snaps, and it offers a new way to package, distribute, update and run OS components and applications on a Linux system. See more about snaps on snapcraft.io.

Installation via web installer on a VPS or web space

When you don't have access to the command line, for example at a web hosting or VMPS, an easy option is to use our web installer. This script can be found on our server installation page here.

The script checks the dependencies, downloads Nextcloud from the official server, unpacks it with the right permissions and the right user account. Finally, you will be redirected to the Nextcloud installer. Here a quick how-to:

1. Get the file from the installation page
2. Upload setup-nextcloud.php to your web space
3. Point your web browser to setup-nextcloud.php on your webspace
4. Follow the instructions and configure Nextcloud
5. Login to your newly created Nextcloud instance!

Note

that the installer uses the same Nextcloud version as available for the built in updater in Nextcloud. After a major release it can take up to a month before it becomes available through the web installer and the updater. This is done to spread the deployment of new major releases out over time.

Installation on TrueNAS

See the TrueNAS installation documentation.

Installation via install script

One of the easiest ways of installing is to use the Nextcloud VM scripts. It's basically just two steps:

1. Download the latest installation script.
2. Run the script with:

```
sudo bash nextcloud_install_production.sh
```

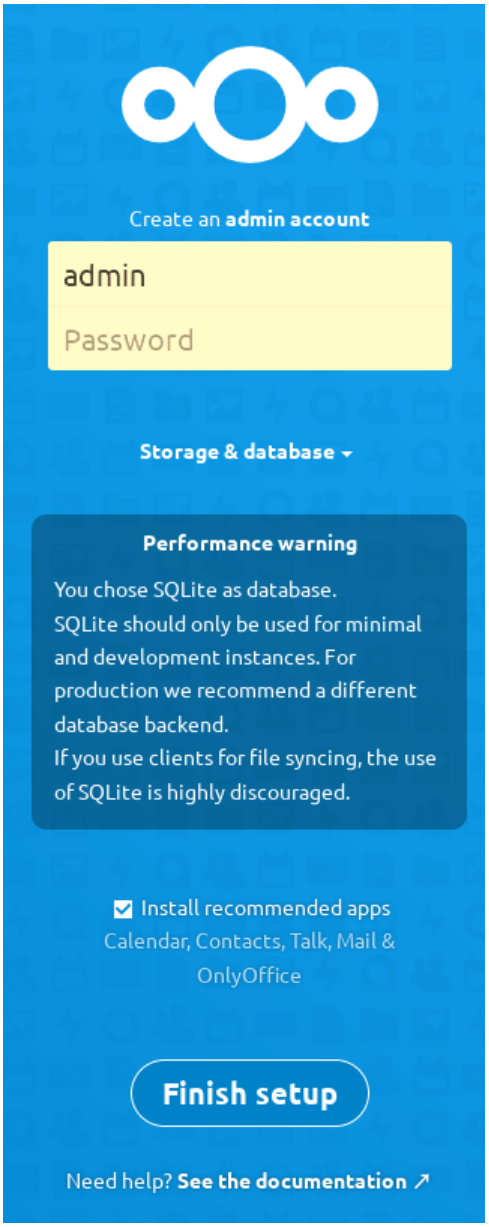
A guided setup will follow and the only thing you have to do it to follow the on screen instructions, when given to you.

Installation wizard

Quick start

When Nextcloud prerequisites are fulfilled and all Nextcloud files are installed, the last step to completing the installation is running the Installation Wizard. This is just three steps:

1. Point your Web browser to `http://localhost/nextcloud`
2. Enter your desired administrator's username and password.
3. Click **Finish Setup**.



You're finished and can start using your new Nextcloud server.

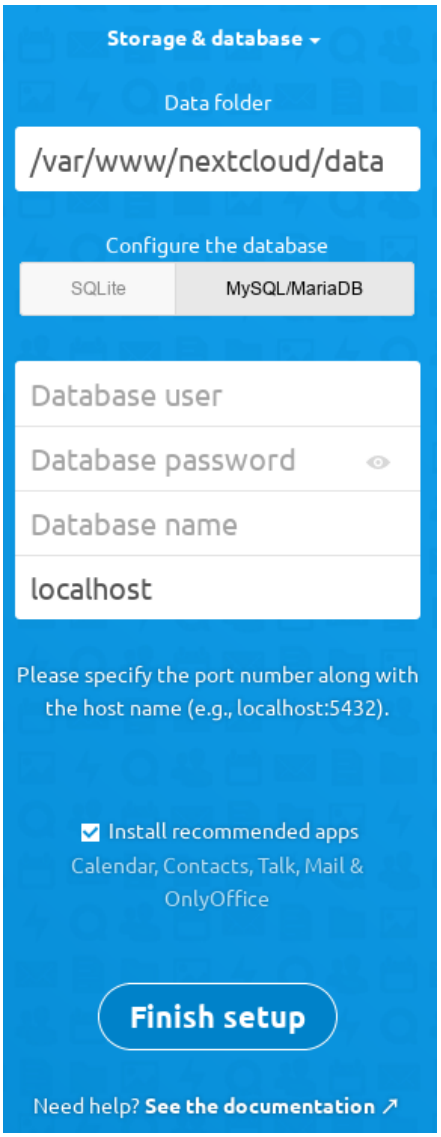
Of course, there is much more that you can do to set up your Nextcloud server for best performance and security. In the following sections we will cover important installation and post-installation steps.

- Data Directory Location

- Database Choice
- Trusted Domains

Data directory location

Click **Storage and Database** to expose additional installation configuration options for your Nextcloud data directory and database.



You should locate your Nextcloud data directory outside of your Web root if you are using an HTTP server other than Apache, or you may wish to store your Nextcloud data in a different location for other reasons (e.g. on a storage server). It is best to configure your data directory location at installation, as it is difficult to move after installation. You may put it anywhere; in this example is it located in `/opt/nextcloud/`. This directory must already exist, and must be owned by your HTTP user.

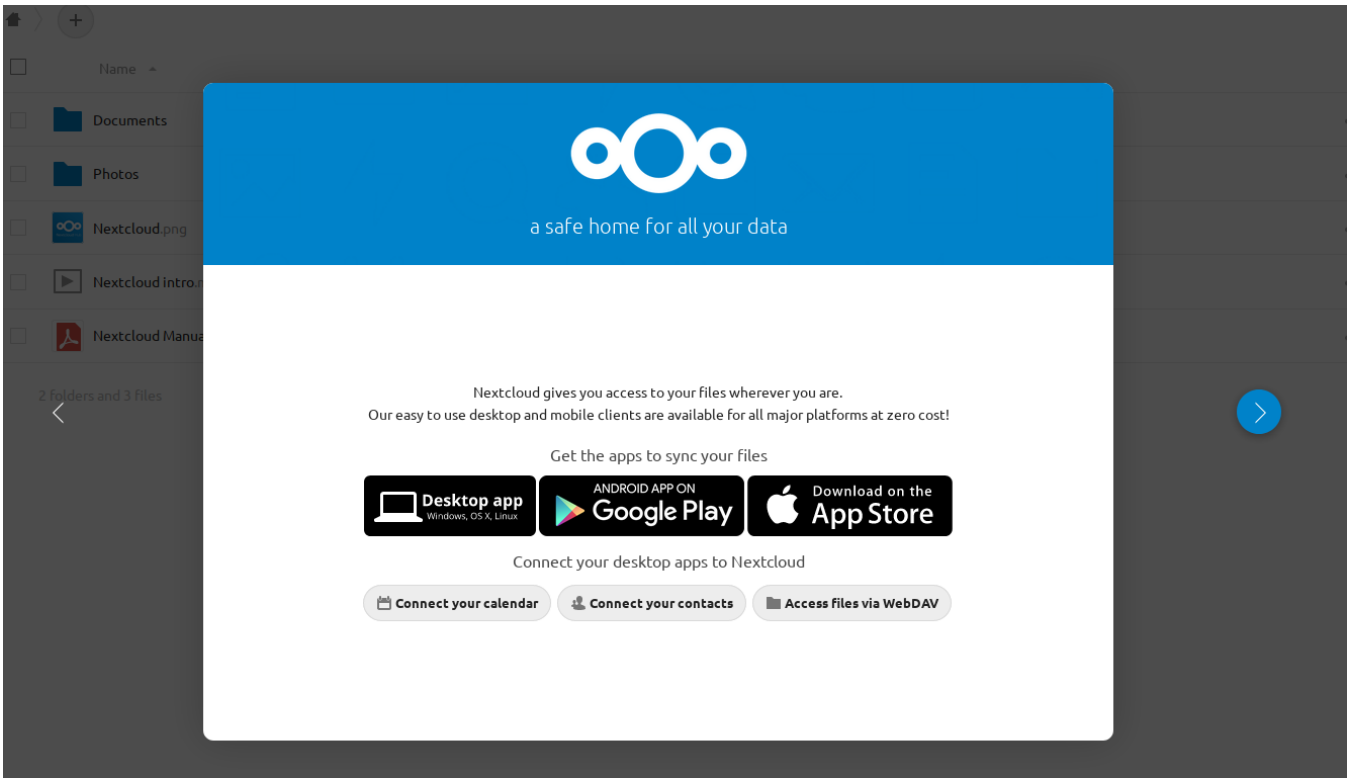
Database choice

SQLite is the default database for Nextcloud Server and it is good only for testing and lightweight single-user setups without client synchronization. Supported databases are MySQL, MariaDB, Oracle 11g, and PostgreSQL, and we recommend MySQL/MariaDB. Your database and PHP connectors must be installed before you run the Installation Wizard. When you install Nextcloud from packages all the necessary dependencies will be satisfied (see Installation on Linux for a detailed listing of required and optional PHP modules). You will need the root database login, or any administrator login , and then enter any name you want for your Nextcloud database. Be careful your administrator login needs to have the permissions to create and modify databases and they needs to have the permissions to grant permissions to other users.

After you enter your root or administrator login for your database, the installer creates a special database user with privileges limited to the Nextcloud database. Then Nextcloud needs only the special Nextcloud database user, and drops the root dB login. This user is named for your Nextcloud admin user, with an `oc_` prefix, and then given a random password. The Nextcloud database user and password are written into `config.php`:

```
'dbuser' => 'oc_molly',
'dbpassword' => 'pX65Ty5DrHQkYPE5HRsDvyFHlZZHcm',
```

Click Finish Setup, and start using your new Nextcloud server.



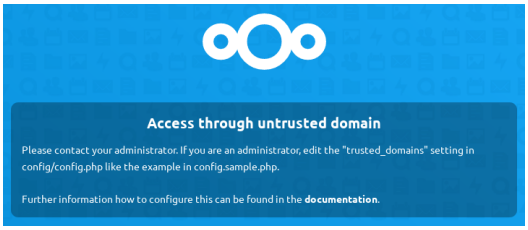
Now we will look at some important post-installation steps.

Trusted domains

All URLs used to access your Nextcloud server must be whitelisted in your `config.php` file, under the `trusted_domains` setting. Users are allowed to log into Nextcloud only when they point their browsers to a URL that is listed in the `trusted_domains` setting. You may use IP addresses and domain names. A typical configuration looks like this:

```
'trusted_domains' =>
  array (
    0 => 'localhost',
    1 => 'server1.example.com',
    2 => '192.168.1.50',
    3 => '[fe80::1:50]',
),
```

The loopback address, `127.0.0.1`, is automatically whitelisted, so as long as you have access to the physical server you can always log in. In the event that a load balancer is in place there will be no issues as long as it sends the correct X-Forwarded-Host header. When a user tries a URL that is not whitelisted the following error appears:



Installing from command line

It is now possible to install Nextcloud entirely from the command line. This is convenient for scripted operations, headless servers, and sysadmins who prefer the command line. There are three stages to installing Nextcloud via the command line:

1. Download the Nextcloud code and unpack the tarball in the appropriate directories. (See Installation on Linux.)

2. Change the ownership of your `nextcloud` directory to your HTTP user, like this example for Debian/Ubuntu. You must run `occ` as your HTTP user; see Run occ as your HTTP user:

```
$ sudo chown -R www-data:www-data /var/www/nextcloud/
```

3. Use the `occ` command to complete your installation. This takes the place of running the graphical Installation Wizard:

```
$ cd /var/www/nextcloud/
$ sudo -u www-data php occ  maintenance:install --database \
"mysql" --database-name "nextcloud"  --database-user "root" --database-pass \
"password" --admin-user "admin" --admin-pass "password"
Nextcloud is not installed - only a limited number of commands are available
Nextcloud was successfully installed
```

Note that you must change to the root Nextcloud directory, as in the example above, to run `occ  maintenance:install`, or the installation will fail with a PHP fatal error message.

Supported databases are:

```
- sqlite (SQLite3 - Nextcloud Community edition only)
- mysql (MySQL/MariaDB)
- pgsql (PostgreSQL)
- oci (Oracle 11g currently only possible if you contact us at https://nextcloud.com/enterpr
```

See Command line installation for more information.

Supported apps

Below is the list of apps supported for Nextcloud latest. Supported here means that we'll accept bugreports and resolve them in these apps with regard to functionality and compatibility with Nextcloud latest. To get access to work-arounds, long term support, priority bug fixing and custom consulting, contact Nextcloud GmbH.

All apps are licensed under `AGPLv3`.

Nextcloud Files

- Accessibility
- Activity
- Auditing / Logging
- Antivirus for files
- Brute-force settings
- Circles
- Collaborative Tags
- Comments
- Contacts Interaction
- Data Request
- Default encryption module
- Deleted files
- External Sites

- External storage support
- Federated file sharing
- Federation
- File sharing
- Files access control
- Files
- Files Automated Tagging
- First run wizard
- Flow
- Full Text Search
- Full Text Search - Elasticsearch Platform
- Full Text Search - Files
- Full Text Search - Tesseract OCR
- Group folders
- Guests
- LDAP user and group backend
- Log Reader
- Monitoring
- Nextcloud announcements
- Notifications
- Password policy
- PDF Viewer
- Photos
- Provisioning API
- Recommendations
- Right click
- Share by mail
- SharePoint Backend
- SSO & SAML authentication
- **Social sharing via:**
    - Diaspora
    - Email
    - Facebook
    - Twitter
- Terms of Service
- Text
- Theming
- Two factor backup codes
- Two factor TOTP Provider
- Two factor U2F

- Update Notifications
- Versions
- Video Player
- WebDAV endpoint

Nextcloud Groupware

- Calendar
- Contacts
- Deck
- Mail

Nextcloud Talk

- Talk

Collaborative editing

- Collabora Online
- ONLYOFFICE

Global Scale

- Global Site Selector
- Lookup Server Connector

SELinux configuration

When you have SELinux enabled on your Linux distribution, you may run into permissions problems after a new Nextcloud installation, and see `permission denied` errors in your Nextcloud logs.

The following settings should work for most SELinux systems that use the default distro profiles. Run these commands as root, and remember to adjust the filepaths in these examples for your installation:

```
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/html/nextcloud/data(/.*)?'
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/html/nextcloud/config(/.*)?'
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/html/nextcloud/apps(/.*)?'
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/html/nextcloud/.htaccess'
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/html/nextcloud/.user.ini'
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/html/nextcloud/3rdparty/aws/aws-sdk

restorecon -Rv '/var/www/html/nextcloud/'
```

If you uninstall Nextcloud you need to remove the Nextcloud directory labels. To do this execute the following commands as root after uninstalling Nextcloud:

```
semanage fcontext -d '/var/www/html/nextcloud/data(/.*)?'
semanage fcontext -d '/var/www/html/nextcloud/config(/.*)?'
semanage fcontext -d '/var/www/html/nextcloud/apps(/.*)?'
semanage fcontext -d '/var/www/html/nextcloud/.htaccess'
semanage fcontext -d '/var/www/html/nextcloud/.user.ini'
semanage fcontext -d '/var/www/html/nextcloud/3rdparty/aws/aws-sdk-php/src/data/logs(/.*)?'

restorecon -Rv '/var/www/html/nextcloud/'
```

If you have customized SELinux policies and these examples do not work, you must give the HTTP server write access to these directories:

```
/var/www/html/nextcloud/data
/var/www/html/nextcloud/config
/var/www/html/nextcloud/apps
```

Enable updates via the web interface

To enable updates via the web interface, you may need this to enable writing to the directories:

```
setsebool httpd_unified on
```

When the update is completed, disable write access:

```
setsebool -P  httpd_unified  off
```

Disallow write access to the whole web directory

For security reasons it's suggested to disable write access to all folders in /var/www/ (default):

```
setsebool -P  httpd_unified  off
```

Allow access to a remote database

An additional setting is needed if your installation is connecting to a remote database:

```
setsebool -P httpd_can_network_connect_db on
```

Allow access to LDAP server

Use this setting to allow LDAP connections:

```
setsebool -P httpd_can_connect_ldap on
```

Allow access to remote network

Nextcloud requires access to remote networks for functions such as Server-to-Server sharing, external storages or the app store. To allow this access use the following setting:

```
setsebool -P httpd_can_network_connect on
```

Allow access to network memcache

This setting is not required if `httpd_can_network_connect` is already on:

```
setsebool -P httpd_can_network_memcache on
```

Allow access to SMTP/sendmail

If you want to allow Nextcloud to send out e-mail notifications via sendmail you need to use the following setting:

```
setsebool -P httpd_can_sendmail on
```

Allow access to CIFS/SMB

If you have placed your datadir on a CIFS/SMB share use the following setting:

```
setsebool -P httpd_use_cifs on
```

Allow access to FuseFS

If your data folder resides on a Fuse Filesystem (e.g. EncFS etc), this setting is required as well:

```
setsebool -P httpd_use_fusefs on
```

Allow access to GPG for Rainloop

If you use a the rainloop webmail client app which supports GPG/PGP, you might need this:

```
setsebool -P httpd_use_gpg on
```

Troubleshooting

For general Troubleshooting of SELinux and its profiles try to install the package `setroubleshoot` and run:

```
sealert -a /var/log/audit/audit.log > /path/to/mylogfile.txt
```

to get a report which helps you configuring your SELinux profiles.

Another tool for troubleshooting is to enable a single ruleset for your Nextcloud directory:

```
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/html/nextcloud(/.*)?'
restorecon -RF /var/www/html/nextcloud
```

It is much stronger security to have a more fine-grained ruleset as in the examples at the beginning, so use this only for testing and troubleshooting. It has a similar effect to disabling SELinux, so don't use it on production systems.

NGINX configuration

> **Warning**
>
> Please note that webservers other than Apache 2.x are not officially supported.

> **Note**
>
> This page covers example NGINX configurations to run a Nextcloud server. These configurations examples were originally provided by @josh4trunks and are exclusively community-maintained. (Thank you contributors!)

- You need to insert the following code into **your Nginx configuration file.**
- Adjust **server_name**, **root**, **ssl_certificate** and **ssl_certificate_key** to suit your needs.
- Make sure your SSL certificates are readable by the server (see nginx HTTP SSL Module documentation).
- Be careful about line breaks if you copy the examples, as long lines may be broken for page formatting.
- 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/`

```
upstream php-handler {
    server 127.0.0.1:9000;
    #server unix:/var/run/php/php7.4-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;" al

    # set max upload size and increase upload timeout:
    client_max_body_size 512M;
    client_body_timeout 300s;
    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+j

    # 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(?:$|/) {
        # Required for legacy support
        rewrite ^/(?!index|remote|public|cron|core\/ajax\/update|status|ocs\/v[12]|updater\/

        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
        fastcgi_param front_controller_active true;     # Enable pretty urls
```

```
        fastcgi_pass php-handler;

        fastcgi_intercept_errors on;
        fastcgi_request_buffering off;

        fastcgi_max_temp_file_size 0;
    }

    location ~ \.(?:css|js|svg|gif|png|jpg|ico|wasm|tflite|map)$ {
        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 ~ \.wasm$ {
            default_type application/wasm;
        }
    }

    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;
    }
}
```

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.

```
upstream php-handler {
    server 127.0.0.1:9000;
    #server unix:/var/run/php/php7.4-fpm.sock;
}

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

```
    server_name cloud.example.com;

    # Enforce HTTPS just for `/nextcloud`
    location /nextcloud {
        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;" al

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

    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;

        # 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/

    # 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;

    # 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`, `/ocm-provider`, `/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)(?:$|/)
    location ~ ^/nextcloud/(?:\.|autotest|occ|issue|indie|db_|console)

    # 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
    # `/nextcloud/index.php` to the URI, resulting in a HTTP 500 error response.
    location ~ \.php(?:$|/) {
        # Required for legacy support
        rewrite ^/nextcloud/(?!index|remote|public|cron|core\/ajax\/update|status|ocs\/v

        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 hea
        fastcgi_param front_controller_active true;     # Enable pretty urls
        fastcgi_pass php-handler;

        fastcgi_intercept_errors on;
        fastcgi_request_buffering off;

        fastcgi_max_temp_file_size 0;
    }

    location ~ \.(?:css|js|svg|gif|png|jpg|ico|wasm|tflite|map)$ {
        try_files $uri /nextcloud/index.php$request_uri;
        expires 6M;         # Cache-Control policy borrowed from `.htaccess`
        access_log off;     # Optional: Don't log access to assets

        location ~ \.wasm$ {
            default_type application/wasm;
        }
    }

    location ~ \.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

Suppressing log messages

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

```
location = /data/htaccesstest.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 an 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 <https://github.com/nextcloud/server/issues/8802>* for more information.

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 wsdlcache directory. Try to check permissions and execute change if needed:

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

Hardening and security guidance

Nextcloud aims to ship with secure defaults that do not need to get modified by administrators. However, in some cases some additional security hardening can be applied in scenarios were the administrator has complete control over the Nextcloud instance. This page assumes that you run Nextcloud Server on Apache2 in a Linux environment.

Note

Nextcloud will warn you in the administration interface if some critical security-relevant options are missing. However, it is still up to the server administrator to review and maintain system security.

Passwords

Storage of access tokens

Upon successful authentication, Nextcloud issues an access token that clients will use for all future HTTP requests. This access token uniquely identifies a user and should not be stored on any system other than the client requesting it. The user password is also stored encrypted in the Nextcloud database. For encryption of the password, the token and an instance-specific secret is used.

Leakage of the access token can have negative security consequences. Depending on the data access by the actor, the risk here is different:

- An actor with access to only the access token can impersonate users and login as them.
- An actor with access to the access token, the Nextcloud config file, and the Nextcloud database can decrypt user passwords stored in the database.

Limit on password length

Nextcloud uses the bcrypt algorithm, and thus for security and performance reasons, e.g. Denial of Service as CPU demand increases exponentially, it only verifies the first 72 characters of passwords. This applies to all passwords that you use in Nextcloud: user passwords, passwords on link shares, and passwords on external shares.

Operating system

Give PHP read access to `/dev/urandom`

Nextcloud uses a RFC 4086 ("Randomness Requirements for Security") compliant mixer to generate cryptographically secure pseudo-random numbers. This means that when generating a random number Nextcloud will request multiple random numbers from different sources and derive from these the final random number.

The random number generation also tries to request random numbers from `/dev/urandom`, thus it is highly recommended to configure your setup in such a way that PHP is able to read random data from it.

Note

When having an `open_basedir` configured within your `php.ini` file, make sure to include `/dev/urandom`.

Enable hardening modules such as SELinux

It is highly recommended to enable hardening modules such as SELinux where possible. See SELinux configuration to learn more about SELinux.

Deployment

Place data directory outside of the web root

It is highly recommended to place your data directory outside of the Web root (i.e. outside of `/var/www`). It is easiest to do this on a new installation.

Disable preview image generation

Nextcloud is able to generate preview images of common filetypes such as images or text files. By default the preview generation for some file types that we consider secure enough for deployment is enabled by default. However, administrators should be aware that these previews are generated using PHP libraries written in C which might be vulnerable to attack vectors.

For high security deployments we recommend disabling the preview generation by setting the `enable_previews` switch to `false` in `config.php`. As an administrator you are also able to manage which preview providers are enabled by modifying the `enabledPreviewProviders` option switch.

Use HTTPS

Using Nextcloud without using an encrypted HTTPS connection opens up your server to a man-in-the-middle (MITM) attack, and risks the interception of user data and passwords. It is a best practice, and highly recommended, to always use HTTPS on production servers, and to never allow unencrypted HTTP.

How to setup HTTPS on your Web server depends on your setup; please consult the documentation for your HTTP server. The following examples are for Apache.

Redirect all unencrypted traffic to HTTPS

To redirect all HTTP traffic to HTTPS administrators are encouraged to issue a permanent redirect using the 301 status code. When using Apache this can be achieved by a setting such as the following in the Apache VirtualHosts configuration:

```
<VirtualHost *:80>
   ServerName cloud.nextcloud.com
   Redirect permanent / https://cloud.nextcloud.com/
</VirtualHost>
```

Enable HTTP Strict Transport Security

While redirecting all traffic to HTTPS is good, it may not completely prevent man-in-the-middle attacks. Thus administrators are encouraged to set the HTTP Strict Transport Security header, which instructs browsers to not allow any connection to the Nextcloud instance using HTTP, and it attempts to prevent site visitors from bypassing invalid certificate warnings.

This can be achieved by setting the following settings within the Apache VirtualHost file:

```
<VirtualHost *:443>
  ServerName cloud.nextcloud.com
    <IfModule mod_headers.c>
      Header always set Strict-Transport-Security "max-age=15552000; includeSubDomains"
    </IfModule>
 </VirtualHost>
```

Warning

We recommend the additional setting `; preload` to be added to that header. Then the domain will be added to a hardcoded list that is shipped with all major browsers and enforce HTTPS upon those domains. See the HSTS preload website for more information. Due to the policy of this list you need to add it to the above example for yourself once you are sure that this is what you want. Removing the domain from this list could take some months until it reaches all installed browsers.

This example configuration will make all subdomains only accessible via HTTPS. If you have subdomains not accessible via HTTPS, remove `includeSubDomains`.

This requires the `mod_headers` extension in Apache.

Proper SSL configuration

Default SSL configurations by Web servers are often not state-of-the-art, and require fine-tuning for an optimal performance and security experience. The available SSL ciphers and options depend completely on your environment and thus giving a generic recommendation is not really possible.

We recommend using the Mozilla SSL Configuration Generator to generate a suitable configuration suited for your environment. To verify your configuration you can use the free Web TLS Profiler service. This service gives detailed error messages, if your server's TLS settings deviate from the Mozilla Configuration. Another useful tool to check your server's TLS configuration is the free Qualys SSL Labs Test which provides general information about the TLS settings.

Also ensure that HTTP compression is disabled to mitigate the BREACH attack.

Use a dedicated domain for Nextcloud

Administrators are encouraged to install Nextcloud on a dedicated domain such as cloud.domain.tld instead of domain.tld to gain all the benefits offered by the Same-Origin-Policy.

Ensure that your Nextcloud instance is installed in a DMZ

As Nextcloud supports features such as Federated File Sharing we do not consider Server Side Request Forgery (SSRF) part of our threat model. In fact, given all our external storage adapters this can be considered a feature and not a vulnerability.

This means that a user on your Nextcloud instance could probe whether other hosts are accessible from the Nextcloud network. If you do not want this you need to ensure that your Nextcloud is properly installed in a segregated network and proper firewall rules are in place.

Serve security related headers by the Web server

Basic security headers are served by Nextcloud already in a default environment. These include:

- `X-Content-Type-Options: nosniff`
  - Instructs some browsers to not sniff the mimetype of files. This is used for example to prevent browsers from interpreting text files as JavaScript.
- `X-XSS-Protection: 1; mode=block`
  - Instructs browsers to enable their browser side Cross-Site-Scripting filter.
- `X-Robots-Tag: none`
  - Instructs search machines to not index these pages.
- `X-Frame-Options: SAMEORIGIN`
  - Prevents embedding of the Nextcloud instance within an iframe from other domains to prevent Clickjacking and other similar attacks.
- `Referrer-Policy: no-referrer`
  - The default *no-referrer* policy instructs the browser not to send referrer information along with requests to any origin.

These headers are hard-coded into the Nextcloud server, and need no intervention by the server administrator.

For optimal security, administrators are encouraged to serve these basic HTTP headers by the Web server to enforce them on response. To do this Apache has to be configured to use the `.htaccess` file and the following Apache modules need to be enabled:

- mod_headers
- mod_env

Administrators can verify whether this security change is active by accessing a static resource served by the Web server and verify that the above mentioned security headers are shipped.

Connections to remote servers

Some Nextcloud functionality requires connecting to remote servers. Depending on your server setup those are possible connections:

- www.nextcloud.com, www.startpage.com, www.eff.org, www.edri.org for checking the internet connection
- apps.nextcloud.com for the available apps
- updates.nextcloud.com for Nextcloud updates
- lookup.nextcloud.com For updating and lookup in the federated sharing addressbook
- push-notifications.nextcloud.com for sending push notifications to mobile clients
- surveyserver.nextcloud.com if the admin has agreed to share anonymized data
- Any remote Nextcloud server that is connected with federated sharing

Setup fail2ban

Exposing your server to the internet will inevitably lead to the exposure of the services running on the internet-exposed ports to brute force login attempts.

Fail2ban is a service that uses iptables to automatically drop connections for a pre-defined amount of time from IPs that continuously failed to authenticate to the configured services.

In order to setup fail2ban, you first need to download and install it on your server. Downloads for several distributions can be found on fail2ban download page. It is often available from most distributions' package managers (e.g. `apt-get`).

The standard path for fail2ban's configuration is `/etc/fail2ban`.

Setup a filter and a jail for Nextcloud

A filter defines regex rules to identify when users fail to authenticate on Nextcloud's user interface, WebDAV, or use an untrusted domain to access the server.

Create a file in `/etc/fail2ban/filter.d` named `nextcloud.conf` with the following contents:

```
[Definition]
_groupsre = (?:(?:,?\s*"\w+":(?:"[^"]+"|\w+))*)
failregex = ^\{%(_groupsre)s,?\s*"remoteAddr":"<HOST>"%(_groupsre)s,?\s*"message":"Login fai
            ^\{%(_groupsre)s,?\s*"remoteAddr":"<HOST>"%(_groupsre)s,?\s*"message":"Trusted d
datepattern = ,?\s*"time"\s*:\s*"%%Y-%%m-%%d[T ]%%H:%%M:%%S(%%z)?"
```

The jail file defines how to handle the failed authentication attempts found by the Nextcloud filter.

Create a file in `/etc/fail2ban/jail.d` named `nextcloud.local` with the following contents:

```
[nextcloud]
backend = auto
enabled = true
port = 80,443
protocol = tcp
filter = nextcloud
maxretry = 3
bantime = 86400
findtime = 43200
logpath = /path/to/data/directory/nextcloud.log
```

Ensure to replace `logpath` with your installation's `nextcloud.log` location. If you are using ports other than `80` and `443` for your Web server you should replace those too. The `bantime` and `findtime` are defined in seconds.

Restart the fail2ban service. You can check the status of your Nextcloud jail by running:

```
fail2ban-client status nextcloud
```

Server tuning

Using cron to perform background jobs

See Background jobs for a description and the benefits.

Reducing system load

High system load will slow down Nextcloud and might also lead to other unwanted side effects. To reduce load you should first identify the source of the problem. Tools such as htop, iotop, netdata or glances will help to identify the process or the drive that slows down your system. First you should make sure that you installed/assigned enough RAM. Swap usage should be prevented by all means. If you run your database inside a VM, you should not store it inside a VM image file. Better put it on a dedicated block device to reduce latency due to multiple abstraction layers.

Caching

Caching improves performance by storing data, code, and other objects in memory. Memory cache configuration for the Nextcloud server must be installed and configured. See Memory caching.

Using MariaDB/MySQL instead of SQLite

MySQL or MariaDB are preferred because of the performance limitations of SQLite with highly concurrent applications, like Nextcloud.

See the section Database configuration for how to configure Nextcloud for MySQL or MariaDB. If your installation is already running on SQLite then it is possible to convert to MySQL or MariaDB using the steps provided in Converting database type.

For more details and help tuning your database, check this article at MariaDB.

Using Redis-based transactional file locking

File locking is enabled by default, using the database locking backend. This places a significant load on your database. See the section Transactional file locking for how to configure Nextcloud to use Redis-based Transactional File Locking.

TLS / encryption app

TLS (HTTPS) and file encryption/decryption can be offloaded to a processor's AES-NI extension. This can both speed up these operations while lowering processing overhead. This requires a processor with the AES-NI instruction set.

Here are some examples how to check if your CPU / environment supports the AES-NI extension:

- For each CPU core present: `grep flags /proc/cpuinfo` or as a summary for all cores: `grep -m 1 '^flags' /proc/cpuinfo` If the result contains any `aes`, the extension is present.
- Search eg. on the Intel web if the processor used supports the extension Intel Processor Feature Filter You may set a filter by `"AES New Instructions"` to get a reduced result set.
- For versions of openssl >= 1.0.1, AES-NI does not work via an engine and will not show up in the `openssl engine` command. It is active by default on the supported hardware. You can check the openssl version via `openssl version -a`
- If your processor supports AES-NI but it does not show up eg via grep or coreinfo, it is maybe disabled in the BIOS.
- If your environment runs virtualized, check the virtualization vendor for support.

Enable HTTP/2 for faster loading

HTTP/2 has huge speed improvements over HTTP with multiple request. Most browsers already support HTTP/2 over TLS (HTTPS). Refer to your web server manual for guides on how to enable HTTP/2.

Tune PHP-FPM

If you are using a default installation of PHP-FPM you might have noticed excessive load times on the web interface or even sync issues. This is due to the fact that each simultaneous request of an element is handled by a separate PHP-FPM process. So even on a small installation you should allow more processes to run in parallel to handle the requests.

This link can help you calculate the good values for your system.

Enable PHP OPcache

The OPcache improves the performance of PHP applications by caching precompiled bytecode. The default OPcache settings are usually sufficient for Nextcloud code to be fully cached. If any cache size limit is reached by more than 90%, the admin panel will show a related warning. Nextcloud strictly requires code comments to be preserved in opcode, which is the default. But in case PHP settings are changed on your system, you may need set the following:

```
opcache.save_comments = 1
```

By default, cached scripts are revalidated on access to ensure that changes on disk take effect after at most 2 seconds. Since Nextcloud handles cache revalidation internally when required, the revalidation frequency can be reduced or completely disabled to enhance performance. Note, however, that it affects manual changes to scripts, including `config.php`. To check for changes at most every `60` seconds, use the following setting:

```
opcache.revalidate_freq = 60
```

To disable the revalidation completely:

```
opcache.validate_timestamps = 0
```

Any change to `config.php` will then require either restarting PHP, manually clearing the cache, or invalidating this particular script.

For more details check out the official documentation. To monitor OPcache usage, clear individual or all cache entries, opcache-gui can be used.

Example installation on Ubuntu 20.04 LTS

You can use .deb packages to install the required and recommended modules for a typical Nextcloud installation, using Apache and MariaDB, by issuing the following commands in a terminal:

```
sudo apt update
sudo apt install apache2 mariadb-server libapache2-mod-php7.4
sudo apt install php7.4-gd php7.4-mysql php7.4-curl php7.4-mbstring php7.4-intl
sudo apt install php7.4-gmp php7.4-bcmath php-imagick php7.4-xml php7.4-zip
```

- This installs the packages for the Nextcloud core system. If you are planning on running additional apps, keep in mind that they might require additional packages. See Prerequisites for manual installation for details.

Now you need to create a database user and the database itself by using the MySQL command line interface. The database tables will be created by Nextcloud when you login for the first time.

To start the MySQL command line mode use the following command and press the enter key when prompted for a password:

```
sudo /etc/init.d/mysql start
sudo mysql -uroot -p
```

Then a **MariaDB [root]>** prompt will appear. Now enter the following lines, replacing username and password with appropriate values, and confirm them with the enter key:

```
CREATE USER 'username'@'localhost' IDENTIFIED BY 'password';
CREATE DATABASE IF NOT EXISTS nextcloud CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;
GRANT ALL PRIVILEGES ON nextcloud.* TO 'username'@'localhost';
FLUSH PRIVILEGES;
```

You can quit the prompt by entering:

```
quit;
```

Now download the archive of the latest Nextcloud version:

- Go to the Nextcloud Download Page.
- Go to **Download Nextcloud Server > Download > Archive file for server owners** and download either the tar.bz2 or .zip archive.

- This downloads a file named nextcloud-x.y.z.tar.bz2 or nextcloud-x.y.z.zip (where x.y.z is the version number).
- Download its corresponding checksum file, e.g. nextcloud-x.y.z.tar.bz2.md5, or nextcloud-x.y.z.tar.bz2.sha256.
- Verify the MD5 or SHA256 sum:

```
md5sum -c nextcloud-x.y.z.tar.bz2.md5 < nextcloud-x.y.z.tar.bz2
sha256sum -c nextcloud-x.y.z.tar.bz2.sha256 < nextcloud-x.y.z.tar.bz2
md5sum  -c nextcloud-x.y.z.zip.md5 < nextcloud-x.y.z.zip
sha256sum  -c nextcloud-x.y.z.zip.sha256 < nextcloud-x.y.z.zip
```

- You may also verify the PGP signature:

```
wget https://download.nextcloud.com/server/releases/nextcloud-x.y.z.tar.bz2.asc
wget https://nextcloud.com/nextcloud.asc
gpg --import nextcloud.asc
gpg --verify nextcloud-x.y.z.tar.bz2.asc nextcloud-x.y.z.tar.bz2
```

- Now you can extract the archive contents. Run the appropriate unpacking command for your archive type:

```
tar -xjvf nextcloud-x.y.z.tar.bz2
unzip nextcloud-x.y.z.zip
```

- This unpacks to a single `nextcloud` directory. Copy the Nextcloud directory to its final destination. When you are running the Apache HTTP server you may safely install Nextcloud in your Apache document root:

```
cp -r nextcloud /path/to/webserver/document-root
```

  where `/path/to/webserver/document-root` is replaced by the document root of your Web server:

```
cp -r nextcloud /var/www
```

On other HTTP servers it is recommended to install Nextcloud outside of the document root.

Next steps

After installing the prerequisites and extracting the nextcloud directory, you should follow the instructions for Apache configuration at Apache Web server configuration. Once Apache is installed, you can optionally follow the Installation on Linux guide from Pretty URLs until Other Web servers

Example installation on CentOS 8

In this install tutorial we will be deploying CentOS 8, PHP 7.4, MariaDB, Redis as memcache and Nextcloud running on Apache.

Start off by installing a CentOS 8 minimal install. This should provide a sufficient platform to run a successful Nextcloud instance.

First install some dependencies you will be needing during installation, but which will also be useful in every day use situations:

```
dnf install -y epel-release yum-utils unzip curl wget \
bash-completion policycoreutils-python-utils mlocate bzip2
```

Now make sure your system is up to date:

```
dnf update -y
```

Apache

```
dnf install -y httpd
```

Create a virtualhost in `/etc/httpd/conf.d/nextcloud.conf` and add the following content to it:

```
<VirtualHost *:80>
  DocumentRoot /var/www/html/nextcloud/
  ServerName  your.server.com
```

```
  <Directory /var/www/html/nextcloud/>
    Require all granted
    AllowOverride All
    Options FollowSymLinks MultiViews

    <IfModule mod_dav.c>
      Dav off
    </IfModule>

  </Directory>
</VirtualHost>
```

See Apache Web server configuration for further details.

Make sure the apache web service is enabled and started:

```
systemctl enable httpd.service
systemctl start httpd.service
```

PHP

> **Note**
>
> CentOS 8 doesn't come with packages for the redis and imagick php extensions. Those can either be installed using pecl. Apart from the official PHP packages there are 3rdparty repositories available at `https://rpms.remirepo.net`. Using remirepo you can also install the latest PHP version instead of the standard shipped one.

Setting up remirepo with PHP 7.4

More details can be found on `https://blog.remirepo.net/pages/Config-en`

Command to install the Remi repository configuration package:

```
dnf install https://rpms.remirepo.net/enterprise/remi-release-8.rpm
```

Command to install the yum-utils package (for the yum-config-manager command):

```
dnf install yum-utils
```

You want a single version which means replacing base packages from the distribution. Packages have the same name than the base repository, ie php-*. Some common dependencies are available in remi-safe repository, which is enabled by default.

You have to enable the module stream for 7.4:

```
dnf module reset php
dnf module install php:remi-7.4
dnf update
```

Installing PHP and the required modules

Next install the PHP modules needed for this install. Remember, because this is a limited basic install, we only install the neccessary modules, not all of them. If you are making a more complete install, please refer to PHP module list in the source installation documentation, Installation on Linux.:

```
dnf install -y php php-gd php-mbstring php-intl php-pecl-apcu\
     php-mysqlnd php-opcache php-json php-zip
```

Installing optional modules redis/imagick

```
dnf install -y php-redis php-imagick
```

Database

As mentioned, we will be using MySQL/MariaDB as our database.:

```
dnf install -y mariadb mariadb-server
```

Make sure the database service is enabled to start at boot time.:

```
systemctl enable mariadb.service
systemctl start mariadb.service
```

Improve MariaDB security.:

```
mysql_secure_installation
```

After you have done this, make sure you create a database with a username and password so that Nextcloud will have access to it. For further details on database setup and configuration, see the Database configuration documentation.

Redis

```
dnf install -y redis
systemctl enable redis.service
systemctl start redis.service
```

Installing Nextcloud

Nearly there, so keep at it, you are doing great!

Now download the archive of the latest Nextcloud version:

- Go to the Nextcloud Download Page.
- Go to **Download Nextcloud Server > Download > Archive file for server owners** and download either the tar.bz2 or .zip archive.
- This downloads a file named nextcloud-x.y.z.tar.bz2 or nextcloud-x.y.z.zip (where x.y.z is the version number).
- Download its corresponding checksum file, e.g. nextcloud-x.y.z.tar.bz2.md5, or nextcloud-x.y.z.tar.bz2.sha256.
- Verify the MD5 or SHA256 sum:

```
md5sum -c nextcloud-x.y.z.tar.bz2.md5 < nextcloud-x.y.z.tar.bz2
sha256sum -c nextcloud-x.y.z.tar.bz2.sha256 < nextcloud-x.y.z.tar.bz2
md5sum  -c nextcloud-x.y.z.zip.md5 < nextcloud-x.y.z.zip
sha256sum  -c nextcloud-x.y.z.zip.sha256 < nextcloud-x.y.z.zip
```

- You may also verify the PGP signature:

```
wget https://download.nextcloud.com/server/releases/nextcloud-x.y.z.tar.bz2.asc
wget https://nextcloud.com/nextcloud.asc
gpg --import nextcloud.asc
gpg --verify nextcloud-x.y.z.tar.bz2.asc nextcloud-x.y.z.tar.bz2
```

For the sake of the walk-through, we grabbed the latest version of Nextcloud in the form a zip file, confirmed the download with the above-mentioned command, and now we will extract it:

```
unzip nextcloud-*.zip
```

Copy the content over to the root directory of your webserver. In our case, we are using apache so it will be `/var/www/html/`:

```
cp -R nextcloud/ /var/www/html/
```

During the install process, no data folder is created, so we will create one manually to help with the installation wizard:

```
mkdir /var/www/html/nextcloud/data
```

Make sure that apache has read and write access to the whole nextcloud folder:

```
chown -R apache:apache /var/www/html/nextcloud
```

Restart apache:

```
systemctl restart httpd.service
```

Create a firewall rule for access to apache:

```
firewall-cmd --zone=public --add-service=http --permanent
firewall-cmd --reload
```

SELinux

Again, there is an extensive write-up done on SELinux which can be found at SELinux configuration, so if you are using SELinux in Enforcing mode, please run the commands suggested on that page. The following commands only refers to this tutorial:

```
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/html/nextcloud/data(/.*)?'
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/html/nextcloud/config(/.*)?'
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/html/nextcloud/apps(/.*)?'
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/html/nextcloud/.htaccess'
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/html/nextcloud/.user.ini'
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/html/nextcloud/3rdparty/aws/aws-sdk

restorecon -R '/var/www/html/nextcloud/'

setsebool -P httpd_can_network_connect on
```

If you need more SELinux configs, refer to the above-mentioned URL, return to this tutorial.

Once done with with SELinux, please head over to `http://your.server.com/nextcloud` and follow the steps as found Installation wizard, where it will explain to you exactly how to proceed with the final part of the install, which is done as admin user through your web browser.

Note

If you use this tutorial, and you see warnings in the web browser after installation about `OPcache` not being enabled or configured correctly, you need to make the suggested changes in `/etc/opt/rh/rh-php74/php.d/10-opcache.ini` for the errors to disappear. These warnings will be on the Admin page, under Basic settings.

Because we used `Redis` as a memcache, you will need a config similar to the following example in `/var/www/html/nextcloud/config/config.php` which is auto-generated when you run the online installation wizard mentioned earlier.

Example config:

```
'memcache.distributed' => '\OC\Memcache\Redis',
'memcache.locking' => '\OC\Memcache\Redis',
'memcache.local' => '\OC\Memcache\APCu',
'redis' => array(
  'host' => 'localhost',
  'port' => 6379,
),
```

Remember, this tutorial is only for a basic setup of Nextcloud on CentOS 8, with PHP 7.4. If you are going to use more features like LDAP or Single Sign On, you will need additional PHP modules as well as extra configurations. So please visit the rest of the Admin manual, Introduction, for detailed descriptions on how to get this done.

Example installation on OpenBSD

Warning

Nextcloud does not have official OpenBSD or other BSDs support

In this install tutorial we will be deploying Nextcloud on a minimal OpenBSD with our own httpd(8), PHP, PostgreSQL and redis (for -stable or -current are the same steps).

From a base installed OpenBSD system you can just do:

```
# pkg_add nextcloud
```

The extra packages:

```
# pkg_add postgresql-server redis pecl74-redis php-pdo_pgsql
```

This will take care of your dependencies and give you the options to choose which PHP version do you want.

HTTPD(8)

Create a virtualhost in `/etc/httpd.conf` and add the following content to it:

```
server "domain.tld" {
    listen on egress tls port 443
    hsts max-age 15768000

      tls {
               certificate "/etc/ssl/domain.tld_fullchain.pem"
               key "/etc/ssl/private/domain.tld_private.pem"
      }

      # Set max upload size to 513M (in bytes)
      connection max request body 537919488
      connection max requests 1000
      connection request timeout 3600
      connection timeout 3600

      block drop

      # Ensure that no '*.php*' files can be fetched from these directories
      location "/nextcloud/config/*" {
              block drop
      }

      location "/nextcloud/data/*" {
              block drop
      }

      # Note that this matches "*.php*" anywhere in the request path.
      location "/nextcloud/*.php*" {
              root "/nextcloud"
              request strip 1
              fastcgi socket "/run/php-fpm.sock"
              pass
      }

      location "/nextcloud/apps/*" {
              root "/nextcloud"
              request strip 1
```

```
                pass
        }

        location "/nextcloud/core/*" {
                root "/nextcloud"
                request strip 1
                pass
        }

        location "/nextcloud" {
                block return 301 "$DOCUMENT_URI/index.php"
        }

        location "/nextcloud/" {
                block return 301 "$DOCUMENT_URI/index.php"
        }

        location "/.well-known/carddav" {
                block return 301 "https://$SERVER_NAME/nextcloud/remote.php/dav"
        }

        location "/.well-known/caldav" {
                block return 301 "https://$SERVER_NAME/nextcloud/remote.php/dav"
        }

        location "/.well-known/webfinger" {
                block return 301 "https://$SERVER_NAME/nextcloud/public.php?service=webfinge
        }

        location match "/nextcloud/oc[ms]%-provider/*" {
                directory index index.php
                pass
        }
}
```

Make sure that httpd(8) is enabled and started:

```
# rcctl enable httpd
# rcctl start httpd
```

PHP

Assuming that you are on OpenBSD -current (or >= 6.8-stable) you could use PHP 7.4 so I will keep this version, but the concept is the same for other version.

The PHP packages will be available since you installed Nextcloud with pkg_add, so you just need to adjust a bit your php.ini.

It is recommended to add opcache to it:

```
[opcache]
opcache.enable=1
opcache.enable_cli=1
opcache.memory_consumption=512
opcache.interned_strings_buffer=8
opcache.max_accelerated_files=10000
opcache.revalidate_freq=1
opcache.save_comments=1
```

And increase some limits:

```
post_max_size = 513M
upload_max_filesize = 513M
```

We can enable the PHP modules with:

```
# cd /etc/php-7.4.sample
# for i in *; do ln -sf ../php-7.4.sample/$i ../php-7.4/; done
```

And then we just enable and start PHP:

```
# rcctl enable php74_fpm
# rcctl start php74_fpm
```

Database

As mentioned, we will be using PostgreSQL as our database, and we already installed it, now we need to initialised:

```
$ su - _postgresql
$ mkdir /var/postgresql/data
$ initdb -D /var/postgresql/data -U postgres -A md5 -E UTF8 -W
...
Enter new superuser password: PASSWORD
Enter it again: PASSWORD
...
Success. You can now start the database server using:

pg_ctl -D /var/postgresql/data -l logfile start

$ pg_ctl -D /var/postgresql/data -l logfile start
server starting
$ exit
```

We need to check, enable and start postgres:

```
# rcctl check postgresql
# rcctl enable postgresql
# rcctl start postgresql
```

You can follow the README on `/usr/local/share/doc/pkg-readmes/postgresql-server` to create users and permission.

Redis

We installed redis before, we need to enable it and start it and also add it to the Nextcloud conf:

```
# rcctl enable redis
# rcctl start redis
# mg /var/www/nextcloud/config/config.php
...
  'memcache.local' => '\OC\Memcache\Redis',
  'redis' => array(
  'host' => 'localhost',
  'port' => 6379,
  'timeout' => 0.0,
),
...
```

Cron job

We need to add the Nextcloud cron job to get some tasks done by adding this entry on your cronjob:

```
*/5 * * * * /usr/bin/ftp -Vo - https://domain.tld/cron.php >/dev/null
```

Chroot

Since in OpenBSD httpd(8) works with a chroot(8) by default, we need to be sure that we have the relevant files into the /var/www jail:

```
# mkdir -p /var/www/etc/ssl
# install -m 444 -o root -g bin /etc/ssl/cert.pem /etc/ssl/openssl.cnf \
        /var/www/etc/ssl/
# cp /etc/resolv.conf /var/www/etc
```

Nextcloud final steps

Now that we have all in place, you should go to your browser with your URL (I am asuming you have an SSL already installed):

```
https://domain.tld
```

Now you just need to follow the steps and put in place your DB name, usr and passwords.

Keep in mind that the upgrades for Nextcloud you can do it by running on -current:

```
# pkg_add -u -Dsnap
```

And on -stable:

```
# pkg_add -u
```

Then you just follow the steps from your browser.

NOTE

Remember always to read all the README**S** from the OpenBSD packages on:

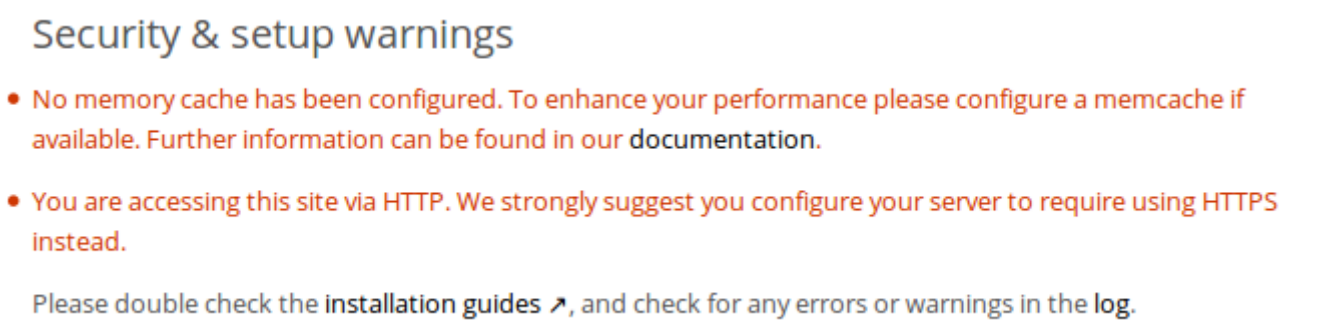
```
/usr/local/share/doc/pkg-readmes/
```

All this information and more is available for you there.

Nextcloud configuration

Warnings on admin page

Your Nextcloud server has a built-in configuration checker, and it reports its findings at the top of your Admin page. These are some of the warnings you might see, and what to do about them.



You can use the Nextcloud Security Scan to see if your system is up to date and well secured. We have ran this scan over public IP addresses in the past to try and reach out to extremely outdated systems and might again in the future. Please, protect your privacy and keep your server up to date! Privacy means little without security.

Cache warnings

“No memory cache has been configured. To enhance your performance please configure a memcache if available.” Nextcloud supports multiple php caching extensions:

- APCu (minimum required PHP extension version 4.0.6)
- Memcached

- Redis (minimum required PHP extension version: 2.2.6)

You will see this warning if you have no caches installed and enabled, or if your cache does not have the required minimum version installed; older versions are disabled because of performance problems.

If you see "*{Cache}* below version *{Version}* is installed. for stability and performance reasons we recommend to update to a newer *{Cache}* version" then you need to upgrade, or, if you're not using it, remove it.

You are not required to use any caches, but caches improve server performance. See Memory caching.

Transactional file locking is disabled

"Transactional file locking is disabled, this might lead to issues with race conditions."

Please see Transactional file locking on how to correctly configure your environment for transactional file locking.

You are accessing this site via HTTP

"You are accessing this site via HTTP. We strongly suggest you configure your server to require using HTTPS instead." Please take this warning seriously; using HTTPS is a fundamental security measure. You must configure your Web server to support it, and then there are some settings in the **Security** section of your Nextcloud Admin page to enable. The following pages describe how to enable HTTPS on the Apache and Nginx Web servers.

Enabling SSL (on Apache)

Use HTTPS

NGINX configuration

The test with getenv("PATH") only returns an empty response

Some environments are not passing a valid PATH variable to Nextcloud. The php-fpm configuration notes provides the information about how to configure your environment.

The "Strict-Transport-Security" HTTP header is not configured

"The "Strict-Transport-Security" HTTP header is not configured to least "15552000" seconds. For enhanced security we recommend enabling HSTS as described in our security tips."

The HSTS header needs to be configured within your Web server by following the Enable HTTP Strict Transport Security documentation

/dev/urandom is not readable by PHP

"/dev/urandom is not readable by PHP which is highly discouraged for security reasons. Further information can be found in our documentation."

This message is another one which needs to be taken seriously. Please have a look at the Give PHP read access to /dev/urandom documentation.

Your Web server is not yet set up properly to allow file synchronization

"Your web server is not yet set up properly to allow file synchronization because the WebDAV interface seems to be broken."

At the ownCloud community forums a larger FAQ is maintained containing various information and debugging hints.

Outdated NSS / OpenSSL version

"cURL is using an outdated OpenSSL version (OpenSSL/$version). Please update your operating system or features such as installing and updating apps via the app store or Federated Cloud Sharing will not work reliably."

"cURL is using an outdated NSS version (NSS/$version). Please update your operating system or features such as installing and updating apps via the app store or Federated Cloud Sharing will not work reliably."

There are known bugs in older OpenSSL and NSS versions leading to misbehavior in combination with remote hosts using SNI. A technology used by most of the HTTPS websites. To ensure that Nextcloud will work properly you need to update OpenSSL to at least 1.0.2b or 1.0.1d. For NSS the patch version depends on your distribution and an heuristic is running the test which actually reproduces the bug. There are distributions such as RHEL/CentOS which have this backport still pending.

Your Web server is not set up properly to resolve /.well-known/caldav/ or /.well-known/carddav/

Both URLs need to be correctly redirected to the DAV endpoint of Nextcloud. Please refer to Service discovery for more info.

Some files have not passed the integrity check

Please refer to the Fixing invalid code integrity messages documentation how to debug this issue.

Your database does not run with “READ COMMITED” transaction isolation level

“Your database does not run with “READ COMMITED” transaction isolation level. This can cause problems when multiple actions are executed in parallel.”

Please refer to Database “READ COMMITTED” transaction isolation level how to configure your database for this requirement.

Using the occ command

Nextcloud’s `occ` command (origins from “ownCloud Console”) is Nextcloud’s command-line interface. You can perform many common server operations with `occ`, such as installing and upgrading Nextcloud, manage users, encryption, passwords, LDAP setting, and more.

`occ` is in the nextcloud/ directory; for example /var/www/nextcloud on Ubuntu Linux. `occ` is a PHP script. **You must run it as your HTTP user** to ensure that the correct permissions are maintained on your Nextcloud files and directories.

occ command Directory

- Run occ as your HTTP user
- Apps commands
- Background jobs selector
- Config commands
- Dav commands
- Database conversion
- Add missing indices
- Encryption
- Federation sync
- File operations
- Files external
- Integrity check
- l10n, create JavaScript translation files for apps
- LDAP commands
- Logging commands
- Maintenance commands

- Security
- Trashbin
- User commands
- Group commands
- Versions
- Command line installation
- Command line upgrade
- Two-factor authentication
- Disable users
- System Tags
- Debugging

Run occ as your HTTP user

The HTTP user is different on the various Linux distributions:

- The HTTP user and group in Debian/Ubuntu is www-data.
- The HTTP user and group in Fedora/CentOS is apache.
- The HTTP user and group in Arch Linux is http.
- The HTTP user in openSUSE is wwwrun, and the HTTP group is www.

If your HTTP server is configured to use a different PHP version than the default (/usr/bin/php), `occ` should be run with the same version. For example, in CentOS 6.5 with SCL-PHP70 installed, the command looks like this:

```
sudo -u apache /opt/rh/php70/root/usr/bin/php /var/www/html/nextcloud/occ
```

Note

Although the following examples make use of the `sudo -u ... /path/to/php /path/to/occ` method, your environment may require use of a different wrapper utility than `sudo` to execute the command as the appropriate user. Other common wrappers:

- `su --command '/path/to/php ...' username` – Note here that the target user specification comes at the end, and the command to execute is specified first.
- `runuser --user username -- /path/to/php ...` – This wrapper might be used in container contexts (ex: Docker / `arm32v7/nextcloud`) where both `sudo` and `su` wrapper utilities cannot be used.

Running `occ` with no options lists all commands and options, like this example on Ubuntu:

```
sudo -u www-data php occ
Nextcloud version 19.0.0

Usage:
 command [options] [arguments]

Options:
 -h, --help            Display this help message
 -q, --quiet           Do not output any message
 -V, --version         Display this application version
     --ansi            Force ANSI output
     --no-ansi         Disable ANSI output
 -n, --no-interaction  Do not ask any interactive question
     --no-warnings     Skip global warnings, show command output only
```

```
  -v|vv|vvv, --verbose  Increase the verbosity of messages: 1 for normal output,
                        2 for more verbose output and 3 for debug

 Available commands:
  check                 check dependencies of the server
                        environment
  help                  Displays help for a command
  list                  Lists commands
  status                show some status information
  upgrade               run upgrade routines after installation of
                        a new release. The release has to be
                        installed before.
```

This is the same as `sudo -u www-data php occ list`.

Run it with the `-h` option for syntax help:

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

Display your Nextcloud version:

```
sudo -u www-data php occ -V
  Nextcloud version 19.0.0
```

Query your Nextcloud server status:

```
sudo -u www-data php occ status
  - installed: true
  - version: 19.0.0.12
  - versionstring: 19.0.0
  - edition:
```

`occ` has options, commands, and arguments. Options and arguments are optional, while commands are required. The syntax is:

```
occ [options] command [arguments]
```

Get detailed information on individual commands with the `help` command, like this example for the `maintenance:mode` command:

```
sudo -u www-data php occ help maintenance:mode
Usage:
 maintenance:mode [options]

Options:
     --on              enable maintenance mode
     --off             disable maintenance mode
 -h, --help            Display this help message
 -q, --quiet           Do not output any message
 -V, --version         Display this application version
     --ansi            Force ANSI output
     --no-ansi         Disable ANSI output
 -n, --no-interaction  Do not ask any interactive question
     --no-warnings     Skip global warnings, show command output only
 -v|vv|vvv, --verbose  Increase the verbosity of messages: 1 for normal output,
                       2 for more verbose output and 3 for debug
```

The `status` command from above has an option to define the output format. The default is plain text, but it can also be `json`:

```
sudo -u www-data php occ status --output=json
{"installed":true,"version":"19.0.0.9","versionstring":"19.0.0","edition":""}
```

or `json_pretty`:

```
sudo -u www-data php occ status --output=json_pretty
{
   "installed": true,
   "version": "19.0.0.12",
   "versionstring": "19.0.0",
   "edition": ""
}
```

This output option is available on all list and list-like commands: `status`, `check`, `app:list`, `config:list`, `encryption:status` and `encryption:list-modules`

Enabling autocompletion

Note

Command autocompletion currently only works if the user you use to execute the occ commands has a profile. `www-data` in most cases is `nologon` and therefor **cannot** use this feature.

Autocompletion is available for bash (and bash based consoles). To enable it, you have to run **one** of the following commands:

```
# BASH ~4.x, ZSH
source <(/var/www/html/nextcloud/occ _completion --generate-hook)

# BASH ~3.x, ZSH
/var/www/html/nextcloud/occ _completion --generate-hook | source /dev/stdin

# BASH (any version)
eval $(/var/www/html/nextcloud/occ _completion --generate-hook)
```

This will allow you to use autocompletion with the full path `/var/www/html/nextcloud/occ <tab>`.

If you also want to use autocompletion on occ from within the directory without using the full path, you need to specify `--program occ` after the `--generate-hook`.

If you want the completion to apply automatically for all new shell sessions, add the command to your shell's profile (eg. `~/.bash_profile` or `~/.zshrc`).

Apps commands

The app commands list, enable, and disable apps:

```
app
  app:install      install selected app
  app:disable      disable an app
  app:enable       enable an app
  app:getpath      get an absolute path to the app directory
  app:list         list all available apps
  app:update       update an app or all apps
  app:remove       disable and remove an app
```

Download and install an app:

```
sudo -u www-data php occ app:install twofactor_totp
```

Install but don't enable:

```
sudo -u www-data php occ app:install --keep-disabled twofactor_totp
```

List all of your installed apps, and show whether they are enabled or disabled:

```
sudo -u www-data php occ app:list
```

Enable an app, for example the External Storage Support app:

```
sudo -u www-data php occ app:enable files_external
files_external enabled
```

Disable an app:

```
sudo -u www-data php occ app:disable files_external
files_external disabled
```

You can get the full filepath to an app:

```
sudo -u www-data php occ app:getpath notifications
/var/www/nextcloud/apps/notifications
```

To update an app, for instance Contacts:

```
sudo -u www-data php occ app:update contacts
```

To update all apps:

```
sudo -u www-data php occ app:update --all
```

Background jobs selector

Use the `background` command to select which scheduler you want to use for controlling background jobs, Ajax, Webcron, or Cron. This is the same as using the **Cron** section on your Nextcloud Admin page:

```
background
 background:ajax       Use ajax to run background jobs
 background:cron       Use cron to run background jobs
 background:webcron    Use webcron to run background jobs
```

This example selects Ajax:

```
sudo -u www-data php occ background:ajax
  Set mode for background jobs to 'ajax'
```

The other two commands are:

- `background:cron`
- `background:webcron`

See Background jobs to learn more.

Config commands

The `config` commands are used to configure the Nextcloud server:

```
config
 config:app:delete      Delete an app config value
 config:app:get         Get an app config value
 config:app:set         Set an app config value
 config:import          Import a list of configs
 config:list            List all configs
 config:system:delete   Delete a system config value
 config:system:get      Get a system config value
 config:system:set      Set a system config value
```

You can list all configuration values with one command:

```
sudo -u www-data php occ config:list
```

By default, passwords and other sensitive data are omitted from the report, so the output can be posted publicly (e.g. as part of a bug report). In order to generate a full backport of all configuration values the `--private` flag needs to be set:

```
sudo -u www-data php occ config:list --private
```

The exported content can also be imported again to allow the fast setup of similar instances. The import command will only add or update values. Values that exist in the current configuration, but not in the one that is being imported are left untouched:

```
sudo -u www-data php occ config:import filename.json
```

It is also possible to import remote files, by piping the input:

```
sudo -u www-data php occ config:import < local-backup.json
```

Note

While it is possible to update/set/delete the versions and installation statuses of apps and Nextcloud itself, it is **not** recommended to do this directly. Use the `occ app:enable`, `occ app:disable` and `occ app:update` commands instead.

Getting a single configuration value

These commands get the value of a single app or system configuration:

```
sudo -u www-data php occ config:system:get version
19.0.0.12

sudo -u www-data php occ config:app:get activity installed_version
2.2.1
```

Setting a single configuration value

These commands set the value of a single app or system configuration:

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

sudo -u www-data php occ config:app:set files_sharing
incoming_server2server_share_enabled --value="yes" --type=boolean
Config value incoming_server2server_share_enabled for app files_sharing set to yes
```

The `config:system:set` command creates the value, if it does not already exist. To update an existing value, set `--update-only`:

```
sudo -u www-data php occ config:system:set doesnotexist --value="true"
--type=boolean --update-only
Value not updated, as it has not been set before.
```

Note that in order to write a Boolean, float, or integer value to the configuration file, you need to specify the type on your command. This applies only to the `config:system:set` command. The following values are known:

- `boolean`
- `integer`
- `float`
- `string` (default)

When you want to e.g. disable the maintenance mode run the following command:

```
sudo -u www-data php occ config:system:set maintenance --value=false
--type=boolean
Nextcloud is in maintenance mode - no app have been loaded
System config value maintenance set to boolean false
```

Setting an array configuration value

Some configurations (e.g. the trusted domain setting) are an array of data. In this case, `config:system:get` for this key will return multiple values:

```
sudo -u www-data php occ config:system:get trusted_domains
localhost
nextcloud.local
sample.tld
```

To set one of multiple values, you need to specify the array index as the second `name` in the `config:system:set` command, separated by a space. For example, to replace `sample.tld` with `example.com`, `trusted_domains => 2` needs to be set:

```
sudo -u www-data php occ config:system:set trusted_domains 2
--value=example.com
System config value trusted_domains => 2 set to string example.com

sudo -u www-data php occ config:system:get trusted_domains
localhost
nextcloud.local
example.com
```

Setting a hierarchical configuration value

Some configurations use hierarchical data. For example, the settings for the Redis cache would look like this in the `config.php` file:

```
'redis' => array(
  'host' => '/var/run/redis/redis.sock',
  'port' => 0,
  'dbindex' => 0,
  'password' => 'secret',
  'timeout' => 1.5,
)
```

Setting such hierarchical values works similarly to setting an array value above. For this Redis example, use the following commands:

```
sudo -u www-data php occ config:system:set redis host \
--value=/var/run/redis/redis.sock
sudo -u www-data php occ config:system:set redis port --value=0
sudo -u www-data php occ config:system:set redis dbindex --value=0
sudo -u www-data php occ config:system:set redis password --value=secret
sudo -u www-data php occ config:system:set redis timeout --value=1.5
```

Deleting a single configuration value

These commands delete the configuration of an app or system configuration:

```
sudo -u www-data php occ config:system:delete maintenance:mode
System config value maintenance:mode deleted

sudo -u www-data php occ config:app:delete appname provisioning_api
Config value provisioning_api of app appname deleted
```

The delete command will by default not complain if the configuration was not set before. If you want to be notified in that case, set the `--error-if-not-exists` flag:

```
sudo -u www-data php occ config:system:delete doesnotexist
--error-if-not-exists
Config provisioning_api of app appname could not be deleted because it did not
exist
```

Dav commands

A set of commands to create and manage addressbooks and calendars:

```
dav
 dav:create-addressbook        Create a dav addressbook
 dav:create-calendar           Create a dav calendar
 dav:delete-calendar           Delete a dav calendar
 dav:list-calendars            List all calendars of a user
 dav:move-calendar             Move a calendar from a user to another
 dav:remove-invalid-shares     Remove invalid dav shares
 dav:send-event-reminders      Sends event reminders
 dav:sync-birthday-calendar    Synchronizes the birthday calendar
 dav:sync-system-addressbook   Synchronizes users to the system
                               addressbook
```

The syntax for `dav:create-addressbook` and `dav:create-calendar` is `dav:create-addressbook [user] [name]`. This example creates the addressbook `mollybook` for the user molly:

```
sudo -u www-data php occ dav:create-addressbook molly mollybook
```

This example creates a new calendar for molly:

```
sudo -u www-data php occ dav:create-calendar molly mollycal
```

Molly will immediately see these in the Calendar and Contacts apps.

`dav:delete-calendar [--birthday] [-f|--force] <uid> [<name>]` deletes the calendar named `name` (or the birthday calendar if `--birthday` is specified) of the user `uid`. You can use the force option `-f` or `--force` to delete the calendar instead of moving it to the trashbin.

This example will delete the calendar mollycal of user molly:

```
sudo -u www-data php occ dav:delete-calendar molly mollycal
```

This example will delete the birthday calendar of user molly:

```
sudo -u www-data php occ dav:delete-calendar --birthday molly
```

`dav:lists-calendars [user]` will display a table listing the calendars for a given user. This example will list all calendars for user annie:

```
sudo -u www-data php occ dav:list-calendars annie
```

`dav::move-calendar [name] [sourceuid] [destinationuid]` allows the admin to move a calendar named `name` from a user `sourceuid` to the user `destinationuid`. You can use the force option *-f* to enforce the move if there are conflicts with existing shares. The system will also generate a new unique calendar name in case there is a conflict over the destination user.

This example will move calendar named personal from user dennis to user sabine:

```
sudo -u www-data php occ dav:move-calendar personal dennis sabine
```

`dav:remove-invalid-shares` will remove invalid shares created by a bug into the calendar app

`dav:send-event-reminders` is a command that should be called regularly through a dedicated cron job to send event reminder notifications.

See Calendar / CalDAV for more information on how to use this command.

`dav:sync-birthday-calendar` adds all birthdays to your calendar from addressbooks shared with you. This example syncs to your calendar from user bernie:

```
sudo -u www-data php occ dav:sync-birthday-calendar bernie
```

`dav:sync-system-addressbook` synchronizes all users to the system addressbook:

```
sudo -u www-data php occ dav:sync-system-addressbook
```

Database conversion

The SQLite database is good for testing, and for Nextcloud servers with small single-user workloads that do not use sync clients, but production servers with multiple users should use MariaDB, MySQL, or PostgreSQL. You can use `occ` to convert from SQLite to one of these other databases.

```
db
  db:convert-type           Convert the Nextcloud database to the newly
                            configured one
  db:generate-change-script generates the change script from the current
                            connected db to db_structure.xml
```

You need:

- Your desired database and its PHP connector installed.
- The login and password of a database admin user.
- The database port number, if it is a non-standard port.

This is example converts SQLite to MySQL/MariaDB:

```
sudo -u www-data php occ db:convert-type mysql oc_dbuser 127.0.0.1
oc_database
```

For a more detailed explanation see Converting database type

Add missing indices

It might happen that we add from time to time new indices to already existing database tables, for example to improve performance. In order to check your database for missing indices run following command:

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

Encryption

`occ` includes a complete set of commands for managing encryption:

```
encryption
  encryption:change-key-storage-root   Change key storage root
  encryption:decrypt-all               Disable server-side encryption and
                                       decrypt all files
  encryption:disable                   Disable encryption
  encryption:enable                    Enable encryption
  encryption:enable-master-key         Enable the master key. Only available
                                       for fresh installations with no existing
                                       encrypted data! There is also no way to
                                       disable it again.
  encryption:encrypt-all               Encrypt all files for all users
  encryption:list-modules              List all available encryption modules
  encryption:set-default-module        Set the encryption default module
  encryption:show-key-storage-root     Show current key storage root
  encryption:status                    Lists the current status of encryption
```

`encryption:status` shows whether you have active encryption, and your default encryption module. To enable encryption you must first enable the Encryption app, and then run `encryption:enable`:

```
sudo -u www-data php occ app:enable encryption
sudo -u www-data php occ encryption:enable
sudo -u www-data php occ encryption:status
  - enabled: true
  - defaultModule: OC_DEFAULT_MODULE
```

`encryption:change-key-storage-root` is for moving your encryption keys to a different folder. It takes one argument, `newRoot`, which defines your new root folder:

```
sudo -u www-data php occ encryption:change-key-storage-root /etc/oc-keys
```

You can see the current location of your keys folder:

```
sudo -u www-data php occ encryption:show-key-storage-root
Current key storage root:  default storage location (data/)
```

`encryption:list-modules` displays your available encryption modules. You will see a list of modules only if you have enabled the Encryption app. Use `encryption:set-default-module [module name]` to set your desired module.

`encryption:encrypt-all` encrypts all data files for all users. You must first put your Nextcloud server into maintenance mode to prevent any user activity until encryption is completed.

`encryption:decrypt-all` decrypts all user data files, or optionally a single user:

```
sudo -u www-data php occ encryption:decrypt freda
```

Users must have enabled recovery keys on their Personal pages. You must first put your Nextcloud server into maintenance mode to prevent any user activity until decryption is completed.

Note that if you do not have master key/recovery key enabled, you can ONLY decrypt files per user, one user at a time and NOT when in maintenance mode. You will need the users' password to decrypt the files.

Use `encryption:disable` to disable your encryption module. You must first put your Nextcloud server into maintenance mode to prevent any user activity.

`encryption:enable-master-key` creates a new master key, which is used for all user data instead of individual user keys. This is especially useful to enable single-sign on. Use this only on fresh installations with no existing data, or on systems where encryption has not already been enabled. It is not possible to disable it.

See Encryption configuration to learn more.

Federation sync

Note

This command is only available when the "Federation" app (`federation`) is enabled.

Synchronize the addressbooks of all federated Nextcloud servers:

```
federation:sync-addressbooks  Synchronizes addressbooks of all
                              federated clouds
```

In Nextcloud, servers connected with federation shares can share user address books, and auto-complete usernames in share dialogs. Use this command to synchronize federated servers:

```
sudo -u www-data php occ federation:sync-addressbooks
```

File operations

occ has three commands for managing files in Nextcloud:

```
files
  files:cleanup              Cleanup filecache
  files:repair-tree          Try and repair malformed filesystem tree structures
  files:scan                 Rescan filesystem
  files:scan-app-data        Rescan the AppData folder
  files:transfer-ownership   All files' and folders' ownerships are moved to another
                             user. Outgoing shares are moved as well.
                             Incoming shares are not moved by default because the
                             sharing user holds the ownership of the respective files.
                             There is however an option to enable moving incoming shares.
```

Scan

The `files:scan` command scans for new files and updates the file cache. You may rescan all files, per-user, a space-delimited list of users, and limit the search path. If not using `--quiet`, statistics will be shown at the end of the scan:

```
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 files
```

Verbosity levels of `-vv` or `-vvv` are automatically reset to `-v`

Note for option `--unscanned`: In general there is a background job (through cron) that will do that scan periodically. The `--unscanned` option makes it possible to trigger this from the CLI.

When using the `--path` option, the path must consist of following components:

```
"user_id/files/path"
  or
"user_id/files/mount_name"
  or
"user_id/files/mount_name/path"
```

where the term `files` is mandatory.

Example:

```
--path="/alice/files/Music"
```

In the example above, the user_id `alice` is determined implicitly from the path component given.

The `--path`, `--all` and `[user_id]` parameters are exclusive - only one must be specified.

Scan appdata

Appdata is a folder inside of the data directory which contains files that are shared between users and can be put there by the server or apps like avatar images, file previews and cached CSS files for example.

Since the regular files scan only operates on user files the `occ files:scan-app-data` command will check the appdata directory and make sure that the filecache is consistent with the files on the actual storage.:

```
Usage:
  files:scan-app-data [options] [--] [<folder>]

Arguments:
  folder                The appdata subfolder to scan [default: ""]
```

Cleanup

`files:cleanup` tidies up the server's file cache by deleting all file entries that have no matching entries in the storage table.

Repair-Tree

`files:repair-tree` try and repair malformed filesystem tree structures. If for any reason the path of an entry in the filecache doesn't match with it's expected path, based on the path of it's parent node, you end up with an entry in the filecache that exists in different places based on how the entry is generated. For example, if while listing folder `/foo` it contains a file `bar.txt`, but when trying to do anything with `/foo/bar.txt` the file doesn't exists.

This command attempts to repair such entries by querying for entries where the path doesn't match the expected path based on it's parent path and filename and resets it's path to the expected one.

Transfer

You may transfer all files and shares from one user to another. This is useful before removing a user:

```
sudo -u www-data php occ files:transfer-ownership <source-user> <destination-user>
```

It is also possible to transfer only one directory along with it's contents. This can be useful to restructure your organization or quotas. The `--path` argument is given as the path to the directory as seen from the source user:

```
sudo -u www-data php occ files:transfer-ownership --path="path_to_dir" <source-user> <destin
```

In case the incoming shares must be transferred as well, use the argument `--transfer-incoming-shares` with `0` or `1` as parameters

```
sudo -u www-data php occ files:transfer-ownership --transfer-incoming-shares=1 --path="path_
```

As an alternative, the system configuration option `transferIncomingShares` in config.php can be set to `true` to always transfer incoming shares.

The command line option `--transfer-incoming-shares` overwrites the config.php option `transferIncomingShares`. For example, `'transferIncomingShares => true` can be overwritten by:

```
sudo -u www-data php occ files:transfer-ownership --transfer-incoming-shares=0 <source-user>
```

Users may also transfer files or folders selectively by themselves. See user documentation for details.

Files Sharing

Commands for handling shares:

```
sharing
  sharing:cleanup-remote-storages  Cleanup shared storage entries that have no matching entry
  sharing:expiration-notification  Notify share initiators when a share will expire the next
```

Files external

Note

These commands are only available when the "External storage support" app (`files_external`) is enabled.

Commands for managing external storage:

```
files_external
  files_external:applicable  Manage applicable users and groups for a mount
  files_external:backends    Show available authentication and storage backends
  files_external:config      Manage backend configuration for a mount
  files_external:create      Create a new mount configuration
  files_external:delete      Delete an external mount
  files_external:export      Export mount configurations
  files_external:import      Import mount configurations
  files_external:list        List configured mounts
  files_external:option      Manage mount options for a mount
```

```
  files_external:verify      Verify mount configuration
  files_external:notify      Listen for active update notifications for a configured external
```

These commands replicate the functionality in the Nextcloud Web GUI, plus two new features: `files_external:export` and `files_external:import`.

Use `files_external:export` to export all admin mounts to stdout, and `files_external:export [user_id]` to export the mounts of the specified Nextcloud user.

Use `files_external:import [filename]` to import legacy JSON configurations, and to copy external mount configurations to another Nextcloud server.

Integrity check

Apps which have a `Featured` tag MUST be code signed with Nextcloud. Unsigned featured apps won't be installable anymore. Code signing is optional for all third-party applications:

```
integrity
  integrity:check-app                Check app integrity using a signature.
  integrity:check-core               Check core integrity using a signature.
  integrity:sign-app                 Signs an app using a private key.
  integrity:sign-core                Sign core using a private key
```

After creating your signing key, sign your app like this example:

```
sudo -u www-data php occ integrity:sign-app --privateKey=/Users/lukasreschke/contacts.key --
```

Verify your app:

```
sudo -u www-data php occ integrity:check-app --path=/pathto/app appname
```

When it returns nothing, your app is signed correctly. When it returns a message then there is an error. See Code Signing in the Developer manual for more detailed information.

`integrity:sign-core` is for Nextcloud core developers only.

See Code signing to learn more.

l10n, create JavaScript translation files for apps

This command is for app developers to update their translation mechanism from ownCloud 7 to Nextcloud.

LDAP commands

Note

These commands are only available when the "LDAP user and group backend" app (`user_ldap`) is enabled.

These LDAP commands appear only when you have enabled the LDAP app. Then you can run the following LDAP commands with `occ`:

```
ldap
  ldap:check-user              checks whether a user exists on LDAP.
  ldap:create-empty-config     creates an empty LDAP configuration
  ldap:delete-config           deletes an existing LDAP configuration
  ldap:search                  executes a user or group search
  ldap:set-config              modifies an LDAP configuration
  ldap:show-config             shows the LDAP configuration
  ldap:show-remnants           shows which users are not available on
                               LDAP anymore, but have remnants in
                               Nextcloud.
  ldap:test-config             tests an LDAP configuration
```

Search for an LDAP user, using this syntax:

```
sudo -u www-data php occ ldap:search [--group] [--offset="..."]
[--limit="..."] search
```

Searches will match at the beginning of the attribute value only. This example searches for givenNames that start with “rob”:

```
sudo -u www-data php occ ldap:search "rob"
```

This will find robbie, roberta, and robin. Broaden the search to find, for example, `jeroboam` with the asterisk wildcard:

```
sudo -u www-data php occ ldap:search "*rob"
```

User search attributes are set with `ldap:set-config` (below). For example, if your search attributes are `givenName` and `sn` you can find users by first name + last name very quickly. For example, you’ll find Terri Hanson by searching for `te ha`. Trailing whitespaces are ignored.

Check if an LDAP user exists. This works only if the Nextcloud server is connected to an LDAP server:

```
sudo -u www-data php occ ldap:check-user robert
```

`ldap:check-user` will not run a check when it finds a disabled LDAP connection. This prevents users that exist on disabled LDAP connections from being marked as deleted. If you know for certain that the user you are searching for is not in one of the disabled connections, and exists on an active connection, use the `--force` option to force it to check all active LDAP connections:

```
sudo -u www-data php occ ldap:check-user --force robert
```

`ldap:create-empty-config` creates an empty LDAP configuration. The first one you create has `configID s01`, and all subsequent configurations that you create are automatically assigned IDs:

```
sudo -u www-data php occ ldap:create-empty-config
   Created new configuration with configID 's01'
```

Then you can list and view your configurations:

```
sudo -u www-data php occ ldap:show-config
```

And view the configuration for a single configID:

```
sudo -u www-data php occ ldap:show-config s01
```

`ldap:delete-config [configID]` deletes an existing LDAP configuration:

```
sudo -u www-data php occ ldap:delete  s01
Deleted configuration with configID 's01'
```

The `ldap:set-config` command is for manipulating configurations, like this example that sets search attributes:

```
sudo -u www-data php occ ldap:set-config s01 ldapAttributesForUserSearch
"cn;givenname;sn;displayname;mail"
```

`ldap:test-config` tests whether your configuration is correct and can bind to the server:

```
sudo -u www-data php occ ldap:test-config s01
The configuration is valid and the connection could be established!
```

`ldap:show-remnants` is for cleaning up the LDAP mappings table, and is documented in LDAP user cleanup.

Logging commands

These commands view and configure your Nextcloud logging preferences:

```
log
  log:file        manipulate Nextcloud logging backend
  log:manage      manage logging configuration
  log:tail        tail the nextcloud logfile [requires app "Log Reader" to be enabled]
  log:watch       watch the nextcloud logfile live [requires app "Log Reader" to be enabled]
```

Run `log:file [--] [--enable] [--file] [--rotate-size]` to see your current logging status:

```
sudo -u www-data php occ log:file
Log backend Nextcloud: enabled
Log file: /opt/nextcloud/data/nextcloud.log
Rotate at: disabled
```

- `--enable` turns on logging.
- `--file` sets a different log file path.
- `--rotate-size` sets your rotation by log file size in bytes with; 0 disables rotation.

`log:manage [--backend] [--level] [--timezone]` sets your logging backend, log level, and timezone. The defaults are `file`, `warning`, and `UTC`. Available options are:

- `--backend [file, syslog, errorlog, systemd]`
- `--level [debug|info|warning|error|fatal]`
- `--timezone` according to https://www.php.net/manual/en/timezones.php

Maintenance commands

Use these commands when you upgrade Nextcloud, manage encryption, perform backups and other tasks that require locking users out until you are finished:

```
maintenance
 maintenance:data-fingerprint        update the systems data-fingerprint after a backup is r
 maintenance:mimetype:update-db      Update database mimetypes and update filecache
 maintenance:mimetype:update-js      Update mimetypelist.js
 maintenance:mode                    set maintenance mode
 maintenance:repair                  repair this installation
 maintenance:theme:update            Apply custom theme changes
 maintenance:update:htaccess         Updates the .htaccess file
```

`maintenance:mode` locks the sessions of all logged-in users, including administrators, and displays a status screen warning that the server is in maintenance mode. Users who are not already logged in cannot log in until maintenance mode is turned off. When you take the server out of maintenance mode logged-in users must refresh their Web browsers to continue working:

```
sudo -u www-data php occ maintenance:mode --on
sudo -u www-data php occ maintenance:mode --off
```

After restoring a backup of your data directory or the database, you should always call `maintenance:data-fingerprint` once. This changes the ETag for all files in the communication with sync clients, allowing them to realize a file was modified.

The `maintenance:repair` command runs automatically during upgrades to clean up the database, so while you can run it manually there usually isn't a need to:

```
sudo -u www-data php occ maintenance:repair
```

`maintenance:mimetype:update-db` updates the Nextcloud database and file cache with changed mimetypes found in `config/mimetypemapping.json`. Run this command after modifying `config/mimetypemapping.json`. If you change a mimetype, run `maintenance:mimetype:update-db --repair-filecache` to apply the change to existing files.

Run the `maintenance:theme:update` command if the icons of your custom theme are not updated correctly. This updates the mimetypelist.js and cleares the image cache.

Security

Use these commands to manage server-wide SSL certificates. These are useful when you create federation shares with other Nextcloud servers that use self-signed certificates:

```
security
 security:certificates         list trusted certificates
 security:certificates:import  import trusted certificate
 security:certificates:remove  remove trusted certificate
```

This example lists your installed certificates:

```
sudo -u www-data php occ security:certificates
```

Import a new certificate:

```
sudo -u www-data php occ security:certificates:import /path/to/certificate
```

Remove a certificate:

```
sudo -u www-data php occ security:certificates:remove [certificate name]
```

Trashbin

Note

This command is only available when the “Deleted files” app (`files_trashbin`) is enabled.

The `trashbin:cleanup  [--all-users] [--] [<user_id>...]` command removes the deleted files of the specified users in a space-delimited list, or all users if –all-users is specified.

```
trashbin
 trashbin:cleanup  [--all-users] [--] [<user_id>...]  Remove deleted files
```

This example removes the deleted files of all users:

```
sudo -u www-data php occ trashbin:cleanup --all-users
Remove all deleted files for all users
Remove deleted files for users on backend Database
 freda
 molly
 stash
 rosa
 edward
```

This example removes the deleted files of users molly and freda:

```
sudo -u www-data php occ trashbin:cleanup molly freda
Remove deleted files of   molly
Remove deleted files of   freda
```

User commands

The `user` commands create and remove users, reset passwords, display a simple report showing how many users you have, and when a user was last logged in:

```
user
 user:add                        adds a user
 user:delete                     deletes the specified user
 user:disable                    disables the specified user
 user:enable                     enables the specified user
 user:lastseen                   shows when the user was logged in last time
 user:list                       shows list of all registered users
 user:report                     shows how many users have access
 user:resetpassword              Resets the password of the named user
 user:setting                    Read and modify user settings
```

You can create a new user with their display name, login name, and any group memberships with the `user:add` command. The syntax is:

```
user:add [--password-from-env] [--display-name["..."]] [-g|--group["..."]]
uid
```

The `display-name` corresponds to the **Full Name** on the Users page in your Nextcloud Web UI, and the `uid` is their **Username**, which is their login name. This example adds new user Layla Smith, and adds them to the **users** and **db-admins** groups. Any groups that do not exist are created:

```
sudo -u www-data php occ user:add --display-name="Layla Smith"
  --group="users" --group="db-admins" layla
  Enter password:
  Confirm password:
  The user "layla" was created successfully
  Display name set to "Layla Smith"
  User "layla" added to group "users"
  User "layla" added to group "db-admins"
```

Go to your Users page, and you will see your new user.

`password-from-env` allows you to set the user's password from an environment variable. This prevents the password from being exposed to all users via the process list, and will only be visible in the history of the user (root) running the command. This also permits creating scripts for adding multiple new users.

To use `password-from-env` you must run as "real" root, rather than `sudo`, because `sudo` strips environment variables. This example adds new user Fred Jones:

```
export OC_PASS=newpassword
su -s /bin/sh www-data -c 'php occ user:add --password-from-env
  --display-name="Fred Jones" --group="users" fred'
The user "fred" was created successfully
Display name set to "Fred Jones"
User "fred" added to group "users"
```

You can reset any user's password, including administrators (see Resetting a lost admin password):

```
sudo -u www-data php occ user:resetpassword layla
  Enter a new password:
  Confirm the new password:
  Successfully reset password for layla
```

You may also use `password-from-env` to reset passwords:

```
export OC_PASS=newpassword
su -s /bin/sh www-data -c 'php occ user:resetpassword --password-from-env
  layla'
  Successfully reset password for layla
```

You can delete users:

```
sudo -u www-data php occ user:delete fred
```

View a user's most recent login:

```
sudo -u www-data php occ user:lastseen layla
  layla's last login: 09.01.2020 18:46
```

Read user settings:

```
sudo -u www-data php occ user:setting layla
  - core:
    - lang: en
  - login:
    - lastLogin: 1465910968
  - settings:
    - email: layla@example.tld
```

Filter by app:

```
sudo -u www-data php occ user:setting layla core
  - core:
    - lang: en
```

Get a single setting:

```
sudo -u www-data php occ user:setting layla core lang
en
```

Set a setting:

```
sudo -u www-data php occ user:setting layla settings email "new-layla@example.tld"
```

Delete a setting:

```
sudo -u www-data php occ user:setting layla settings email --delete
```

Generate a simple report that counts all users, including users on external user authentication servers such as LDAP:

```
sudo -u www-data php occ user:report
+------------------+----+
| User Report      |    |
+------------------+----+
| Database         | 12 |
| LDAP             | 86 |
|                  |    |
| total users      | 98 |
|                  |    |
| user directories | 2  |
+------------------+----+
```

Group commands

The `group` commands create and remove groups, add and remove users in groups, display a list of all users in a group:

```
group
 group:add                          add a group
 group:delete                       remove a group
 group:adduser                      add a user to a group
 group:removeuser                   remove a user from a group
 group:list                         list configured groups
```

You can create a new group with the `group:add` command. The syntax is:

```
group:add [gid]
```

The `gid` corresponds to the group name you entering after clicking “Add group” on the Users page in your Nextcloud Web UI. This example adds new group “beer”:

```
sudo -u www-data php occ group:add beer
```

Add an existing user to the specified group with the `group:adduser` command. The syntax is:

```
group:adduser [gid] [uid]
```

This example adds the user “denis” to the existing group “beer”:

```
sudo -u www-data php occ group:adduser beer denis
```

You can remove user from the group with the `group:removeuser` command. This example removes the existing user “denis” from the existing group “beer”:

```
sudo -u www-data php occ group:removeuser beer denis
```

Remove a group with the `group:delete` command. Removing a group doesn't remove users in a group. You cannot remove the "admin" group. This example removes the existing group "beer":

```
sudo -u www-data php occ group:delete beer
```

List configured groups via the `group:list` command. The syntax is:

```
group:list [-l|--limit] [-o|--offset] [--output="..."]
```

`limit` allows you to specify the number of groups to retrieve.

`offset` is an offset for retrieving groups.

`output` specifies the output format (plain, json or json_pretty). Default is plain.

Versions

Note

This command is only available when the "Versions" app (`files_versions`) is enabled.

Use this command to delete file versions for specific users, or for all users when none are specified:

```
versions
 versions:cleanup   Delete versions
```

This example deletes all versions for all users:

```
sudo -u www-data php occ versions:cleanup
Delete all versions
Delete versions for users on backend Database
  freda
  molly
  stash
  rosa
  edward
```

You can delete versions for specific users in a space-delimited list:

```
sudo -u www-data php occ versions:cleanup freda molly
Delete versions of   freda
Delete versions of   molly
```

Command line installation

These commands are available only after you have downloaded and unpacked the Nextcloud archive, and taken no further installation steps.

You can install Nextcloud entirely from the command line. After downloading the tarball and copying Nextcloud into the appropriate directories you can use `occ` commands in place of running the graphical Installation Wizard.

Then choose your `occ` options. This lists your available options:

```
sudo -u www-data php /var/www/nextcloud/occ
Nextcloud is not installed - only a limited number of commands are available
Nextcloud version 19.0.0

Usage:
 [options] command [arguments]

Options:
 --help (-h)           Display this help message
 --quiet (-q)          Do not output any message
 --verbose (-v|vv|vvv) Increase the verbosity of messages: 1 for normal
```

```
  output,  2 for more verbose output and 3 for debug
  --version (-V)        Display this application version
  --ansi                Force ANSI output
  --no-ansi             Disable ANSI output
  --no-interaction (-n) Do not ask any interactive question

 Available commands:
  check                 check dependencies of the server environment
  help                  Displays help for a command
  list                  Lists commands
  status                show some status information
  app
  l10n
  l10n:createjs         Create javascript translation files for a given app
  maintenance
  maintenance:install   install Nextcloud
```

Display your `maintenance:install` options:

```
sudo -u www-data php occ help maintenance:install
Nextcloud is not installed - only a limited number of commands are available
Usage:
 maintenance:install [--database="..."] [--database-name="..."]
[--database-host="..."] [--database-user="..."] [--database-pass[="..."]]
[--database-table-prefix[="..."]] [--admin-user="..."] [--admin-pass="..."]
[--data-dir="..."]

Options:
 --database               Supported database type (default: "sqlite")
 --database-name          Name of the database
 --database-host          Hostname of the database (default: "localhost")
 --database-user          User name to connect to the database
 --database-pass          Password of the database user
 --admin-user             User name of the admin account (default: "admin")
 --admin-pass             Password of the admin account
 --data-dir               Path to data directory (default:
                          "/var/www/nextcloud/data")
 --help (-h)              Display this help message
 --quiet (-q)             Do not output any message
 --verbose (-v|vv|vvv)    Increase the verbosity of messages: 1 for normal
  output, 2 for more verbose output and 3 for debug
 --version (-V)           Display this application version
 --ansi                   Force ANSI output
 --no-ansi                Disable ANSI output
 --no-interaction (-n)    Do not ask any interactive question
```

This example completes the installation:

```
cd /var/www/nextcloud/
sudo -u www-data php occ maintenance:install --database
"mysql" --database-name "nextcloud"  --database-user "root" --database-pass
"password" --admin-user "admin" --admin-pass "password"
Nextcloud is not installed - only a limited number of commands are available
Nextcloud was successfully installed
```

Supported databases are:

```
- sqlite (SQLite3 - Nextcloud Community edition only)
- mysql (MySQL/MariaDB)
- pgsql (PostgreSQL)
- oci (Oracle - Nextcloud Enterprise edition only)
```

Command line upgrade

These commands are available only after you have downloaded upgraded packages or tar archives, and before you complete the upgrade.

List all options, like this example on CentOS Linux:

```
sudo -u apache php occ upgrade -h
Usage:
upgrade [--quiet]

Options:
--help (-h)           Display this help message.
--quiet (-q)          Do not output any message.
--verbose (-v|vv|vvv)  Increase the verbosity of messages: 1 for normal output,
  2 for more verbose output and 3 for debug.
--version (-V)        Display this application version.
--ansi                Force ANSI output.
--no-ansi             Disable ANSI output.
--no-interaction (-n)  Do not ask any interactive question
```

When you are performing an update or upgrade on your Nextcloud server (see the Maintenance section of this manual), it is better to use `occ` to perform the database upgrade step, rather than the Web GUI, in order to avoid timeouts. PHP scripts invoked from the Web interface are limited to 3600 seconds. In larger environments this may not be enough, leaving the system in an inconsistent state. After performing all the preliminary steps (see How to upgrade) use this command to upgrade your databases, like this example on CentOS Linux. Note how it details the steps:

```
sudo -u www-data php occ upgrade
Nextcloud or one of the apps require upgrade - only a limited number of
commands are available
Turned on maintenance mode
Checked database schema update
Checked database schema update for apps
Updated database
Updating <gallery> ...
Updated <gallery> to 0.6.1
Updating <activity> ...
Updated <activity> to 2.1.0
Update successful
Turned off maintenance mode
```

Enabling verbosity displays timestamps:

```
sudo -u www-data php occ upgrade -v
Nextcloud or one of the apps require upgrade - only a limited number of commands are availab
2015-06-23T09:06:15+0000 Turned on maintenance mode
2015-06-23T09:06:15+0000 Checked database schema update
2015-06-23T09:06:15+0000 Checked database schema update for apps
2015-06-23T09:06:15+0000 Updated database
2015-06-23T09:06:15+0000 Updated <files_sharing> to 0.6.6
2015-06-23T09:06:15+0000 Update successful
2015-06-23T09:06:15+0000 Turned off maintenance mode
```

If there is an error it throws an exception, and the error is detailed in your Nextcloud logfile, so you can use the log output to figure out what went wrong, or to use in a bug report:

```
Turned on maintenance mode
Checked database schema update
Checked database schema update for apps
Updated database
Updating <files_sharing> ...
Exception
ServerNotAvailableException: LDAP server is not available
```

```
Update failed
Turned off maintenance mode
```

Two-factor authentication

If a two-factor provider app is enabled, it is enabled for all users by default (though the provider can decide whether or not the user has to pass the challenge). In the case of a user losing access to the second factor (e.g. lost phone with two-factor SMS verification), the admin can try to disable the two-factor check for that user via the occ command:

```
sudo -u www-data php occ twofactor:disable <uid> <provider_id>
```

Note

This is not supported by all providers.

To re-enable two-factor auth again use the following commmand:

```
sudo -u www-data php occ twofactor:enable <uid> <provider_id>
```

Note

This is not supported by all providers.

Disable users

Admins can disable users via the occ command too:

```
sudo -u www-data php occ user:disable <username>
```

Use the following command to enable the user again:

```
sudo -u www-data php occ user:enable <username>
```

Note that once users are disabled, their connected browsers will be disconnected.

System Tags

List tags:

```
sudo -u www-data php occ tag:list
```

Add a tag:

```
sudo -u www-data php occ tag:add <name> <access>
```

Edit a tag:

```
sudo -u www-data php occ tag:edit --name <name> --access <access> <id>
```

–name and *–access* are optional.

Delete a tag:

```
sudo -u www-data php occ tag:delete <id>
```

Access level

Level	Visible[1]	Assignable[2]
public	Yes	Yes

restricted	Yes	No
invisible	No	No

[1] User can see the tag
[2] User can assign the tag to a file

Debugging

In certain situations it's necessary to generate debugging information, e.g. before submitting a bug report. You can run `occ` with debug logging:

```
sudo -u www-data NC_loglevel=0 php occ -h
```

Activity app

You can configure your Nextcloud server to automatically send out e-mail notifications to your users for various events like:

- A file or folder has been shared
- A new file or folder has been created
- A file or folder has been changed
- A file or folder has been deleted

Users can see actions (delete, add, modify) that happen to files they have access to. Sharing actions are only visible to the sharer and sharee.

Enabling the activity app

The Activity App is shipped and enabled by default. If it is not enabled simply go to your Nextcloud Apps page to enable it.

Configuring your Nextcloud for the activity app

To configure your Nextcloud to send out e-mail notifications a working Email is mandatory.

Furthermore it is recommended to configure the background job `Webcron` or `Cron` as described in Background jobs.

There is also a configuration option `activity_expire_days` available in your `config.php` (See Activity app) which allows you to clean-up older activities from the database.

Activities in groupfolders or external storages

By default activities in groupfolders or external storages are only generated for the current user. This is due to the logic of groupfolders and external storages. There is a config flag `activity_use_cached_mountpoints` that makes activities in groupfolders and external storages work like in normal shares when set to `true`.

Warning

This config option comes with the following limitations:

1. Users that had access to a groupfolder, share or external storage can see activities in their stream and emails that happen after they are removed until they login again
2. Users that are newly added to a groupfolder, share or external storage can not see activities in their stream nor emails that happen after they are added until they login again

Better scheduling of activity emails

In certain scenarios it makes sense to send the activity emails out more regularly, e.g. you want to send the hourly emails always at the full hour, daily emails before people start to work in the morning and weekly mails shall be send on monday morning, so people can read up when starting into the week.

A console command is available to trigger sending those emails. This allows to set up special cron jobs on your server with the known granularity, instead of relying on the Nextcloud cron feature which is not very flexible on scheduling.

To implement the samples mentioned above, the following three entries are necessary:

```
# crontab -u www-data -e
 0  *  *  *  *    php -f /var/www/nextcloud/occ activity:send-mails hourly
30  7  *  *  *    php -f /var/www/nextcloud/occ activity:send-mails daily
30  7  *  *  MON  php -f /var/www/nextcloud/occ activity:send-mails weekly
```

If you want to manually send out all activity emails which are queued, you can run `occ activity:send-mails` without any argument.

Memory caching

You can significantly improve your Nextcloud server performance with memory caching, where frequently-requested objects are stored in memory for faster retrieval. There are two types of caches to use: a PHP opcode cache, which is commonly called *opcache*, and data caching for your Web server. If you do not install and enable a local memcache you will see a warning on your Nextcloud admin page. **A memcache is not required and you may safely ignore the warning if you prefer.**

Note

If you enable only a distributed cache in your `config.php` (`memcache.distributed`) and not a local cache (`memcache.local`) you will still see the cache warning.

A PHP opcache stores compiled PHP scripts so they don't need to be re-compiled every time they are called. PHP bundles the Zend OPcache in core since version 5.5, so you don't need to install an opcache manually.

Data caching is supplied by the user (APCu), Memcached or Redis.

Nextcloud supports multiple memory caching backends, so you can choose the type of memcache that best fits your needs. The supported caching backends are:

- **APCu, APCu 4.0.6 and up required.**

  A local cache for systems.

- **Redis, PHP module 2.2.6 and up required.**

  For local and distributed caching as well as transactional file locking.

- **Memcached**

  For distributed caching.

Memcaches must be explicitly configured in Nextcloud by installing and enabling your desired cache, and then adding the appropriate entry to `config.php` (See Configuration Parameters for an overview of all possible config parameters).

You may use both a local and a distributed cache. Recommended caches are APCu and Redis. After installing and enabling your chosen memcache, verify that it is active by running PHP version and information.

Note

If you run multiple web servers and enable a distributed cache in your `config.php` (`memcache.distributed`) or a file locking provider (`memcache.locking`) you need to make sure that they are referring to the very same memcache server and not to `localhost` or a unix socket.

APCu

APCu is a data cache, and it is available in most Linux distributions. On Red Hat/CentOS/Fedora systems install `php-pecl-apcu`. On Debian/Ubuntu/Mint systems install `php-apcu`.

After restarting your Web server, add this line to your `config.php` file:

```
'memcache.local' => '\OC\Memcache\APCu',
```

Refresh your Nextcloud admin page, and the cache warning should disappear.

Warning

APCu is disabled by default on CLI which could cause issues with nextcloud's cron jobs. Please make sure you set the `apc.enable_cli` to `1` on your `php.ini` config file or append `--define apc.enable_cli=1` to the cron job call.

Redis

Redis is an excellent modern memcache to use for distributed caching, and as a key-value store for Transactional File Locking because it guarantees that cached objects are available for as long as they are needed.

The Redis PHP module must be version 2.2.6+. If you are running a Linux distribution that does not package the supported versions of this module, or does not package Redis at all, see Additional Redis installation help.

On Debian/Ubuntu/Mint install `redis-server` and `php-redis`. The installer will automatically launch `redis-server` and configure it to launch at startup.

On CentOS and Fedora install `redis` and `php-pecl-redis`. It will not start automatically, so you must use your service manager to start `redis`, and to launch it at boot as a daemon.

You can verify that the Redis daemon is running with `ps ax`:

```
ps ax | grep redis
22203 ? Ssl    0:00 /usr/bin/redis-server 127.0.0.1:6379
```

Restart your Web server, add the appropriate entries to your `config.php`, and refresh your Nextcloud admin page. This example `config.php` configuration uses Redis for the distributed server cache:

```
'memcache.distributed' => '\OC\Memcache\Redis',
'redis' => [
     'host' => 'redis-host.example.com',
     'port' => 6379,
],
```

For best performance, use Redis for file locking by adding this:

```
'memcache.locking' => '\OC\Memcache\Redis',
```

If you want to connect to Redis configured to listen on an Unix socket (which is recommended if Redis is running on the same system as Nextcloud) use this example `config.php` configuration:

```
'memcache.local' => '\OC\Memcache\APCu',
'memcache.distributed' => '\OC\Memcache\Redis',
'redis' => [
     'host'     => '/run/redis/redis-server.sock',
     'port'     => 0,
     'dbindex'  => 0,
     'password' => 'secret',
     'timeout'  => 1.5,
],
```

Only “host” and “port” variables are required, the other ones are optional.

Update the redis configuration in `/etc/redis/redis.conf` accordingly: uncomment Unix socket options and ensure the “socket” and “port” settings match your Nextcloud configuration.

Be sure to set the right permissions on redis.sock so that your webserver can read and write to it. For this you typically have to add the webserver user to the redis group:

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

You might need to restart apache for the changes to take effect:

```
systemctl restart apache2
```

Redis is very configurable; consult the Redis documentation to learn more.

Using the Redis session handler: If you are using Redis for locking and/or caching, you may also wish to use Redis for session management. Redis can be used for centralized session management across multiple Nextcloud application servers, unlike the standard *files* handler. If you use the Redis handler, though, you *MUST* ensure that session locking is enabled. As of this writing, the Redis session handler does *NOT* enable session locking by default, which can lead to session corruption in some Nextcloud apps that make heavy use of session writes such as Talk. In addition, even when session locking is enabled, if the application fails to acquire a lock, the Redis session handler does not currently return an error. Adding the following settings in your *php.ini* file will prevent session corruption when using Redis as your session handler:

```
redis.session.locking_enabled=1
redis.session.lock_retries=-1
redis.session.lock_wait_time=10000
```

More information on configuration of phpredis session handler can be found on the PhpRedis GitHub page

Memcached

Memcached is a reliable oldtimer for shared caching on distributed servers, and performs well with Nextcloud with one exception: it is not suitable to use with Transactional File Locking because it does not store locks, and data can disappear from the cache at any time (Redis is the best memcache for this).

Note

Be sure to install the **memcached** PHP module, and not memcache, as in the following examples. Nextcloud supports only the **memcached** PHP module.

Setting up Memcached is easy. On Debian/Ubuntu/Mint install `memcached` and `php-memcached`. The installer will automatically start `memcached` and configure it to launch at startup.

On Red Hat/CentOS/Fedora install `memcached` and `php-pecl-memcached`. It will not start automatically, so you must use your service manager to start `memcached`, and to launch it at boot as a daemon.

You can verify that the Memcached daemon is running with `ps ax`:

```
ps ax | grep memcached
19563 ? Sl 0:02 /usr/bin/memcached -m 64 -p 11211 -u memcache -l
127.0.0.1
```

Restart your Web server, add the appropriate entries to your `config.php`, and refresh your Nextcloud admin page. This example uses APCu for the local cache, Memcached as the distributed memcache, and lists all the servers in the shared cache pool with their port numbers:

```
'memcache.local' => '\OC\Memcache\APCu',
'memcache.distributed' => '\OC\Memcache\Memcached',
'memcached_servers' => [
     [ 'server0.example.com', 11211 ],
     [ 'server1.example.com', 11211 ],
     [ 'server2.example.com', 11211 ],
 ],
```

Cache Directory location

The cache directory defaults to `data/$user/cache` where `$user` is the current user. You may use the `'cache_path'` directive in `config.php` (See Configuration Parameters) to select a different location.

Recommendations based on type of deployment

Small/Private home server

Only use APCu:

```
'memcache.local' => '\OC\Memcache\APCu',
```

Organizations with single-server and clustered setups

Use Redis for everything except local memcache:

```
'memcache.local' => '\OC\Memcache\APCu',
'memcache.distributed' => '\OC\Memcache\Redis',
'memcache.locking' => '\OC\Memcache\Redis',
'redis' => [
     'host' => 'redis-host.example.com',
     'port' => 6379,
],
```

Additional notes for Redis vs. APCu on memory caching

APCu is faster at local caching than Redis. If you have enough memory, use APCu for Memory Caching and Redis for File Locking. If you are low on memory, use Redis for both.

Additional Redis installation help

If your version of Mint or Ubuntu does not package the required version of `php-redis`, then try this Redis guide on Tech and Me for a complete Redis installation on Ubuntu 14.04 using PECL. These instructions are adaptable for any distro that does not package the supported version, or that does not package Redis at all, such as SUSE Linux Enterprise Server and Red Hat Enterprise Linux.

For PHP 7.0 and PHP 7.1 use Redis PHP module 3.1.x or later.

See https://pecl.php.net/package/redis

On Debian/Mint/Ubuntu, use `apt-cache` to see the available `php-redis` version, or the version of your installed package:

```
apt-cache policy php-redis
```

On CentOS and Fedora, the `yum` command shows available and installed version information:

```
yum search php-pecl-redis
```

Background jobs

A system like Nextcloud sometimes requires tasks to be done on a regular basis without the need for user interaction or hindering Nextcloud performance. For that purpose, as a system administrator, you can define background jobs (for example, database clean-ups) which are executed without any need for user interaction.

These jobs are typically referred to as *cron jobs*. Cron jobs are commands or shell-based scripts that are scheduled to run periodically at fixed times, dates, or intervals. `cron.php` is a Nextcloud internal process that runs such background jobs on demand.

Nextcloud apps register actions with `cron.php` automatically to take care of typical housekeeping operations, such as garbage collecting of temporary files or checking for newly updated files using `filescan()` for externally mounted file systems.

Parameters

In the admin settings menu you can configure how cron-jobs should be executed. You can choose between the following options:

- AJAX
- Webcron
- Cron

Cron jobs

You can schedule cron jobs in three ways – using AJAX, Webcron, or cron. The default method is to use AJAX. However, the recommended method is to use cron. The following sections describe the differences between each method.

AJAX

Usecase: Single user instance

The AJAX scheduling method is the default option. Unfortunately, however, it is also the least reliable. Each time a user visits the Nextcloud page, a single background job is executed. The advantage of this mechanism is that it does not require access to the system nor registration with a third party service. The disadvantage of this mechanism, when compared to the Webcron service, is that it requires regular visits to the page for it to be triggered.

> **Warning**
>
> Especially when using the Activity app or external storages, where new files are added, updated or deleted, or when **multiple users** use the server, it is recommended to use `cron`.

Webcron

Usecase: Very small instance (1-5 users depending on the usage)

By registering your Nextcloud `cron.php` script address at an external webcron service (for example, easyCron), you ensure that background jobs are executed regularly. To use this type of service with your server, you must be able to access your server using the Internet. For example:

```
URL to call: http[s]://<domain-of-your-server>/nextcloud/cron.php
```

> **Warning**
>
> Since WebCron is still executed via web, the webserver in most case limits the resources on the execution. To avoid interrupts inside jobs only 1 jobs is executed per call. When webcron is called once every 5 minutes this limits your instance to 288 background jobs per day, which is only suitable for very small instance. For bigger instances it is recommended to use `cron`.

Cron

Using the operating system cron feature is the preferred method for executing regular tasks. This method enables the execution of scheduled jobs without the inherent limitations the Web server might have.

To run a cron job on a *nix system, every 5 minutes, under the default Web server user (often, `www-data` or `wwwrun`), you must set up the following cron job to call the **cron.php** script:

```
# crontab -u www-data -e
```

And append this line:

```
*/5  *  *  *  * php -f /var/www/nextcloud/cron.php
```

You can verify if the cron job has been added and scheduled by executing:

```
# crontab -u www-data -l
```

Which returns:

```
[snip]
*/5  *  *  *  * php -f /var/www/nextcloud/cron.php
```

Note

You have to replace the path `/var/www/nextcloud/cron.php` with the path to your current Nextcloud installation.

Note

On some systems it might be required to call **php-cli** instead of **php**.

Note

For some configurations, it might be neccessary to append `--define apc.enable_cli=1` to the cron command. Please refer to Memory caching (section APCu).

Note

Please refer to the crontab man page for the exact command syntax.

systemd

If systemd is installed on the system, a systemd timer could be an alternative to a cronjob.

This approach requires two files: **nextcloudcron.service** and **nextcloudcron.timer**. Create these two files in `/etc/systemd/system/`.

nextcloudcron.service should look like this:

```
[Unit]
Description=Nextcloud cron.php job

[Service]
User=www-data
ExecStart=/usr/bin/php -f /var/www/nextcloud/cron.php
KillMode=process
```

Replace the user `www-data` with the user of your http server and `/var/www/nextcloud/cron.php` with the location of **cron.php** in your nextcloud directory.

The `KillMode=process` setting is necessary for external programs that are started by the cron job to keep running after the cron job has finished.

Note that the **.service** unit file does not need an `[Install]` section. Please check your setup because we recommended it in earlier versions of this admin manual.

nextcloudcron.timer should look like this:

```
[Unit]
Description=Run Nextcloud cron.php every 5 minutes

[Timer]
OnBootSec=5min
OnUnitActiveSec=5min
Unit=nextcloudcron.service

[Install]
WantedBy=timers.target
```

The important parts in the timer-unit are `OnBootSec` and `OnUnitActiveSec`. `OnBootSec` will start the timer 5 minutes after boot, otherwise you would have to start it manually after every boot. `OnUnitActiveSec` will set a 5 minute timer after the service-unit was last activated.

Now all that is left is to start and enable the timer by running this command:

```
systemctl enable --now nextcloudcron.timer
```

When the option `--now` is used with `enable`, the respective unit will also be started.

Note

Selecting the option `Cron` in the admin menu for background jobs is not mandatory, because once *cron.php* is executed from the command line or cron service it will set it automatically to `Cron`.

Configuration Parameters

Nextcloud uses the `config/config.php` file to control server operations. `config/config.sample.php` lists all the configurable parameters within Nextcloud, along with example or default values. This document provides a more detailed reference. Most options are configurable on your Admin page, so it is usually not necessary to edit `config/config.php`.

Note

The installer creates a configuration containing the essential parameters. Only manually add configuration parameters to `config/config.php` if you need to use a special value for a parameter. **Do not copy everything from** `config/config.sample.php` **. Only enter the parameters you wish to modify!**

Multiple config.php file

Nextcloud supports loading configuration parameters from multiple files. You can add arbitrary files ending with .config.php in the config/ directory, for example you could place your email server configuration in email.config.php. This allows you to easily create and manage custom configurations, or to divide a large complex configuration file into a set of smaller files. These custom files are not overwritten by Nextcloud, and the values in these files take precedence over config.php.

Default Parameters

These parameters are configured by the Nextcloud installer, and are required for your Nextcloud server to operate.

```
'instanceid' => '',
```

This is a unique identifier for your Nextcloud installation, created automatically by the installer. This example is for documentation only, and you should never use it because it will not work. A valid `instanceid` is created when you install Nextcloud.

‘instanceid’ => ‘d3c944a9a’,

```
'passwordsalt' => '',
```

The salt used to hash all passwords, auto-generated by the Nextcloud installer. (There are also per-user salts.) If you lose this salt you lose all your passwords. This example is for documentation only, and you should never use it.

```
'trusted_domains' =>
   [
    'demo.example.org',
    'otherdomain.example.org',
    '10.111.112.113',
    '[2001:db8::1]'
  ],
```

Your list of trusted domains that users can log into. Specifying trusted domains prevents host header poisoning. Do not remove this, as it performs necessary security checks.

You can specify:

- the exact hostname of your host or virtual host, e.g. demo.example.org.
- the exact hostname with permitted port, e.g. demo.example.org:443. This disallows all other ports on this host
- use * as a wildcard, e.g. ubos-raspberry-pi*.local will allow ubos-raspberry-pi.local and ubos-raspberry-pi-2.local
- the IP address with or without permitted port, e.g. [2001:db8::1]:8080 Using TLS certificates where commonName=<IP address> is deprecated

```
'datadirectory' => '/var/www/nextcloud/data',
```

Where user files are stored. The SQLite database is also stored here, when you use SQLite.

Default to `data/` in the Nextcloud directory.

```
'version' => '',
```

The current version number of your Nextcloud installation. This is set up during installation and update, so you shouldn’t need to change it.

```
'dbtype' => 'sqlite3',
```

Identifies the database used with this installation. See also config option `supportedDatabases`

Available:

- sqlite3 (SQLite3)
- mysql (MySQL/MariaDB)
- pgsql (PostgreSQL)

Defaults to `sqlite3`

```
'dbhost' => '',
```

Your host server name, for example `localhost`, `hostname`, `hostname.example.com`, or the IP address. To specify a port use `hostname:####`; to specify a Unix socket use `/path/to/directory/containing/socket` e.g. `/run/postgresql/`.

```
'dbname' => 'nextcloud',
```

The name of the Nextcloud database, which is set during installation. You should not need to change this.

```
'dbuser' => '',
```

The user that Nextcloud uses to write to the database. This must be unique across Nextcloud instances using the same SQL database. This is set up during installation, so you shouldn’t need to change it.

```
'dbpassword' => '',
```

The password for the database user. This is set up during installation, so you shouldn’t need to change it.

```
'dbtableprefix' => '',
```

Prefix for the Nextcloud tables in the database.

Default to oc_

```
'installed' => false,
```

Indicates whether the Nextcloud instance was installed successfully; `true` indicates a successful installation, and `false` indicates an unsuccessful installation.

Defaults to `false`

Default config.php Examples

When you use SQLite as your Nextcloud database, your `config.php` looks like this after installation. The SQLite database is stored in your Nextcloud `data/` directory. SQLite is a simple, lightweight embedded database that is good for testing and for simple installations, but for production Nextcloud systems you should use MySQL, MariaDB, or PosgreSQL.

```
<?php
$CONFIG = array (
  'instanceid' => 'occ6f7365735',
  'passwordsalt' => '2c57784763467863063O3',
  'trusted_domains' =>
  array (
    0 => 'localhost',
    1 => 'studio',
  ),
  'datadirectory' => '/var/www/nextcloud/data',
  'dbtype' => 'sqlite3',
  'version' => '7.0.2.1',
  'installed' => true,
);
```

This example is from a new Nextcloud installation using MariaDB:

```
<?php
$CONFIG = array (
  'instanceid' => 'oc8c0fd71e03',
  'passwordsalt' => '515a13302a6b3950a9d0fdb970191a',
  'trusted_domains' =>
  array (
    0 => 'localhost',
    1 => 'studio',
    2 => '192.168.10.155'
  ),
  'datadirectory' => '/var/www/nextcloud/data',
  'dbtype' => 'mysql',
   'version' => '7.0.2.1',
  'dbname' => 'nextcloud',
  'dbhost' => 'localhost',
  'dbtableprefix' => 'oc_',
  'dbuser' => 'oc_carla',
  'dbpassword' => '67336bcdf7630dd80b2b81a413d07',
  'installed' => true,
);
```

User Experience

These optional parameters control some aspects of the user interface. Default values, where present, are shown.

```
'default_language' => 'en',
```

This sets the default language on your Nextcloud server, using ISO_639-1 language codes such as `en` for English, `de` for German, and `fr` for French. It overrides automatic language detection on public pages like login or shared items. User's language preferences configured under "personal -> language" override this setting after they have logged in. Nextcloud has two distinguished language codes for German, 'de' and 'de_DE'. 'de' is used for informal German and 'de_DE' for formal German. By setting this value to 'de_DE' you can enforce the formal version of German unless the user has chosen something different explicitly.

Defaults to `en`

```
'force_language' => 'en',
```

With this setting a language can be forced for all users. If a language is forced, the users are also unable to change their language in the personal settings. If users shall be unable to change their language, but users have different languages, this value can be set to `true` instead of a language code.

Defaults to `false`

```
'default_locale' => 'en_US',
```

This sets the default locale on your Nextcloud server, using ISO_639 language codes such as `en` for English, `de` for German, and `fr` for French, and ISO-3166 country codes such as `GB`, `US`, `CA`, as defined in RFC 5646. It overrides automatic locale detection on public pages like login or shared items. User's locale preferences configured under "personal -> locale" override this setting after they have logged in.

Defaults to `en`

```
'default_phone_region' => 'GB',
```

This sets the default region for phone numbers on your Nextcloud server, using ISO 3166-1 country codes such as `DE` for Germany, `FR` for France, … It is required to allow inserting phone numbers in the user profiles starting without the country code (e.g. +49 for Germany).

No default value!

```
'force_locale' => 'en_US',
```

With this setting a locale can be forced for all users. If a locale is forced, the users are also unable to change their locale in the personal settings. If users shall be unable to change their locale, but users have different languages, this value can be set to `true` instead of a locale code.

Defaults to `false`

```
'defaultapp' => 'dashboard,files',
```

Set the default app to open on login. Use the app names as they appear in the URL after clicking them in the Apps menu, such as documents, calendar, and gallery. You can use a comma-separated list of app names, so if the first app is not enabled for a user then Nextcloud will try the second one, and so on. If no enabled apps are found it defaults to the dashboard app.

Defaults to `dashboard,files`

```
'knowledgebaseenabled' => true,
```

`true` enables the Help menu item in the user menu (top right of the Nextcloud Web interface). `false` removes the Help item.

```
'allow_user_to_change_display_name' => true,
```

`true` allows users to change their display names (on their Personal pages), and `false` prevents them from changing their display names.

```
'remember_login_cookie_lifetime' => 60*60*24*15,
```

Lifetime of the remember login cookie. This should be larger than the session_lifetime. If it is set to 0 remember me is disabled.

Defaults to `60*60*24*15` seconds (15 days)

```
'session_lifetime' => 60 * 60 * 24,
```

The lifetime of a session after inactivity.

The maximum possible time is limited by the session.gc_maxlifetime php.ini setting which would overwrite this option if it is less than the value in the config.php

Defaults to `60*60*24` seconds (24 hours)

```
'session_keepalive' => true,
```

Enable or disable session keep-alive when a user is logged in to the Web UI.

Enabling this sends a “heartbeat” to the server to keep it from timing out.

Defaults to `true`

```
'auto_logout' => false,
```

Enable or disable the automatic logout after session_lifetime, even if session keepalive is enabled. This will make sure that an inactive browser will be logged out even if requests to the server might extend the session lifetime.

Defaults to `false`

```
'token_auth_enforced' => false,
```

Enforce token authentication for clients, which blocks requests using the user password for enhanced security. Users need to generate tokens in personal settings which can be used as passwords on their clients.

Defaults to `false`

```
'token_auth_activity_update' => 60,
```

The interval at which token activity should be updated.

Increasing this value means that the last activity on the security page gets more outdated.

Tokens are still checked every 5 minutes for validity max value: 300

Defaults to `300`

```
'auth.bruteforce.protection.enabled' => true,
```

Whether the bruteforce protection shipped with Nextcloud should be enabled or not.

Disabling this is discouraged for security reasons.

Defaults to `true`

```
'ratelimit.protection.enabled' => true,
```

Whether the rate limit protection shipped with Nextcloud should be enabled or not.

Disabling this is discouraged for security reasons.

Defaults to `true`

```
'auth.authtoken.v1.disabled' => false,
```

Whether the authtoken v1 provider should be skipped

The v1 provider is deprecated and removed in Nextcloud 24 onwards. It can be disabled already when the instance was installed after Nextcloud 14.

Defaults to `false`

```
'auth.webauthn.enabled' => true,
```

By default WebAuthn is available but it can be explicitly disabled by admins

```
'auth.storeCryptedPassword' => true,
```

Whether encrypted password should be stored in the database

The passwords are only decrypted using the login token stored uniquely in the clients and allow to connect to external storages, autoconfigure mail account in the mail app and periodically check if the password it still valid.

This might be desirable to disable this functionality when using one time passwords or when having a password policy enforcing long passwords (> 300 characters).

By default the passwords are stored encrypted in the database.

```
'hide_login_form' => false,
```

By default the login form is always available. There are cases (SSO) where an admin wants to avoid users entering their credentials to the system if the SSO app is unavailable.

This will show an error. But the the direct login still works with adding ?direct=1

```
'skeletondirectory' => '/path/to/nextcloud/core/skeleton',
```

The directory where the skeleton files are located. These files will be copied to the data directory of new users. Leave empty to not copy any skeleton files.

`{lang}` can be used as a placeholder for the language of the user. If the directory does not exist, it falls back to non dialect (from `de_DE` to `de`). If that does not exist either, it falls back to `default`

Defaults to `core/skeleton` in the Nextcloud directory.

```
'templatedirectory' => '/path/to/nextcloud/templates',
```

The directory where the template files are located. These files will be copied to the template directory of new users. Leave empty to not copy any template files.

`{lang}` can be used as a placeholder for the language of the user. If the directory does not exist, it falls back to non dialect (from `de_DE` to `de`). If that does not exist either, it falls back to `default`

If this is not set creating a template directory will only happen if no custom `skeletondirectory` is defined, otherwise the shipped templates will be used to create a template directory for the user.

```
'lost_password_link' => 'https://example.org/link/to/password/reset',
```

If your user backend does not allow password resets (e.g. when it's a read-only user backend like LDAP), you can specify a custom link, where the user is redirected to, when clicking the "reset password" link after a failed login-attempt.

In case you do not want to provide any link, replace the url with 'disabled'

Mail Parameters

These configure the email settings for Nextcloud notifications and password resets.

```
'mail_domain' => 'example.com',
```

The return address that you want to appear on emails sent by the Nextcloud server, for example `nc-admin@example.com`, substituting your own domain, of course.

```
'mail_from_address' => 'nextcloud',
```

FROM address that overrides the built-in `sharing-noreply` and `lostpassword-noreply` FROM addresses.

Defaults to different from addresses depending on the feature.

```
'mail_smtpdebug' => false,
```

Enable SMTP class debugging.

Defaults to `false`

```
'mail_smtpmode' => 'smtp',
```

Which mode to use for sending mail: `sendmail`, `smtp` or `qmail`.

If you are using local or remote SMTP, set this to `smtp`.

For the `sendmail` option you need an installed and working email system on the server, with `/usr/sbin/sendmail` installed on your Unix system.

For `qmail` the binary is /var/qmail/bin/sendmail, and it must be installed on your Unix system.

Defaults to `smtp`

```
'mail_smtphost' => '127.0.0.1',
```

This depends on `mail_smtpmode`. Specify the IP address of your mail server host. This may contain multiple hosts separated by a semi-colon. If you need to specify the port number append it to the IP address separated by a colon, like this: `127.0.0.1:24`.

Defaults to `127.0.0.1`

```
'mail_smtpport' => 25,
```

This depends on `mail_smtpmode`. Specify the port for sending mail.

Defaults to `25`

```
'mail_smtptimeout' => 10,
```

This depends on `mail_smtpmode`. This sets the SMTP server timeout, in seconds. You may need to increase this if you are running an anti-malware or spam scanner.

Defaults to `10` seconds

```
'mail_smtpsecure' => '',
```

This depends on `mail_smtpmode`. Specify when you are using `ssl` for SSL/TLS or `tls` for STARTTLS, or leave empty for no encryption.

Defaults to `''` (empty string)

```
'mail_smtpauth' => false,
```

This depends on `mail_smtpmode`. Change this to `true` if your mail server requires authentication.

Defaults to `false`

```
'mail_smtpauthtype' => 'LOGIN',
```

This depends on `mail_smtpmode`. If SMTP authentication is required, choose the authentication type as `LOGIN` or `PLAIN`.

Defaults to `LOGIN`

```
'mail_smtpname' => '',
```

This depends on `mail_smtpauth`. Specify the username for authenticating to the SMTP server.

Defaults to `''` (empty string)

```
'mail_smtppassword' => '',
```

This depends on `mail_smtpauth`. Specify the password for authenticating to the SMTP server.

Default to `''` (empty string)

```
'mail_template_class' => '\OC\Mail\EMailTemplate',
```

Replaces the default mail template layout. This can be utilized if the options to modify the mail texts with the theming app is not enough.

The class must extend `\OC\Mail\EMailTemplate`

```
'mail_send_plaintext_only' => false,
```

Email will be send by default with an HTML and a plain text body. This option allows to only send plain text emails.

```
'mail_smtpstreamoptions' => [],
```

This depends on `mail_smtpmode`. Array of additional streams options that will be passed to underlying Swift mailer implementation.

Defaults to an empty array.

```
'mail_sendmailmode' => 'smtp',
```

Which mode is used for sendmail/qmail: `smtp` or `pipe`.

For `smtp` the sendmail binary is started with the parameter `-bs`:

- Use the SMTP protocol on standard input and output.

For `pipe` the binary is started with the parameters `-t`:

- Read message from STDIN and extract recipients.

Defaults to `smtp`

Proxy Configurations

```
'overwritehost' => '',
```

The automatic hostname detection of Nextcloud can fail in certain reverse proxy and CLI/cron situations. This option allows you to manually override the automatic detection; for example `www.example.com`, or specify the port `www.example.com:8080`.

```
'overwriteprotocol' => '',
```

When generating URLs, Nextcloud attempts to detect whether the server is accessed via `https` or `http`. However, if Nextcloud is behind a proxy and the proxy handles the `https` calls, Nextcloud would not know that `ssl` is in use, which would result in incorrect URLs being generated.

Valid values are `http` and `https`.

```
'overwritewebroot' => '',
```

Nextcloud attempts to detect the webroot for generating URLs automatically.

For example, if `www.example.com/nextcloud` is the URL pointing to the Nextcloud instance, the webroot is `/nextcloud`. When proxies are in use, it may be difficult for Nextcloud to detect this parameter, resulting in invalid URLs.

```
'overwritecondaddr' => '',
```

This option allows you to define a manual override condition as a regular expression for the remote IP address. For example, defining a range of IP addresses starting with `10.0.0.` and ending with 1 to 3: `^10.0.0.[1-3]$`

Defaults to `''` (empty string)

```
'overwrite.cli.url' => '',
```

Use this configuration parameter to specify the base URL for any URLs which are generated within Nextcloud using any kind of command line tools (cron or occ). The value should contain the full base URL: `https://www.example.com/nextcloud`

Defaults to `''` (empty string)

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

To have clean URLs without */index.php* this parameter needs to be configured.

This parameter will be written as “RewriteBase” on update and installation of Nextcloud to your *.htaccess* file. While this value is often simply the URL path of the Nextcloud installation it cannot be set automatically properly in every scenario and needs thus some manual configuration.

In a standard Apache setup this usually equals the folder that Nextcloud is accessible at. So if Nextcloud is accessible via “https://mycloud.org/nextcloud” the correct value would most likely be “/nextcloud”. If Nextcloud is running under “https://mycloud.org/” then it would be “/”.

Note that the above rule is not valid in every case, as there are some rare setup cases where this may not apply. However, to avoid any update problems this configuration value is explicitly opt-in.

After setting this value run *occ maintenance:update:htaccess*. Now, when the following conditions are met Nextcloud URLs won’t contain *index.php*:

- *mod_rewrite* is installed
- *mod_env* is installed

Defaults to `''` (empty string)

```
'htaccess.IgnoreFrontController' => false,
```

For server setups, that don't have *mod_env* enabled or restricted (e.g. suEXEC) this parameter has to be set to true and will assume mod_rewrite.

Please check, if *mod_rewrite* is active and functional before setting this parameter and you updated your .htaccess with *occ maintenance:update:htaccess*. Otherwise your nextcloud installation might not be reachable anymore. For example, try accessing resources by leaving out *index.php* in the URL.

```
'proxy' => '',
```

The URL of your proxy server, for example `proxy.example.com:8081`.

Note: Guzzle (the http library used by Nextcloud) is reading the environment variables HTTP_PROXY (only for cli request), HTTPS_PROXY, and NO_PROXY by default.

If you configure proxy with Nextcloud any default configuration by Guzzle is overwritten. Make sure to set `proxyexclude` accordingly if necessary.

Defaults to `''` (empty string)

```
'proxyuserpwd' => '',
```

The optional authentication for the proxy to use to connect to the internet.

The format is: `username:password`.

Defaults to `''` (empty string)

```
'proxyexclude' => [],
```

List of host names that should not be proxied to.

For example: `['.mit.edu', 'foo.com']`.

Hint: Use something like `explode(',', getenv('NO_PROXY'))` to sync this value with the global NO_PROXY option.

Defaults to empty array.

```
'allow_local_remote_servers' => true,
```

Allow remote servers with local addresses e.g. in federated shares, webcal services and more

Defaults to false

Deleted Items (trash bin)

These parameters control the Deleted files app.

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

If the trash bin app is enabled (default), this setting defines the policy for when files and folders in the trash bin will be permanently deleted.

The app allows for two settings, a minimum time for trash bin retention, and a maximum time for trash bin retention. Minimum time is the number of days a file will be kept, after which it may be deleted. Maximum time is the number of days at which it is guaranteed to be deleted. Both minimum and maximum times can be set together to explicitly define file and folder deletion. For migration purposes, this setting is installed initially set to "auto", which is equivalent to the default setting in Nextcloud.

Available values:

- `auto`

  default setting. keeps files and folders in the trash bin for 30 days and automatically deletes anytime after that if space is needed (note: files may not be deleted if space is not needed).

- `D, auto`

  keeps files and folders in the trash bin for D+ days, delete anytime if space needed (note: files may not be deleted if space is not needed)

- `auto, D`
  delete all files in the trash bin that are older than D days automatically, delete other files anytime if space needed
- `D1, D2`
  keep files and folders in the trash bin for at least D1 days and delete when exceeds D2 days (note: files will not be deleted automatically if space is needed)
- `disabled`
  trash bin auto clean disabled, files and folders will be kept forever

Defaults to `auto`

File versions

These parameters control the Versions app.

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

If the versions app is enabled (default), this setting defines the policy for when versions will be permanently deleted.

The app allows for two settings, a minimum time for version retention, and a maximum time for version retention. Minimum time is the number of days a version will be kept, after which it may be deleted. Maximum time is the number of days at which it is guaranteed to be deleted. Both minimum and maximum times can be set together to explicitly define version deletion. For migration purposes, this setting is installed initially set to “auto”, which is equivalent to the default setting in Nextcloud.

Available values:

- `auto`
  default setting. Automatically expire versions according to expire rules. Please refer to Controlling file versions and aging for more information.
- `D, auto`
  keep versions at least for D days, apply expire rules to all versions that are older than D days
- `auto, D`
  delete all versions that are older than D days automatically, delete other versions according to expire rules
- `D1, D2`
  keep versions for at least D1 days and delete when exceeds D2 days
- `disabled`
  versions auto clean disabled, versions will be kept forever

Defaults to `auto`

Nextcloud Verifications

Nextcloud performs several verification checks. There are two options, `true` and `false`.

```
'appcodechecker' => true,
```

Checks an app before install whether it uses private APIs instead of the proper public APIs. If this is set to true it will only allow to install or enable apps that pass this check.

Defaults to `false`

```
'updatechecker' => true,
```

Check if Nextcloud is up-to-date and shows a notification if a new version is available. It sends current version, php version, installation and last update time and release channel to the updater server which responds with the latest available version based on those metrics.

Defaults to `true`

```
'updater.server.url' => 'https://updates.nextcloud.com/updater_server/',
```

URL that Nextcloud should use to look for updates

Defaults to `https://updates.nextcloud.com/updater_server/`

```
'updater.release.channel' => 'stable',
```

The channel that Nextcloud should use to look for updates

Supported values:

- `daily`
- `beta`
- `stable`

```
'has_internet_connection' => true,
```

Is Nextcloud connected to the Internet or running in a closed network?

Defaults to `true`

```
'connectivity_check_domains' => [
        'www.nextcloud.com',
        'www.startpage.com',
        'www.eff.org',
        'www.edri.org'
],
```

Which domains to request to determine the availability of an Internet connection. If none of these hosts are reachable, the administration panel will show a warning. Set to an empty list to not do any such checks (warning will still be shown).

If no protocol is provided, both http and https will be tested. For example, 'http://www.nextcloud.com' and 'https://www.nextcloud.com' will be tested for 'www.nextcloud.com' If a protocol is provided, only this one will be tested.

Defaults to the following domains:

- www.nextcloud.com
- www.startpage.com
- www.eff.org
- www.edri.org

```
'check_for_working_wellknown_setup' => true,
```

Allows Nextcloud to verify a working .well-known URL redirects. This is done by attempting to make a request from JS to https://your-domain.com/.well-known/caldav/

Defaults to `true`

```
'check_for_working_htaccess' => true,
```

This is a crucial security check on Apache servers that should always be set to `true`. This verifies that the `.htaccess` file is writable and works.

If it is not, then any options controlled by `.htaccess`, such as large file uploads, will not work. It also runs checks on the `data/` directory, which verifies that it can't be accessed directly through the Web server.

Defaults to `true`

```
'check_data_directory_permissions' => true,
```

In rare setups (e.g. on Openshift or docker on windows) the permissions check might block the installation while the underlying system offers no means to "correct" the permissions. In this case, set the value to false.

In regular cases, if issues with permissions are encountered they should be adjusted accordingly. Changing the flag is discouraged.

Defaults to `true`

```
'config_is_read_only' => false,
```

In certain environments it is desired to have a read-only configuration file.

When this switch is set to `true` Nextcloud will not verify whether the configuration is writable. However, it will not be possible to configure all options via the Web interface. Furthermore, when updating Nextcloud it is required to make the configuration file writable again for the update process.

Defaults to `false`

Logging

```
'log_type' => 'file',
```

This parameter determines where the Nextcloud logs are sent.

`file`: the logs are written to file `nextcloud.log` in the default Nextcloud data directory. The log file can be changed with parameter `logfile`. `syslog`: the logs are sent to the system log. This requires a syslog daemon to be active. `errorlog`: the logs are sent to the PHP `error_log` function. `systemd`: the logs are sent to the Systemd journal. This requires a system that runs Systemd and the Systemd journal. The PHP extension `systemd` must be installed and active.

Defaults to `file`

```
'log_type_audit' => 'file',
```

This parameter determines where the audit logs are sent. See `log_type` for more information.

Defaults to `file`

```
'logfile' => '/var/log/nextcloud.log',
```

Name of the file to which the Nextcloud logs are written if parameter `log_type` is set to `file`.

Defaults to `[datadirectory]/nextcloud.log`

```
'logfile_audit' => '/var/log/audit.log',
```

Name of the file to which the audit logs are written if parameter `log_type` is set to `file`.

Defaults to `[datadirectory]/audit.log`

```
'logfilemode' => 0640,
```

Log file mode for the Nextcloud logging type in octal notation.

Defaults to 0640 (writeable by user, readable by group).

```
'loglevel' => 2,
```

Loglevel to start logging at. Valid values are: 0 = Debug, 1 = Info, 2 = Warning, 3 = Error, and 4 = Fatal. The default value is Warning.

Defaults to 2

```
'syslog_tag' => 'Nextcloud',
```

If you maintain different instances and aggregate the logs, you may want to distinguish between them. `syslog_tag` can be set per instance with a unique id. Only available if `log_type` is set to `syslog` or `systemd`.

The default value is `Nextcloud`.

```
'syslog_tag_audit' => 'Nextcloud',
```

If you maintain different instances and aggregate the logs, you may want to distinguish between them. `syslog_tag_audit` can be set per instance with a unique id. Only available if `log_type` is set to `syslog` or `systemd`.

The default value is the value of `syslog_tag`.

```
'log.condition' => [
        'shared_secret' => '57b58edb6637fe3059b3595cf9c41b9',
        'users' => ['sample-user'],
        'apps' => ['files'],
],
```

Log condition for log level increase based on conditions. Once one of these conditions is met, the required log level is set to debug. This allows to debug specific requests, users or apps

Supported conditions:

- `shared_secret`**: if a request parameter with the name** *log_secret* **is set to**
  this value the condition is met
- `users`**: if the current request is done by one of the specified users,**
  this condition is met
- `apps`**: if the log message is invoked by one of the specified apps,**
  this condition is met

Defaults to an empty array.

```
'logdateformat' => 'F d, Y H:i:s',
```

This uses PHP.date formatting; see https://www.php.net/manual/en/function.date.php

Defaults to ISO 8601 `2005-08-15T15:52:01+00:00` - see DateTime::ATOM (https://www.php.net/manual/en/class.datetime.php#datetime.constants.atom)

```
'logtimezone' => 'Europe/Berlin',
```

The timezone for logfiles. You may change this; see https://www.php.net/manual/en/timezones.php

Defaults to `UTC`

```
'log_query' => false,
```

Append all database queries and parameters to the log file. Use this only for debugging, as your logfile will become huge.

```
'log_rotate_size' => 100 * 1024 * 1024,
```

Enables log rotation and limits the total size of logfiles. Set it to 0 for no rotation. Specify a size in bytes, for example 104857600 (100 megabytes = 100 * 1024 * 1024 bytes). A new logfile is created with a new name when the old logfile reaches your limit. If a rotated log file is already present, it will be overwritten.

Defaults to 100 MB

Alternate Code Locations

Some of the Nextcloud code may be stored in alternate locations.

```
'customclient_desktop' =>
        'https://nextcloud.com/install/#install-clients',
'customclient_android' =>
        'https://play.google.com/store/apps/details?id=com.nextcloud.client',
'customclient_ios' =>
        'https://itunes.apple.com/us/app/nextcloud/id1125420102?mt=8',
'customclient_ios_appid' =>
                '1125420102',
```

This section is for configuring the download links for Nextcloud clients, as seen in the first-run wizard and on Personal pages.

Defaults to:

- Desktop client: `https://nextcloud.com/install/#install-clients`

- Android client: `https://play.google.com/store/apps/details?id=com.nextcloud.client`
- iOS client: `https://itunes.apple.com/us/app/nextcloud/id1125420102?mt=8`
- iOS client app id: `1125420102`

Apps

Options for the Apps folder, Apps store, and App code checker.

```
'appstoreenabled' => true,
```

When enabled, admins may install apps from the Nextcloud app store.

Defaults to `true`

```
'appstoreurl' => 'https://apps.nextcloud.com/api/v1',
```

Enables the installation of apps from a self hosted apps store.

Requires that at least one of the configured apps directories is writeable.

Defaults to `https://apps.nextcloud.com/api/v1`

```
'appsallowlist' => [],
```

Filters allowed installable apps from the appstore.

Empty array will prevent all apps from the store to be found.

```
'apps_paths' => [
        [
                'path'=> '/var/www/nextcloud/apps',
                'url' => '/apps',
                'writable' => true,
        ],
],
```

Use the `apps_paths` parameter to set the location of the Apps directory, which should be scanned for available apps, and where user-specific apps should be installed from the Apps store. The `path` defines the absolute file system path to the app folder. The key `url` defines the HTTP Web path to that folder, starting from the Nextcloud webroot. The key `writable` indicates if a Web server can write files to that folder.

```
'appcodechecker' => true,
```

Checks an app before install whether it uses private APIs instead of the proper public APIs. If this is set to true it will only allow to install or enable apps that pass this check.

Defaults to `false`

Previews

Nextcloud supports previews of image files, the covers of MP3 files, and text files. These options control enabling and disabling previews, and thumbnail size.

```
'enable_previews' => true,
```

By default, Nextcloud can generate previews for the following filetypes:

- Image files
- Covers of MP3 files
- Text documents

Valid values are `true`, to enable previews, or `false`, to disable previews

Defaults to `true`

```
'preview_max_x' => 4096,
```

The maximum width, in pixels, of a preview. A value of `null` means there is no limit.

Defaults to `4096`

```
'preview_max_y' => 4096,
```

The maximum height, in pixels, of a preview. A value of `null` means there is no limit.

Defaults to `4096`

```
'preview_max_filesize_image' => 50,
```

max file size for generating image previews with imagegd (default behavior) If the image is bigger, it'll try other preview generators, but will most likely show the default mimetype icon. Set to -1 for no limit.

Defaults to `50` megabytes

```
'preview_max_memory' => 128,
```

max memory for generating image previews with imagegd (default behavior) Reads the image dimensions from the header and assumes 32 bits per pixel.

If creating the image would allocate more memory, preview generation will be disabled and the default mimetype icon is shown. Set to -1 for no limit.

Defaults to `128` megabytes

```
'preview_libreoffice_path' => '/usr/bin/libreoffice',
```

custom path for LibreOffice/OpenOffice binary

Defaults to `''` (empty string)

```
'preview_office_cl_parameters' =>
        ' --headless --nologo --nofirststartwizard --invisible --norestore '.
        '--convert-to png --outdir ',
```

Use this if LibreOffice/OpenOffice requires additional arguments.

Defaults to `''` (empty string)

```
'enabledPreviewProviders' => [
        'OC\Preview\PNG',
        'OC\Preview\JPEG',
        'OC\Preview\GIF',
        'OC\Preview\BMP',
        'OC\Preview\XBitmap',
        'OC\Preview\MP3',
        'OC\Preview\TXT',
        'OC\Preview\MarkDown',
        'OC\Preview\OpenDocument',
        'OC\Preview\Krita',
],
```

Only register providers that have been explicitly enabled

The following providers are disabled by default due to performance or privacy concerns:

- OC\Preview\Illustrator
- OC\Preview\HEIC
- OC\Preview\Movie
- OC\Preview\MSOffice2003
- OC\Preview\MSOffice2007
- OC\Preview\MSOfficeDoc
- OC\Preview\PDF
- OC\Preview\Photoshop

- OC\Preview\Postscript
- OC\Preview\StarOffice
- OC\Preview\SVG
- OC\Preview\TIFF
- OC\Preview\Font

Defaults to the following providers:

- OC\Preview\BMP
- OC\Preview\GIF
- OC\Preview\JPEG
- OC\Preview\MarkDown
- OC\Preview\MP3
- OC\Preview\PNG
- OC\Preview\TXT
- OC\Preview\XBitmap
- OC\Preview\OpenDocument
- OC\Preview\Krita

LDAP

Global settings used by LDAP User and Group Backend

```
'ldapUserCleanupInterval' => 51,
```

defines the interval in minutes for the background job that checks user existence and marks them as ready to be cleaned up. The number is always minutes. Setting it to 0 disables the feature.

See command line (occ) methods `ldap:show-remnants` and `user:delete`

Defaults to `51` minutes

```
'sort_groups_by_name' => false,
```

Sort groups in the user settings by name instead of the user count

By enabling this the user count beside the group name is disabled as well.

Comments

Global settings for the Comments infrastructure

```
'comments.managerFactory' => '\OC\Comments\ManagerFactory',
```

Replaces the default Comments Manager Factory. This can be utilized if an own or 3rdParty CommentsManager should be used that – for instance – uses the filesystem instead of the database to keep the comments.

Defaults to `\OC\Comments\ManagerFactory`

```
'systemtags.managerFactory' => '\OC\SystemTag\ManagerFactory',
```

Replaces the default System Tags Manager Factory. This can be utilized if an own or 3rdParty SystemTagsManager should be used that – for instance – uses the filesystem instead of the database to keep the tags.

Defaults to `\OC\SystemTag\ManagerFactory`

Maintenance

These options are for halting user activity when you are performing server maintenance.

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

Enable maintenance mode to disable Nextcloud

If you want to prevent users from logging in to Nextcloud before you start doing some maintenance work, you need to set the value of the maintenance parameter to true. Please keep in mind that users who are already logged-in are kicked out of Nextcloud instantly.

Defaults to `false`

```
'maintenance_window_start' => 1,
```

UTC Hour for maintenance windows

Some background jobs only run once a day. When an hour is defined for this config, the background jobs which advertise themselves as not time sensitive will be delayed during the “working” hours and only run in the 4 hours after the given time. This is e.g. used for activity expiration, suspicious login training and update checks.

A value of 1 e.g. will only run these background jobs between 01:00am UTC and 05:00am UTC.

Defaults to `100` which disables the feature

SSL

```
'openssl' => [
        'config' => '/absolute/location/of/openssl.cnf',
],
```

Extra SSL options to be used for configuration.

Defaults to an empty array.

Memory caching backend configuration

Available cache backends:

- `\OC\Memcache\APCu` APC user backend
- `\OC\Memcache\ArrayCache` In-memory array-based backend (not recommended)
- `\OC\Memcache\Memcached` Memcached backend
- `\OC\Memcache\Redis` Redis backend

Advice on choosing between the various backends:

- APCu should be easiest to install. Almost all distributions have packages. Use this for single user environment for all caches.
- Use Redis or Memcached for distributed environments. For the local cache (you can configure two) take APCu.

```
'memcache.local' => '\OC\Memcache\APCu',
```

Memory caching backend for locally stored data

- Used for host-specific data, e.g. file paths

Defaults to `none`

```
'memcache.distributed' => '\OC\Memcache\Memcached',
```

Memory caching backend for distributed data

- Used for installation-specific data, e.g. database caching
- If unset, defaults to the value of memcache.local

Defaults to `none`

```
'redis' => [
        'host' => 'localhost', // can also be a unix domain socket: '/tmp/redis.sock'
        'port' => 6379,
```

```
        'timeout' => 0.0,
        'read_timeout' => 0.0,
        'user' =>  '', // Optional, if not defined no password will be used.
        'password' => '', // Optional, if not defined no password will be used.
        'dbindex' => 0, // Optional, if undefined SELECT will not run and will use Redis Ser
        // If redis in-transit encryption is enabled, provide certificates
        // SSL context https://www.php.net/manual/en/context.ssl.php
        'ssl_context' => [
                'local_cert' => '/certs/redis.crt',
                'local_pk' => '/certs/redis.key',
                'cafile' => '/certs/ca.crt'
        ]
 ],
```

Connection details for redis to use for memory caching in a single server configuration.

For enhanced security it is recommended to configure Redis to require a password. See http://redis.io/topics/security for more information.

We also support redis SSL/TLS encryption as of version 6. See https://redis.io/topics/encryption for more information.

```
 'redis.cluster' => [
        'seeds' => [ // provide some/all of the cluster servers to bootstrap discovery, port
                'localhost:7000',
                'localhost:7001',
        ],
        'timeout' => 0.0,
        'read_timeout' => 0.0,
        'failover_mode' => \RedisCluster::FAILOVER_ERROR,
        'user' =>  '', // Optional, if not defined no password will be used.
        'password' => '', // Optional, if not defined no password will be used.
        // If redis in-transit encryption is enabled, provide certificates
        // SSL context https://www.php.net/manual/en/context.ssl.php
        'ssl_context' => [
                'local_cert' => '/certs/redis.crt',
                'local_pk' => '/certs/redis.key',
                'cafile' => '/certs/ca.crt'
        ]
 ],
```

Connection details for a Redis Cluster

Only for use with Redis Clustering, for Sentinel-based setups use the single server configuration above, and perform HA on the hostname.

Redis Cluster support requires the php module phpredis in version 3.0.0 or higher.

Available failover modes:

- \RedisCluster::FAILOVER_NONE - only send commands to master nodes (default)
- \RedisCluster::FAILOVER_ERROR - failover to slaves for read commands if master is unavailable (recommended)
- \RedisCluster::FAILOVER_DISTRIBUTE - randomly distribute read commands across master and slaves

WARNING: FAILOVER_DISTRIBUTE is a not recommended setting and we strongly suggest to not use it if you use Redis for file locking. Due to the way Redis is synchronized it could happen, that the read for an existing lock is scheduled to a slave that is not fully synchronized with the connected master which then causes a FileLocked exception.

See https://redis.io/topics/cluster-spec for details about the Redis cluster

Authentication works with phpredis version 4.2.1+. See https://github.com/phpredis/phpredis/commit/c5994f2a42b8a348af92d3acb4edff1328ad8ce1

```
'memcached_servers' => [
        // hostname, port and optional weight. Also see:
        // https://www.php.net/manual/en/memcached.addservers.php
        // https://www.php.net/manual/en/memcached.addserver.php
        ['localhost', 11211],
        //array('other.host.local', 11211),
],
```

Server details for one or more memcached servers to use for memory caching.

```
'memcached_options' => [
        // Set timeouts to 50ms
        \Memcached::OPT_CONNECT_TIMEOUT => 50,
        \Memcached::OPT_RETRY_TIMEOUT =>   50,
        \Memcached::OPT_SEND_TIMEOUT =>    50,
        \Memcached::OPT_RECV_TIMEOUT =>    50,
        \Memcached::OPT_POLL_TIMEOUT =>    50,

        // Enable compression
        \Memcached::OPT_COMPRESSION =>          true,

        // Turn on consistent hashing
        \Memcached::OPT_LIBKETAMA_COMPATIBLE => true,

        // Enable Binary Protocol
        \Memcached::OPT_BINARY_PROTOCOL =>      true,

        // Binary serializer vill be enabled if the igbinary PECL module is available
        //\Memcached::OPT_SERIALIZER => \Memcached::SERIALIZER_IGBINARY,
],
```

Connection options for memcached

```
'cache_path' => '',
```

Location of the cache folder, defaults to `data/$user/cache` where `$user` is the current user. When specified, the format will change to `$cache_path/$user` where `$cache_path` is the configured cache directory and `$user` is the user.

Defaults to `''` (empty string)

```
'cache_chunk_gc_ttl' => 60*60*24,
```

TTL of chunks located in the cache folder before they’re removed by garbage collection (in seconds). Increase this value if users have issues uploading very large files via the Nextcloud Client as upload isn’t completed within one day.

Defaults to `60*60*24` (1 day)

Using Object Store with Nextcloud

```
'objectstore' => [
        'class' => 'OC\\Files\\ObjectStore\\Swift',
        'arguments' => [
                // trystack will use your facebook id as the user name
                'username' => 'facebook100000123456789',
                // in the trystack dashboard go to user -> settings -> API Password to
                // generate a password
                'password' => 'Secr3tPaSSWoRdt7',
                // must already exist in the objectstore, name can be different
                'container' => 'nextcloud',
                // prefix to prepend to the fileid, default is 'oid:urn:'
                'objectPrefix' => 'oid:urn:',
                // create the container if it does not exist. default is false
                'autocreate' => true,
                // required, dev-/trystack defaults to 'RegionOne'
                'region' => 'RegionOne',
                // The Identity / Keystone endpoint
                'url' => 'http://8.21.28.222:5000/v2.0',
                // required on dev-/trystack
                'tenantName' => 'facebook100000123456789',
                // dev-/trystack uses swift by default, the lib defaults to 'cloudFiles'
                // if omitted
                'serviceName' => 'swift',
                // The Interface / url Type, optional
                'urlType' => 'internal'
        ],
],
```

This example shows how to configure Nextcloud to store all files in a swift object storage.

It is important to note that Nextcloud in object store mode will expect exclusive access to the object store container because it only stores the binary data for each file. The metadata is currently kept in the local database for performance reasons.

WARNING: The current implementation is incompatible with any app that uses direct file IO and circumvents our virtual filesystem. That includes Encryption and Gallery. Gallery will store thumbnails directly in the filesystem and encryption will cause severe overhead because key files need to be fetched in addition to any requested file.

One way to test is applying for a trystack account at http://trystack.org/

```
'objectstore' => [
        'class' => 'OC\\Files\\ObjectStore\\Swift',
        'arguments' => [
                'autocreate' => true,
                'user' => [
                        'name' => 'swift',
                        'password' => 'swift',
                        'domain' => [
                                'name' => 'default',
                        ],
                ],
                'scope' => [
                        'project' => [
                                'name' => 'service',
                                'domain' => [
                                        'name' => 'default',
                                ],
                        ],
                ],
                'tenantName' => 'service',
                'serviceName' => 'swift',
```

```
                'region' => 'regionOne',
                'url' => 'http://yourswifthost:5000/v3',
                'bucket' => 'nextcloud',
        ],
],
```

To use swift V3

```
'objectstore.multibucket.preview-distribution' => false,
```

If this is set to true and a multibucket object store is configured then newly created previews are put into 256 dedicated buckets.

Those buckets are named like the mulibucket version but with the postfix `-preview-NUMBER` where NUMBER is between 0 and 255.

Keep in mind that only previews of files are put in there that don't have some already. Otherwise the old bucket will be used.

To migrate existing previews to this new multibucket distribution of previews use the occ command `preview:repair`. For now this will only migrate previews that were generated before Nextcloud 19 in the flat `appdata_INSTANCEID/previews/FILEID` folder structure.

Sharing

Global settings for Sharing

```
'sharing.managerFactory' => '\OC\Share20\ProviderFactory',
```

Replaces the default Share Provider Factory. This can be utilized if own or 3rdParty Share Providers are used that – for instance – use the filesystem instead of the database to keep the share information.

Defaults to `\OC\Share20\ProviderFactory`

```
'sharing.maxAutocompleteResults' => 25,
```

Define max number of results returned by the search for auto-completion of users, groups, etc. The value must not be lower than 0 (for unlimited).

If more, different sources are requested (e.g. different user backends; or both users and groups), the value is applied per source and might not be truncated after collecting the results. I.e. more results can appear than configured here.

Default is 25.

```
'sharing.minSearchStringLength' => 0,
```

Define the minimum length of the search string before we start auto-completion Default is no limit (value set to 0)

```
'sharing.enable_share_accept' => false,
```

Set to true to enable that internal shares need to be accepted by the users by default.

Users can change this for their account in their personal sharing settings

```
'sharing.force_share_accept' => false,
```

Set to true to enforce that internal shares need to be accepted

```
'sharing.allow_custom_share_folder' => true,
```

Set to false to prevent users from setting a custom share_folder

```
'sharing.enable_share_mail' => true,
```

Set to false to stop sending a mail when users receive a share

```
'sharing.allow_disabled_password_enforcement_groups' => false,
```

Set to true to enable the feature to add exceptions for share password enforcement

```
'transferIncomingShares' => false,
```

Set to true to always transfer incoming shares by default when running “occ files:transfer-ownership”.

Defaults to false, so incoming shares are not transferred if not specifically requested by a command line argument.

All other configuration options

```
'dbdriveroptions' => [
        PDO::MYSQL_ATTR_SSL_CA => '/file/path/to/ca_cert.pem',
        PDO::MYSQL_ATTR_SSL_KEY => '/file/path/to/mysql-client-key.pem',
        PDO::MYSQL_ATTR_SSL_CERT => '/file/path/to/mysql-client-cert.pem',
        PDO::MYSQL_ATTR_SSL_VERIFY_SERVER_CERT => false,
        PDO::MYSQL_ATTR_INIT_COMMAND => 'SET wait_timeout = 28800'
],
```

Additional driver options for the database connection, eg. to enable SSL encryption in MySQL or specify a custom wait timeout on a cheap hoster.

When setting up TLS/SSL for encrypting the connections, you need to ensure that the passed keys and certificates are readable by the PHP process. In addition PDO::MYSQL_ATTR_SSL_VERIFY_SERVER_CERT might need to be set to false, if the database servers certificates CN does not match with the hostname used to connect. The standard behavior here is different from the MySQL/MariaDB CLI client, which does not verify the server cert except –ssl-verify-server-cert is passed manually.

```
'sqlite.journal_mode' => 'DELETE',
```

sqlite3 journal mode can be specified using this configuration parameter - can be ‘WAL’ or ‘DELETE’ see for more details https://www.sqlite.org/wal.html

```
'mysql.utf8mb4' => false,
```

During setup, if requirements are met (see below), this setting is set to true and MySQL can handle 4 byte characters instead of 3 byte characters.

If you want to convert an existing 3-byte setup into a 4-byte setup please set the parameters in MySQL as mentioned below and run the migration command: ./occ db:convert-mysql-charset The config setting will be set automatically after a successful run.

Consult the documentation for more details.

MySQL requires a special setup for longer indexes (> 767 bytes) which are needed:

[mysqld] innodb_large_prefix=ON innodb_file_format=Barracuda innodb_file_per_table=ON

Tables will be created with

- character set: utf8mb4
- collation: utf8mb4_bin
- row_format: compressed

See: https://dev.mysql.com/doc/refman/5.7/en/charset-unicode-utf8mb4.html
https://dev.mysql.com/doc/refman/5.7/en/innodb-parameters.html#sysvar_innodb_large_prefix
https://mariadb.com/kb/en/mariadb/xtradbinnodb-server-system-variables/#innodb_large_prefix
http://www.tocker.ca/2013/10/31/benchmarking-innodb-page-compression-performance.html
http://mechanics.flite.com/blog/2014/07/29/using-innodb-large-prefix-to-avoid-error-1071/

```
'mysql.collation' => null,
```

For search queries in the database, a default collation – depending on the character set – is chosen. In some cases a different behaviour is desired, for instances when a accent sensitive search is desired.

MariaDB and MySQL have an overlap in available collations, but also incompatible ones, also depending on the version of the database server.

This option allows to override the automatic choice. Example:

‘mysql.collation’ => ‘utf8mb4_0900_as_ci’,

This setting has no effect on setup or creating tables. In those cases always utf8[mb4]_bin is being used. This setting is only taken into consideration in SQL queries that utilize LIKE comparison operators.

```
'supportedDatabases' => [
        'sqlite',
        'mysql',
        'pgsql',
        'oci',
],
```

Database types that are supported for installation.

Available:

- sqlite (SQLite3)
- mysql (MySQL)
- pgsql (PostgreSQL)
- oci (Oracle)

Defaults to the following databases:

- sqlite (SQLite3)
- mysql (MySQL)
- pgsql (PostgreSQL)

```
'tempdirectory' => '/tmp/nextcloudtemp',
```

Override where Nextcloud stores temporary files. Useful in situations where the system temporary directory is on a limited space ramdisk or is otherwise restricted, or if external storage which do not support streaming are in use.

The Web server user must have write access to this directory.

Hashing

```
'hashing_default_password' => false,
```

By default Nextcloud will use the Argon2 password hashing if available.

However if for whatever reason you want to stick with the PASSWORD_DEFAULT of your php version. Then set the setting to true.

Nextcloud uses the Argon2 algorithm (with PHP >= 7.2) to create hashes by its own and exposes its configuration options as following. More information can be found at: https://www.php.net/manual/en/function.password-hash.php

---

```
'hashingThreads' => PASSWORD_ARGON2_DEFAULT_THREADS,
```

The number of CPU threads to be used by the algorithm for computing a hash.

The value must be an integer, and the minimum value is 1. Rationally it does not help to provide a number higher than the available threads on the machine. Values that undershoot the minimum will be ignored in favor of the minimum.

```
'hashingMemoryCost' => PASSWORD_ARGON2_DEFAULT_MEMORY_COST,
```

The memory in KiB to be used by the algorithm for computing a hash. The value must be an integer, and the minimum value is 8 times the number of CPU threads.

Values that undershoot the minimum will be ignored in favor of the minimum.

```
'hashingTimeCost' => PASSWORD_ARGON2_DEFAULT_TIME_COST,
```

The number of iterations that are used by the algorithm for computing a hash.

The value must be an integer, and the minimum value is 1. Values that undershoot the minimum will be ignored in favor of the minimum.

```
'hashingCost' => 10,
```

The hashing cost used by hashes generated by Nextcloud Using a higher value requires more time and CPU power to calculate the hashes

```
'blacklisted_files' => ['.htaccess'],
```

Blacklist a specific file or files and disallow the upload of files with this name. `.htaccess` is blocked by default.

WARNING: USE THIS ONLY IF YOU KNOW WHAT YOU ARE DOING.

Defaults to `array('.htaccess')`

```
'share_folder' => '/',
```

Define a default folder for shared files and folders other than root.

Changes to this value will only have effect on new shares.

Defaults to `/`

```
'theme' => '',
```

If you are applying a theme to Nextcloud, enter the name of the theme here.

The default location for themes is `nextcloud/themes/`.

Defaults to the theming app which is shipped since Nextcloud 9

```
'cipher' => 'AES-256-CTR',
```

The default cipher for encrypting files. Currently supported are:

- AES-256-CTR
- AES-128-CTR
- AES-256-CFB
- AES-128-CFB

Defaults to `AES-256-CTR`

```
'minimum.supported.desktop.version' => '2.0.0',
```

The minimum Nextcloud desktop client version that will be allowed to sync with this server instance. All connections made from earlier clients will be denied by the server. Defaults to the minimum officially supported Nextcloud desktop clientversion at the time of release of this server version.

When changing this, note that older unsupported versions of the Nextcloud desktop client may not function as expected, and could lead to permanent data loss for clients or other unexpected results.

Defaults to `2.0.0`

```
'localstorage.allowsymlinks' => false,
```

Option to allow local storage to contain symlinks.

WARNING: Not recommended. This would make it possible for Nextcloud to access files outside the data directory and could be considered a security risk.

Defaults to `false`

```
'quota_include_external_storage' => false,
```

EXPERIMENTAL: option whether to include external storage in quota calculation, defaults to false.

Defaults to `false`

```
'external_storage.auth_availability_delay' => 1800,
```

When an external storage is unavailable for some reasons, it will be flagged as such for 10 minutes. When the trigger is a failed authentication attempt the delay is higher and can be controlled with this option. The motivation is to make account lock outs at Active Directories (and compatible) more unlikely.

Defaults to `1800` (seconds)

```
'files_external_allow_create_new_local' => true,
```

Allows to create external storages of type “Local” in the web interface and APIs.

When disable, it is still possible to create local storages with occ using the following command:

% php occ files_external:create /mountpoint local null::null -c datadir=/path/to/data

Defaults to `true`

```
'filesystem_check_changes' => 0,
```

Specifies how often the local filesystem (the Nextcloud data/ directory, and NFS mounts in data/) is checked for changes made outside Nextcloud. This does not apply to external storage.

0 -> Never check the filesystem for outside changes, provides a performance increase when it’s certain that no changes are made directly to the filesystem

1 -> Check each file or folder at most once per request, recommended for general use if outside changes might happen.

Defaults to `0`

```
'part_file_in_storage' => true,
```

By default Nextcloud will store the part files created during upload in the same storage as the upload target. Setting this to false will store the part files in the root of the users folder which might be required to work with certain external storage setups that have limited rename capabilities.

Defaults to `true`

```
'mount_file' => '/var/www/nextcloud/data/mount.json',
```

Where `mount.json` file should be stored, defaults to `data/mount.json` in the Nextcloud directory.

Defaults to `data/mount.json` in the Nextcloud directory.

```
'filesystem_cache_readonly' => false,
```

When `true`, prevent Nextcloud from changing the cache due to changes in the filesystem for all storage.

Defaults to `false`

```
'secret' => '',
```

Secret used by Nextcloud for various purposes, e.g. to encrypt data. If you lose this string there will be data corruption.

```
'trusted_proxies' => ['203.0.113.45', '198.51.100.128', '192.168.2.0/24'],
```

List of trusted proxy servers

You may set this to an array containing a combination of - IPv4 addresses, e.g. *192.168.2.123* - IPv4 ranges in CIDR notation, e.g. *192.168.2.0/24* - IPv6 addresses, e.g. *fd9e:21a7:a92c:2323::1*

(CIDR notation for IPv6 is currently work in progress and thus not available as of yet)

When an incoming request’s *REMOTE_ADDR* matches any of the IP addresses specified here, it is assumed to be a proxy instead of a client. Thus, the client IP will be read from the HTTP header specified in *forwarded_for_headers* instead of from *REMOTE_ADDR*.

So if you configure *trusted_proxies*, also consider setting *forwarded_for_headers* which otherwise defaults to *HTTP_X_FORWARDED_FOR* (the *X-Forwarded-For* header).

Defaults to an empty array.

```
'forwarded_for_headers' => ['HTTP_X_FORWARDED', 'HTTP_FORWARDED_FOR'],
```

Headers that should be trusted as client IP address in combination with *trusted_proxies*. If the HTTP header looks like ‘X-Forwarded-For’, then use ‘HTTP_X_FORWARDED_FOR’ here.

If set incorrectly, a client can spoof their IP address as visible to Nextcloud, bypassing access controls and making logs useless!

Defaults to `'HTTP_X_FORWARDED_FOR'`

```
'max_filesize_animated_gifs_public_sharing' => 10,
```

max file size for animating gifs on public-sharing-site.

If the gif is bigger, it’ll show a static preview

Value represents the maximum filesize in megabytes. Set to `-1` for no limit.

Defaults to `10` megabytes

```
'filelocking.enabled' => true,
```

Enables transactional file locking.

This is enabled by default.

Prevents concurrent processes from accessing the same files at the same time. Can help prevent side effects that would be caused by concurrent operations. Mainly relevant for very large installations with many users working with shared files.

Defaults to `true`

```
'filelocking.ttl' => 60*60,
```

Set the lock’s time-to-live in seconds.

Any lock older than this will be automatically cleaned up.

Defaults to `60*60` seconds (1 hour) or the php

max_execution_time, whichever is higher.

```
'memcache.locking' => '\\OC\\Memcache\\Redis',
```

Memory caching backend for file locking

Because most memcache backends can clean values without warning using redis is highly recommended to *avoid data loss*.

Defaults to `none`

```
'filelocking.debug' => false,
```

Enable locking debug logging

Note that this can lead to a very large volume of log items being written which can lead to performance degradation and large log files on busy instance.

Thus enabling this in production for longer periods of time is not recommended or should be used together with the `log.condition` setting.

```
'upgrade.disable-web' => false,
```

Disable the web based updater

```
'debug' => false,
```

Set this Nextcloud instance to debugging mode

Only enable this for local development and not in production environments This will disable the minifier and outputs some additional debug information

Defaults to `false`

```
'data-fingerprint' => '',
```

Sets the data-fingerprint of the current data served

This is a property used by the clients to find out if a backup has been restored on the server. Once a backup is restored run ./occ maintenance:data-fingerprint To set this to a new value.

Updating/Deleting this value can make connected clients stall until the user has resolved conflicts.

Defaults to `''` (empty string)

```
'copied_sample_config' => true,
```

This entry is just here to show a warning in case somebody copied the sample configuration. DO NOT ADD THIS SWITCH TO YOUR CONFIGURATION!

If you, brave person, have read until here be aware that you should not modify *ANY* settings in this file without reading the documentation.

```
'lookup_server' => 'https://lookup.nextcloud.com',
```

use a custom lookup server to publish user data

```
'gs.enabled' => false,
```

set to true if the server is used in a setup based on Nextcloud's Global Scale architecture

```
'gs.federation' => 'internal',
```

by default federation is only used internally in a Global Scale setup If you want to allow federation outside of your environment set it to 'global'

```
'csrf.optout' => [
        '/^WebDAVFS/', // OS X Finder
        '/^Microsoft-WebDAV-MiniRedir/', // Windows webdav drive
],
```

List of incompatible user agents opted out from Same Site Cookie Protection.

Some user agents are notorious and don't really properly follow HTTP specifications. For those, have an opt-out.

WARNING: only use this if you know what you are doing

```
'simpleSignUpLink.shown' => true,
```

By default there is on public pages a link shown that allows users to learn about the "simple sign up" - see https://nextcloud.com/signup/

If this is set to "false" it will not show the link.

```
'login_form_autocomplete' => true,
```

By default autocompletion is enabled for the login form on Nextcloud's login page.

While this is enabled, browsers are allowed to "remember" login names and such. Some companies require it to be disabled to comply with their security policy.

Simply set this property to "false", if you want to turn this feature off.

```
'files_no_background_scan' => false,
```

Disable background scanning of files

By default, a background job runs every 10 minutes and execute a background scan to sync filesystem and database. Only users with unscanned files (size < 0 in filecache) are included. Maximum 500 users per job.

Defaults to `true`

```
'query_log_file' => '',
```

Log all queries into a file

Warning: This heavily decreases the performance of the server and is only meant to debug/profile the query interaction manually. Also, it might log sensitive data into a plain text file.

```
'redis_log_file' => '',
```

Log all redis requests into a file

Warning: This heavily decreases the performance of the server and is only meant to debug/profile the redis interaction manually. Also, it might log sensitive data into a plain text file.

```
'ldap_log_file' => '',
```

Log all LDAP requests into a file

Warning: This heavily decreases the performance of the server and is only meant to debug/profile the LDAP interaction manually. Also, it might log sensitive data into a plain text file.

```
'profile.enabled' => true,
```

Enable profile globally

Defaults to `true`

App config options

Activity app

Retention for activities of the activity app:

```
'activity_expire_days' => 365,
```

Every day a cron job is ran, which deletes all activities for all users which are older then the number of days that is set for `activity_expire_days`

```
'activity_use_cached_mountpoints' => false,
```

Before enabling this, read the warning in Activities in groupfolders or external storages

Settings app

If an email address of a user is changed by an admin, then it triggers an email to the user that states “Your email address on URL was changed by an administrator.”. In some cases this should not be triggered, because it was a normal maintenance change. To disable this specific email the appconfig option `disable_email.email_address_changed_by_admin` can be set to `yes`:

```
occ config:app:set settings disable_activity.email_address_changed_by_admin --value yes
```

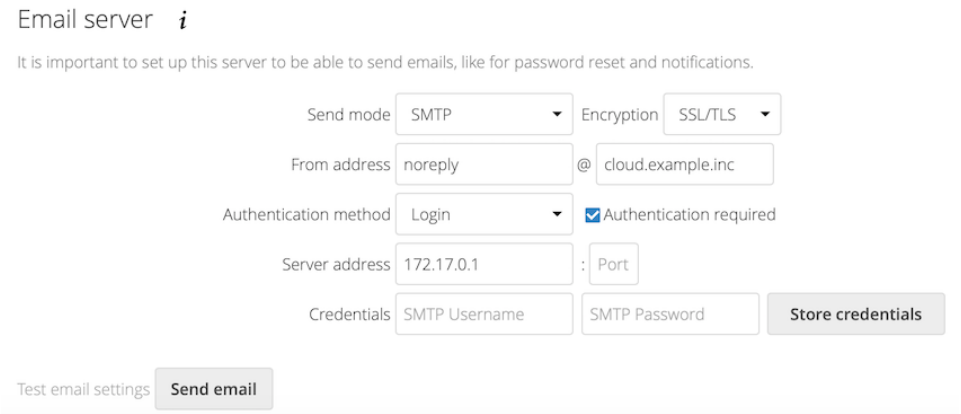
To disable this behaviour change it to any other value or delete the app config:

```
occ config:app:delete settings disable_activity.email_address_changed_by_admin
```

Email

Nextcloud is capable of sending password reset emails, notifying users of new file shares, changes in files, and activity notifications. Your users configure which notifications they want to receive on their Personal pages.

Nextcloud does not contain a full email server, but rather connects to your existing mail server. You must have a functioning mail server for Nextcloud to be able to send emails. You may have a mail server on the same machine as Nextcloud, or it may be a remote server.



With the wizard, connecting Nextcloud to your mail server is fast and easy. The wizard fills in the values in `config/config.php`, so you may use either or both as you prefer.

The Nextcloud Email wizard supports three types of mail server connections: SMTP, qmail, and Sendmail. Use the SMTP configurator for a remote server or Sendmail when your mail server is on the same machine as Nextcloud.

Note

The Sendmail option refers to the Sendmail SMTP server and any drop-in Sendmail replacement such as Postfix, Exim, or Courier. All of these include a `sendmail` binary, and are freely-interchangeable.

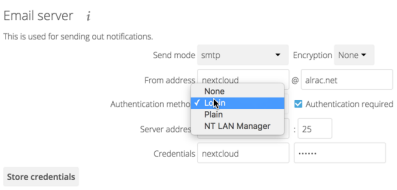
Configuring an SMTP server

You need the following information from your mail server administrator to connect Nextcloud to a remote SMTP server:

- Encryption type: None, SSL/TLS, or STARTTLS
- The From address you want your outgoing Nextcloud mails to use
- Whether authentication is required
- Authentication method: None, Login, Plain, or NT LAN Manager
- The server's IP address or fully-qualified domain name and the SMTP port
- Login credentials (if required)

Note

The `overwrite.cli.url` parameter from `config.php` will be used for the SMTP EHLO.



Your changes are saved immediately, and you can click the Send Email button to test your configuration. This sends a test message to the email address you configured on your Personal page. The test message says:

```
If you received this email, the settings seem to be correct.

--
Nextcloud
a safe home for all your data
```

Configuring Sendmail/qmail

Configuring Sendmail or qmail requires only that you select one of them instead of SMTP, and then enter your desired return email address.

In most cases the SMTP option is best, since you will be able to control all of your mail server options in one place, in your mail server configuration then.

Using email templates

We designed a mechanism that generates emails which follow the theming settings and look the same in all the different email clients out there.

Note

If, for some reason, you need text-only emails, consider simply configuring this on the client side or let the receiving (or even sending) mail server drop the HTML part. Note that there is no security impact from **sending** HTML emails, just from displaying them and thus any security risk can only be mitigated by disabling showing HTML on the client (or removing the HTML part in the mail server).

Modifying the look of emails beyond the theming app capabilities

You can overwrite templates by writing a class that implements the template interface (or extends it to not need to copy over everything). Easiest way is to then put this class into an app and load it so you do not need to patch it on every update.

This is the interface of the class that needs to be implemented: https://github.com/nextcloud/server/blob/master/lib/public/Mail/IEMailTemplate.php

That is the implementation that could be extended and used to see how it works: https://github.com/nextcloud/server/blob/master/lib/private/Mail/EMailTemplate.php

An example from a GitHub issue:

1. Look at the source code of extended class OC\Mail\EMailTemplate::class
2. Then override what you need in your own *OC\Mail\EMailTemplate::class* extension

Example:

Let's assume that we need to override the email header:

```
<?php

namespace \OCA\MyApp;

use OC\Mail\EMailTemplate;

class MyClass extends EMailTemplate
{
   protected $header = <<<EOF
      <table align="center" class="wrapper">
            // your theme email header modification
      </table>
   EOF;
}
```

3. Then in `config/config.php` change `mail_template_class` to your class namespace:

```
    'mail_template_class' => 'OCA\\MyApp\\MyClass',
```

You will find a detailed step by step guide in our support portal.

Setting mail server parameters in config.php

If you prefer, you may set your mail server parameters in `config/config.php`. The following examples are for SMTP, Sendmail, and Qmail.

SMTP

If you want to send email using a local or remote SMTP server it is necessary to enter the name or IP address of the server, optionally followed by a colon separated port number, e.g. **:425**. If this value is not given the default port 25/tcp will be used unless you change that by modifying the **mail_smtpport** parameter.

```
"mail_smtpmode"     => "smtp",
"mail_smtphost"     => "smtp.server.dom:425",
```

or

```
"mail_smtpmode"     => "smtp",
"mail_smtphost"     => "smtp.server.dom",
"mail_smtpport"     => 425,
```

If a malware or SPAM scanner is running on the SMTP server it might be necessary that you increase the SMTP timeout to e.g. 30s:

```
"mail_smtptimeout"  => 30,
```

If the SMTP server accepts insecure connections, the default setting can be used:

```
"mail_smtpsecure"   => '',
```

If the SMTP server only accepts secure connections you can choose between the following two variants:

SSL

A secure connection will be initiated using the outdated SMTPS protocol which uses the port 465/tcp:

```
"mail_smtphost"     => "smtp.server.dom:465",
"mail_smtpsecure"   => 'ssl',
```

TLS

A secure connection will be initiated using the STARTTLS protocol which uses the default port 25/tcp:

```
"mail_smtphost"     => "smtp.server.dom",
"mail_smtpsecure"   => 'tls',
```

And finally it is necessary to configure if the SMTP server requires authentication, if not, the default values can be taken as is.

```
"mail_smtpauth"     => false,
"mail_smtpname"     => "",
"mail_smtppassword" => "",
```

If SMTP authentication is required you have to set the required username and password and can optionally choose between the authentication types **LOGIN** (default) or **PLAIN**.

```
"mail_smtpauth"     => true,
"mail_smtpauthtype" => "LOGIN",
"mail_smtpname"     => "username",
"mail_smtppassword" => "password",
```

Advanced users can add additional stream options in `config/config.php`, which maps directly to Swift Mailer's `streamOptions` configuration parameter:

```
"mail_smtpstreamoptions" => array(),
```

Sendmail

If you want to use the well known Sendmail program to send email, it is necessary to have an installed and working email system on your *nix server. The sendmail binary (**/usr/sbin/sendmail**) is usually part of that system. Nextcloud should be able to send email out of the box.

```
"mail_smtpmode"     => "sendmail",
"mail_smtphost"     => "127.0.0.1",
"mail_smtpport"     => 25,
"mail_smtptimeout"  => 10,
"mail_smtpsecure"   => "",
"mail_smtpauth"     => false,
"mail_smtpauthtype" => "LOGIN",
"mail_smtpname"     => "",
"mail_smtppassword" => "",
```

qmail

If you want to use the qmail program to send email, it is necessary to have an installed and working qmail email system on your server. The qmail binary installed on your server will then be used to send email. Nextcloud should be able to send email out of the box.

```
"mail_smtpmode"     => "qmail",
"mail_smtphost"     => "127.0.0.1",
"mail_smtpport"     => 25,
"mail_smtptimeout"  => 10,
"mail_smtpsecure"   => "",
"mail_smtpauth"     => false,
"mail_smtpauthtype" => "LOGIN",
"mail_smtpname"     => "",
"mail_smtppassword" => "",
```

Send a test email

To test your email configuration, save your email address in your personal settings and then use the **Send email** button in the *Email Server* section of the Admin settings page.

Troubleshooting

If you are unable to send email, try turning on debugging. Do this by enabling the `mail_smtpdebug` parameter in `config/config.php`.

```
"mail_smtpdebug" => true;
```

Note

Immediately after pressing the **Send email** button, as described before, several **SMTP -> get_lines(): …** messages appear on the screen. This is expected behavior and can be ignored.

Question: Why is my web domain different from my mail domain?

Answer: The default domain name used for the sender address is the hostname where your Nextcloud installation is served. If you have a different mail domain name you can override this behavior by setting the following configuration parameter:

```
"mail_domain" => "example.com",
```

This setting results in every email sent by Nextcloud (for example, the password reset email) having the domain part of the sender address appear as follows:

```
no-reply@example.com
```

Question: How can I find out if an SMTP server is reachable?

Answer: Use the ping command to check the server availability:

```
ping smtp.server.dom

PING smtp.server.dom (ip-address) 56(84) bytes of data.
64 bytes from your-server.local.lan (192.168.1.10): icmp_req=1 ttl=64
time=3.64ms
```

Question: How can I find out if the SMTP server is listening on a specific TCP port?

Answer: The best way to get mail server information is to ask your mail server admin. If you are the mail server admin, or need information in a hurry, you can use the `netstat` command. This example shows all active servers on your system, and the ports they are listening on. The SMTP server is listening on localhost port 25.

```
# netstat -pant
```

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address   Foreign Address  State  ID/Program name
tcp    0      0    0.0.0.0:631     0.0.0.0:*        LISTEN   4418/cupsd
tcp    0      0    127.0.0.1:25    0.0.0.0:*        LISTEN   2245/exim4
tcp    0      0    127.0.0.1:3306  0.0.0.0:*        LISTEN   1524/mysqld
```

- 25/tcp is unencrypted smtp
- 110/tcp/udp is unencrypted pop3
- 143/tcp/udp is unencrypted imap4
- 465/tcp is encrypted ssmtp
- 993/tcp/udp is encrypted imaps
- 995/tcp/udp is encrypted pop3s

Question: How can I determine if the SMTP server supports the outdated SMTPS protocol?

Answer: A good indication that the SMTP server supports the SMTPS protocol is that it is listening on port **465**.

Question: How can I determine what authorization and encryption protocols the mail server supports?

Answer: SMTP servers usually announce the availability of STARTTLS immediately after a connection has been established. You can easily check this using the `telnet` command.

Note

You must enter the marked lines to obtain the information displayed.

```
telnet smtp.domain.dom 25

Trying 192.168.1.10...
Connected to smtp.domain.dom.
Escape character is '^]'.
220 smtp.domain.dom ESMTP Exim 4.80.1 Tue, 22 Jan 2013 22:39:55 +0100
EHLO your-server.local.lan                  # <<< enter this command
250-smtp.domain.dom Hello your-server.local.lan [ip-address]
250-SIZE 52428800
250-8BITMIME
250-PIPELINING
250-AUTH PLAIN LOGIN CRAM-MD5               # <<< Supported auth protocols
250-STARTTLS                                # <<< Encryption is supported
250 HELP
QUIT                                        # <<< enter this command
221 smtp.domain.dom closing connection
Connection closed by foreign host.
```

Question: How can I send mail when using self-signed certificates if remote SMTP server do not have options to allow this on their side?

Answer: If you are having remote SMTP setup, you can try adding this to your `config/config.php`:

```
"mail_smtpstreamoptions" => array(
    'ssl' => array(
        'allow_self_signed' => true,
        'verify_peer' => false,
        'verify_peer_name' => false
    )
),
```

Enabling debug mode

If you are unable to send email, it might be useful to activate further debug messages by enabling the `mail_smtpdebug` parameter:

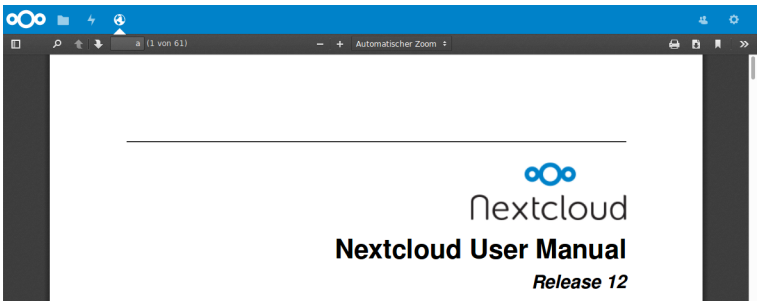
```
"mail_smtpdebug" => true,
```

Note

Immediately after pressing the **Send email** button, as described before, several **SMTP -> get_lines(): ...** messages appear on the screen. This is expected behavior and can be ignored.

Linking external sites

You can embed external websites or documents inside your Nextcloud pages with the **External sites** app, as this screenshot shows.



Click to enlarge

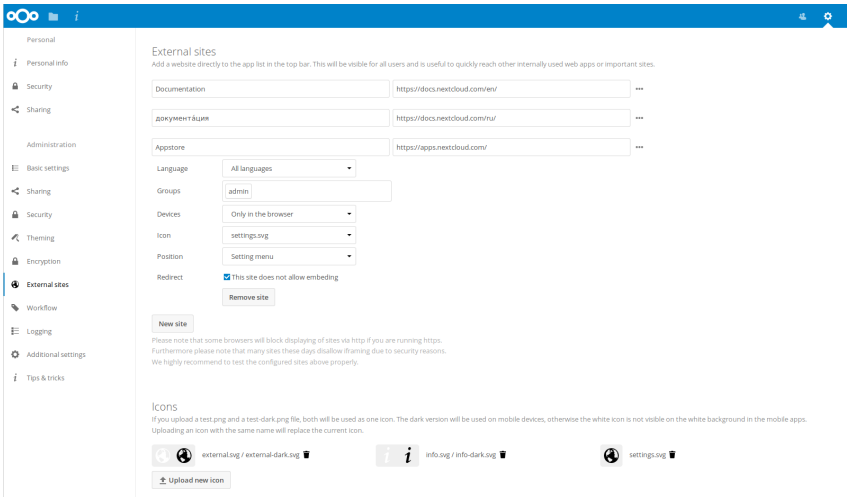
This is useful for quick access to important pages such as the Nextcloud manuals and informational pages for your company, and for presenting external pages inside your custom Nextcloud branding, if you use your own custom themes.

The External sites app can be easily installed from the app store. Go to **Settings > Apps > Customization** to enable it. Then go to your Nextcloud **Settings > Administration > External sites** to create your links, which are saved automatically.

Each link can have a unique icon, which can be uploaded in the admin settings. If you select a language, the link will only be displayed for users with the selected language. This allows you to have different documentation links for users depending on their language.

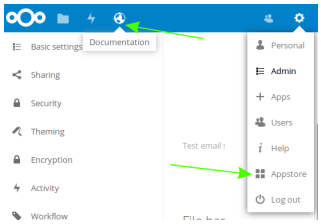
It is also possible to add links for a special device (recognized by the user agent). Currently the following options are available: All devices, Android app, iOS app, Desktop client and all others (Browsers).

It is also possible to add links only for members of a given group.



Click to enlarge

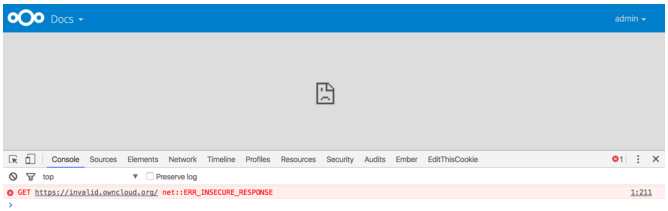
The links appear in the Nextcloud menu on the top or in the settings menu, after reloading the page.



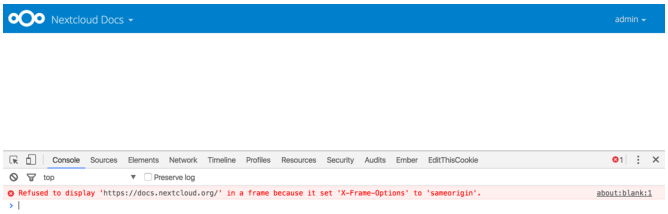
Configurations preventing embedding

Your links may or may not work correctly due to the various ways that Web browsers and Web sites handle HTTP and HTTPS URLs, and because the External Sites app embeds external links in IFrames. Modern Web browsers try very hard to protect Web surfers from dangerous links, and safety apps like Privacy Badger and ad-blockers may block embedded pages. It is strongly recommended to enforce HTTPS on your Nextcloud server; do not weaken this, or any of your security tools, just to make embedded Web pages work. After all, you can freely access them outside of Nextcloud.

Most Web sites that offer login functionalities use the `X-Frame-Options` or `Content-Security-Policy` HTTP header which instructs browsers to not allow their pages to be embedded for security reasons (e.g. “Clickjacking”). You can usually verify the reason why embedding the website is not possible by using your browser’s console tool. For example, this page has an invalid SSL certificate.



On this page, X-Frame-Options prevents the embedding.



There is also a redirect option, which allows that those websites can still be added for quick access. Instead of embedding the website the user will be redirected to it.

Language & Locale

Default language

In normal cases Nextcloud will automatically detect the language of the Web-GUI. If this does not work properly or you want to make sure that Nextcloud always starts with a given language, you can set a **default_language** parameter in the config/config.php.

Note

The default_language parameter is only used, when the browser does not send any language, and the user hasn’t configured own language preferences.

```
"default_language" => "en",
```

Force language

If you want to force a specific language, users will no longer be able to change their language in the personal settings. You can set a **force_language** parameter in the config/config.php.

```
"force_language" => "en",
```

If users shall be unable to change their language, but users have different languages, this value can be set to `true` instead of a language code.

> **Note**
>
> Please check Transifex language codes for the list of valid language codes.

Default locale

The locale is used to define how dates and other formats are displayed. Nextcloud should automatically pick an appropriate locale based on your current language. Users can modify their locale inside their settings panel. If that does not work properly or if you want to make sure that Nextcloud always starts with a given locale, you can set a **default_locale** parameter in the config/config.php.

> **Note**
>
> The default_locale parameter is only used when the user hasn't configured own locale preferences.

```
"default_locale" => "en_US",
```

Force locale

If you want to force a specific locale, users will no longer be able to change their locale in the personal settings. You can set a **force_locale** parameter in the config/config.php.

```
"force_locale" => "en_US",
```

> **Note**
>
> Please check the list of MomentJS supported locales for the list of valid locales.

Logging

Use your Nextcloud log to review system status, or to help debug problems. You may adjust logging levels, and choose between using the Nextcloud log or your syslog. If additional audit information is required, you can optionally activate the **admin_audit** app, which by default generates a separate audit.log file in the data directory.

Log level

Logging levels range from **DEBUG**, which logs all activity, to **FATAL**, which logs only fatal errors.

- **0**: DEBUG: All activity; the most detailed logging.
- **1**: INFO: Activity such as user logins and file activities, plus warnings, errors, and fatal errors.
- **2**: WARN: Operations succeed, but with warnings of potential problems, plus errors and fatal errors.

- **3**: ERROR: An operation fails, but other services and operations continue, plus fatal errors.
- **4**: FATAL: The server stops.

By default the log level is set to **2** (WARN). Use **DEBUG** when you have a problem to diagnose, and then reset your log level to a less-verbose level as **DEBUG** outputs a lot of information, and can affect your server performance.

Logging level parameters are set in the config/config.php file.

Log type

errorlog

All log information will be sent to PHP `error_log()`.

```
"log_type" => "errorlog",
```

file

All log information will be written to a separate log file which can be viewed using the log viewer on your Admin page. By default, a log file named **nextcloud.log** will be created in the directory which has been configured by the **datadirectory** parameter in config/config.php.

The desired date format can optionally be defined using the **logdateformat** parameter in config/config.php. By default the PHP date function parameter c is used, and therefore the date/time is written in the format `2013-01-10T15:20:25+02:00`. By using the date format in the example below, the date/time format will be written in the format `January 10, 2013 15:20:25`.

```
"log_type" => "file",
"logfile" => "nextcloud.log",
"loglevel" => 3,
"logdateformat" => "F d, Y H:i:s",
```

syslog

All log information will be sent to your default syslog daemon.

```
"log_type" => "syslog",
"syslog_tag" => "Nextcloud",
"logfile" => "",
"loglevel" => 3,
```

systemd

All log information will be sent to Systemd journal. Requires php-systemd extension.

```
"log_type" => "systemd",
"syslog_tag" => "Nextcloud",
```

Log fields explained

Example log entries


```
{
    "reqId":"TBsuA2uE86DiOD0S8f9j",
    "level":1,
    "time":"April 13, 2021 16:55:37",
    "remoteAddr":"192.168.56.1",
    "user":"admin",
    "app":"admin_audit",
    "method":"GET",
    "url":"/ocs/v1.php/cloud/users?disabled",
    "message":"Login successful: \"admin\"",
    "userAgent":"curl/7.68.0",
    "version":"21.0.1.1"
}

{
    "reqId":"ByeDVLuwkXKMfLpBgvxC",
    "level":2,
    "time":"April 14, 2021 09:03:29",
    "remoteAddr":"192.168.56.1",
    "user":"--",
    "app":"no app in context",
    "method":"POST",
    "url":"/login",
    "message":"Login failed: asdf (Remote IP: 192.168.56.1)",
    "userAgent":"Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chro
    "version":"21.0.1.1"
}
```


Log field breakdown

- **reqId** (request id): any log lines related to a single request have the same value
- **level**: logged incident's level, always 1 in audit.log
- **time**: date and time (format and timezone can be configured in config.php)
- **remoteAddr**: the IP address of the user (if applicable – empty for occ commands)
- **user**: acting user's id (if applicable)
- **app**: affected app (always admin_audit in audit.log)
- **method**: HTTP method, for example GET, POST, PROPFIND, etc. – empty on occ calls
- **url**: request path (if applicable – empty on occ calls)
- **message**: event information message
- **userAgent**: user agent (if applicable – empty on occ calls)
- **exception**: Full exception with trace (if applicable)
- **data** additional structured data (if applicable)
- **version**: Nextcloud version at the time of request

Empty value are written as two dashes: "–".

Admin audit log

If `loglevel` in `config.php` is set to 2 or higher, audit logging needs to be triggered explicitly by adding the following setting to to `config.php`:

```
'log.condition' => [
        'apps' => ['admin_audit'],
],
```

Similar to the normal logging, the audit log can be written to any of the existing logging mechanism in config/config.php:

```
"log_type_audit" => "syslog",
"syslog_tag_audit" => "Nextcloud",
"logfile_audit" => "",
```

Previously the logfile could be defined in the app config. This config is still used when the system config is not provided:

```
occ config:app:set admin_audit logfile --value=/var/log/nextcloud/audit.log
```

Find detailed documentation on auditable events for enterprises in our customer portal.

Temporary overrides

You can run override the config.php log level of `occ` commands with as documented here.

Antivirus scanner

You can configure your Nextcloud server to automatically run a virus scan on newly-uploaded files with the Antivirus app for Files. The Antivirus app for Files integrates the open source anti-virus engine ClamAV with Nextcloud. ClamAV detects all forms of malware including Trojan horses, viruses, and worms, and it operates on all major file types including Windows, Linux, and Mac files, compressed files, executables, image files, Flash, PDF, and many others. ClamAV's Freshclam daemon automatically updates its malware signature database at scheduled intervals.

ClamAV runs on Linux and any Unix-type operating system, and Microsoft Windows. However, it has only been tested with Nextcloud on Linux, so these instructions are for Linux systems. You must first install ClamAV, and then install and configure the Antivirus app for Files on Nextcloud.

Installing ClamAV

As always, the various Linux distributions manage installing and configuring ClamAV in different ways.

Debian, Ubuntu, Linux Mint

On Debian and Ubuntu systems, and their many variants, install ClamAV with these commands:

```
apt-get install clamav clamav-daemon
```

The installer automatically creates default configuration files and launches the `clamd` and `freshclam` daemons. You don't have to do anything more, though it's a good idea to review the ClamAV documentation and your settings in `/etc/clamav/`. Enable verbose logging in both `clamd.conf` and `freshclam.conf` until you get any kinks worked out.

RedHat Enterprise Linux 7, CentOS 7

On RedHat Enterprise Linux 7 and related systems you must install the Extra Packages for Enterprise Linux (EPEL) repository, and then install ClamAV:

```
yum install epel-release
yum install clamav clamav-scanner clamav-scanner-systemd clamav-server
clamav-server-systemd clamav-update
```

This installs two configuration files: `/etc/freshclam.conf` and `/etc/clamd.d/scan.conf`. You must edit both of these before you can run ClamAV. Both files are well-commented, and `man clamd.conf` and `man freshclam.conf` explain all the options. Refer to `/etc/passwd` and `/etc/group` when you need to verify the ClamAV user and group.

First edit `/etc/freshclam.conf` and configure your options. `freshclam` updates your malware database, so you want it to run frequently to get updated malware signatures. Run it manually post-installation to download your first set of malware signatures:

```
freshclam
```

The EPEL packages do not include an init file for `freshclam`, so the quick and easy way to set it up for regular checks is with a cron job. This example runs it every hour at 47 minutes past the hour:

```
# m   h  dom mon dow  command
  47  *  *   *    *  /usr/bin/freshclam --quiet
```

Please avoid any multiples of 10, because those are when the ClamAV servers are hit the hardest for updates.

Next, edit `/etc/clamd.d/scan.conf`. When you're finished you must enable the `clamd` service file and start `clamd`:

```
systemctl enable clamd@scan.service
systemctl start clamd@scan.service
```

That should take care of everything. Enable verbose logging in `scan.conf` and `freshclam.conf` until it is running the way you want.

Docker, Docker-compose

To install ClamAV via docker or docker compose you can take official image of ClamAV, or build one by yourself. This example is based on docker image from https://github.com/Cisco-Talos/clamav.

You can mount ClamAV Socket from the Docker Container to the host System as volume. In this case you do not need to expose any port outside of container.

For a Docker run this command:

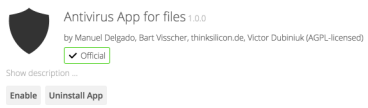
```
docker run --name clamav -d -v /var/run/clamav/:/var/run/clamav/ -v /var/docker/clamav/virus
```

For a Docker-compose use following settings:

```
version: "3.6"
services:
  clamav:
    image: "clamav/clamav:stable_base"
    container_name: "clamav"
    volumes:
      # Socket
      - /var/run/clamav/:/var/run/clamav/
      # Virus DB
      - /var/docker/clamav/virus_db/:/var/lib/clamav/
    restart: unless-stopped
```

Enabling the antivirus app for files

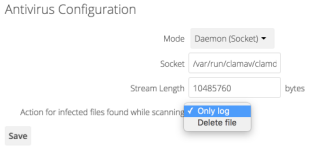
Place the `files_antivirus` app into the `apps` directory of your Nextcloud server. Then the app shows up on the Nextcloud Apps page where it simply can be enabled.



Configuring ClamAV on Nextcloud

Next, go to your Nextcloud Admin page and set your Nextcloud logging level to Everything.

Now find your Antivirus Configuration panel on your Admin page.



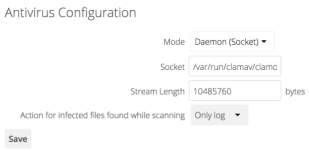
ClamAV runs in one of three modes:

- Daemon (Socket): ClamAV is running on the same server as Nextcloud. The ClamAV daemon, `clamd`, runs in the background. When there is no activity `clamd` places a minimal load on your system. If your users upload large volumes of files you will see high CPU usage.
- Daemon: ClamAV is running on a different server. This is a good option for Nextcloud servers with high volumes of file uploads.
- Executable: ClamAV is running on the same server as Nextcloud, and the `clamscan` command is started and then stopped with each file upload. `clamscan` is slow and not always reliable for on-demand usage; it is better to use one of the daemon modes.

Daemon (Socket)

Nextcloud should detect your `clamd` socket and fill in the `Socket` field. This is the `LocalSocket` option in `clamd.conf`. You can run `netstat` to verify:

```
netstat -a|grep clam
unix 2 [ ACC ] STREAM LISTENING 15857 /var/run/clamav/clamd.ctl
```

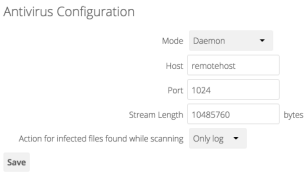


The `Stream Length` value sets the number of bytes read in one pass. 10485760 bytes, or ten megabytes, is the default. This value should be no larger than the PHP `memory_limit` settings, or physical memory if `memory_limit` is set to -1 (no limit).

`Action for infected files found while scanning` gives you the choice of logging any alerts without deleting the files, or immediately deleting infected files.

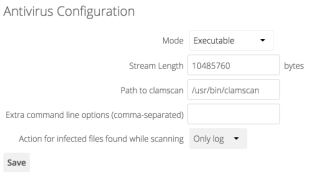
Daemon

For the Daemon option you need the hostname or IP address of the remote server running ClamAV, and the server's port number.



Executable

The Executable option requires the path to `clamscan`, which is the interactive ClamAV scanning command. Nextcloud should find it automatically.



When you are satisfied with how ClamAV is operating, you might want to go back and change all of your logging to less verbose levels.

Reverse proxy

Nextcloud can be run through a reverse proxy, which can cache static assets such as images, CSS or JS files, move the load of handling HTTPS to a different server or load balance between multiple servers.

Defining trusted proxies

For security, you must explicitly define the proxy servers that Nextcloud is to trust. Connections from trusted proxies will be specially treated to get the real client information, for use in access control and logging. Parameters are configured in config/config.php

Set the trusted_proxies parameter as an array of:

- IPv4 addresses,
- IPv4 ranges in CIDR notation
- IPv6 addresses

to define the servers Nextcloud should trust as proxies. This parameter provides protection against client spoofing, and you should secure those servers as you would your Nextcloud server.

A reverse proxy can define HTTP headers with the original client IP address, and Nextcloud can use those headers to retrieve that IP address. Nextcloud uses the de-facto standard header 'X-Forwarded-For' by default, but this can be configured with the **forwarded_for_headers** parameter. This parameter is an array of PHP lookup strings, for example 'X-Forwarded-For' becomes 'HTTP_X_FORWARDED_FOR'. Incorrectly setting this parameter may allow clients to spoof their IP address as visible to Nextcloud, even when going through the trusted proxy! The correct value for this parameter is dependent on your proxy software.

Overwrite parameters

The automatic hostname, protocol or webroot detection of Nextcloud can fail in certain reverse proxy situations. This configuration allows the automatic detection to be manually overridden. If Nextcloud fails to automatically detect the hostname, protocol or webroot you can use the **overwrite** parameters inside the config/config.php.

- overwritehost set the hostname of the proxy. You can also specify a port.
- overwriteprotocol set the protocol of the proxy. You can choose between the two options **http** and **https**.
- overwritewebroot set the absolute web path of the proxy to the Nextcloud folder.
- overwritecondaddr overwrite the values dependent on the remote address. The value must be a **regular expression** of the IP addresses of the proxy. This is useful when you use a reverse SSL proxy only for https access and you want to use the automatic detection for http access.

Leave the value empty or omit the parameter to keep the automatic detection.

Service Discovery

The redirects for CalDAV or CardDAV does not work if Nextcloud is running behind a reverse proxy. The recommended solution is that your reverse proxy does the redirects.

Apache2

```
RewriteEngine On
RewriteRule ^/\.well-known/carddav https://%{SERVER_NAME}/remote.php/dav/ [R=301,L]
RewriteRule ^/\.well-known/caldav https://%{SERVER_NAME}/remote.php/dav/ [R=301,L]
```

Thanks to @ffried for apache2 example.

Traefik 1

Using docker tags:

```
traefik.frontend.redirect.permanent: 'true'
traefik.frontend.redirect.regex: https://(.*)/.well-known/(card|cal)dav
traefik.frontend.redirect.replacement: https://$1/remote.php/dav/
```

Using traefik.toml:

```
[frontends.frontend1.redirect]
  regex = "https://(.*)/.well-known/(card|cal)dav"
```

```
  replacement = "https://$1/remote.php/dav/
  permanent = true
```

Thanks to @pauvos and @mrtumnus for traefik examples.

Traefik 2

```
[http.middlewares]
  [http.middlewares.nextcloud-redirectregex.redirectRegex]
    permanent = true
    regex = "https://(.*)/.well-known/(card|cal)dav"
    replacement = "https://${1}/remote.php/dav/"
```

HAProxy

```
acl url_discovery path /.well-known/caldav /.well-known/carddav
http-request redirect location /remote.php/dav/ code 301 if url_discovery
```

NGINX

```
location /.well-known/carddav {
    return 301 $scheme://$host/remote.php/dav;
}

location /.well-known/caldav {
    return 301 $scheme://$host/remote.php/dav;
}
```

or

```
rewrite ^/\.well-known/carddav https://$server_name/remote.php/dav/ redirect;
rewrite ^/\.well-known/caldav https://$server_name/remote.php/dav/ redirect;
```

Caddy

```
subdomain.example.com {
    rewrite /.well-known/carddav /remote.php/dav
    rewrite /.well-known/caldav /remote.php/dav

    reverse_proxy {$NEXTCLOUD_HOST:localhost}
}
```

Example

Multiple domains reverse SSL proxy

If you want to access your Nextcloud installation **http://domain.tld/nextcloud** via a multiple domains reverse SSL proxy **https://ssl-proxy.tld/domain.tld/nextcloud** with the IP address **10.0.0.1** you can set the following parameters inside the config/config.php.

```
<?php
$CONFIG = array (
  'trusted_proxies'   => ['10.0.0.1'],
  'overwritehost'     => 'ssl-proxy.tld',
  'overwriteprotocol' => 'https',
  'overwritewebroot'  => '/domain.tld/nextcloud',
  'overwritecondaddr' => '^10\.0\.0\.1$',
);
```

Note

If you want to use the SSL proxy during installation you have to create the config/config.php otherwise you have to extend the existing **$CONFIG** array.

Brute force protection

Nextcloud has built-in protection against brute force attempts. This protects your system from attackers trying for example a lot of different passwords.

Brute force protection is enabled by default on Nextcloud.

How it works

The brute force protection is easiest to see in action at the login page. If you try to log in the first time with an invalid username and/or password you will not notice anything. But if you do this a few times you start to notice that the verification of the login is taking longer each time. This is the brute force protection kicking in.

The maximum delay is 25 seconds.

After a successful login the attempts will be cleared. And once a user is properly authenticated they will not longer be hit by the delay.

Troubleshooting

On most setups Nextcloud will work out of the box without any issues. If you run into a situation where login is often very slow for all users the first step is to inspect the *bruteforce_attempts* table. There you can see which IP addresses are actually throttled.

If you are behind a reverse proxy or load balancer it is important you make sure it is setup properly. Especially the **trusted_proxies** and **forwarded_for_headers** *config.php* variables need to be set correctly. Otherwise it can happen that Nextcloud actually starts throttling all traffic coming from the reverse proxy or load balancer. For more information see Reverse proxy.

Automatic setup

If you need to install Nextcloud on multiple servers, you normally do not want to set up each instance separately as described in Database configuration. For this reason, Nextcloud provides an automatic configuration feature.

To take advantage of this feature, you must create a configuration file, called config/autoconfig.php, and set the file parameters as required. You can specify any number of parameters in this file. Any unspecified parameters appear on the “Finish setup” screen when you first launch Nextcloud.

The config/autoconfig.php is automatically removed after the initial configuration has been applied.

Note

Keep in mind that the automatic configuration does not eliminate the need for creating the database user and database in advance, as described in Database configuration.

Parameters

When configuring parameters, you must understand that two parameters are named differently in this configuration file when compared to the standard config.php file.

autoconfig.php	config.php
directory	datadirectory
dbpass	dbpassword

Automatic configurations examples

The following sections provide sample automatic configuration examples and what information is requested at the end of the configuration.

Data Directory

Using the following parameter settings, the “Finish setup” screen requests database and admin credentials settings.

```
<?php
$AUTOCONFIG = [
  "directory"     => "/www/htdocs/nextcloud/data",
];
```

SQLite database

Using the following parameter settings, the “Finish setup” screen requests data directory and admin credentials settings.

```
<?php
$AUTOCONFIG = [
  "dbtype"        => "sqlite",
  "dbname"        => "nextcloud",
  "dbtableprefix" => "",
];
```

MySQL database

Using the following parameter settings, the “Finish setup” screen requests data directory and admin credentials settings.

```
<?php
$AUTOCONFIG = array(
  "dbtype"        => "mysql",
  "dbname"        => "nextcloud",
  "dbuser"        => "username",
  "dbpass"        => "password",
  "dbhost"        => "localhost",
  "dbtableprefix" => "",
);
```

PostgreSQL database

Using the following parameter settings, the “Finish setup” screen requests data directory and admin credentials settings.

```
<?php
$AUTOCONFIG = array(
  "dbtype"        => "pgsql",
  "dbname"        => "nextcloud",
  "dbuser"        => "username",
  "dbpass"        => "password",
  "dbhost"        => "localhost",
  "dbtableprefix" => "",
);
```

Note

Keep in mind that the automatic configuration does not eliminate the need for creating the database user and database in advance, as described in Database configuration.

All parameters

Using the following parameter settings, because all parameters are already configured in the file, the Nextcloud installation skips the “Finish setup” screen.

```
<?php
$AUTOCONFIG = array(
  "dbtype"        => "mysql",
  "dbname"        => "nextcloud",
  "dbuser"        => "username",
  "dbpass"        => "password",
  "dbhost"        => "localhost",
  "dbtableprefix" => "",
  "adminlogin"    => "root",
  "adminpass"     => "root-password",
  "directory"     => "/www/htdocs/nextcloud/data",
);
```

Note

Keep in mind that the automatic configuration does not eliminate the need for creating the database user and database in advance, as described in Database configuration.

Theming

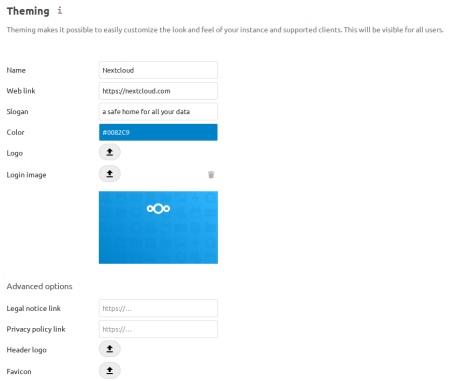
With our theming feature you are able to customize the look and feel of your Nextcloud instance according to the corporate design of your organization by replacing Nextcloud logo and color with your own assets.

The theming-app is enabled by default so the section should appear by default in your admin-settings. If not, check in the apps management that this app is enabled.

Modify the appearance of Nextcloud

You can change the following parameters of the look and feel on your instance:

- Name (e.g. ACME Inc. Cloud)
- Web Address (e.g. https://acme.inc/)
- Slogan
- Color: The color of header bar, checkboxes and folder icon
- Logo: The logo will appear in the header and on the log in page. Default has 62/34 px.
- Login image: The background image of the login page
- Additional legal links (Legal notice and Privacy policy link)
- Custom header logo and favicon as alternative to auto-generation based on logo



Configure theming through CLI

Theming configuration can also be adjusted through the `occ theming:config` command.

The following values are available to be set through this:

- name, url, imprintUrl, privacyUrl, slogan, color `occ theming:config name "My Example Cloud"`
- background, logo, favicon, logoheader `occ theming:config logo /tmp/mylogo.png`

> **Note**
>
> Images require to be read from a local file on the Nextcloud server

Theming of icons

According to the parameters you have set, Nextcloud will automatically generate favicons and a header logo depending on the current logo and theming color.

This requires the following additional dependencies:

- PHP module `imagick`
- SVG support for imagick (e.g. `libmagickcore-6.q16-3-extra` on Debian 9 and Ubuntu 18.04)

> **Note**
>
> In the advanced options of the theming app you are able to set a custom favicon in case you do not want to use the same logo resources you have set above or you do not want to install the mentioned dependencies.

Branded clients

> **Note**
>
> Nextcloud GmbH provides branding services, delivering sync clients (mobile and desktop) which use your corporate identity and are pre-configured to help your users get up and running in no time. If you are interested in our advanced branding & support subscription, contact our sales team.

The theming app supports to change the URLs to the mobile apps (Android & iOS) that is shown when the webinterface is opened on one of those devices. Then there was a header shown, that redirects the user to the app in the app store. By default this redirects to the Nextcloud apps. In some cases it is wanted that this links to branded versions of those apps. In those cases the IDs and URLs can be set via the `occ`-command:

```
occ config:app:set theming AndroidClientUrl --value "https://play.google.com/store/apps/deta
occ config:app:set theming iTunesAppId --value "1125420102"
occ config:app:set theming iOSClientUrl --value "https://itunes.apple.com/us/app/nextcloud/i
```

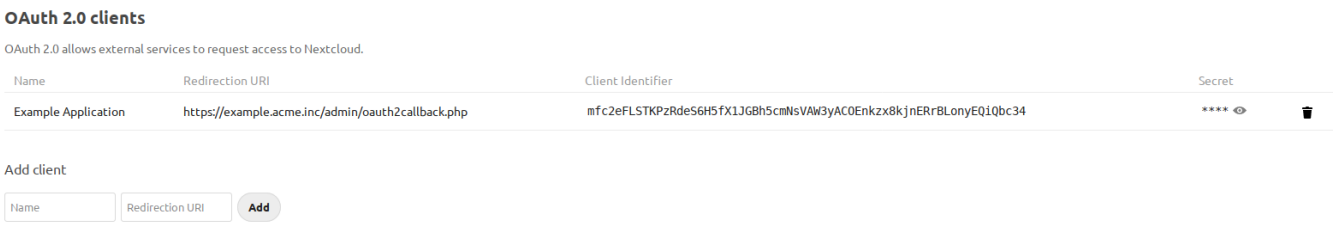
OAuth2

Nextcloud allows connecting external services (for example Moodle) to your Nextcloud. This is done via `OAuth2`. See RFC6749 for the OAuth2 specification.

> **Note**
>
> Nextcloud does only support confidential clients.

Add an OAuth2 Application

Head over to your Administrator Security Settings. Here you can add a new `OAuth2` client.



Enter the name of your application and provide a redirection url. You should now have a Client Identifier and Secret. Enter those into your `OAuth2` client.

Please provide the OAuth2 application the following details:

- Authorization endpoint: `https://cloud.example.org/apps/oauth2/authorize`
- Token endpoint: `https://cloud.example.org/apps/oauth2/api/v1/token`

Note that you must include `index.php` if pretty URL is not configured - i.e. `https://cloud.example.org/index.php/apps/oauth2/api/v1/token`.

The access token

The access token obtained is a so called Bearer token. Which means that for request to the Nextcloud server you will have to send the proper authorization header.

Authorization: Bearer <TOKEN>

Note that apache by default strips this. Make sure you have `mod_headers`, `mod_rewrite` and `mod_env` enabled.

Security considerations

Nextcloud `OAuth2` implementation currently does not support scoped access. This means that every token has full access to the complete account including read and write permission to the stored files. It is essential to store the `OAuth2` tokens in a safe way!

Without scopes and restrictable access it is not recommended to use a Nextcloud instance as a user authentication service.

Admin right priviledge

By default only members of the admin group can access and edit the admin settings. It is sometimes needed to give some group of users access to a setting page while not giving them access to everything. For this you can use the *Admin right priviledge* settings.

Note

Not every setting pages support this features. This is due to either the feature not being implemented yet for the specific setting page or due to possible priviledge escalations.

Configuring Admin right priviledge

Go to the *Admin right priviledge* Admin page, you should be presented with the list of settings that support this features.

By clicking on the combobox, you will be able to choose which user groups are able to access the selected setting. You can revoke the access at any time by removing the group from the selection.

Domain Change

Changing the domain after the first setup is currently not supported by Nextcloud. That is mainly because Nextcloud's apps don't support changing the domain after they are set up.

Note

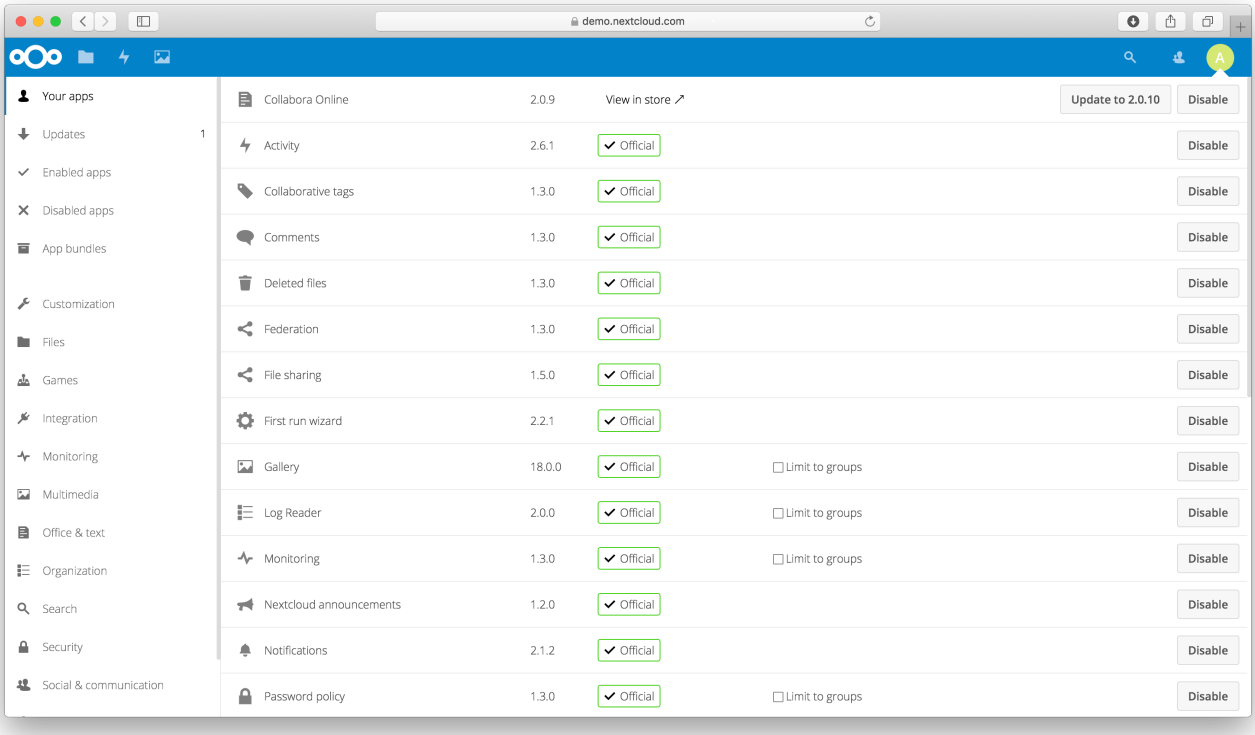
This documentation will get updated with steps what needs to be done if you want to change the domain as soon as this feature is available.

Apps management

Nextcloud apps can enhance, customize or even restrict the features and experience you and your users has with the Nextcloud server. Next to default enabled functions like Files, Activity and Photos there are other apps like Calendar, Contacts, Talk and more which are enhancing the features of your Nextcloud server.

After installing the Nextcloud server, you might want to consider about enabling, disabling or even restricting some apps to groups depending on your and your users' needs.

Apps



During the Nextcloud server installation, some apps are enabled by default. To see which apps are enabled go to your Apps page.

Those apps are supported and developed by Nextcloud GmbH directly and have an **Featured**-tag. See Supported apps for a list of supported apps.

Note

Your Nextcloud server needs to be able to communicate with `https://apps.nextcloud.com` to list and download apps. Please make sure to whitelist this target in your firewall or proxy if necessary.

Note

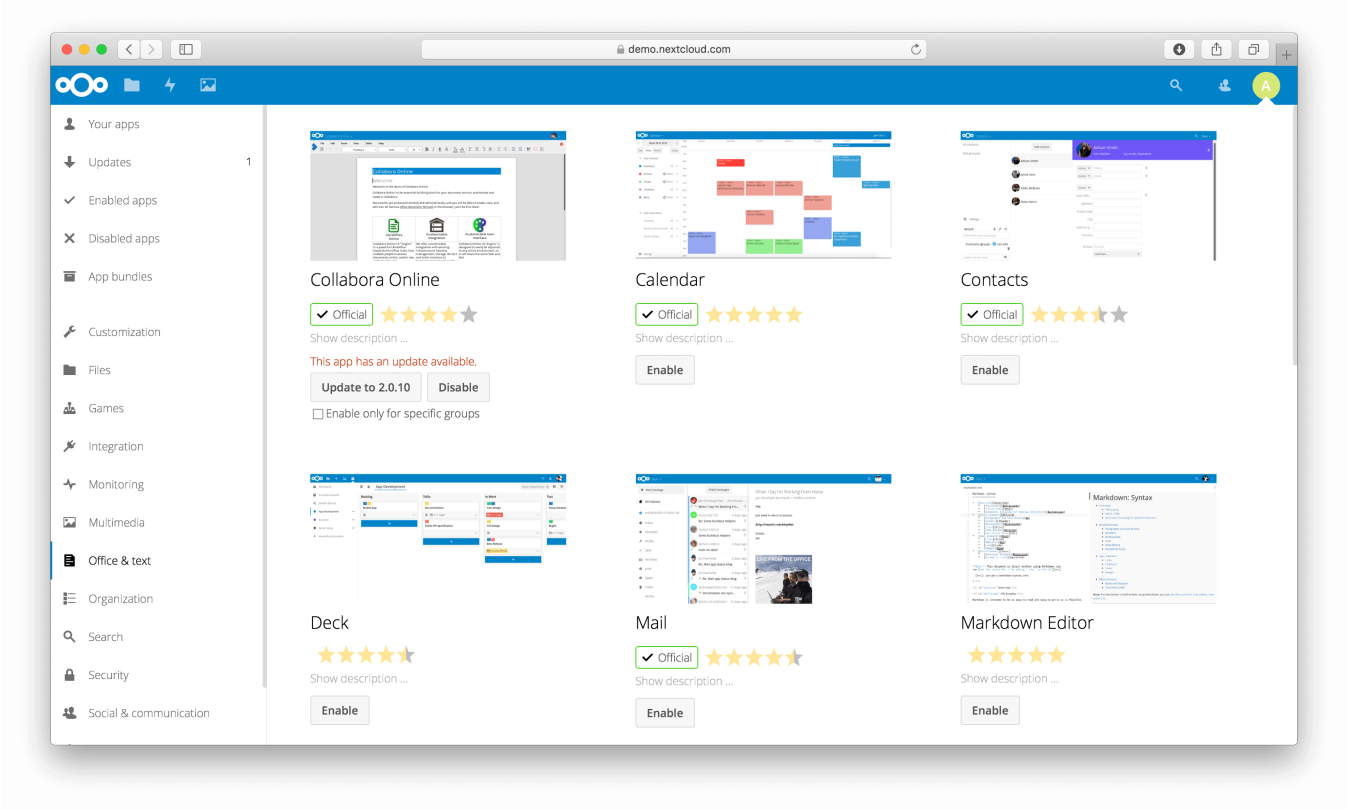
To get access to work-arounds, long-term-support, priority bug fixing and custom consulting for supported apps, contact our sales team.

Note

If you would like to develop your own Nextcloud app, you can find out more information in our developer manual.

All apps must be licensed under AGPLv3+ or any compatible license.

Managing apps



You will see which apps are enabled, disabled and available. You'll also see additional app bundles and filters, such as Customization, Security and Monitoring for finding more apps quickly.

In the Apps page you can enable or disable applications. Some apps have configurable options on the Apps page, such as **Enable only for specific groups**, but mainly they are enabled or disabled here, and are configured in your Nextcloud settings (admin and/or user-settings) or in the `config.php`.

Click the app name to view a description of the app and any of the app settings in the Application View field. Clicking the **Enable** button will enable the app. If the app is not part of the Nextcloud installation, it will be downloaded from the app store, installed and enabled.

App updates will also be offered to you on this page. Simply click on the **Update** button to update a specific app or use the **Update all** button on top of the page to update all apps.

Note

Beta releases: You can also install beta releases of apps directly from here by switching your Nextcloud to the beta channel in the admin overview.

Using private API

If private API, rather than the public APIs are used in a third-party app, the installation fails, if `'appcodechecker' => true,` is set in `config.php`.

Using custom app directories

Use the **apps_paths** array in `config.php` to set any custom apps directory locations. The key **path** defines the absolute file system path to the app folder. The key **url** defines the HTTP web path to that folder, starting at the Nextcloud web root. The key **writable** indicates if a user can install apps in that folder.

Note

To ensure that the default **/apps/** folder only contains apps shipped with Nextcloud, follow this example to setup an **/apps2/** folder which will be used to store all other apps.

```
"apps_paths" => [
    [
            "path"     => OC::$SERVERROOT . "/apps",
            "url"      => "/apps",
            "writable" => false,
    ],
    [
            "path"     => OC::$SERVERROOT . "/apps2",
            "url"      => "/apps2",
            "writable" => true,
    ],
],
```

Note

Apps paths can be located outside the server root. However, for any **path** outside the server root, you need to create a symlink in the server root that points **url** to **path**. For instance, if **path** is `/var/local/lib/nextcloud/apps`, and **url** is `/apps2`, then you would do this in the server root: `ln -sf /var/local/lib/nextcloud/apps ./apps2`

Using a self hosted apps store

Enables the installation of apps from a self hosted apps store. Requires that at least one of the configured apps directories is writeable.

To enable a self hosted apps store:

1. Set the **appstoreenabled** parameter to “true”.

   This parameter is used to enable the apps store in Nextcloud.

2. Set the **appstoreurl** to the URL of your Nextcloud apps store.

   This parameter is used to set the http path to your self hosted Nextcloud apps store.

```
"appstoreenabled" => true,
"appstoreurl" => "https://my.appstore.instance/v1",
```

By default the apps store is enabled and configured to use `https://apps.nextcloud.com/api/v1` as apps store url. Nextcloud will fetch `apps.json` and `categories.json` from there. To use the defaults again remove **appstoreenabled** and **appstoreurl** from the configuration.

Example: If `categories.json` is available at `https://apps.nextcloud.com/api/v1/categories.json` the apps store url is `https://apps.nextcloud.com/api/v1`.

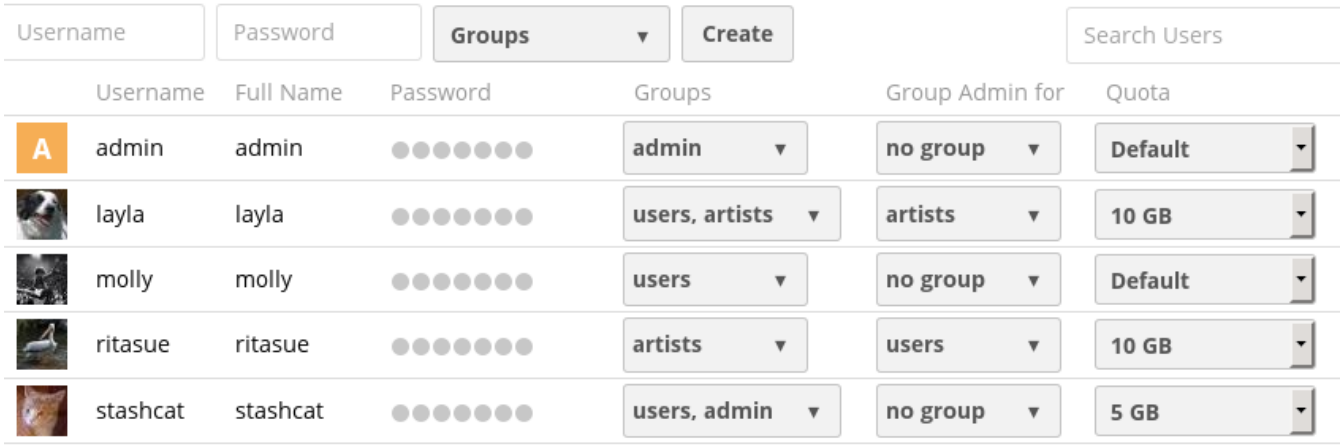
User management

User management

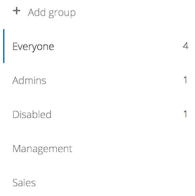
On the User management page of your Nextcloud Web UI you can:

- Create new users
- View all of your users in a single scrolling window
- Filter users by group
- See what groups they belong to
- Edit their full names and passwords
- See their data storage locations
- View and set quotas
- Create and edit their email addresses
- Send an automatic email notification to new users
- Disable and Enable users
- Delete them with a single click

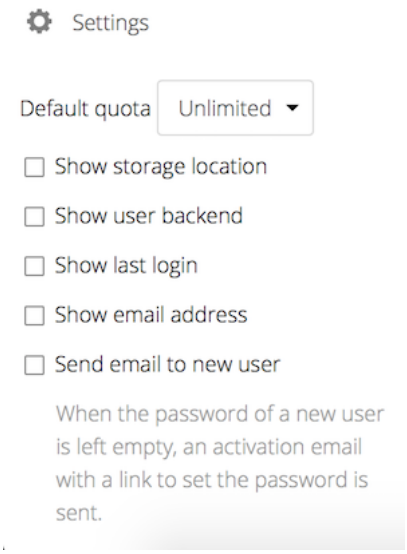
The default view displays basic information about your users.



The Group filters on the left sidebar lets you quickly filter users by their group memberships, and create new groups.



Click the gear icon on the lower left sidebar to set a default storage quota, and to display additional fields: **Show storage location, Show last log in, Show user backend, Send email to new users,** and **Show email address**.



User accounts have the following properties:

Login Name (Username)

The unique ID of a Nextcloud user, and it cannot be changed.

Full Name

The user's display name that appears on file shares, the Nextcloud Web interface, and emails. Admins and users may change the Full Name anytime. If the Full Name is not set it defaults to the login name.

Password

The admin sets the new user's first password. Both the user and the admin can change the user's password at anytime.

Groups

You may create groups, and assign group memberships to users. By default new users are not assigned to any groups.

Group Admin

Group admins are granted administrative privileges on specific groups, and can add and remove users from their groups.

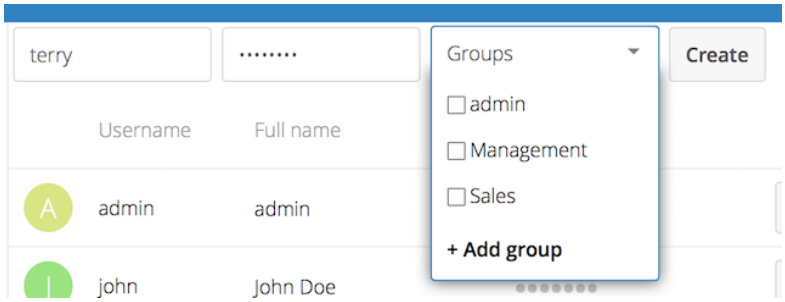
Quota

The maximum disk space assigned to each user. Any user that exceeds the quota cannot upload or sync data. You have the the option to include external storage in user quotas.

Creating a new user

To create a user account:

- Enter the new user's **Login Name** and their initial **Password**
- Optionally, assign **Groups** memberships
- Click the **Create** button



Login names may contain letters (a-z, A-Z), numbers (0-9), dashes (-), underscores (_), periods (.), spaces ( ) and at signs (@). After creating the user, you may fill in their **Full Name** if it is different than the login name, or leave it for the user to complete.

If you have checked **Send email to new user** in the control panel on the lower left sidebar, you may also enter the new user's email address, and Nextcloud will automatically send them a notification with their new login information. You may edit this email using the email template editor on your Admin page (see Email).

Set the **Send email to new user**-checkbox allows you to leave the **Password** field empty. The user will get an activation-email to set their own password.

Reset a user's password

You cannot recover a user's password, but you can set a new one:

- Hover your cursor over the user's **Password** field
- Click on the **pencil icon**
- Enter the user's new password in the password field, and remember to provide the user with their password

If you have encryption enabled, there are special considerations for user password resets. Please see Encryption configuration.

Renaming a user

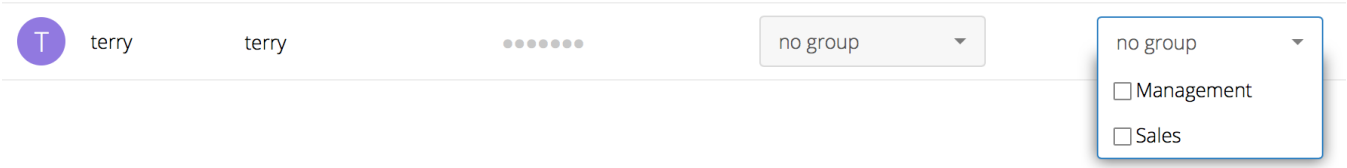
Each Nextcloud user has two names: a unique **Login Name** used for authentication, and a **Full Name**, which is their display name. You can edit the display name of a user, but you cannot change the login name of any user.

To set or change a user's display name:

- Hover your cursor over the user's **Full Name** field
- Click on the **Pencil icon**
- Enter the user's new display name

Granting administrator privileges to a user

Nextcloud has two types of administrators: **Super Administrators** and **Group Administrators**. Group administrators have the rights to create, edit and delete users in their assigned groups. Group administrators cannot access system settings, or add or modify users in the groups that they are not **Group Administrators** for. Use the dropdown menus in the **Group Admin** column to assign group admin privileges.



Super Administrators have full rights on your Nextcloud server, and can access and modify all settings. To assign the **Super Administrators** role to a user, simply add them to the `admin` group.

Managing groups

You can assign new users to groups when you create them, and create new groups when you create new users. You may also use the **Add Group** button at the top of the left pane to create new groups. New group members will immediately have access to file shares that belong to their new groups.

Setting Storage quotas

Click the gear on the lower left pane to set a default storage quota. This is automatically applied to new users. You may assign a different quota to any user by selecting from the **Quota** dropdown, selecting either a preset value or entering a custom value. When you create custom quotas, use the normal abbreviations for your storage values such as 500 MB, 5 GB, 5 TB, and so on.

You now have a configurable option in `config.php` that controls whether external storage is counted against user's quotas. This is still experimental, and may not work as expected. The default is to not count external storage as part of user storage quotas. If you prefer to include it, then change the default `false` to `true`.

```
'quota_include_external_storage' => false,
```

Note

If an external storage is defined as root, the quota will not be calculable and will be **ignored**.

Metadata (such as thumbnails, temporary files, and encryption keys) takes up about 10% of disk space, but is not counted against user quotas. Users can check their used and available space on their Personal pages. Only files that originate with users count against their quotas, and not files shared with them that originate from other users. For example, if you upload files to a different user's share, those files count against your quota. If you re-share a file that another user shared with you, that file does not count against your quota, but the originating user's.

Encrypted files are a little larger than unencrypted files; the unencrypted size is calculated against the user's quota.

Deleted files that are still in the trash bin do not count against quotas. The trash bin is set at 50% of quota. Deleted file aging is set at 30 days. When deleted files exceed 50% of quota then the oldest files are removed until the total is below 50%.

When version control is enabled, the older file versions are not counted against quotas.

When a user creates a public share via URL, and allows uploads, any uploaded files count against that user's quota.

Disable and enable users

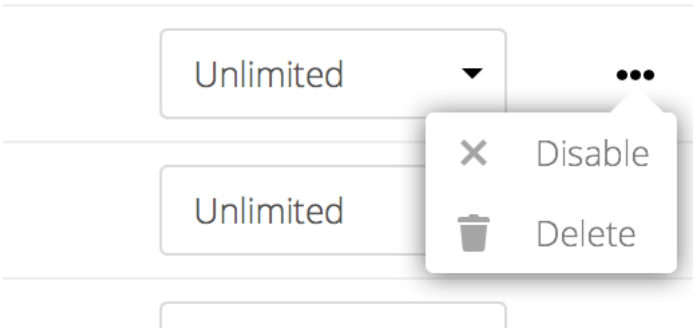
Sometimes you may want to disable a user without permanently deleting their settings and files. The user can be activated any time again, without data-loss.

Hover your cursor over their name on the **Users** page until the "…"-menu icon appears at the far right. After clicking on it, you will see the **Disable** option.

The user will not longer be able to access their Nextcloud until you enable them again. Keep in mind that the files, which were shared by this user will not longer be accessible.

You will find all disabled users in the **disabled**-section on the left pane. Enabling users is as easy as disabling them. Just click on the "…"-menu, and select **Enable**.

Deleting users



Deleting a user is easy: hover your cursor over their name on the **Users** page until the "…"-menu icon appears at the far right. After clicking on it, you will see the **Delete** option. Clicking on it, delets a user with all their data immediately.

You'll see an undo button at the top of the page, which remains for some seconds. When the undo button is gone you cannot recover the deleted user.

All of the files owned by the user are deleted as well, including all files they have shared. If you need to preserve the user's files and shares, you must first download them from your Nextcloud Files page, which compresses them into a zip file, or use a sync client to copy them to your local computer. See File Sharing to learn how to create persistent file shares that survive user deletions.

Resetting a lost admin password

The normal ways to recover a lost password are:

1. Click the password reset link on the login screen; this appears after a failed login attempt. This works only if you have entered your email address on your Personal page in the Nextcloud Web interface, so that the Nextcloud server can email a reset link to you.
2. Ask another Nextcloud server admin to reset it for you.

If neither of these is an option, then you have a third option, and that is using the `occ` command. See Using the occ command to learn more about using the `occ` command.

```
$ sudo -u www-data php /var/www/nextcloud/occ user:resetpassword admin
Enter a new password:
Confirm the new password:
Successfully reset password for admin
```

If your Nextcloud username is not `admin`, then substitute your Nextcloud username.

Resetting a user password

The Nextcloud login screen displays a **Wrong password. Reset it?** message after a user enters an incorrect password, and then Nextcloud automatically resets their password. However, if you are using a read-only authentication backend such as LDAP or Active Directory, this will not work. In this case you may specify a custom URL in your `config.php` file to direct your user to a server than can handle an automatic reset:

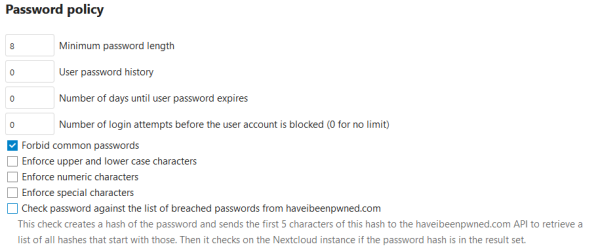
```
'lost_password_link' => 'https://example.org/link/to/password/reset',
```

User password policy

A password policy is a set of rules designed to enhance computer security by encouraging users to employ strong passwords and use them properly.

In the security-section of your administrator-settings you can configure

- a minimal length of a password. Default is 8 characters.
- a password history
- a password expiration period
- a lockout policy
- to forbid common passwords like 'password' or 'login'.
- to enforce upper and lower case characters
- to enforce numeric characters
- to enforce special characters like ! or :
- to check the password against the list of breached passwords from haveibeenpwnd.com (hashed check via haveibeenpwnd.com-API)



Two-factor authentication

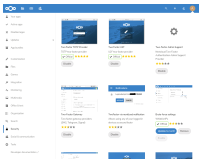
Two-factor authentication adds an additional layer of security to user accounts. In order to log in on an account with two-factor authentication (2FA) enabled, it is necessary to provide both the login password and another factor. 2FA in Nextcloud is pluggable, meaning that they are not part of the Nextcloud Server component but provided by featured and 3rd-party Nextcloud apps.

Several 2FA apps are already available including TOTP, a Telegram/Signal/SMS gateway and U2F.

Developers can build new two-factor provider apps.

Enabling two-factor authentication

You can enable 2FA by installing and enabling a 2FA app like TOTP which works with Google Authenticator and compatible apps. The apps are available in the Nextcloud App store so by navigating there and clicking **enable** for the app you want, 2FA will be installed and enabled on your Nextcloud server.



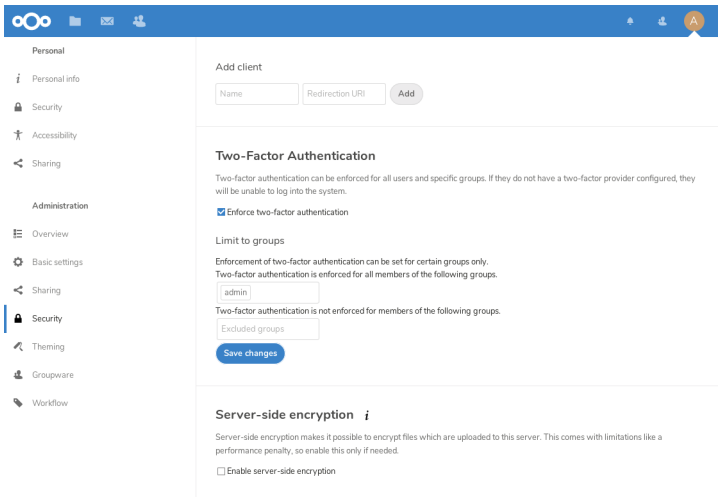
Once 2FA has been enabled, users have to activate it in their personal settings.

Enforcing two-factor authentication

By default 2FA is *optional*, hence users are given the choice whether to enable it for their account. Admins may enforce the use of 2FA.

Enforcement is possible systemwide (all users), for selected groups only and can also be excluded for certain groups.

These settings can be found in the administrator's security settings.



When groups are selected/excluded, they use the following logic to determine if a user has 2FA enforced:

- If no groups are selected, 2FA is enabled for everyone except members of the excluded groups
- If groups are selected, 2FA is enabled for all members of these. If a user is both in a selected *and* excluded group, the selected takes precedence and 2FA is enforced.

User authentication with IMAP, SMB, FTP and others

You may configure additional user backends with the External User Backends (user_external) app.

See https://github.com/nextcloud/user_external#readme for an up to date documentation.

User authentication with LDAP

Nextcloud ships with an LDAP application to allow LDAP users (including Active Directory) to appear in your Nextcloud user listings. These users will authenticate to Nextcloud with their LDAP credentials, so you don't have to create separate Nextcloud user accounts for them. You will manage their Nextcloud group memberships, quotas, and sharing permissions just like any other Nextcloud user.

Note

The PHP LDAP module is required; this is supplied by `php-ldap` on most distributions.

The LDAP application supports:

- LDAP group support
- File sharing with Nextcloud users and groups
- Access via WebDAV and Nextcloud Desktop Client
- Versioning, external Storage and all other Nextcloud features
- Seamless connectivity to Active Directory, with no extra configuration required
- Support for primary groups in Active Directory
- Auto-detection of LDAP attributes such as base DN, email, and the LDAP server port number
- Only read access to your LDAP (edit or delete of users on your LDAP is not supported)
- Optional: Allow users to change their LDAP password from Nextcloud

Note

A non-blocking or correctly configured SELinux setup is needed for the LDAP backend to work. Please refer to the SELinux configuration.

Configuration

First enable the `LDAP user and group backend` app on the Apps page in Nextcloud. Then go to your Admin page to configure it.

The LDAP configuration panel has four tabs. A correctly completed first tab ("Server") is mandatory to access the other tabs. A green indicator lights when the configuration is correct. Hover your cursor over the fields to see some pop-up tooltips.

Server tab

Start with the Server tab. You may configure multiple servers if you have them.

Note

Do not configure any failover LDAP hosts here. See Advanced settings for instructions instead.

At a minimum you must supply the LDAP server's hostname. If your server requires authentication, enter your credentials on this tab. Nextcloud will then attempt to auto-detect the server's port and base DN. The base DN and port are mandatory, so if Nextcloud cannot detect them you must enter them manually.

Server configuration:

Configure one or more LDAP servers. Click the **Delete Configuration** button to remove the active configuration.

Host:

The host name or IP address of the LDAP server. It can also be a **ldaps://** URI. If you enter the port number, it speeds up server detection.

Examples:

- *directory.my-company.com*
- *ldaps://directory.my-company.com*
- *directory.my-company.com:9876*

Port:

The port on which to connect to the LDAP server. The field is disabled in the beginning of a new configuration. If the LDAP server is running on a standard port, the port will be detected automatically. If you are using a non-standard port, Nextcloud will attempt to detect it. If this fails you must enter the port number manually.

Example:

- *389*

User DN:

The name as DN of a user who has permissions to do searches in the LDAP directory. Leave it empty for anonymous access. We recommend that you have a special LDAP system user for this.

Example:

- *uid=nextcloudsystemuser,cn=sysusers,dc=my-company,dc=com*

Password:

The password for the user given above. Empty for anonymous access.

Base DN:

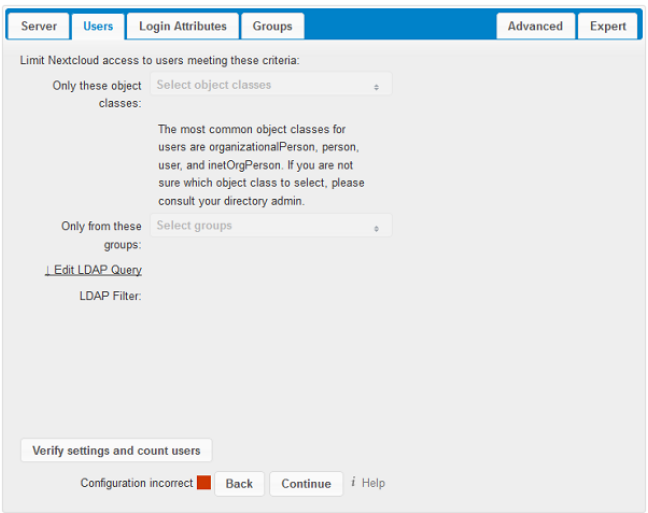
The base DN of LDAP, from where all users and groups can be reached. You may enter multiple base DNs, one per line. (Base DNs for users and groups can be set in the Advanced tab.) This field is mandatory. Nextcloud attempts to determine the Base DN according to the provided User DN or the provided Host, and you must enter it manually if Nextcloud does not detect it.

Example:

- *dc=my-company,dc=com*

Users tab

Use this to control which LDAP users are listed as Nextcloud users on your Nextcloud server. In order to control which LDAP users can login to your Nextcloud server use the **Login Attributes** tab. Those LDAP users who have access but are not listed as users (if there are any) will be hidden users. You may bypass the form fields and enter a raw LDAP filter if you prefer.



Only those object classes:

Nextcloud will determine the object classes that are typically available for user objects in your LDAP. Nextcloud will automatically select the object class that returns the highest amount of users. You may select multiple object classes.

Only from those groups:

If your LDAP server supports the `member-of-overlay` in LDAP filters, you can define that only users from one or more certain groups are allowed to appear in user listings in Nextcloud. By default, no value will be selected. You may select multiple groups.

If your LDAP server does not support the `member-of-overlay` in LDAP filters, the input field is disabled. Please contact your LDAP administrator.

Edit LDAP Query:

Clicking on this text toggles the filter mode and you can enter the raw LDAP filter directly. Example:

```
(&(objectClass=inetOrgPerson)(memberOf=cn=nextcloudusers,ou=groups,
dc=example,dc=com))
```

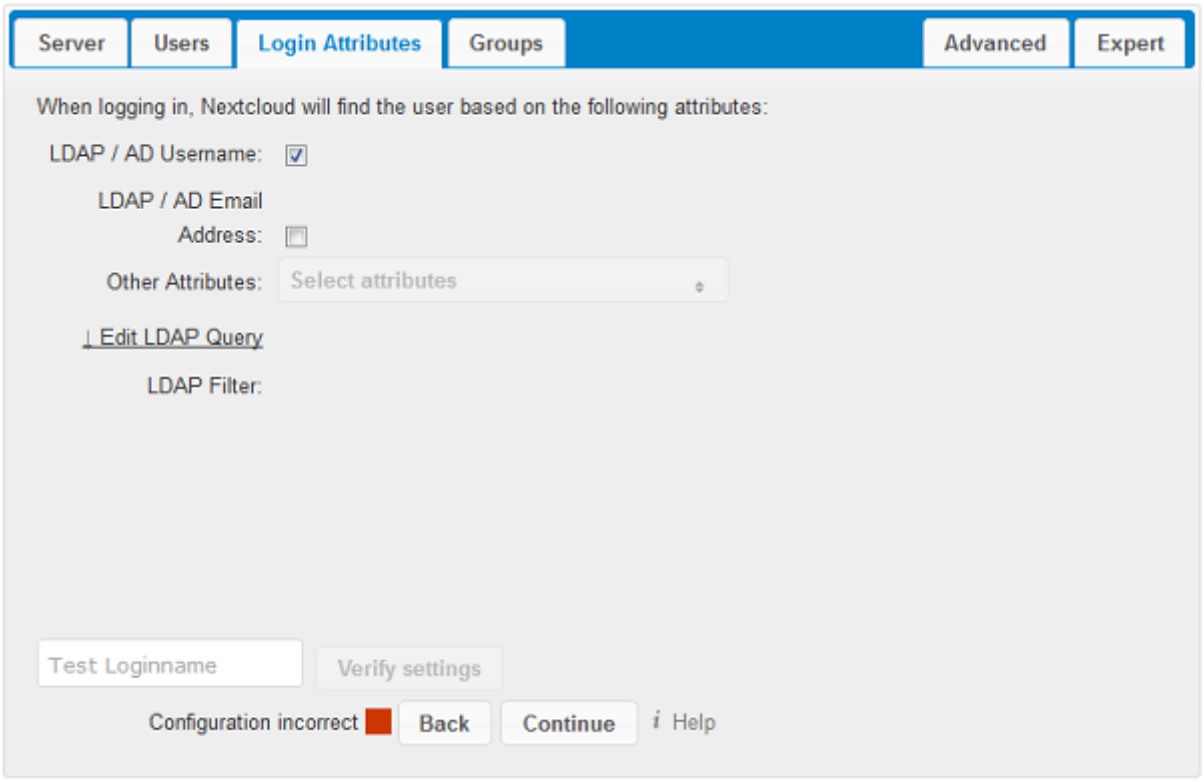
x users found:

This is an indicator that tells you approximately how many users will be listed in Nextcloud. The number updates automatically after any changes.

Login attributes tab

The settings in the Login Attributes tab determine which LDAP users can log in to your Nextcloud system and which attribute or attributes the provided login name is matched against (e.g. LDAP/AD username, email address). You may select multiple user details. (You may bypass the form fields and enter a raw LDAP filter if you prefer.)

You may override your User Filter settings on the Users tab by using a raw LDAP filter.



LDAP Username:

If this value is checked, the login value will be compared to the username in the LDAP directory. The corresponding attribute, usually *uid* or *samaccountname* will be detected automatically by Nextcloud.

LDAP Email Address:

If this value is checked, the login value will be compared to an email address in the LDAP directory; specifically, the *mailPrimaryAddress* and *mail* attributes.

Other Attributes:

This multi-select box allows you to select other attributes for the comparison. The list is generated automatically from the user object attributes in your LDAP server.

Edit LDAP Query:

Clicking on this text toggles the filter mode and you can enter the raw LDAP filter directly.

The **%uid** placeholder is replaced with the login name entered by the user upon login.

Examples:

- only username:

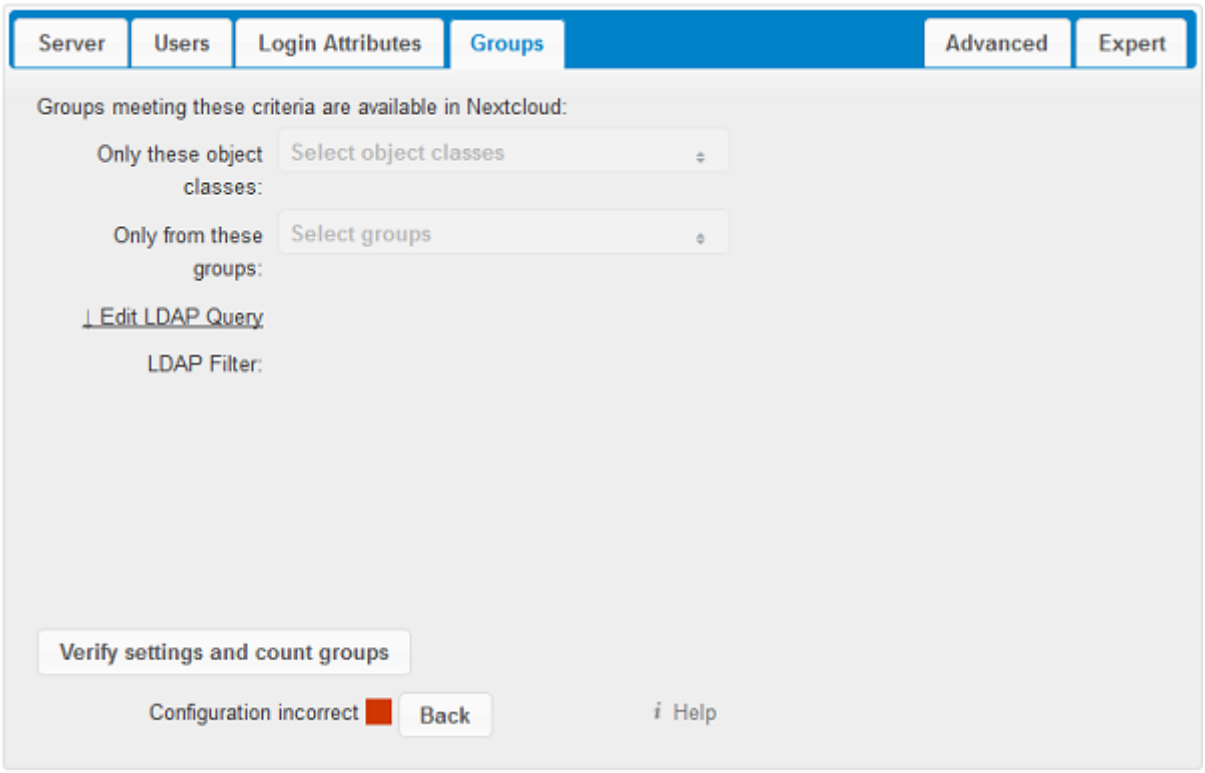
```
(&(objectClass=inetOrgPerson)(memberOf=cn=nextcloudusers,ou=groups,
dc=example,dc=com)(uid=%uid)
```

- username or email address:

```
((&(objectClass=inetOrgPerson)(memberOf=cn=nextcloudusers,ou=groups,
dc=example,dc=com)(|(uid=%uid)(mail=%uid)))
```

Groups tab

By default, no LDAP groups will be available in Nextcloud. The settings in the Groups tab determine which groups will be available in Nextcloud. You may also elect to enter a raw LDAP filter instead.



Only these object classes:

Nextcloud will determine the object classes that are typically available for group objects in your LDAP server. Nextcloud will only list object classes that return at least one group object. You can select multiple object classes. A typical object class is “group”, or “posixGroup”.

Only from these groups:

Nextcloud will generate a list of available groups found in your LDAP server. Then you select the group or groups that get access to your Nextcloud server.

Edit LDAP Query:

Clicking on this text toggles the filter mode and you can enter the raw LDAP filter directly.

Example:

- *objectClass=group*

- *objectClass=posixGroup*

y groups found:

This tells you approximately how many groups will be available in Nextcloud. The number updates automatically after any change.

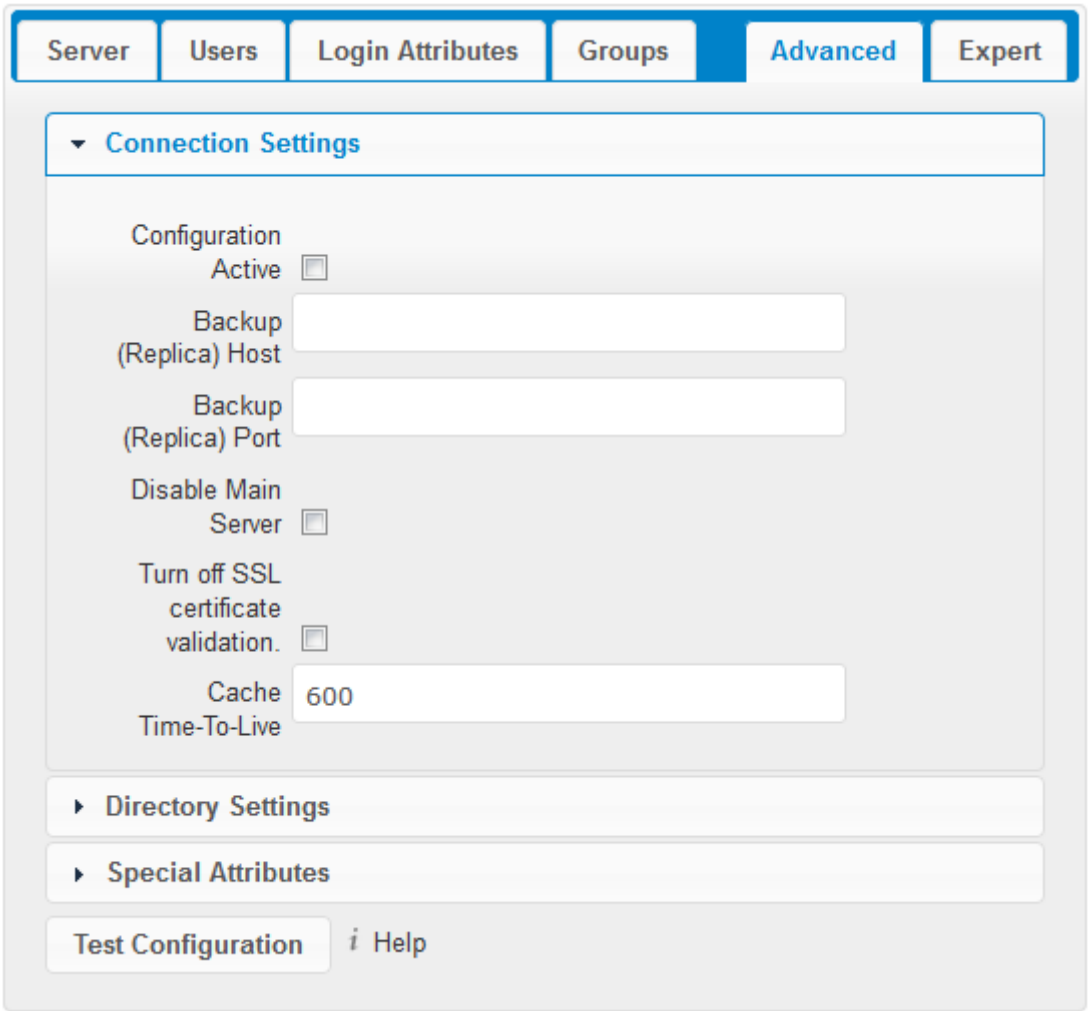
Advanced settings

The LDAP Advanced Setting section contains options that are not needed for a working connection. This provides controls to disable the current configuration, configure replica hosts, and various performance-enhancing options.

The Advanced Settings are structured into three parts:

- Connection Settings
- Directory Settings
- Special Attributes

Connection settings



Configuration Active:

Enables or Disables the current configuration. By default, it is turned off. When Nextcloud makes a successful test connection it is automatically turned on.

Backup (Replica) Host:

If you have a backup LDAP server, enter the connection settings here. Nextcloud will then automatically connect to the backup when the main server cannot be reached. The backup server must be a replica of the main server so that the object UUIDs match.

Example:

- *directory2.my-company.com*

Backup (Replica) Port:

The connection port of the backup LDAP server. If no port is given, but only a host, then the main port (as specified above) will be used.

Example:

- *389*

Disable Main Server:

You can manually override the main server and make Nextcloud only connect to the backup server. This is useful for planned downtimes.

Turn off SSL certificate validation:

Turns off SSL certificate checking. Use it for testing only! *Note*: The effect of this setting depdends on the PHP system configuration. It does for example not work with the [official Nextcloud container image](https://github.com/nextcloud/docker). To disable certificate verification for a particular use, append the following configuration line to your */etc/ldap/ldap.conf*:

```
TLS_REQCERT ALLOW
```

Cache Time-To-Live:

A cache is introduced to avoid unnecessary LDAP traffic, for example caching usernames so they don't have to be looked up for every page, and speeding up loading of the Users page. Saving the configuration empties the cache. The time is given in seconds.

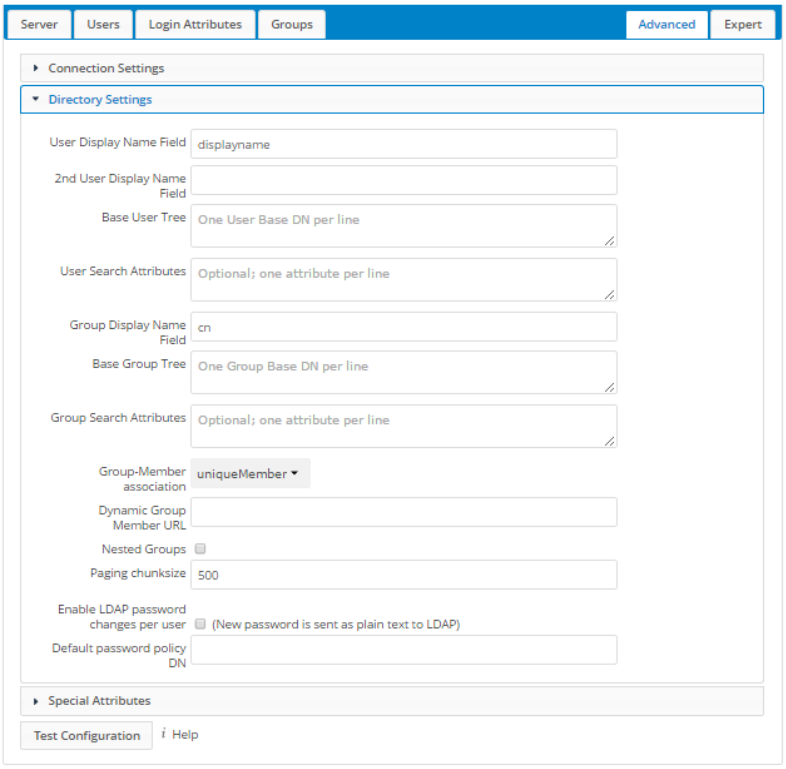
Note that almost every PHP request requires a new connection to the LDAP server. If you require fresh PHP requests we recommend defining a minimum lifetime of 15s or so, rather than completely eliminating the cache.

Examples:

- ten minutes: *600*
- one hour: *3600*

See the Caching section below for detailed information on how the cache operates.

Directory settings



User Display Name Field:

The attribute that should be used as display name in Nextcloud.

- Example: *displayName*

2nd User Display Name Field:

An optional second attribute displayed in brackets after the display name, for example using the `mail` attribute displays as `Molly Foo (molly@example.com)`.

Base User Tree:

The base DN of LDAP, from where all users can be reached. This must be a complete DN, regardless of what you have entered for your Base DN in the Basic setting. You can specify multiple base trees, one on each line.

- Example:

  cn=programmers,dc=my-company,dc=com
  cn=designers,dc=my-company,dc=com

User Search Attributes:

These attributes are used when searches for users are performed, for example in the share dialogue. The user display name attribute is the default. You may list multiple attributes, one per line.

If an attribute is not available on a user object, the user will not be listed, and will be unable to login. This also affects the display name attribute. If you override the default you must specify the display name attribute here.

- Example:

  displayName
  mail

Group Display Name Field:

The attribute that should be used as Nextcloud group name. Nextcloud allows a limited set of characters (a-zA-Z0-9.-_@). Once a group name is assigned it cannot be changed.

- Example: *cn*

Base Group Tree:

The base DN of LDAP, from where all groups can be reached. This must be a complete DN, regardless of what you have entered for your Base DN in the Basic setting. You can specify multiple base trees, one in each line.

- Example:

  cn=barcelona,dc=my-company,dc=com
  cn=madrid,dc=my-company,dc=com

Group Search Attributes:

These attributes are used when a search for groups is done, for example in the share dialogue. By default the group display name attribute as specified above is used. Multiple attributes can be given, one in each line.

If you override the default, the group display name attribute will not be taken into account, unless you specify it as well.

- Example:

  cn
  description

Group Member association:

The attribute that is used to indicate group memberships, i.e. the attribute used by LDAP groups to refer to their users.

Nextcloud detects the value automatically. You should only change it if you have a very valid reason and know what you are doing.

- Example: *uniquemember*

Nested groups:

Enable group member retrieval from sub groups.

To allow user listing and login from nested groups, please see **User listing and login per nested groups** in the section **Troubleshooting, Tips and Tricks**.

Enable LDAP password changes per user:

Allow LDAP users to change their password and allow Super Administrators and Group Administrators to change the password of their LDAP users.

To enable this feature, the following requirements have to be met:

- General requirements:
  - Access control policies must be configured on the LDAP server to grant permissions for password changes. The User DN as configured in *Server Settings* needs to have write permissions in order to update the userPassword attribute.
  - Passwords are sent in plaintext to the LDAP server. Therefore, transport encryption must be used for the communication between Nextcloud and the LDAP server, e.g. employ LDAPS.
  - Enabling password hashing on the LDAP server is highly recommended. While Active Directory stores passwords in a one-way format by default, OpenLDAP users could configure the `ppolicy_hash_cleartext` directive of the ppolicy overlay that ships with OpenLDAP.
- Additional requirements for Active Directory:
  - At least a 128-bit transport encryption must be used for the communication between Nextcloud and the LDAP server.
  - Make sure that the `fUserPwdSupport` char of the dSHeuristics is configured to employ the `userPassword` attribute as `unicodePwd` alias. While this is set accordingly on AD LDS by default, this is not the case on AD DS.

Default password policy DN:

This feature requires OpenLDAP with ppolicy. The DN of a default password policy will be used for password expiry handling in the absence of any user specific password policy. Password expiry handling features the following:

- When a LDAP password is about to expire, display a warning message to the user showing the number of days left before it expires. Password expiry warnings are displayed through the notifications app for Nextcloud.
- Prompt LDAP users with expired passwords to reset their password during login, provided that an adequate number of grace logins is still available.

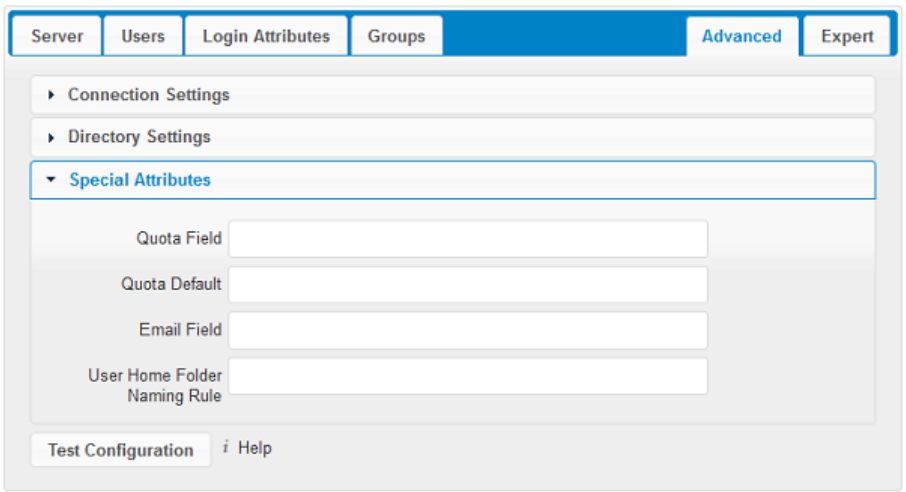
Leave the setting empty to keep password expiry handling disabled.

For the password expiry handling feature to work, LDAP password changes per user must be enabled and the LDAP server must be running OpenLDAP with its ppolicy module configured accordingly.

- Example:

  cn=default,ou=policies,dc=my-company,dc=com

Special attributes



Quota Field:

Nextcloud can read an LDAP attribute and set the user quota according to its value. Specify the attribute here, and it will return human-readable values, e.g. “2 GB”. Any quota set in LDAP overrides quotas set on the Nextcloud user management page.

- Example: *NextcloudQuota*

Quota Default:

Override Nextcloud default quota for LDAP users who do not have a quota set in the Quota Field.

- Example: *15 GB*

Email Field:

Set the user’s email from their LDAP attribute. Leave it empty for default behavior.

- Example: *mail*

User Home Folder Naming Rule:

By default, the Nextcloud server creates the user directory in your Nextcloud data directory and gives it the Nextcloud username, .e.g `/var/www/nextcloud/data/alice`. You may want to override this setting and name it after an LDAP attribute value. The attribute can also return an absolute path, e.g. `/mnt/storage43/alice`. Leave it empty for default behavior.

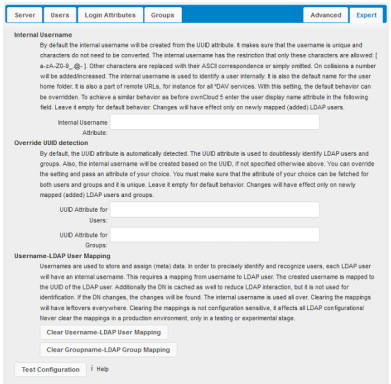
- Example: *cn*

In new Nextcloud installations the home folder rule is enforced. This means that once you set a home folder naming rule (get a home folder from an LDAP attribute), it must be available for all users. If it isn’t available for a user, then that user will not be able to login. Also, the filesystem will not be set up for that user, so their file shares will not be available to other users.

In migrated Nextcloud installations the old behavior still applies, which is using the Nextcloud username as the home folder when an LDAP attribute is not set. You may change this enforcing the home folder rule with the `occ` command in Nextcloud, like this example on Ubuntu:

```
sudo -u www-data php occ config:app:set user_ldap enforce_home_folder_naming_rule --value=1
```

Expert settings



In the Expert Settings fundamental behavior can be adjusted to your needs. The configuration should be well-tested before starting production use.

Internal Username:

The internal username is the identifier in Nextcloud for LDAP users. By default it will be created from the UUID attribute. The UUID attribute ensures that the username is unique, and that characters do not need to be converted. Only these characters are allowed: [a-zA-Z0-9_.@-]. Other characters are replaced with their ASCII equivalents, or are simply omitted.

The LDAP backend ensures that there are no duplicate internal usernames in Nextcloud, i.e. that it is checking all other activated user backends (including local Nextcloud users). On collisions a random number (between 1000 and 9999) will be attached to the retrieved value. For example, if “alice” exists, the next username may be “alice_1337”.

The internal username is the default name for the user home folder in Nextcloud. It is also a part of remote URLs, for instance for all *DAV services.

You can override all of this with the Internal Username setting. Leave it empty for default behavior. Changes will affect only newly mapped LDAP users.

When configuring this, be aware that the username in Nextcloud is considered immutable and cannot be changed afterwards. This can cause issues when using an attribute that might change, e.g. the email address of a user that will get changed during name change.

- Example: *uid*

Override UUID detection

By default, Nextcloud auto-detects the UUID attribute. The UUID attribute is used to uniquely identify LDAP users and groups. The internal username will be created based on the UUID, if not specified otherwise.

You can override the setting and pass an attribute of your choice. You must make sure that the attribute of your choice can be fetched for both users and groups and it is unique. Leave it empty for default behavior. Changes will have effect only on newly mapped LDAP users and groups. It also will have effect when a user's or group's DN changes and an old UUID was cached, which will result in a new user. Because of this, the setting should be applied before putting Nextcloud in production use and clearing the bindings (see the `User and Group Mapping` section below).

- Example: *cn*

Username-LDAP User Mapping

Nextcloud uses usernames as keys to store and assign data. In order to precisely identify and recognize users, each LDAP user will have a internal username in Nextcloud. This requires a mapping from Nextcloud username to LDAP user. The created username is mapped to the UUID of the LDAP user. Additionally the DN is cached as well to reduce LDAP interaction, but it is not used for identification. If the DN changes, the change will be detected by Nextcloud by checking the UUID value.

The same is valid for groups.

The internal Nextcloud name is used all over in Nextcloud. Clearing the Mappings will have leftovers everywhere. Never clear the mappings in a production environment, but only in a testing or experimental server.

Warning

Clearing the Mappings is not configuration sensitive, it affects all LDAP configurations!

Testing the configuration

The **Test Configuration** button checks the values as currently given in the input fields. You do not need to save before testing. By clicking on the button, Nextcloud will try to bind to the Nextcloud server using the settings currently given in the input fields. If the binding fails you'll see a yellow banner with the error message "The configuration is invalid. Please have a look at the logs for further details."

When the configuration test reports success, save your settings and check if the users and groups are fetched correctly on the Users page.

Additional configuration options via occ

Few configuration settings can only be set on command line via `occ`.

Attribute update interval

The LDAP backend will update user information that is used within Nextcloud with the values provided by the LDAP server. For instance these are email, quota or the avatar. This happens on every login, the first detection of a user from LDAP and regularly by a background job.

The interval value determines the time between updates of the values and is used to avoid frequent overhead, including time-expensive write actions to the database.

The interval is described in seconds and it defaults to 86400 equalling a day. It is not a per-configuration option.

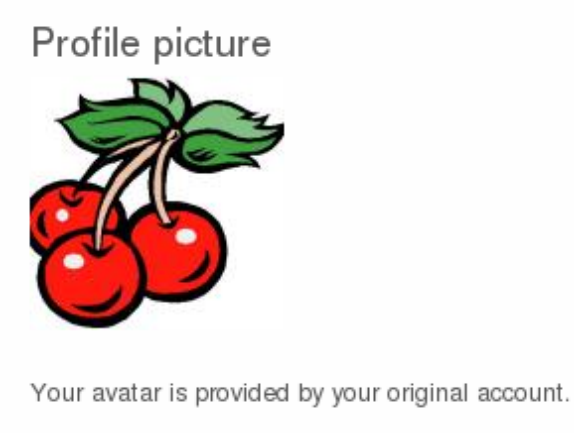
The value can be modified by:

```
sudo -u www-data php occ config:app:set user_ldap updateAttributesInterval --value=86400
```

A value of 0 will update it on every of the named occasions.

Nextcloud avatar integration

Nextcloud supports user profile pictures, which are also called avatars. If a user has a photo stored in the *jpegPhoto* or *thumbnailPhoto* attribute on your LDAP server, it will be used as their avatar. In this case the user cannot alter their avatar (on their Personal page) as it must be changed in LDAP. *jpegPhoto* is preferred over *thumbnailPhoto*.



If the *jpegPhoto* or *thumbnailPhoto* attribute is not set or empty, then users can upload and manage their avatars on their Nextcloud Personal pages. Avatars managed in Nextcloud are not stored in LDAP.

The *jpegPhoto* or *thumbnailPhoto* attribute is fetched once a day to make sure the current photo from LDAP is used in Nextcloud. LDAP avatars override Nextcloud avatars, and when an LDAP avatar is deleted then the most recent Nextcloud avatar replaces it.

Photos served from LDAP are automatically cropped and resized in Nextcloud. This affects only the presentation, and the original image is not changed.

Use a specific attribute or turn off loading of images

It is possible to turn off the avatar integration or specify a single, different attribute to read the image from. It is expected to contain image data just like *jpegPhoto* or *thumbnailPhoto* do.

The behaviour can be changed using the occ command line tool only. Essentially those options are available:

- The default behaviour as described above should be used

  ```
  occ ldap:set-config "s01" "ldapUserAvatarRule" "default"
  ```

- User images shall not be fetched from LDAP

  ```
  occ ldap:set-config "s01" "ldapUserAvatarRule" "none"
  ```

- The image should be read from the attribute “selfiePhoto”

  ```
  occ ldap:set-config "s01" "ldapUserAvatarRule" "data:selfiePhoto"
  ```

The “s01” refers to the configuration ID as can be retrieved per `occ ldap:show-config`.

Troubleshooting, tips and tricks

SSL certificate verification (LDAPS, TLS)

A common mistake with SSL certificates is that they may not be known to PHP. If you have trouble with certificate validation make sure that

- You have the certificate of the server installed on the Nextcloud server

- The certificate is announced in the system's LDAP configuration file (usually */etc/ldap/ldap.conf*)
- Using LDAPS, also make sure that the port is correctly configured (by default 636)

Microsoft Active Directory

Compared to earlier Nextcloud versions, no further tweaks need to be done to make Nextcloud work with Active Directory. Nextcloud will automatically find the correct configuration in the set-up process.

memberOf / read memberof permissions

If you want to use `memberOf` within your filter you might need to give your querying user the permissions to use it. For Microsoft Active Directory this is described here.

User listing and login per nested groups

When it is intended to allow user listing and login based on a specific group having subgroups ("nested groups"), checking **Nested groups** on **Directory Settings** is not enough. Also the User (and Login) filter need to be changed, by specifying the `LDAP_MATCHING_RULE_IN_CHAIN` matching rule. Change the filter parts containing the *memberof* condition according to this example:

- (memberof=cn=Nextcloud Users Group,ou=Groups,…)

to

- (memberof:1.2.840.113556.1.4.1941:=cn=Nextcloud Users Group,ou=Groups,…)

Duplicating server configurations

In case you have a working configuration and want to create a similar one or "snapshot" configurations before modifying them you can do the following:

1. Go to the **Server** tab
2. On **Server Configuration** choose *Add Server Configuration*
3. Answer the question *Take over settings from recent server configuration?* with *yes*.
4. (optional) Switch to **Advanced** tab and uncheck **Configuration Active** in the *Connection Settings*, so the new configuration is not used on Save
5. Click on **Save**

Now you can modify and enable the configuration.

Nextcloud LDAP internals

Some parts of how the LDAP backend works are described here.

User and group mapping

In Nextcloud the user or group name is used to have all relevant information in the database assigned. To work reliably a permanent internal user name and group name is created and mapped to the LDAP DN and UUID. If the DN changes in LDAP it will be detected, and there will be no conflicts.

Those mappings are done in the database table `ldap_user_mapping` and `ldap_group_mapping`. The user name is also used for the user's folder (except if something else is specified in *User Home Folder Naming Rule*), which contains files and meta data.

The internal user name and a visible display name are separated. This is not the case for group names, yet, i.e. a group name cannot be altered.

That means that your LDAP configuration should be good and ready before putting it into production. The mapping tables are filled early, but as long as you are testing, you can empty the tables any time. Do not do this in production.

The attributes of users are fetched on demand (i.e. for sharing autocompletion or in the user management) and then stored inside the Nextcloud database to allow a better performance on our side. They are typically checked twice a

day in batches from all users again. Beside that they are also refreshed during a login for this user or can be fetched manually via the occ command `occ ldap:check-user --update USERID` where `USERID` is Nextcloud's user id.

Caching

The LDAP information is cached in Nextcloud memory cache, and you must install and configure the memory cache (see Memory caching). The Nextcloud **Cache** helps to speed up user interactions and sharing. It is populated on demand, and remains populated until the **Cache Time-To-Live** for each unique request expires. User logins are not cached, so if you need to improve login times set up a slave LDAP server to share the load.

You can adjust the **Cache Time-To-Live** value to balance performance and freshness of LDAP data. All LDAP requests will be cached for 10 minutes by default, and you can alter this with the **Cache Time-To-Live** setting. The cache answers each request that is identical to a previous request, within the time-to-live of the original request, rather than hitting the LDAP server.

The **Cache Time-To-Live** is related to each single request. After a cache entry expires there is no automatic trigger for re-populating the information, as the cache is populated only by new requests, for example by opening the User administration page, or searching in a sharing dialog.

There is one trigger which is automatically triggered by a certain background job which keeps the `user-group-mappings` up-to-date, and always in cache.

Under normal circumstances, all users are never loaded at the same time. Typically the loading of users happens while page results are generated, in steps of 30 until the limit is reached or no results are left. For this to work on a Nextcloud-Server and LDAP-Server, **Paged Results** must be supported.

Nextcloud remembers which user belongs to which LDAP-configuration. That means each request will always be directed to the right server unless a user is defunct, for example due to a server migration or unreachable server. In this case the other servers will also receive the request.

Handling with backup server

When Nextcloud is not able to contact the main LDAP server, Nextcloud assumes it is offline and will not try to connect again for the time specified in **Cache Time-To-Live**. If you have a backup server configured Nextcloud will connect to it instead. When you have scheduled downtime, check **Disable Main Server** to avoid unnecessary connection attempts.

Note

When a LDAP object's name or surname, that is display name attribute, by default "displayname", is left empty, Nextcloud will treat it as an empty object, therefore no results from this user or AD-Object will be shown to avoid gathering of technical accounts.

LDAP user cleanup

LDAP User Cleanup is a new feature in the `LDAP user and group backend` application. LDAP User Cleanup is a background process that automatically searches the Nextcloud LDAP mappings table, and verifies if the LDAP users are still available. Any users that are not available are marked as `deleted` in the `oc_preferences` database table. Then you can run a command to display this table, displaying only the users marked as `deleted`, and then you have the option of removing their data from your Nextcloud data directory.

These items are removed upon cleanup:

- Local Nextcloud group assignments
- User preferences (DB table `oc_preferences`)
- User's Nextcloud home folder
- User's corresponding entry in `oc_storages`

There are two prerequisites for LDAP User Cleanup to operate:

1. Set `ldapUserCleanupInterval` in `config.php` to your desired check interval in minutes. The default is 51 minutes.

2. All configured LDAP connections are enabled and operating correctly. As users can exist on multiple LDAP servers, you want to be sure that all of your LDAP servers are available so that a user on a temporarily disconnected LDAP server is not marked as `deleted`.

The background process examines 50 users at a time, and runs at the interval you configured with `ldapUserCleanupInterval`. For example, if you have 200 LDAP users and your `ldapUserCleanupInterval` is 20 minutes, the process will examine the first 50 users, then 20 minutes later the next 50 users, and 20 minutes later the next 50, and so on.

The amount of users to check can be set to a custom value via occ command. The following example sets it to 300:

```
sudo -u www-data php occ config:app:set --value=300 user_ldap cleanUpJobChunkSize
```

There are two `occ` commands to use for examining a table of users marked as deleted, and then manually deleting them. The `occ` command is in your Nextcloud directory, for example `/var/www/nextcloud/occ`, and it must be run as your HTTP user. To learn more about `occ`, see Using the occ command.

These examples are for Ubuntu Linux:

1. `sudo -u www-data php occ ldap:show-remnants` displays a table with all users that have been marked as deleted, and their LDAP data.
2. `sudo -u www-data php occ user:delete [user]` removes the user's data from the Nextcloud data directory.

This example shows what the table of users marked as `deleted` looks like:

```
$ sudo -u www-data php occ ldap:show-remnants
+-----------------+-----------------+------------------+-----------------------------------
| Nextcloud name  | Display Name    | LDAP UID         | LDAP DN
+-----------------+-----------------+------------------+-----------------------------------
| aaliyah_brown   | aaliyah brown   | aaliyah_brown    | uid=aaliyah_brown,ou=people,dc=com
| aaliyah_hammes  | aaliyah hammes  | aaliyah_hammes   | uid=aaliyah_hammes,ou=people,dc=com
| aaliyah_johnston| aaliyah johnston| aaliyah_johnston | uid=aaliyah_johnston,ou=people,dc=c
| aaliyah_kunze   | aaliyah kunze   | aaliyah_kunze    | uid=aaliyah_kunze,ou=people,dc=com
+-----------------+-----------------+------------------+-----------------------------------
```

Following flags can be specified additionally:

- `--short-date`: formats the dates for `Last login` and `Detected on` in a short Y-m-d format (e.g. 2019-01-14)
- `--json`: instead of a table, the output is json-encoded. This makes it easy to process the data programmatically.

Then you can run `sudo -u www-data php occ user:delete aaliyah_brown` to delete user aaliyah_brown. You must use the user's Nextcloud name.

Deleting local Nextcloud users

You may also use `occ user:delete [user]` to remove a local Nextcloud user; this removes their user account and their data.

The LDAP configuration API

All methods require that the "OCS-APIREQUEST" header be set to "true". Methods take an optional "format" parameter, which may be "xml" (the default) or "json".

Creating a configuration

Creates a new and empty LDAP configuration. It returns its ID. Authentication is done by sending a basic HTTP authentication header.

Syntax: ocs/v2.php/apps/user_ldap/api/v1/config

- HTTP method: POST

Example

```
$ curl -X POST https://admin:secret@example.com/ocs/v2.php/apps/user_ldap/api/v1/config -H "
```

- Creates a new, empty configuration

XML output

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message>OK</message>
 </meta>
 <data>
  <configID>s01</configID>
 </data>
</ocs>
```

Deleting a configuration

Deletes a given LDAP configuration. Authentication is done by sending a basic HTTP authentication header.

Syntax: ocs/v2.php/apps/user_ldap/api/v1/config/{configID}

- HTTP method: DELETE

Example

```
$ curl -X DELETE ``https://admin:secret@example.com/ocs/v2.php/apps/user_ldap/api/v1/config/
```

- deletes the LDAP configuration

XML output

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message>OK</message>
 </meta>
 <data/>
</ocs>
```

Reading a configuration

Returns all keys and values of the specified LDAP configuration. Authentication is done by sending a basic HTTP authentication header.

Syntax: ocs/v2.php/apps/user_ldap/api/v1/config/{configID}

- HTTP method: GET
- url argument: showPassword - int, optional, default 0, whether to return the password in clear text

Example

```
$ curl -X GET https://admin:secret@example.com/ocs/v2.php/apps/user_ldap/api/v1/config/s02?s
```

- fetches the LDAP configuration

XML output

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message>OK</message>
 </meta>
 <data>
  <ldapHost>ldap://ldap.server.tld</ldapHost>
  <ldapPort>389</ldapPort>
  <ldapBackupHost></ldapBackupHost>
  <ldapBackupPort></ldapBackupPort>
  <ldapBase>ou=Department XLII,dc=example,dc=com</ldapBase>
  <ldapBaseUsers>ou=users,ou=Department XLII,dc=example,dc=com</ldapBaseUsers>
  <ldapBaseGroups>ou=Department XLII,dc=example,dc=com</ldapBaseGroups>
  <ldapAgentName>cn=root,dc=example,dc=com</ldapAgentName>
  <ldapAgentPassword>Secret</ldapAgentPassword>
  <ldapTLS>1</ldapTLS>
  <turnOffCertCheck>0</turnOffCertCheck>
  <ldapIgnoreNamingRules/>
  <ldapUserDisplayName>displayname</ldapUserDisplayName>
  <ldapUserDisplayName2>uid</ldapUserDisplayName2>
  <ldapGidNumber>gidNumber</ldapGidNumber>
  <ldapUserFilterObjectclass>inetOrgPerson</ldapUserFilterObjectclass>
  <ldapUserFilterGroups></ldapUserFilterGroups>
  <ldapUserFilter>(&amp;(objectclass=nextcloudUser)(nextcloudEnabled=TRUE))</ldapUserFilter>
  <ldapUserFilterMode>1</ldapUserFilterMode>
  <ldapGroupFilter>(&amp;(|(objectclass=nextcloudGroup)))</ldapGroupFilter>
  <ldapGroupFilterMode>0</ldapGroupFilterMode>
  <ldapGroupFilterObjectclass>nextcloudGroup</ldapGroupFilterObjectclass>
  <ldapGroupFilterGroups></ldapGroupFilterGroups>
  <ldapGroupMemberAssocAttr>memberUid</ldapGroupMemberAssocAttr>
  <ldapGroupDisplayName>cn</ldapGroupDisplayName>
  <ldapLoginFilter>(&amp;(|(objectclass=inetOrgPerson))(uid=%uid))</ldapLoginFilter>
  <ldapLoginFilterMode>0</ldapLoginFilterMode>
  <ldapLoginFilterEmail>0</ldapLoginFilterEmail>
  <ldapLoginFilterUsername>1</ldapLoginFilterUsername>
  <ldapLoginFilterAttributes></ldapLoginFilterAttributes>
  <ldapQuotaAttribute></ldapQuotaAttribute>
  <ldapQuotaDefault>20 MB</ldapQuotaDefault>
  <ldapEmailAttribute>mail</ldapEmailAttribute>
  <ldapCacheTTL>600</ldapCacheTTL>
  <ldapUuidUserAttribute>auto</ldapUuidUserAttribute>
  <ldapUuidGroupAttribute>auto</ldapUuidGroupAttribute>
  <ldapOverrideMainServer></ldapOverrideMainServer>
  <ldapConfigurationActive>1</ldapConfigurationActive>
  <ldapAttributesForUserSearch>uid;sn;givenname</ldapAttributesForUserSearch>
  <ldapAttributesForGroupSearch></ldapAttributesForGroupSearch>
  <ldapExperiencedAdmin>0</ldapExperiencedAdmin>
  <homeFolderNamingRule>attr:mail</homeFolderNamingRule>
  <hasPagedResultSupport></hasPagedResultSupport>
  <hasMemberOfFilterSupport>1</hasMemberOfFilterSupport>
  <useMemberOfToDetectMembership>1</useMemberOfToDetectMembership>
  <ldapExpertUsernameAttr></ldapExpertUsernameAttr>
  <ldapExpertUUIDUserAttr></ldapExpertUUIDUserAttr>
  <ldapExpertUUIDGroupAttr></ldapExpertUUIDGroupAttr>
  <lastJpegPhotoLookup>0</lastJpegPhotoLookup>
  <ldapNestedGroups>0</ldapNestedGroups>
```

```
   <ldapPagingSize>500</ldapPagingSize>
   <turnOnPasswordChange>1</turnOnPasswordChange>
   <ldapDynamicGroupMemberURL></ldapDynamicGroupMemberURL>
   <ldapDefaultPPolicyDN></ldapDefaultPPolicyDN>
  </data>
 </ocs>
```

Modifying a configuration

Updates a configuration with the provided values. Authentication is done by sending a basic HTTP authentication header.

Syntax: ocs/v2.php/apps/user_ldap/api/v1/config/{configID}

- HTTP method: PUT
- url argument: configData - array, see table below for the fields. All fields are optional. The values must be url-encoded.

Example

```
$ curl -X PUT https://admin:secret@example.com/ocs/v2.php/apps/user_ldap/api/v1/config/s01 -
```

- updates the LDAP configuration

XML output

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message>OK</message>
 </meta>
 <data/>
</ocs>
```

Configuration keys

Key	Mode	Required	Description
ldapHost	rw	yes	LDAP server host, supports protocol
ldapPort	rw	yes	LDAP server port
ldapBackupHost	rw	no	LDAP replica host
ldapBackupPort	rw	no	LDAP replica port
ldapOverrideMainServer	rw	no	Whether replica should be used instead
ldapBase	rw	yes	Base

ldapBaseUsers	rw	no	Base for users, defaults to general base if not specified
ldapBaseGroups	rw	no	Base for groups, defaults to general base if not specified
ldapAgentName	rw	no	DN for the (service) user to connect to LDAP
ldapAgentPassword	rw	no	Password for the service user
ldapTLS	rw	no	Whether to use StartTLS
turnOffCertCheck	rw	no	Turns off certificate validation for TLS connections
ldapIgnoreNamingRules	rw	no	Backwards compatibility, do not set it.
ldapUserDisplayName	rw	yes	Attribute used as display name for users
ldapUserDisplayName2	rw	no	Additional attribute, if set show on brackets next to the main attribute
ldapUserAvatarRule	rw	no	Specify the avatar integration behavior, possible values: “default”, “none”, “data:$ATTRIBUTENAME”
ldapGidNumber	rw	no	group ID attribute, needed for primary groups on OpenLDAP (and compatible)
ldapUserFilterObjectclass	rw	no	set by the Settings Wizard (web UI)
ldapUserFilterGroups	rw	no	set by the Settings Wizard (web UI)
ldapUserFilter	rw	yes	LDAP Filter used to retrieve user
ldapUserFilterMode	rw	no	used by the Settings Wizard, set to 1 for manual editing
ldapAttributesForUserSearch	rw	no	attributes to be matched when searching for users. separate by ;
ldapGroupFilter	rw	no	LDAP Filter used to retrieve groups
ldapGroupFilterMode	rw	no	used by the Settings Wizard, set to 1 for manual editing
ldapGroupFilterObjectclass	rw	no	set by the Settings Wizard (web UI)
ldapGroupFilterGroups	rw	no	set by the Settings Wizard (web UI)
ldapGroupMemberAssocAttr	rw	no	attribute that indicates group members, one of: member, memberUid, uniqueMember, gidNumber
ldapGroupDisplayName	rw	no	Attribute used as display name for groups, required if groups are used
ldapAttributesForGroupSearch	rw	no	attributes to be matched when searching for groups. separate by ;
ldapLoginFilter	rw	yes	LDAP Filter used to authenticate users

ldapLoginFilterMode	rw	no	used by the Settings Wizard, set to 1 for manual editing
ldapLoginFilterEmail	rw	no	set by the Settings Wizard (web UI)
ldapLoginFilterUsername	rw	no	set by the Settings Wizard (web UI)
ldapLoginFilterAttributes	rw	no	set by the Settings Wizard (web UI)
ldapQuotaAttribute	rw	no	LDAP attribute containing the quote value (per user)
ldapQuotaDefault	rw	no	Default Quota, if specified quota attribute is empty
ldapEmailAttribute	rw	no	LDAP attribute containing the email address (takes first if multiple are stored)
ldapCacheTTL	rw	no	How long results from LDAP are cached, defaults to 10min
ldapUuidUserAttribute	r	no	set in runtime
ldapUuidGroupAttribute	r	no	set in runtime
ldapConfigurationActive	rw	no	whether this configuration is active. 1 is on, 0 is off.
ldapExperiencedAdmin	rw	no	used by the Settings Wizard, set to 1 for manual editing
homeFolderNamingRule	rw	no	LDAP attribute to use a user folder name
hasPagedResultSupport	r	no	set in runtime
hasMemberOfFilterSupport	r	no	set in runtime
useMemberOfToDetectMembership	rw	no	Whether to use memberOf to detect group memberships
ldapExpertUsernameAttr	rw	no	LDAP attribute to use as internal username. Might be modified (e.g. to avoid name collisions, character restrictions)
ldapExpertUUIDUserAttr	rw	no	override the LDAP servers UUID attribute to identify LDAP user records
ldapExpertUUIDGroupAttr	rw	no	override the LDAP servers UUID attribute to identify LDAP group records
lastJpegPhotoLookup	r	no	set in runtime
ldapNestedGroups	rw	no	Whether LDAP supports nested groups
ldapPagingSize	rw	no	Number of results to return per page
turnOnPasswordChange	rw	no	Whether users are allowed to change passwords (hashing must happen on LDAP!)
ldapDynamicGroupMemberURL	rw	no	URL for dynamic groups

ldapDefaultPPolicyDN	rw	no	PPolicy DN for password rules

User provisioning API

The Provisioning API application enables a set of APIs that external systems can use to create, edit, delete and query user attributes, query, set and remove groups, set quota and query total storage used in Nextcloud. Group admin users can also query Nextcloud and perform the same functions as an admin for groups they manage. The API also enables an admin to query for active Nextcloud applications, application info, and to enable or disable an app remotely. HTTP requests can be used via a Basic Auth header to perform any of the functions listed above. The Provisioning API app is enabled by default.

The base URL for all calls to the share API is `https://cloud.example.com/ocs/v1.php/cloud`.

All calls to OCS endpoints require the `OCS-APIRequest` header to be set to `true`.

All POST requests require the `Content-Type: application/x-www-form-urlencoded` header. (Note: Some libraries like cURL set this header automatically, others require setting the header explicitly.)

Instruction set for users

Add a new user

Create a new user on the Nextcloud server. Authentication is done by sending a basic HTTP authentication header.

Syntax: ocs/v1.php/cloud/users

- HTTP method: POST
- POST argument: userid - string, the required username for the new user
- POST argument: password - string, the password for the new user, leave empty to send welcome mail
- POST argument: displayName - string, the display name for the new user
- POST argument: email - string, the email for the new user, required if password empty
- POST argument: groups - array, the groups for the new user
- POST argument: subadmin - array, the groups in which the new user is subadmin
- POST argument: quota - string, quota for the new user
- POST argument: language - string, language for the new user

Status codes:

- 100 - successful
- 101 - invalid input data
- 102 - username already exists
- 103 - unknown error occurred whilst adding the user
- 104 - group does not exist
- 105 - insufficient privileges for group
- 106 - no group specified (required for subadmins)
- 107 - all errors that contain a hint - for example “Password is among the 1,000,000 most common ones. Please make it unique.” (this code was added in 12.0.6 & 13.0.1)
- 108 - password and email empty. Must set password or an email
- 109 - invitation email cannot be send

Example

```
$ curl -X POST http://admin:secret@example.com/ocs/v1.php/cloud/users -d userid="Frank" -d p
```

- Creates the user `Frank` with password `frankspassword`
- optionally groups can be specified by one or more `groups[]` query parameters: `URL -d groups[]="admin" -D groups[]="Team1"`

XML output

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>100</statuscode>
  <message/>
 </meta>
 <data/>
</ocs>
```

Search/get users

Retrieves a list of users from the Nextcloud server. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users

- HTTP method: GET
- url arguments: search - string, optional search string
- url arguments: limit - int, optional limit value
- url arguments: offset - int, optional offset value

Status codes:

- 100 - successful

Example

```
$ curl -X GET http://admin:secret@example.com/ocs/v1.php/cloud/users?search=Frank
```

- Returns list of users matching the search string.

XML output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>100</statuscode>
    <status>ok</status>
  </meta>
  <data>
    <users>
      <element>Frank</element>
     </users>
  </data>
</ocs>
```

Get data of a single user

Retrieves information about a single user. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}

- HTTP method: GET

Status codes:

- 100 - successful

Example

```
$ curl -X GET http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank
```

- Returns information on the user `Frank`

XML output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>100</statuscode>
    <status>ok</status>
  </meta>
  <data>
    <enabled>true</enabled>
    <id>Frank</id>
    <quota>0</quota>
    <email>frank@example.org</email>
    <displayname>Frank K.</displayname>
    <phone>0123 / 456 789</phone>
    <address>Foobar 12, 12345 Town</address>
    <website>https://nextcloud.com</website>
    <twitter>Nextcloud</twitter>
    <groups>
     <element>group1</element>
     <element>group2</element>
    </groups>
  </data>
</ocs>
```

Edit data of a single user

Edits attributes related to a user. Users are able to edit email, displayname and password; admins can also edit the quota value. Further restrictions may apply, check the List of editable data fields endpoint. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}

- HTTP method: PUT
- PUT argument: key, the field to edit:
  - email
  - quota
  - displayname
  - display (**deprecated** use *displayname* instead)
  - phone
  - address
  - website
  - twitter
  - password
- PUT argument: value, the new value for the field

Status codes:

- 100 - successful
- 101 - user not found
- 102 - invalid input data

Examples

```
$ curl -X PUT http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank -d key="email" -d
```

- Updates the email address for the user `Frank`

```
$ curl -X PUT http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank -d key="quota" -d
```

- Updates the quota for the user `Frank`

XML output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>100</statuscode>
    <status>ok</status>
  </meta>
  <data/>
</ocs>
```

List of editable data fields

Edits attributes related to a user. Users are able to edit email, displayname and password; admins can also edit the quota value. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/user/fields

- HTTP method: GET

Status codes:

- 100 - successful

Examples

```
$ curl -X GET http://admin:secret@example.com/ocs/v1.php/cloud/user/fields
```

- Gets the list of fields

XML output

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>100</statuscode>
  <message>OK</message>
 </meta>
 <data>
  <element>displayname</element>
  <element>email</element>
  <element>phone</element>
  <element>address</element>
  <element>website</element>
  <element>twitter</element>
 </data>
</ocs>
```

Disable a user

Disables a user on the Nextcloud server so that the user cannot login anymore. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}/disable

- HTTP method: PUT

Statuscodes:

- 100 - successful
- 101 - failure

Example

```
$ curl -X PUT http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/disable
```

- Disables the user `Frank`

XML output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <status>ok</status>
    <statuscode>100</statuscode>
    <message/>
  </meta>
  <data/>
</ocs>
```

Enable a user

Enables a user on the Nextcloud server so that the user can login again. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}/enable

- HTTP method: PUT

Statuscodes:

- 100 - successful

- 101 - failure

Example

```
$ curl -X PUT http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/enable
```

- Enables the user `Frank`

XML output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <status>ok</status>
    <statuscode>100</statuscode>
    <message/>
  </meta>
  <data/>
</ocs>
```

Delete a user

Deletes a user from the Nextcloud server. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}

- HTTP method: DELETE

Statuscodes:

- 100 - successful
- 101 - failure

Example

```
$ curl -X DELETE http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank
```

- Deletes the user `Frank`

XML output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>100</statuscode>
    <status>ok</status>
  </meta>
  <data/>
</ocs>
```

Get user's groups

Retrieves a list of groups the specified user is a member of. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}/groups

- HTTP method: GET

Status codes:

- 100 - successful

Example

```
$ curl -X GET http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/groups
```

- Retrieves a list of groups of which `Frank` is a member

XML output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>100</statuscode>
    <status>ok</status>
  </meta>
  <data>
    <groups>
      <element>admin</element>
      <element>group1</element>
    </groups>
  </data>
</ocs>
```

Add user to group

Adds the specified user to the specified group. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}/groups

- HTTP method: POST
- POST argument: groupid, string - the group to add the user to

Status codes:

- 100 - successful
- 101 - no group specified
- 102 - group does not exist
- 103 - user does not exist
- 104 - insufficient privileges
- 105 - failed to add user to group

Example

```
$ curl -X POST http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/groups -d groupi
```

- Adds the user `Frank` to the group `newgroup`

XML output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>100</statuscode>
    <status>ok</status>
  </meta>
  <data/>
</ocs>
```

Remove user from group

Removes the specified user from the specified group. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}/groups

- HTTP method: DELETE
- DELETE argument: groupid, string - the group to remove the user from

Status codes:

- 100 - successful
- 101 - no group specified
- 102 - group does not exist
- 103 - user does not exist
- 104 - insufficient privileges
- 105 - failed to remove user from group

Example

```
$ curl -X DELETE http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/groups -d grou
```

- Removes the user `Frank` from the group `newgroup`

XML output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>100</statuscode>
    <status>ok</status>
  </meta>
  <data/>
</ocs>
```

Promote user to subadmin

Makes a user the subadmin of a group. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}/subadmins

- HTTP method: POST
- POST argument: groupid, string - the group of which to make the user a subadmin

Status codes:

- 100 - successful
- 101 - user does not exist
- 102 - group does not exist
- 103 - unknown failure

Example

```
$ curl -X POST https://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/subadmins -d gr
```

- Makes the user `Frank` a subadmin of the `group` group

XML output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>100</statuscode>
    <status>ok</status>
  </meta>
  <data/>
</ocs>
```

Demote user from subadmin

Removes the subadmin rights for the user specified from the group specified. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}/subadmins

- HTTP method: DELETE
- DELETE argument: groupid, string - the group from which to remove the user's subadmin rights

Status codes:

- 100 - successful
- 101 - user does not exist
- 102 - user is not a subadmin of the group / group does not exist
- 103 - unknown failure

Example

```
$ curl -X DELETE https://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/subadmins -d
```

- Removes `Frank's` subadmin rights from the `oldgroup` group

XML output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>100</statuscode>
    <status>ok</status>
  </meta>
  <data/>
</ocs>
```

Get user's subadmin groups

Returns the groups in which the user is a subadmin. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}/subadmins

- HTTP method: GET

Status codes:

- 100 - successful
- 101 - user does not exist
- 102 - unknown failure

Example

```
$ curl -X GET https://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/subadmins
```

- Returns the groups of which `Frank` is a subadmin

XML output

```
<?xml version="1.0"?>
<ocs>
  <meta>
      <status>ok</status>
      <statuscode>100</statuscode>
    <message/>
  </meta>
  <data>
    <element>testgroup</element>
  </data>
</ocs>
```

Resend the welcome email

The request to this endpoint triggers the welcome email for this user again.

Syntax: ocs/v1.php/cloud/users/{userid}/welcome

- HTTP method: POST

Status codes:

- 100 - successful
- 101 - email address not available
- 102 - sending email failed

Example

```
$ curl -X POST https://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/welcome
```

- Sends the welcome email to `Frank`

XML output

```
<?xml version="1.0"?>
<ocs>
  <meta>
      <status>ok</status>
      <statuscode>100</statuscode>
    <message/>
  </meta>
  <data/>
</ocs>
```

Instruction set for groups

Search/get groups

Retrieves a list of groups from the Nextcloud server. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/groups

- HTTP method: GET
- url arguments: search - string, optional search string
- url arguments: limit - int, optional limit value
- url arguments: offset - int, optional offset value

Status codes:

- 100 - successful

Example

```
$ curl -X GET http://admin:secret@example.com/ocs/v1.php/cloud/groups?search=adm
```

- Returns list of groups matching the search string.

XML output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>100</statuscode>
    <status>ok</status>
  </meta>
  <data>
    <groups>
      <element>admin</element>
    </groups>
  </data>
</ocs>
```

Create a group

Adds a new group. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/groups

- HTTP method: POST
- POST argument: groupid, string - the new groups name

Status codes:

- 100 - successful
- 101 - invalid input data
- 102 - group already exists
- 103 - failed to add the group

Example

```
$ curl -X POST http://admin:secret@example.com/ocs/v1.php/cloud/groups -d groupid="newgroup"
```

- Adds a new group called `newgroup`

XML output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>100</statuscode>
    <status>ok</status>
  </meta>
  <data/>
</ocs>
```

Get members of a group

Retrieves a list of group members. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/groups/{groupid}

- HTTP method: GET

Status codes:

- 100 - successful

Example

```
$ curl -X GET http://admin:secret@example.com/ocs/v1.php/cloud/groups/admin
```

- Returns a list of users in the `admin` group

XML output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>100</statuscode>
    <status>ok</status>
  </meta>
  <data>
    <users>
      <element>Frank</element>
    </users>
  </data>
</ocs>
```

Get subadmins of a group

Returns subadmins of the group. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/groups/{groupid}/subadmins

- HTTP method: GET

Status codes:

- 100 - successful
- 101 - group does not exist
- 102 - unknown failure

Example

```
$ curl -X GET https://admin:secret@example.com/ocs/v1.php/cloud/groups/mygroup/subadmins
```

- Return the subadmins of the group: `mygroup`

XML output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <status>ok</status>
    <statuscode>100</statuscode>
    <message/>
  </meta>
  <data>
    <element>Tom</element>
  </data>
</ocs>
```

Edit data of a single group

Edits attributes related to a group. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/groups/{groupid}

- HTTP method: PUT
- PUT argument: key, string - the field to edit:
  - displayname
- PUT argument: value, string - the new value for the field

Status codes:

- 100 - successful
- 101 - not supported by backend

Examples

```
$ curl -X PUT http://admin:secret@example.com/ocs/v1.php/cloud/groups/mygroup -d key="displa
```

- Updates the display name for the group `mygroup`

XML output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>100</statuscode>
    <status>ok</status>
  </meta>
  <data/>
</ocs>
```

Delete a group

Removes a group. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/groups/{groupid}

- HTTP method: DELETE

Status codes:

- 100 - successful
- 101 - group does not exist

- 102 - failed to delete group

Example

```
$ curl -X DELETE http://admin:secret@example.com/ocs/v1.php/cloud/groups/mygroup
```

- Delete the group `mygroup`

XML output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>100</statuscode>
    <status>ok</status>
  </meta>
  <data/>
</ocs>
```

Instruction set for apps

Getlist of apps

Returns a list of apps installed on the Nextcloud server. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/apps/

- HTTP method: GET
- url argument: filter, string - optional (`enabled` or `disabled`)

Status codes:

- 100 - successful
- 101 - invalid input data

Example

```
$ curl -X GET http://admin:secret@example.com/ocs/v1.php/cloud/apps?filter=enabled
```

- Gets enabled apps

XML output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>100</statuscode>
    <status>ok</status>
  </meta>
  <data>
    <apps>
      <element>files</element>
      <element>provisioning_api</element>
    </apps>
  </data>
</ocs>
```

Get app info

Provides information on a specific application. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/apps/{appid}

- HTTP method: GET

Status codes:

- 100 - successful

Example

```
$ curl -X GET http://admin:secret@example.com/ocs/v1.php/cloud/apps/files
```

- Get app info for the `files` app

XML output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>100</statuscode>
    <status>ok</status>
  </meta>
  <data>
    <info/>
    <remote>
      <files>appinfo/remote.php</files>
      <webdav>appinfo/remote.php</webdav>
      <filesync>appinfo/filesync.php</filesync>
    </remote>
    <public/>
    <id>files</id>
    <name>Files</name>
    <description>File Management</description>
    <licence>AGPL</licence>
    <author>Robin Appelman</author>
    <require>4.9</require>
    <shipped>true</shipped>
    <standalone></standalone>
    <default_enable></default_enable>
    <types>
      <element>filesystem</element>
    </types>
  </data>
</ocs>
```

Enable an app

Enable an app. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/apps/{appid}

- HTTP method: POST

Status codes:

- 100 - successful

Example

```
$ curl -X POST http://admin:secret@example.com/ocs/v1.php/cloud/apps/files_texteditor
```

- Enable the `files_texteditor` app

XML output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>100</statuscode>
    <status>ok</status>
  </meta>
</ocs>
```

Disable an app

Disables the specified app. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/apps/{appid}

- HTTP method: DELETE

Status codes:

- 100 - successful

Example

```
$ curl -X DELETE http://admin:secret@example.com/ocs/v1.php/cloud/apps/files_texteditor
```

- Disable the `files_texteditor` app

XML output

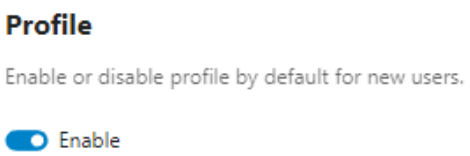
```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>100</statuscode>
    <status>ok</status>
  </meta>
</ocs>
```

Profile configuration

The user profile presents the information of a user and is enabled by default for all users. Users may individually enable or disable their profile in their Personal info settings under the Personal settings section.

As an administrator you may change the default for new users and may also disable profile globally to remove all profile functionality.

To enable or disable profile by default for new users switch the toggle in Basic settings under the Administration settings section.



Note

If you are upgrading from Nextcloud 22 to 23 and want profile to be disabled by default for all users, you may run the `occ` command below before upgrading or upgrade first then switch the toggle in Basic settings before letting any users log in.

```
occ config:app:set settings profile_enabled_by_default --value="0"
```

Please refer to Using the occ command for all available `occ` commands.

To disable profile globally add the following line to your `config.php`

```
'profile.enabled' => false,
```

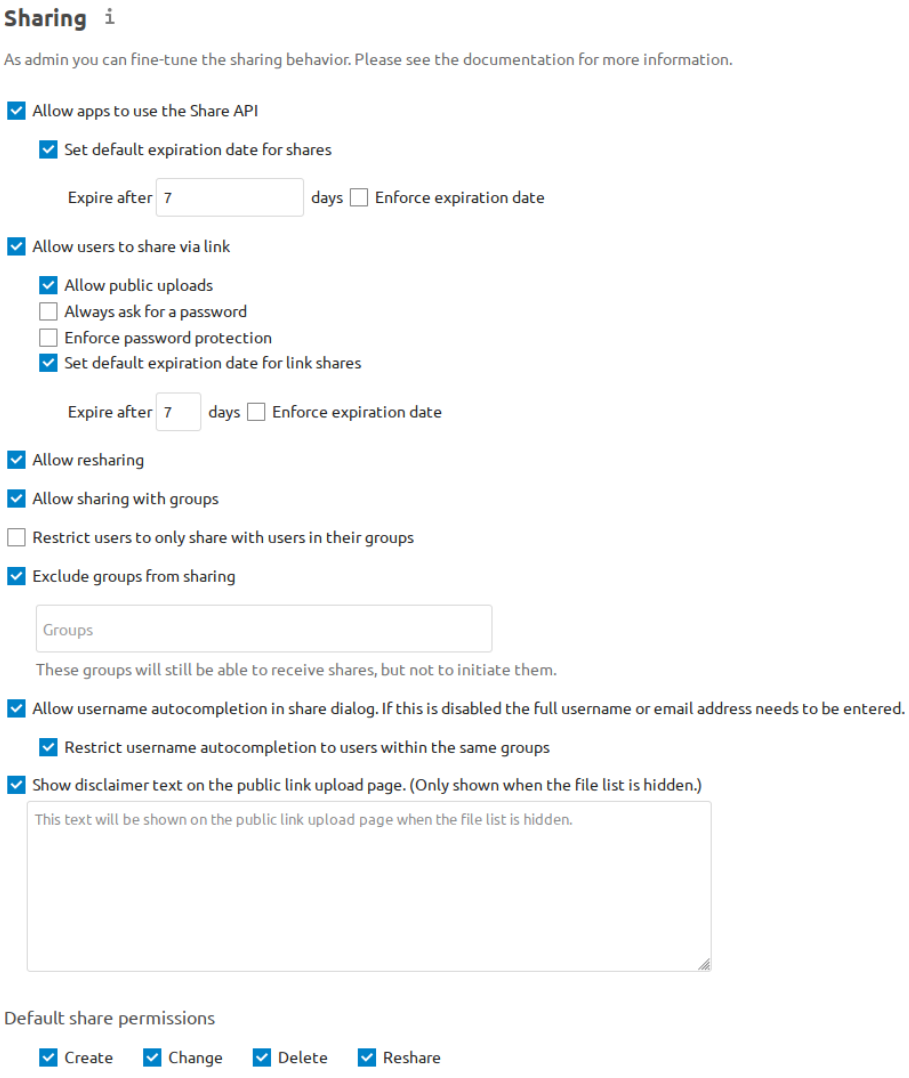
Please refer to Configuration Parameters for all available `config.php` options.

File sharing and management

File Sharing

Nextcloud users can share files with their Nextcloud groups and other users on the same Nextcloud server, with Nextcloud users on other Nextcloud servers, and create public shares for people who are not Nextcloud users. You have control of a number of user permissions on file shares.

Configure your sharing policy on your Admin page in the Sharing section.



- Check `Allow apps to use the Share API` to enable users to share files. If this is not checked, no users can create file shares.
- Check `Set default expiration date for shares` to set a default expiration date on local user and group shares.
- Check `Enforce expiration date` to always enforce the configured expiration date on local user and group shares.

> **Note**
>
> Users will not be able to set the expiration date further in the future than the enforced expiration date, although they will be able to set a more recent date. Also note that users will be able to update the expiration date again at a later point. The expiration date is based on the current date and not on the share creation date. The user will be able to extend the expiration date again whenever a previous expiration date is close to be reached.

- Check `Allow users to share via link` to enable creating public shares for people who are not Nextcloud users via hyperlink.
- Check `Allow public uploads` to allow anyone to upload files to public shares.
- Check `Always ask for a password` to proactively ask a user to set a password for a share link.
- Check `Enforce password protection` to force users to set a password on all public share links. This does not apply to local user and group shares.
- Check `Set default expiration date for link shares` to set a default expiration date on public shares.
- Check `Enforce expiration date` to always enforce the configured expiration date on public shares.

> **Note**
>
> Users will not be able to set the expiration date further in the future than the enforced expiration date, although they will be able to set a more recent date. Also note that users will be able to update the expiration date again at a later point. The expiration date is based on the current date and not on the share creation date. The user will be able to extend the expiration date again whenever a previous expiration date is close to be reached.

- Check `Allow resharing` to enable users to re-share files shared with them.
- Check `Allow sharing with groups` to enable users to share with groups.
- Check `Restrict users to only share with users in their groups` to confine sharing within group memberships.

> **Note**
>
> This setting does not apply to the Federated Cloud sharing feature. If Federated Cloud Sharing is enabled, users can still share items with any users on any instances (including the one they are on) via a remote share.

- Check `Exclude groups from sharing` to prevent members of specific groups from creating any file shares in those groups. When you check this, you'll get a dropdown list of all your groups to choose from. Members of excluded groups can still receive shares, but not create any.
- Check `Allow username autocompletion in share dialog` to enable auto-completion of Nextcloud usernames.

- Check `Restrict username autocompletion to users within the same groups` to limit username autocompletion to users from within the same groups as the share owner.
- Check `Show disclaimer text on the public link upload page` to set and show a disclaimer text on public links with hidden file lists.

With `Default share permissions` you are able to set the default permissions for user-shares (`Create`, `Change`, `Delete` and `Reshare`) without forcing them.

Note

Nextcloud does not preserve the mtime (modification time) of directories, though it does update the mtimes on files. See Wrong folder date when syncing for discussion of this.

Note

There are more sharing options on config.php level available: **`Configuration Parameters`**

Distinguish between max expiration date and default expiration date

The expiration date which can be set and enforced in the settings above are the hard limit and the default value at the same time. Sometimes admins want to have a moderate default expire date, for example 7 days but make sure that the user can't extend it to more than 14 days.

In order to do so the user can set a enforced expiration date in the settings as described above and set the default value to something below the maximal possible expiration date with the following OCC commands:

```
occ config:app:set --value <DAYS> core internal_defaultExpDays
occ config:app:set --value <DAYS> core link_defaultExpDays
```

Get a notification before a share expires

Users can get a notification before a share expires. In order to do so a cronjob need to be configured which calls the following OCC command once a day:

```
occ sharing:expiration-notification
```

A notification will be send for all shares which expire within the next 24 hours.

Transferring files to another user

You may transfer files from one user to another with `occ`. This is useful when you have to remove a user. Be sure to transfer the files before you delete the user! This transfers all files from user1 to user2, and the shares and metadata info associated with those files (shares, tags, comments, etc). Trashbin contents are not transferred:

```
occ files:transfer-ownership user1 user2
```

(See Using the occ command for a complete `occ` reference.)

Users may also transfer files or folders selectively by themselves. See user documentation for details.

Creating persistent file Shares

When a user is deleted, their files are also deleted. As you can imagine, this is a problem if they created file shares that need to be preserved, because these disappear as well. In Nextcloud files are tied to their owners, so whatever happens to the file owner also happens to the files.

One solution is to create persistent shares for your users. You can retain ownership of them, or you could create a special user for the purpose of establishing permanent file shares. Simply create a shared folder in the usual way, and share it with the users or groups who need to use it. Set the appropriate permissions on it, and then no matter

which users come and go, the file shares will remain. Because all files added to the share, or edited in it, automatically become owned by the owner of the share regardless of who adds or edits them.

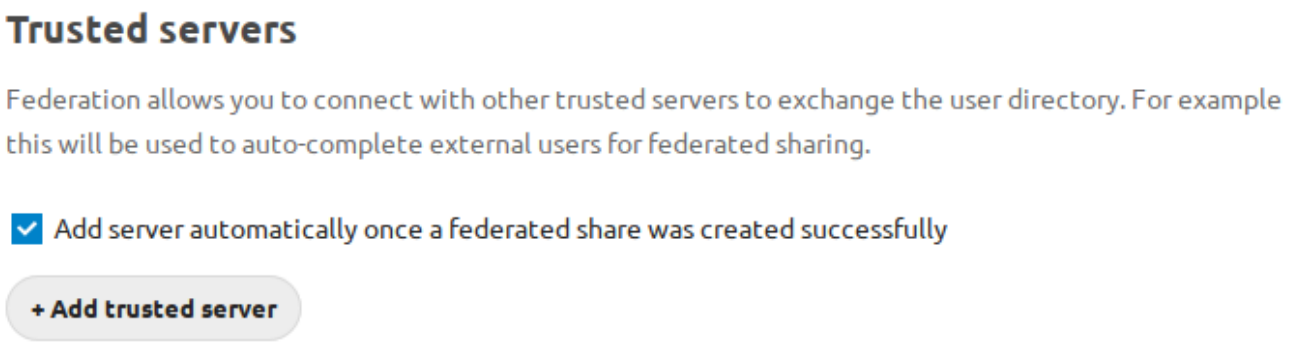
Configuring Federation Sharing

Federated Cloud Sharing is now managed by the Federation app (9.0+), and is now called Federation sharing. When you enable the Federation app you can easily and securely link file shares between Nextcloud servers, in effect creating a cloud of Nextclouds.

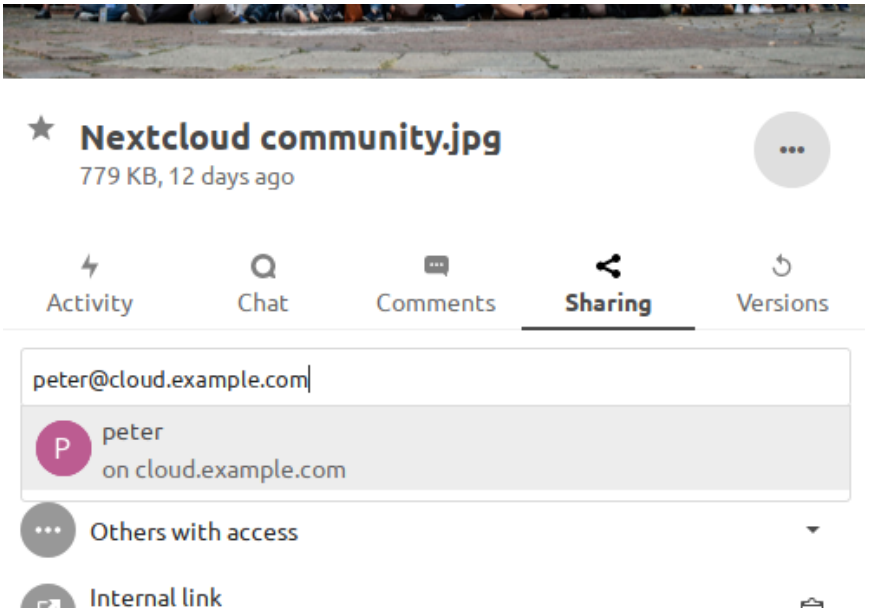
Creating a new Federation Share

Follow these steps to create a new Federation share between two Nextcloud servers. This requires no action by the user on the remote server; all it takes is a few steps on the originating server.

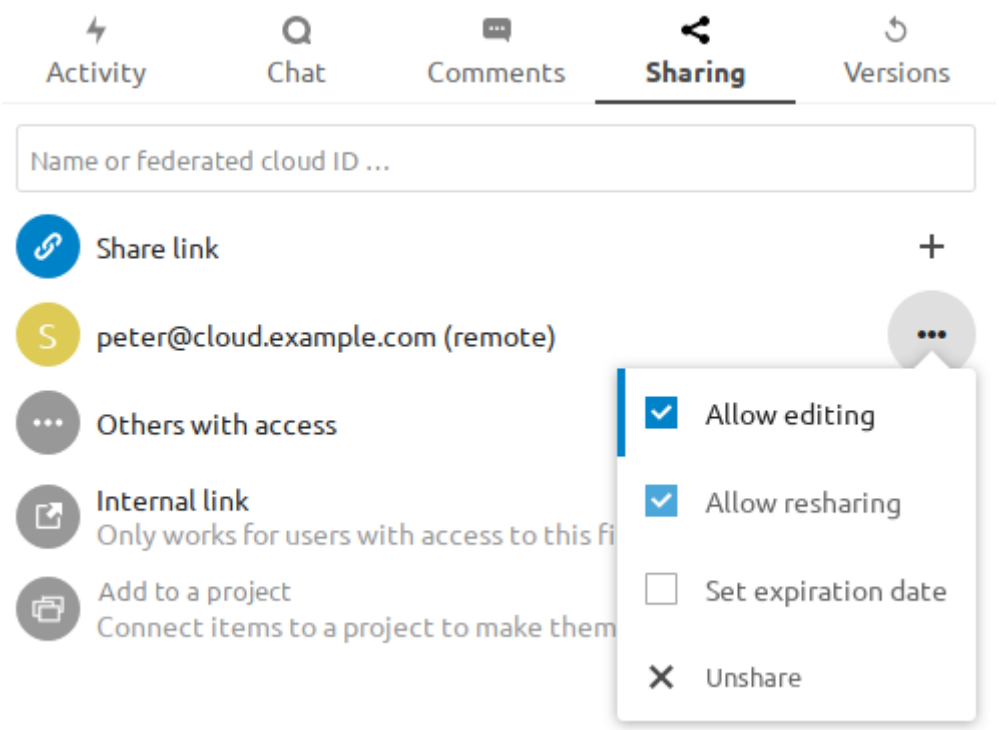
1. Enable the Federation app.
2. Go to your Nextcloud Admin page and scroll to the Sharing section. Verify that **Allow users on this server to send shares to other servers** and **Allow users on this server to receive shares from other servers** are enabled.
3. Now go to the Federation section. By default, **Add server automatically once a federated share was created successfully** is checked. The Federation app supports creating a list of trusted Nextcloud servers, which allows the trusted servers to exchange user directories and auto-complete the names of external users when you create shares. If you do not want this enabled, then un-check it.



4. Now go to your Files page and select a folder to share. Click the share icon, and then enter the username and URL of the user on the remote Nextcloud server. In this example, that is `freda@https://example.com/nextcloud`. When Nextcloud verifies the link, it displays it with the **(remote)** label. Click on this label to establish the link.



5. When the link is successfully completed, you have a single share option, and that is **can edit**.

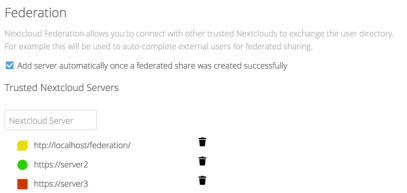


You may disconnect the share at any time by clicking the trash can icon.

Configuring trusted Nextcloud servers

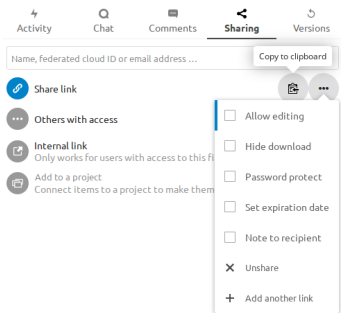
You may create a list of trusted Nextcloud servers for Federation sharing. This allows your linked Nextcloud servers to share user directories, and to auto-fill user names in share dialogs. If **Add server automatically once a federated share was created successfully** is enabled on your Admin page, servers will be automatically added to your trusted list when you create new Federation shares.

You may also enter Nextcloud server URLs in the **Add Nextcloud Server** field. The yellow light indicates a successful connection, with no user names exchanged. The green light indicates a successful connection with user names exchanged. A red light means the connection failed.

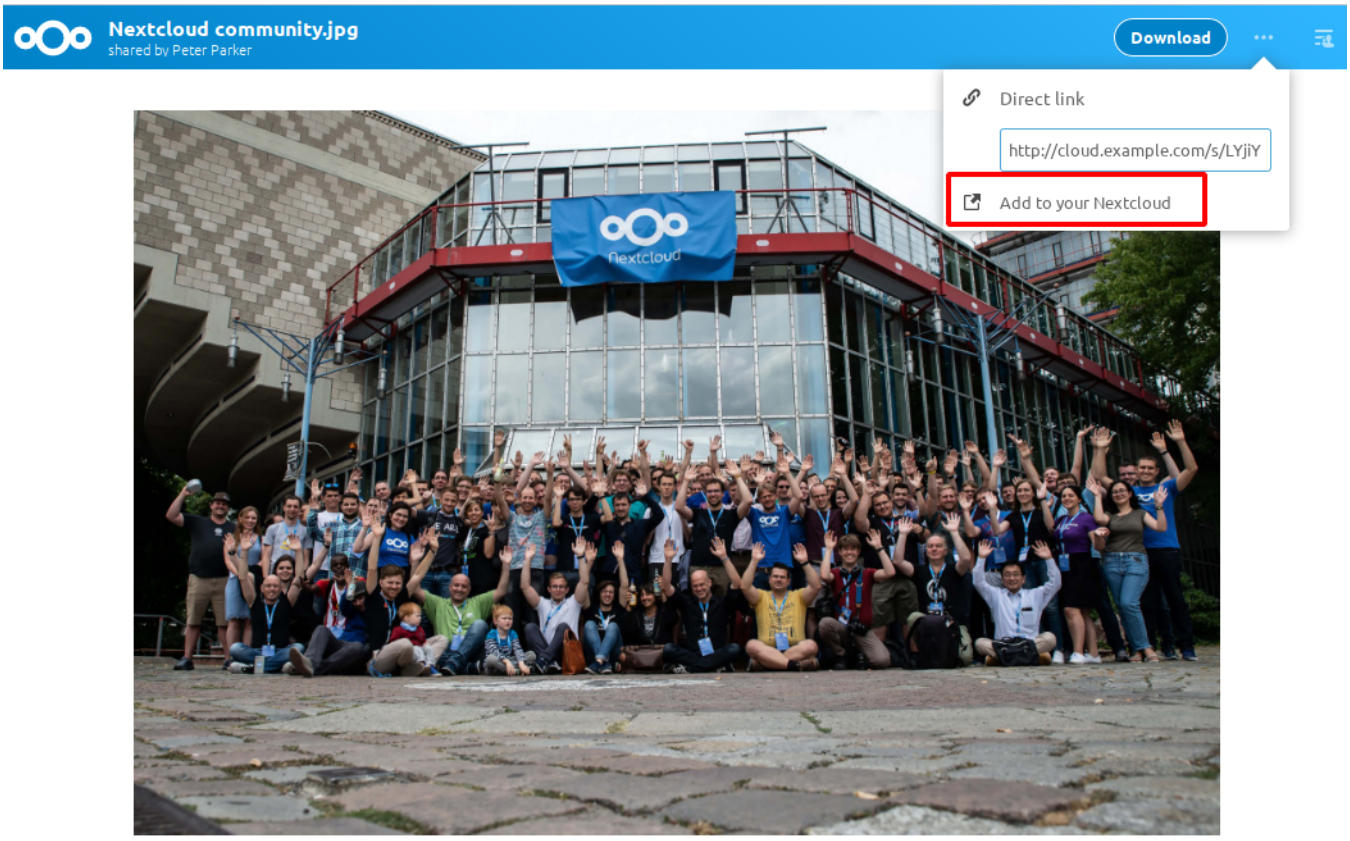


Creating Federation Shares via public Link Share

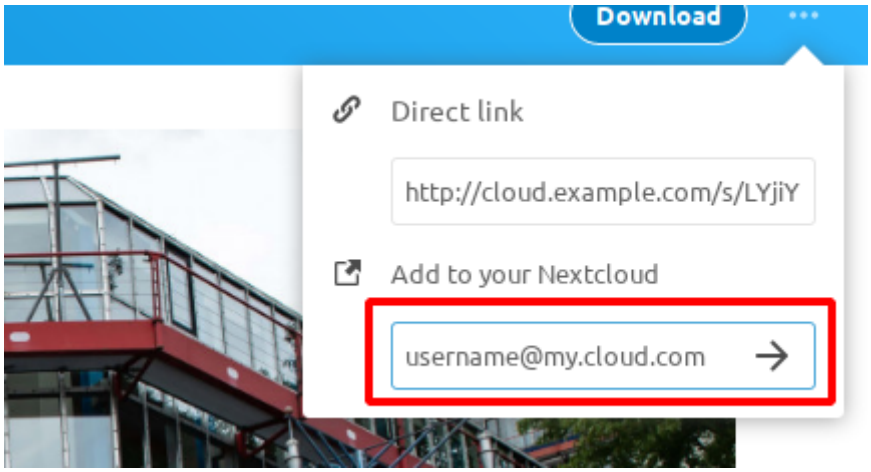
Check the `Share link` entry to expose more sharing options (which are described more fully in File Sharing). You may create a Federation share by allowing Nextcloud to create a public link for you, and then email it to the person you want to create the share with.



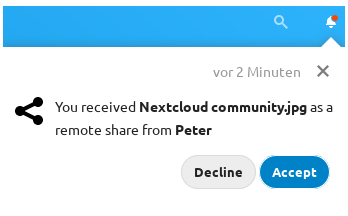
You may optionally set a password and expiration date on it. When your recipient receives your email they must click the link, or copy it to a Web browser. They will see a page displaying a thumbnail of the file, with a button to **Add to your Nextcloud**.



Your recipient should click the **Add to your Nextcloud** button. On the next screen your recipient needs to enter the URL to their Nextcloud server, and then press the return key.



Your recipient has to take one more step, and that is to confirm creating the federated cloud share link by clicking the **Accept** button.



Un-check the `Share link` checkbox to disable any federated cloud share created this way.

Configuration tips

The Sharing section on your Admin page allows you to control how your users manage federated cloud shares:

- Check `Enforce password protection` to require passwords on link shares.
- Check `Set default expiration date` to require an expiration date on link shares.
- Check `Allow public uploads` to allow two-way file sharing.

Your Apache Web server must have `mod_rewrite` enabled, and you must have `trusted_domains` correctly configured in `config.php` to allow external connections (see Installation wizard). Consider also enabling SSL to encrypt all traffic between your servers .

Your Nextcloud server creates the share link from the URL that you used to log into the server, so make sure that you log into your server using a URL that is accessible to your users. For example, if you log in via its LAN IP address, such as `http://192.168.10.50`, then your share URL will be something like `http://192.168.10.50/nextcloud/index.php/s/jWfCfTVztGlWTJe`, which is not accessible outside of your LAN. This also applies to using the server name; for access outside of your LAN you need to use a fully-qualified domain name such as `http://myserver.example.com`, rather than `http://myserver`.

Uploading big files > 512MB

The default maximum file size for uploads is 512MB. You can increase this limit up to what your filesystem and operating system allows. There are certain hard limits that cannot be exceeded:

- < 2GB on 32Bit OS-architecture
- < 2GB with IE6 - IE8
- < 4GB with IE9 - IE11

64-bit filesystems have much higher limits; consult the documentation for your filesystem.

> **Note**
>
> The Nextcloud sync client is not affected by these upload limits as it is uploading files in smaller chunks. See Client documentation for more information on configuration options.

System configuration

- Make sure that the latest version of PHP is installed
- Disable user quotas, which makes them unlimited
- Your temp file or partition has to be big enough to hold multiple parallel uploads from multiple users; e.g. if the max upload size is 10GB and the average number of users uploading at the same time is 100: temp space has to hold at least 10x100 GB

Configuring your Web server

> **Note**
>
> Nextcloud comes with its own `nextcloud/.htaccess` file. Because `php-fpm` can't read PHP settings in `.htaccess` these settings must be set in the `nextcloud/.user.ini` file.

Set the following two parameters inside the corresponding php.ini file (see the **Loaded Configuration File** section of PHP version and information to find your relevant php.ini files)

```
php_value upload_max_filesize 16G
php_value post_max_size 16G
```

The `upload_max_filesize` and `post_max_size` settings may not apply to file uploads through WebDAV single file PUT requests or Chunked file uploads For those, PHP and webserver timeouts are the limiting factor on the upload size.

Adjust these values for your needs. If you see PHP timeouts in your logfiles, increase the timeout values, which are in seconds:

```
php_value max_input_time 3600
php_value max_execution_time 3600
```

The mod_reqtimeout Apache module could also stop large uploads from completing. If you're using this module and getting failed uploads of large files either disable it in your Apache config or raise the configured `RequestReadTimeout` timeouts.

There are also several other configuration options in your Web server config which could prevent the upload of larger files. Please see the manual of your Web server for how to configure those values correctly:

Apache

- LimitRequestBody
- SSLRenegBufferSize
- Timeout

Apache with mod_fcgid

- FcgidMaxRequestInMem
- FcgidMaxRequestLen

Note

If you are using Apache/2.4 with mod_fcgid, as of February/March 2016, `FcgidMaxRequestInMem` still needs to be significantly increased from its default value to avoid the occurrence of segmentation faults when uploading big files. This is not a regular setting but serves as a workaround for Apache with mod_fcgid bug #51747.

Setting `FcgidMaxRequestInMem` significantly higher than normal may no longer be necessary, once bug #51747 is fixed.

Apache with mod_proxy_fcgi

- ProxyTimeout

nginx

- client_max_body_size
- fastcgi_read_timeout
- client_body_temp_path

Since nginx 1.7.11 a new config option fastcgi_request_buffering is available. Setting this option to `fastcgi_request_buffering off;` in your nginx config might help with timeouts during the upload. Furthermore it helps if you're running out of disc space on the tmp partition of your system.

Note

Make sure that `client_body_temp_path` points to a partition with adequate space for your upload file size, and on the same partition as the `upload_tmp_dir` or `tempdirectory` (see below). For optimal performance, place these on a separate hard drive that is dedicated to swap and temp storage.

If your site is behind a nginx frontend (for example a loadbalancer):

By default, downloads will be limited to 1GB due to `proxy_buffering` and `proxy_max_temp_file_size` on the frontend.

- If you can access the frontend's configuration, disable proxy_buffering or increase proxy_max_temp_file_size from the default 1GB.
- If you do not have access to the frontend, set the X-Accel-Buffering header to `add_header X-Accel-Buffering no;` on your backend server.

Configuring PHP

If you don't want to use the Nextcloud `.htaccess` or `.user.ini` file, you may configure PHP instead. Make sure to comment out any lines `.htaccess` pertaining to upload size, if you entered any.

If you are running Nextcloud on a 32-bit system, any `open_basedir` directive in your `php.ini` file needs to be commented out.

Set the following two parameters inside `php.ini`, using your own desired file size values:

```
upload_max_filesize = 16G
post_max_size = 16G
```

Tell PHP which temp directory you want it to use:

```
upload_tmp_dir = /var/big_temp_file/
```

Output Buffering must be turned off in `.htaccess` or `.user.ini` or `php.ini`, or PHP will return memory-related errors:

- `output_buffering = 0`

Configuring Nextcloud

As an alternative to the `upload_tmp_dir` of PHP (e.g. if you don't have access to your `php.ini`) you can also configure a temporary location for uploaded files by using the `tempdirectory` setting in your `config.php` (See Configuration Parameters).

If you have configured the `session_lifetime` setting in your `config.php` (See Configuration Parameters) file then make sure it is not too low. This setting needs to be configured to at least the time (in seconds) that the longest upload will take. If unsure remove this completely from your configuration to reset it to the default shown in the `config.sample.php`.

Adjust chunk size on Nextcloud side

For upload performance improvements in environments with high upload bandwidth, the server's upload chunk size may be adjusted:

```
sudo -u www-data php occ config:app:set files max_chunk_size --value 20971520
```

Put in a value in bytes (in this example, 20MB). Set `--value 0` for no chunking at all.

Default is 10485760 (10 MiB).

Large file upload on object storage

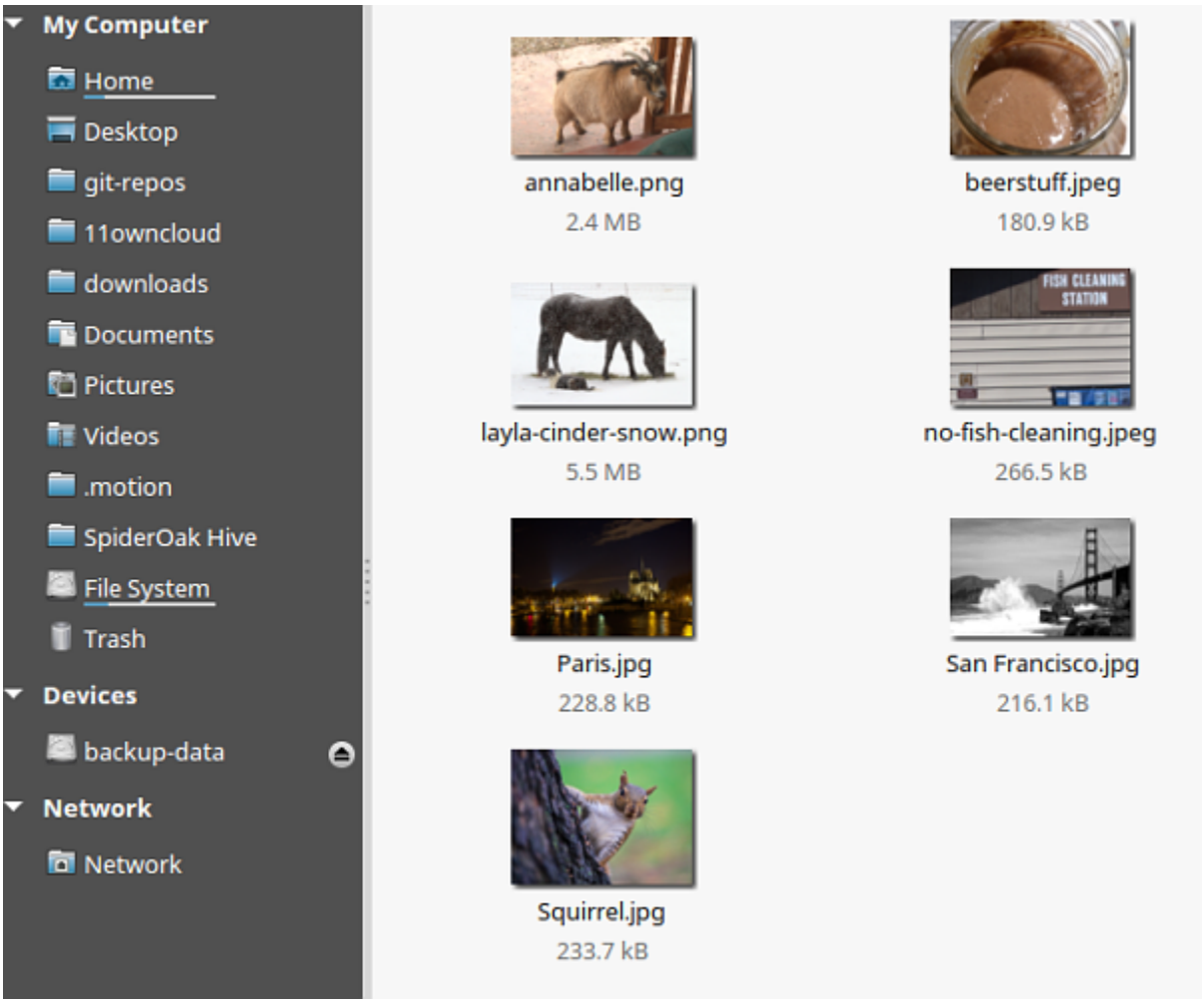
Chunked file uploads do have a larger space consumption on the temporary folder when processing those uploads on object storage as the individual chunks get downloaded from the storage and will be assembled to the actual file on the Nextcloud servers temporary directory. It is recommended to increase the size of your temp directory accordingly and also ensure that request timeouts are high enough for PHP, webservers or any load balancers involved.

Providing default files

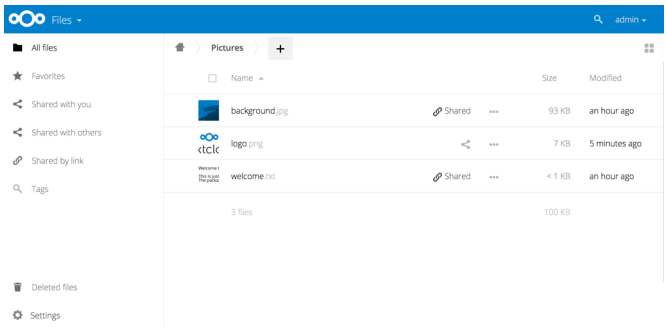
You may distribute a set of default files and folders to all users by placing them in directory that is readable by the webserver user. This allows you to overwrite the files that are shipped by default with Nextcloud in `core/skeleton`. That custom directory should then be configured in the `config.php` via the configuration option `skeletondirectory` (see Configuration Parameters). Leave empty to not copy any skeleton files.

These files will be copied only to new users after their initial login, and existing users will not see files that are added to this directory after their first login. The files in the `skeleton` directory are copied into the users data directories, so they may change and delete the files without affecting the originals.

This screenshot shows a set of photos in the `skeleton` directory.



They appear on the user's Nextcloud Files page just like any other files.



Note

Overwriting the files in `core/skeleton` is not recommended, because those changes will be overwritten on the next update of the Nextcloud server.

Configuring Object Storage as Primary Storage

Nextcloud allows to configure object storages like OpenStack Swift or Amazon Simple Storage Service (S3) or any compatible S3-implementation (e.g. Minio or Ceph Object Gateway) as primary storage replacing the default storage of files.

By default, files are stored in `nextcloud/data` or another directory configured in the `config.php` of your Nextcloud instance. This data directory might still be used for compatibility reasons)

Implications

When using an object store as primary storage, Nextcloud assumes exclusive access over the bucket being used.

Contrary to using an object store as external storage, when an object store is used as primary storage, no metadata (names, directory structures, etc) is stored in the object store. The metadata is only stored in the database and the object store only holds the file content by unique identifier.

Because of this primary object stores usually perform better than when using the same object store as external storage but it restricts being able to access the files from outside of Nextcloud.

Configuration

Primary object stores need to be configured in `config.php` by specifying the objectstore backend and any backend specific configuration.

Note

Configuring a primary object store on an existing Nextcloud instance will make all existing files on the instance inaccessible.

The configuration has the following structure:

```
'objectstore' => [
        'class' => 'Object\\Storage\\Backend\\Class',
        'arguments' => [
                ...
        ],
],
```

OpenStack Swift

The OpenStack Swift backend mounts a container on an OpenStack Object Storage server into the virtual filesystem.

The class to be used is `\OC\Files\ObjectStore\Swift`

Both openstack v2 and v3 authentication are supported,

V2 Authentication:

```
'objectstore' => [
        'class' => '\\OC\\Files\\ObjectStore\\Swift',
        'arguments' => [
                'username' => 'username',
                'password' => 'Secr3tPaSSWoRdt7',
                // the container to store the data in
                'bucket' => 'nextcloud',
                'autocreate' => true,
                'region' => 'RegionOne',
                // The Identity / Keystone endpoint
                'url' => 'http://example.com/v2.0',
                // optional on some swift implementations
                'tenantName' => 'username',
                'serviceName' => 'swift',
                // The Interface / url Type, optional
                'urlType' => 'internal'
        ],
],
```

V3 Authentication:

```
'objectstore' => [
        'class' => 'OC\\Files\\ObjectStore\\Swift',
        'arguments' => [
                'autocreate' => true,
                'user' => [
                        'name' => 'UserName',
                        'password' => 'Secr3tPaSSWoRdt7',
                        'domain' => [
                                'name' => 'Default',
                        ],
                ],
                'scope' => [
                        'project' => [
                                'name' => 'TenantName',
                                'domain' => [
                                        'name' => 'Default',
                                ],
                        ],
                ],
                'serviceName' => 'swift',
                'region' => 'regionOne',
                'url' => 'http://example.com/v3',
                'bucket' => 'nextcloud',
        ],
],
```

Simple Storage Service (S3)

The simple storage service (S3) backend mounts a bucket on an Amazon S3 object storage or compatible implementation (e.g. Minio or Ceph Object Gateway) into the virtual filesystem.

The class to be used is `\OC\Files\ObjectStore\S3`

```
'objectstore' => [
        'class' => '\\OC\\Files\\ObjectStore\\S3',
        'arguments' => [
                'bucket' => 'nextcloud',
```

```
                'autocreate' => true,
                'key'    => 'EJ39ITYZEUH5BGWDRUFY',
                'secret' => 'M5MrXTRjkyMaxXPe2FRXMTfTfbKEnZCu+7uRTVSj',
                'hostname' => 'example.com',
                'port' => 1234,
                'use_ssl' => true,
                'region' => 'optional',
                // required for some non Amazon S3 implementations
                'use_path_style'=>true
        ],
],
```

Note

Not all configuration options are required for all S3 servers. Overriding the hostname, port and region of your S3 server is only required for non-Amazon implementations, which in turn usually don't require the region to be set.

Note

`use_path_style` is usually not required (and is, in fact, incompatible with newer Amazon datacenters), but can be used with non-Amazon servers where the DNS infrastructure cannot be controlled. Ordinarily, requests will be made with http://bucket.hostname.domain/, but with path style enabled, requests are made with http://hostname.domain/bucket instead.

Multibucket Object Store

It's possible to configure Nextcloud to distribute the data over multiple buckets for scalability purposes.

To setup multiple buckets, use `'objectstore_multibucket'` storage backend in `config.php`:

```
'objectstore_multibucket' => [
        'class' => 'Object\\Storage\\Backend\\Class',
        'arguments' => [
                // optional, defaults to 64
                'num_buckets' => 64,
                // will be postfixed by an integer in the range from 0 to (num_nuckets-1)
                'bucket' => 'nextcloud_',
                ...
        ],
],
```

Multibucket object store backend maps every user to a range of buckets and saves all files for that user in their corresponding bucket.

Note

While it is possible to change the number of buckets used by an existing Nextcloud instance, the user-to-buckets mapping is only created once, so only newly created users will be mapped to the updated range of buckets.

You can find out more information about upscaling with object storage and Nextcloud in the Nextcloud customer portal.

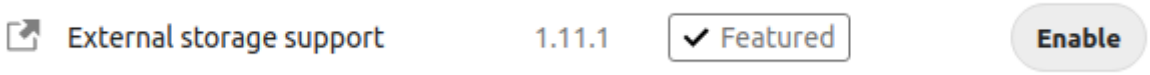
Configuring External Storage (GUI)

The External Storage Support application enables you to mount external storage services and devices as secondary Nextcloud storage devices. You may also allow users to mount their own external storage services.

For configuration of external storages via occ command, see occ documentation.

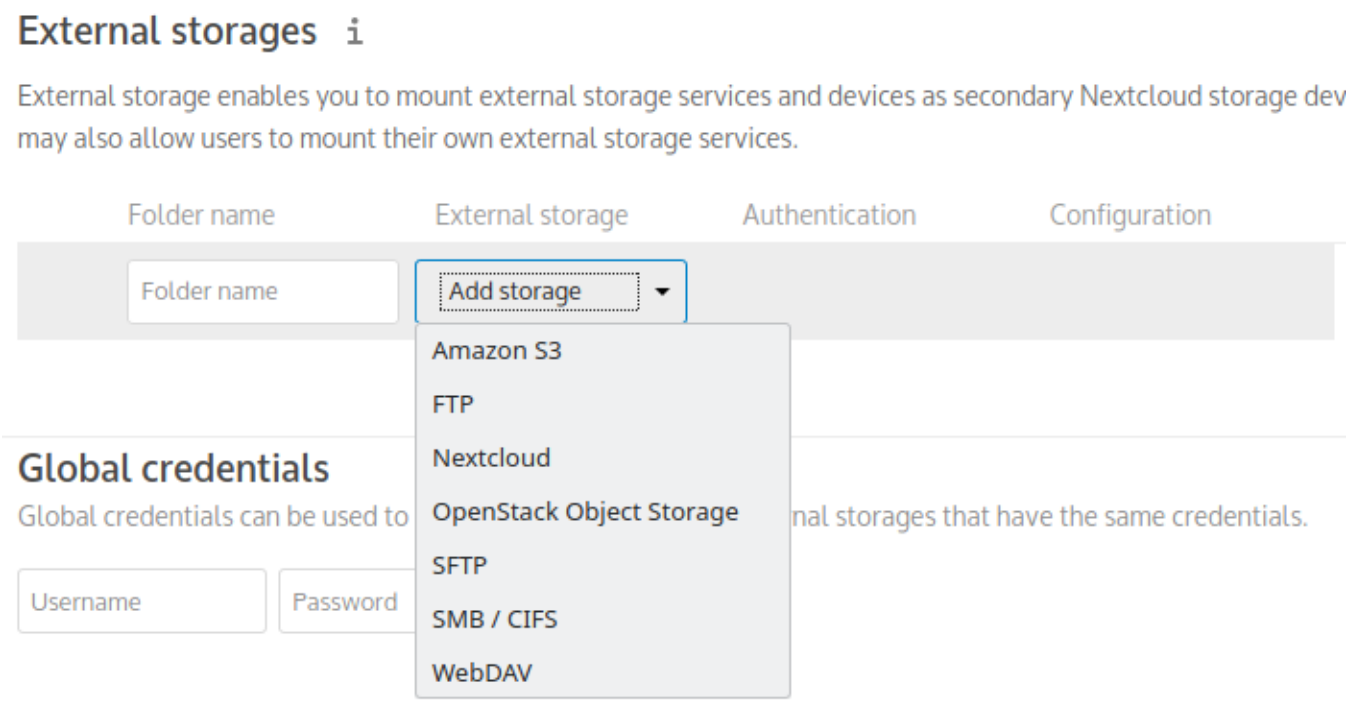
Enabling External Storage Support

The External storage support application is enabled on your Apps page.



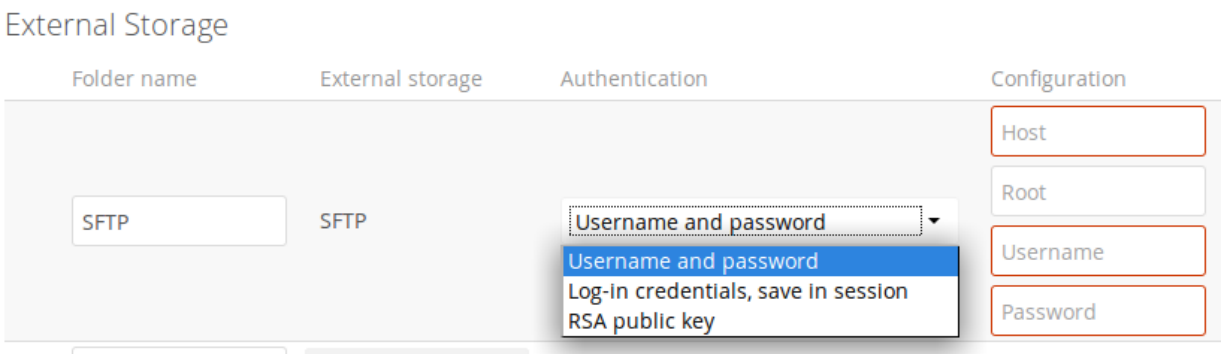
Storage configuration

To create a new external storage mount, select an available backend from the dropdown **Add storage**. Each backend has different required options, which are configured in the configuration fields.



Each backend may also accept multiple authentication methods. These are selected with the dropdown under **Authentication**. Different backends support different authentication mechanisms; some specific to the backend, others are more generic. See External Storage authentication mechanisms for more detailed information.

When you select an authentication mechanism, the configuration fields change as appropriate for the mechanism. The SFTP backend, for one example, supports **username and password**, **Log-in credentials, save in session**, and **RSA public key**.



Required fields are marked with a red border. When all required fields are filled, the storage is automatically saved. A green dot next to the storage row indicates the storage is ready for use. A red or yellow icon indicates that Nextcloud could not connect to the external storage, so you need to re-check your configuration and network availability.

If there is an error on the storage, it will be marked as unavailable for ten minutes. To re-check it, click the colored icon or reload your Admin page.

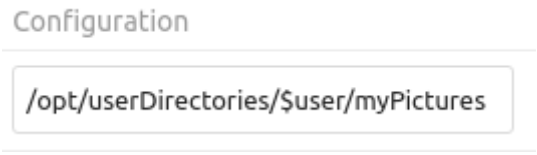
Usage of variables for mount paths

The external storage mounting mechanism accepts variables in the mount path.

Use `$user` for automatic substitution with the logged in user's username.

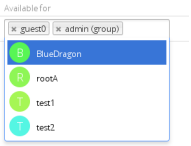
Use `$home` for automatic substitution with a configurable home directory variable (requires LDAP, see Special attributes in the LDAP configuration documentation for details)

In the following example, the mount point for a logged in user "alice" would substitute to `/opt/userDirectories/alice/myPictures`.



User and group permissions

A storage configured in a user's Personal settings is available only to the user that created it. A storage configured in the Admin settings is available to all users by default, and it can be restricted to specific users and groups in the **Available for** field.



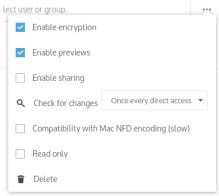
Mount options

The Overflow menu (three dots) exposes the settings and trashcan. Click the trashcan to delete the mountpoint. The settings button allows you to configure each storage mount individually with the following options:

- Encryption
- Previews
- Enable Sharing
- Filesystem check frequency (Never, Once per direct access)
- Mac NFD Compatability
- Read Only

The **Encryption** checkbox is visible only when the Encryption app is enabled. Note that server-side encryption is not available for other Nextcloud servers used as external storage.

Enable Sharing allows the Nextcloud admin to enable or disable sharing on individual mountpoints. When sharing is disabled the shares are retained internally, so that you can re-enable sharing and the previous shares become available again. Sharing is disabled by default.



Using self-signed certificates

When using self-signed certificates for external storage mounts the certificate must be imported into the personal settings of the user. Please refer to Nextcloud HTTPS External Mount for more information.

Available storage backends

The following backends are provided by the external storages app.

Amazon S3

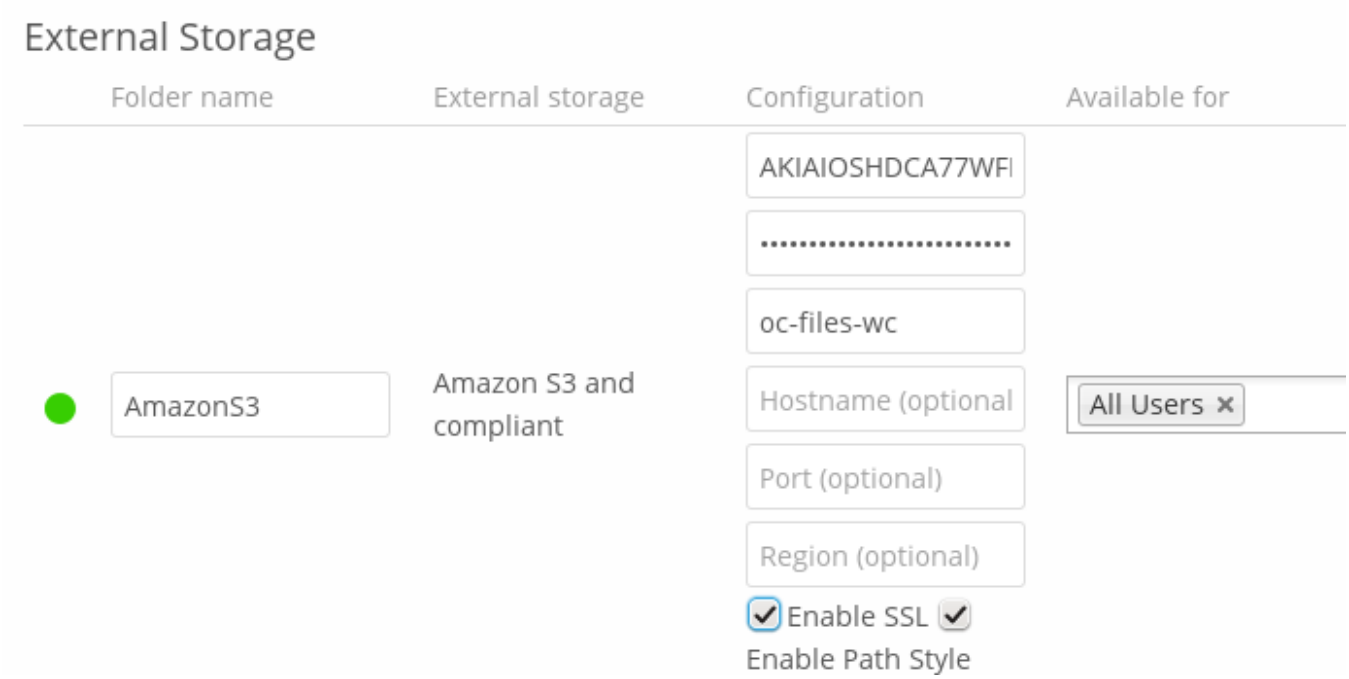
To connect your Amazon S3 buckets to Nextcloud, you will need:

- S3 access key
- S3 secret key
- Bucket name

In the **Folder name** field enter a local folder name for your S3 mountpoint. If this does not exist it will be created.

In the **Available for** field enter the users or groups who have permission to access your S3 mount.

The `Enable SSL` checkbox enables HTTPS connections; using HTTPS is always highly-recommended.



Optionally, you can override the hostname, port and region of your S3 server, which is required for non-Amazon servers such as Ceph Object Gateway.

Enable path style is usually not required (and is, in fact, incompatible with newer Amazon datacenters), but can be used with non-Amazon servers where the DNS infrastructure cannot be controlled. Ordinarily, requests will be made with `http://bucket.hostname.domain/`, but with path style enabled, requests are made with `http://hostname.domain/bucket` instead.

Legacy authentication is only required for S3 servers that only implement version 2 authentication, by default version 4 authentication will be used.

See Configuring External Storage (GUI) for additional mount options and information.

See External Storage authentication mechanisms for more information on authentication schemes.

FTP/FTPS

To connect to an FTP server, you will need:

- A folder name for your local mountpoint; the folder will be created if it does not exist

- The URL of the FTP server
- Port number (default: 21)
- FTP server username and password
- Remote Subfolder, the FTP directory to mount in Nextcloud. Nextcloud defaults to the root directory. If you specify a subfolder you must leave off the leading slash. For example, `public_html/images`

Your new mountpoint is available to all users by default, and you may restrict access by entering specific users or groups in the **Available for** field.

Optionally, Nextcloud can use FTPS (FTP over SSL) by checking **Secure ftps://**. This requires additional configuration with your root certificate if the FTP server uses a self-signed certificate.

Note

The external storage `FTP/FTPS` needs the `allow_url_fopen` PHP setting to be set to `1`. When having connection problems make sure that it is not set to `0` in your `php.ini`. See PHP version and information to learn how to find the right `php.ini` file to edit.

See Configuring External Storage (GUI) for additional mount options and information.

FTP uses the password authentication scheme; see External Storage authentication mechanisms for more information on authentication schemes.

Local

Local storages provide access to any directory on the Nextcloud server. Since this is a significant security risk, Local storage can only be configured in the Nextcloud admin settings. Non-admin users cannot create Local storage mounts.

Use this to mount any directory on your Nextcloud server that is outside of your Nextcloud `data/` directory. This directory must be readable and writable by your HTTP server user. These ownership and permission examples are on Ubuntu Linux:

```
sudo chown -R www-data:www-data /path/to/localdir
sudo chmod -R 0750 /path/to/localdir
```

Important: If you use consecutive commands, make sure, you are user `www-data`:

```
sudo -u www-data bash
cd /path/to/localdir
mkdir data
```

In the **Folder name** field enter the folder name that you want to appear on your Nextcloud Files page.

In the **Configuration** field enter the full filepath of the directory you want to mount.

In the **Available for** field enter the users or groups who have permission to access the mount. By default all users have access.

See Configuring External Storage (GUI) for additional mount options and information.

See External Storage authentication mechanisms for more information on authentication schemes.

Nextcloud

A Nextcloud storage is a specialized WebDAV storage, with optimizations for Nextcloud-Nextcloud communication. See the WebDAV documentation to learn how to configure a Nextcloud external storage.

When filling in the **URL** field, use the path to the root of the Nextcloud installation, rather than the path to the WebDAV endpoint. So, for a server at `https://example.com/nextcloud`, use `https://example.com/nextcloud` and not `https://example.com/nextcloud/remote.php/dav`.

See Configuring External Storage (GUI) for additional mount options and information.

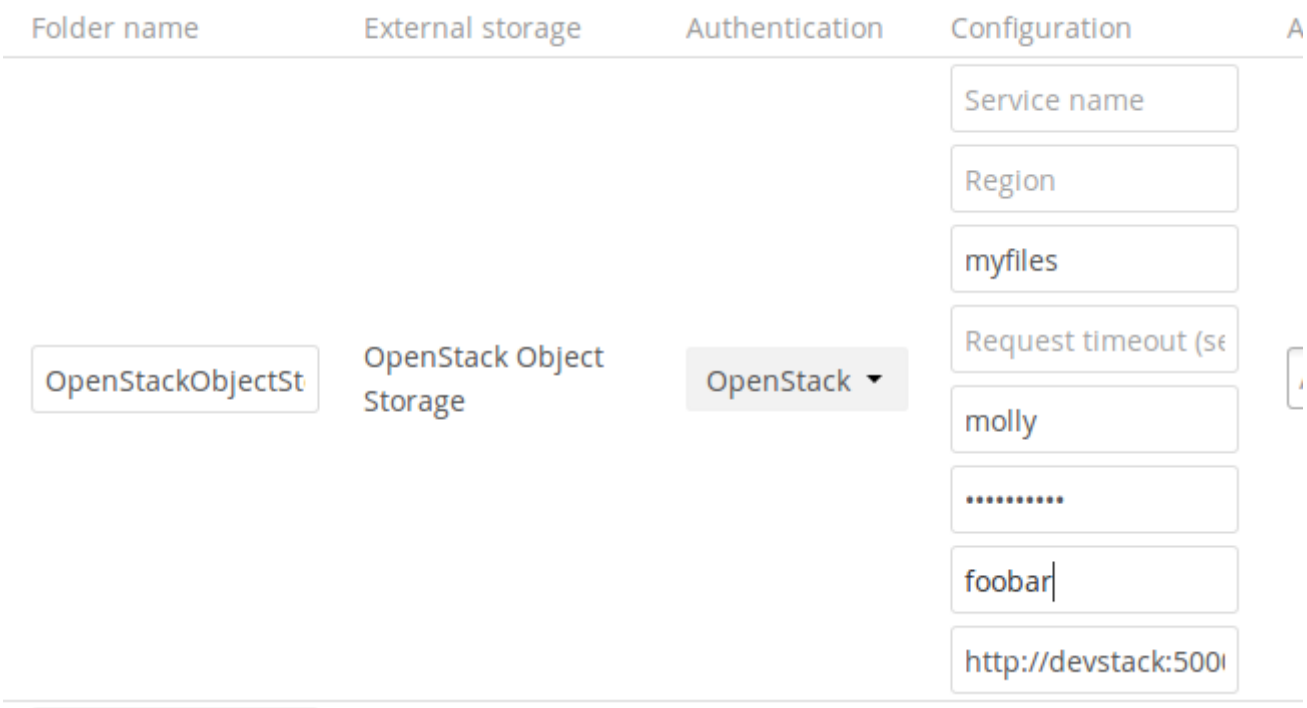
See External Storage authentication mechanisms for more information on authentication schemes.

OpenStack Object Storage

OpenStack Object Storage is used to connect to an OpenStack Swift server, or to Rackspace. Two authentication mechanisms are available: one is the generic OpenStack mechanism, and the other is used exclusively for Rackspace, a provider of object storage that uses the OpenStack Swift protocol.

The OpenStack authentication mechanism uses the OpenStack Keystone v2 protocol. Your Nextcloud configuration needs:

- **Bucket**. This is user-defined; think of it as a subdirectory of your total storage. The bucket will be created if it does not exist.
- **Username** of your account.
- **Password** of your account.
- **Tenant name** of your account. (A tenant is similar to a user group.)
- **Identity Endpoint URL**, the URL to log in to your OpenStack account.

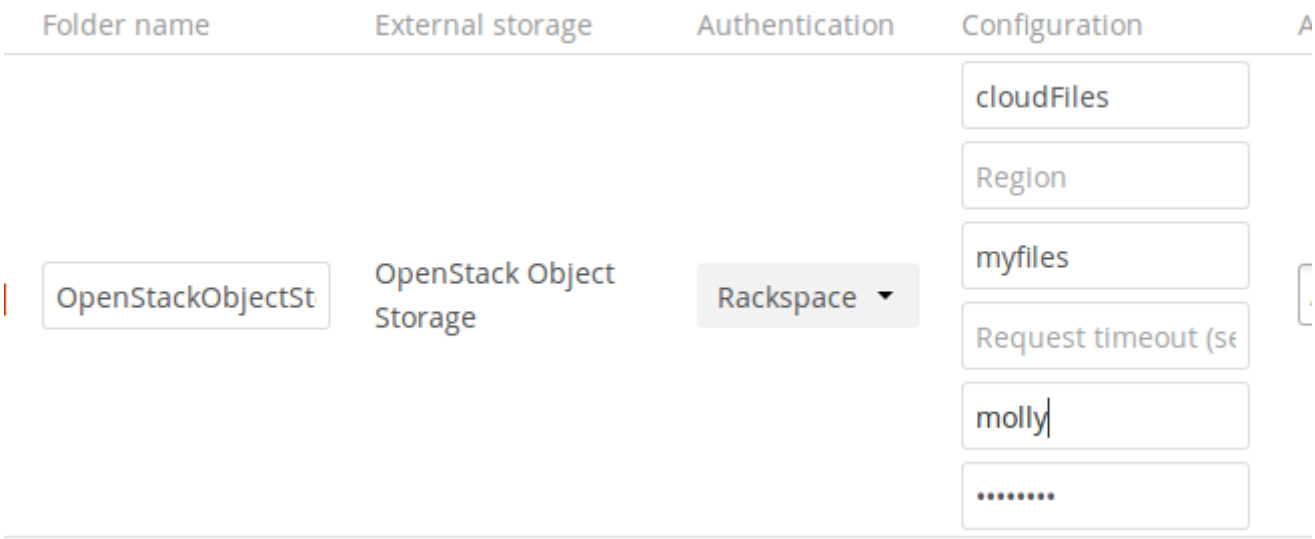


The Rackspace authentication mechanism requires:

- **Bucket**
- **Username**

- **API key**.

You must also enter the term **cloudFiles** in the **Service name** field.



It may be necessary to specify a **Region**. Your region should be named in your account information, and you can read about Rackspace regions at About Regions.

The timeout of HTTP requests is set in the **Request timeout** field, in seconds.

See Configuring External Storage (GUI) for additional mount options and information.

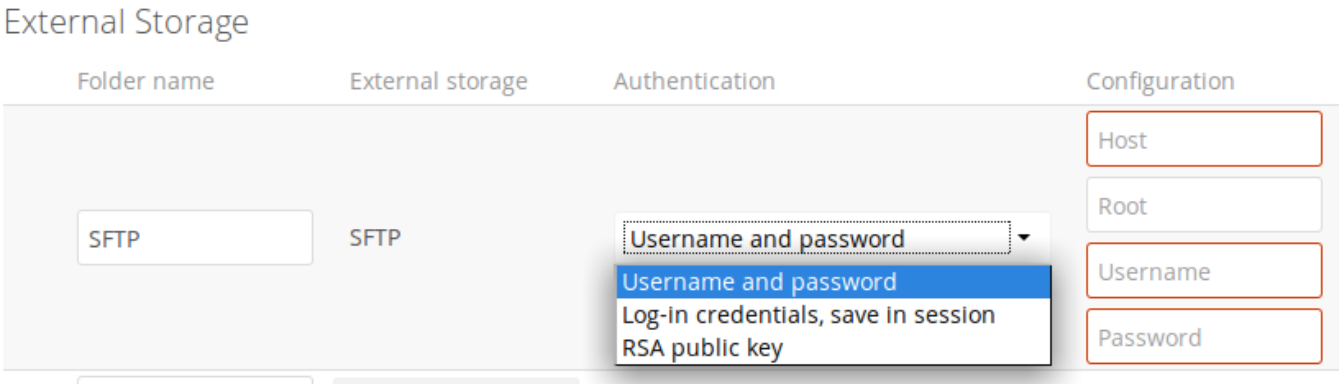
See External Storage authentication mechanisms for more information on authentication schemes.

SFTP

Nextcloud's SFTP (SSH File Transfer Protocol) backend supports both password and public key authentication.

The **Host** field is required; a port can be specified as part of the **Host** field in the following format: `hostname.domain:port`. The default port is 22 (SSH).

For public key authentication, you can generate a public/private key pair from your **SFTP with secret key login** configuration.



After generating your keys, you need to copy your new public key to the destination server to `.ssh/authorized_keys`. Nextcloud will then use its private key to authenticate to the SFTP server.

The default **Remote Subfolder** is the root directory (`/`) of the remote SFTP server, and you may enter any directory you wish.

See Configuring External Storage (GUI) for additional mount options and information.

See External Storage authentication mechanisms for more information on authentication schemes.

SMB/CIFS

Nextcloud can connect to Windows file servers or other SMB-compatible servers with the SMB/CIFS backend.

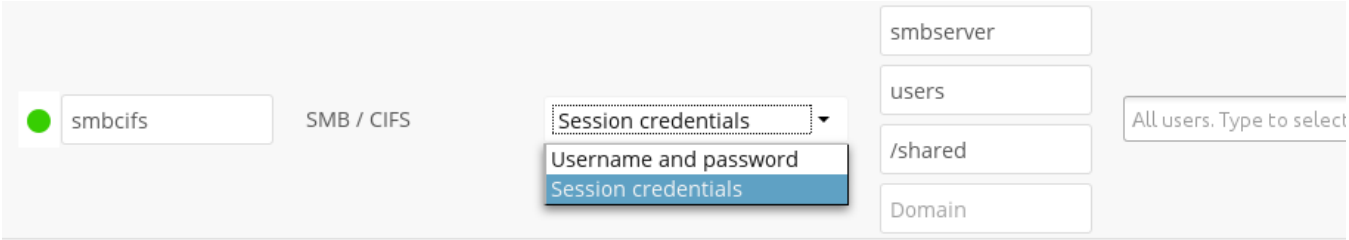
> **Note**
>
> The SMB/CIFS backend requires `smbclient` or the PHP smbclient module to be installed on the Nextcloud server. The PHP smbclient module is preferred, but either will work. These should be included in any Linux distribution. (See PECL smbclient if your distro does not include them.)

You need the following information:

- Folder name for your local mountpoint.
- Host: The URL of the Samba server.
- Username: The username or `domain\username` (see below) used to login to the Samba server.
- Password: the password to login to the Samba server.
- Share: The share on the Samba server to mount.
- Remote Subfolder: The remote subfolder inside the Samba share to mount (optional, defaults to /). To assign the Nextcloud logon username automatically to the subfolder, use `$user` instead of a particular subfolder name.
- And finally, the Nextcloud users and groups who get access to the share.

Optionally, you can specify a `Domain`. This is useful in cases where the SMB server requires a domain and a username, and an advanced authentication mechanism like session credentials is used so that the username cannot be modified. This is concatenated with the username, so the backend gets `domain\username`

> **Note**
>
> For improved reliability and performance, we recommended installing `libsmbclient-php`, a native PHP module for connecting to SMB servers.



See Configuring External Storage (GUI) for additional mount options and information.

See External Storage authentication mechanisms for more information on authentication schemes.

SMB update notifications

Nextcloud can use smb update notifications to listen for changes made to a configured SMB/CIFS storage and detect external changes made to the storage in near real-time.

> **Note**
>
> Due to limitations of linux based SMB servers, this feature only works reliably on Windows SMB servers.

Note

Using update notifications requires `smbclient` 4.x or newer. Due to limitations with the smbclient PHP module, the `smbclient` binary is required even when using the PHP module.

To start listening to update notifications, start the `occ` command like this:

```
occ files_external:notify <mount_id>
```

You can find the mount id for a specific storage using `occ files_external:list`

On default this command shows no output, can you see the list of detected changes by passing the `-v` option to the command.

SMB authentication

Update notifications are not supported when using 'Login credentials, save in session' authentication. Using login credentials is only supported with 'Login credentials, save in database'.

Even when using 'Login credentials, save in database' or 'User entered, stored in database' authentication the notify process can not use the credentials saved to attach to the smb shares because the notify process does not run in the context of a specific user in those cases you can provide the username and password using the `--username` and `--password` arguments.

Decrease sync delay

Any updates detected by the notify command will only be synced to the client after the Nextcloud cron job has been executed (usually every 15 minutes). If this interval is too high for your use case, you can decrease it by running `occ files:scan --unscanned --all` at the desired interval. Note that this might increase the server load and you'll need to ensure that there is no overlap between runs.

Hidden files upload failure or not shown

If you have the configuration `hide dot files = Yes`, you will not be able to upload a hidden file (dot file) nor will you be able to show hidden files on your filelist (even if the 'show hidden file' option is checked on the nextcloud settings. Make sure you have the following option in your configuration: `hide dot files = No`

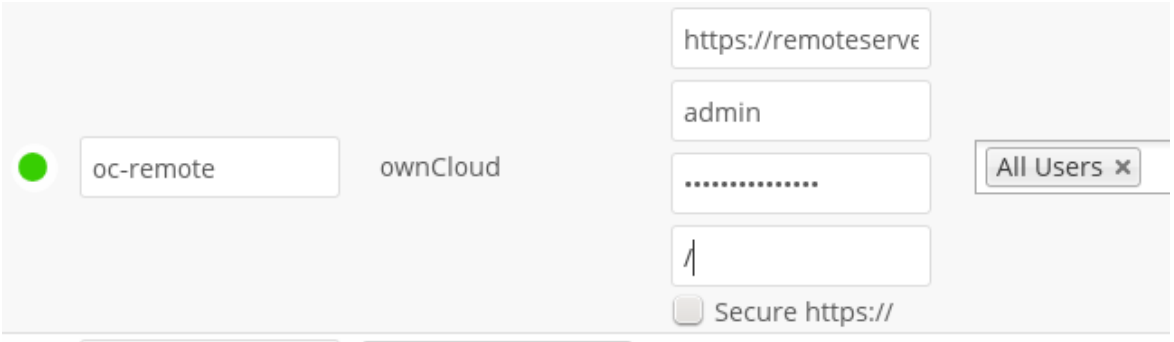
WebDAV

Use this backend to mount a directory from any WebDAV server, or another Nextcloud server.

You need the following information:

- Folder name: The name of your local mountpoint.
- The URL of the WebDAV or Nextcloud server.
- Username and password for the remote server
- Secure https://: We always recommend https:// for security, though you can leave this unchecked for http://.

Optionally, a `Remote Subfolder` can be specified to change the destination directory. The default is to use the whole root.



Note

CPanel users should install Web Disk to enable WebDAV functionality.

See Configuring External Storage (GUI) for additional mount options and information.

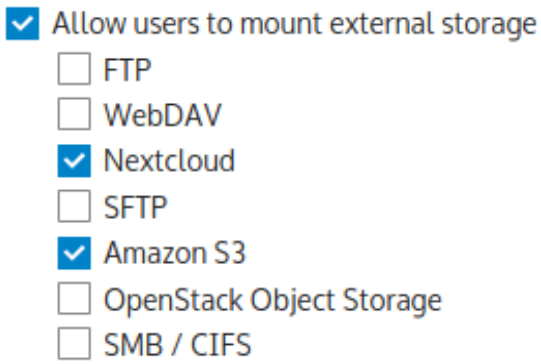
See External Storage authentication mechanisms for more information on authentication schemes.

Note

A non-blocking or correctly configured SELinux setup is needed for these backends to work. Please refer to the SELinux configuration.

Allow users to mount external Storage

Check **Enable User External Storage** to allow your users to mount their own external storage services, and check the backends you want to allow. Beware, as this allows a user to make potentially arbitrary connections to other services on your network!



Adding files to external storages

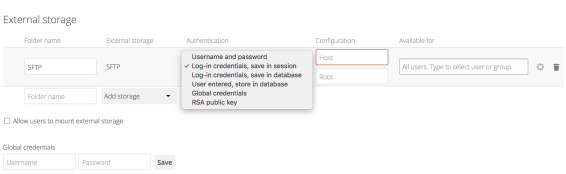
We recommend configuring the background job **Webcron** or **Cron** (see Background jobs) to enable Nextcloud to automatically detect files added to your external storages.

Nextcloud may not always be able to find out what has been changed remotely (files changed without going through Nextcloud), especially when it's very deep in the folder hierarchy of the external storage.

You might need to setup a cron job that runs `sudo -u www-data php occ files:scan --all` (or replace `--all` with the user name, see also Using the occ command) to trigger a rescan of the user's files periodically (for example every 15 minutes), which includes the mounted external storage.

External Storage authentication mechanisms

Nextcloud storage backends accept one or more authentication schemes such as passwords, OAuth, or token-based, to name a few examples. Each authentication scheme may be implemented by combining multiple authentication mechanisms. Different mechanisms require different configuration parameters, depending on their behavior.



Special mechanisms

The **None** authentication mechanism requires no configuration parameters, and is used when a backend requires no authentication.

The **Built-in** authentication mechanism itself requires no configuration parameters, but is used as a placeholder for legacy storages that have not been migrated to the new system and do not take advantage of generic authentication mechanisms. The authentication parameters are provided directly by the backend.

Password-based mechanisms

The **Username and password** mechanism requires a manually-defined username and password. These get passed directly to the backend and are specified during the setup of the mount point.

The **Log-in credentials, save in session** mechanism uses the Nextcloud login credentials of the user to connect to the storage. These are not stored anywhere on the server, but rather in the user session, giving increased security. This method has some important drawbacks, since Nextcloud has no access to the storage credentials and therefore cannot perform any background tasks on the storage:

- Sharing is disabled
- Background file scanning does not work
- Background versions expiration does not work
- Desktop and mobile clients that use tokens to authenticate can not access those shares
- Other services that might request the file through a different request like Collabora Online or OnlyOffice will not be able to open files from that storage
- The method cannot be used with SAML/SSO authentication, because Nextcloud does not get a hold of any credentials whatsoever

The **Log-in credentials, save in database** mechanism uses the Nextcloud login credentials of the user to connect to the storage. These are stored in the database encrypted with the shared secret. This allows to share files from within this mount point.

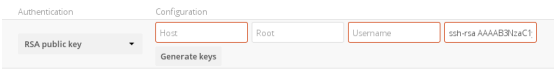
- The method cannot be used with SAML/SSO authentication, because Nextcloud does not get a hold of any credentials whatsoever

The **User entered, store in database** mechanism work in the same way as the “Username and password” mechanism but the credentials need to be specified by each user individually. Before the first access to that mount point the user will be prompted to enter the credentials.

The **Global credentials** mechanism uses the general input field for “Global credentials” in the external storage settings section as source for the credentials instead of individual credentials for a mount point.

Public-key mechanisms

Currently only the RSA mechanism is implemented, where a public/private keypair is generated by Nextcloud and the public half shown in the GUI. The keys are generated in the SSH format, and are currently 1024 bits in length. Keys can be regenerated with a button in the GUI.



Encryption configuration

The primary purpose of the Nextcloud server-side encryption is to protect users’ files on remote storage, such as Dropbox and Google Drive, and to do it easily and seamlessly from within Nextcloud.

Server-side encryption separates encryption of local and remote storage. This allows you to encrypt remote storage, such as Dropbox and Google, without having to also encrypt your home storage on your Nextcloud server.

Note

Nextcloud supports Authenticated Encryption for all newly encrypted files. See https://hackerone.com/reports/108082 for more technical information about the impact.

For maximum security make sure to configure external storage with “Check for changes: Never”. This will let Nextcloud ignore new files not added via Nextcloud, so a malicious external storage administrator could not add new files to the storage without your knowledge. Of course, this is not wise if your external storage is subject to legitimate external changes.

Nextcloud server-side encryption encrypts files stored on the Nextcloud server, and files on remote storage that is connected to your Nextcloud server. Encryption and decryption are performed on the Nextcloud server. All files sent to remote storage will be encrypted by the Nextcloud server, and upon retrieval, decrypted before serving them to you and anyone you have shared them with.

Note

Encrypting files increases their size by roughly 35%, so you must take this into account when you are provisioning storage and setting storage quotas. User’s quotas are based on the unencrypted file size, and not the encrypted file size.

When files on external storage are encrypted in Nextcloud, you cannot share them directly from the external storage services, but only through Nextcloud sharing because the key to decrypt the data never leaves the Nextcloud server.

Nextcloud’s server-side encryption generates a strong encryption key, which is unlocked by user’s passwords. Your users don’t need to track an extra password, but simply log in as they normally do. It encrypts only the contents of files, and not filenames and directory structures.

You should regularly backup all encryption keys to prevent permanent data loss. The encryption keys are stored in the following directories:

`data/<user>/files_encryption`
: Users’ private keys and all other keys necessary to decrypt the users’ files

`data/files_encryption`
: private keys and all other keys necessary to decrypt the files stored on a system wide external storage

When encryption is enabled, all files are encrypted and decrypted by the Nextcloud application, and stored encrypted on your remote storage. This protects your data on externally hosted storage. The Nextcloud admin and the storage admin will see only encrypted files when browsing backend storage.

Warning

Encryption keys are stored only on the Nextcloud server, eliminating exposure of your data to third-party storage providers. The encryption app does **not** protect your data if your Nextcloud server is compromised, and it does not prevent Nextcloud administrators from reading user’s files. This would require client-side encryption, which this app does not provide. If your Nextcloud server is not connected to any external storage services then it is better to use other encryption tools, such as file-level or whole-disk encryption.

Note also that SSL terminates at or before Apache on the Nextcloud server, and all files will exist in an unencrypted state between the SSL connection termination and the Nextcloud code that encrypts and decrypts files. This is also potentially exploitable by anyone with administrator access to your server. Read How Nextcloud uses encryption to protect your data for more information.

Before enabling encryption

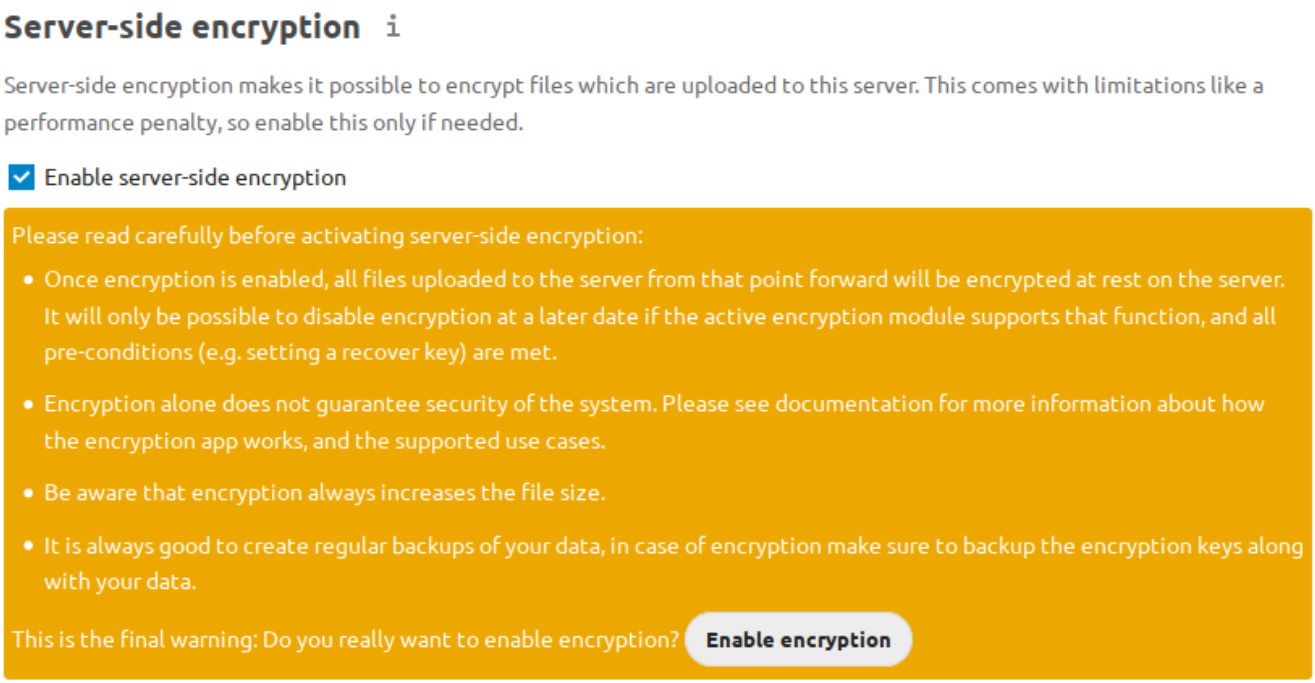
Plan very carefully before enabling encryption because it is not reversible via the Nextcloud Web interface. If you lose your encryption keys your files are not recoverable. Always have backups of your encryption keys stored in a safe location, and consider enabling all recovery options.

You have more options via the `occ` command (see occ encryption commands)

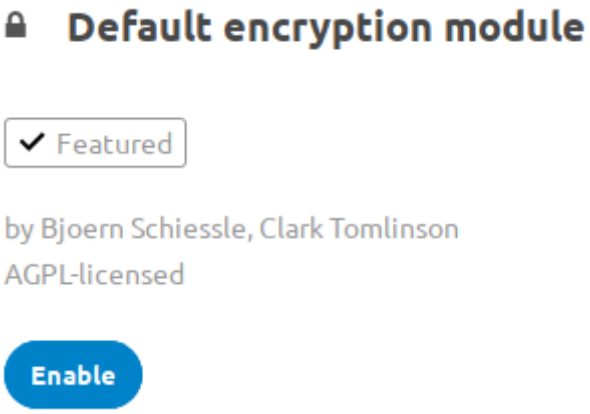
Enabling encryption

Nextcloud encryption consists of two parts. The base encryption system is enabled and disabled on your Admin page. First you must enable this, and then select an encryption module to load. Currently the only available encryption module is the Nextcloud Default Encryption Module.

First go to the **Server-side encryption** section of your Admin page and check **Enable server-side encryption**. You have one last chance to change your mind.



After clicking the **Enable Encryption** button you see the message “No encryption module loaded, please load a encryption module in the app menu”, so go to your Apps page to enable the Nextcloud Default Encryption Module.



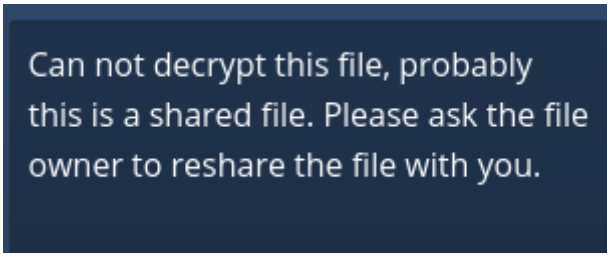
Return to your Admin page to see the Nextcloud Default Encryption Module added to the module selector, and automatically selected. Now you must log out and then log back in to initialize your encryption keys.

When you log back in, there is a checkbox for enabling encryption on your home storage. This is checked by default. Un-check to avoid encrypting your home storage.

Sharing encrypted files

After encryption is enabled your users must also log out and log back in to generate their personal encryption keys. They will see a yellow warning banner that says “Encryption App is enabled but your keys are not initialized, please log-out and log-in again.”

Share owners may need to re-share files after encryption is enabled; users trying to access the share will see a message advising them to ask the share owner to re-share the file with them. For individual shares, un-share and re-share the file. For group shares, share with any individuals who can’t access the share. This updates the encryption, and then the share owner can remove the individual shares.



Encrypting external mountpoints

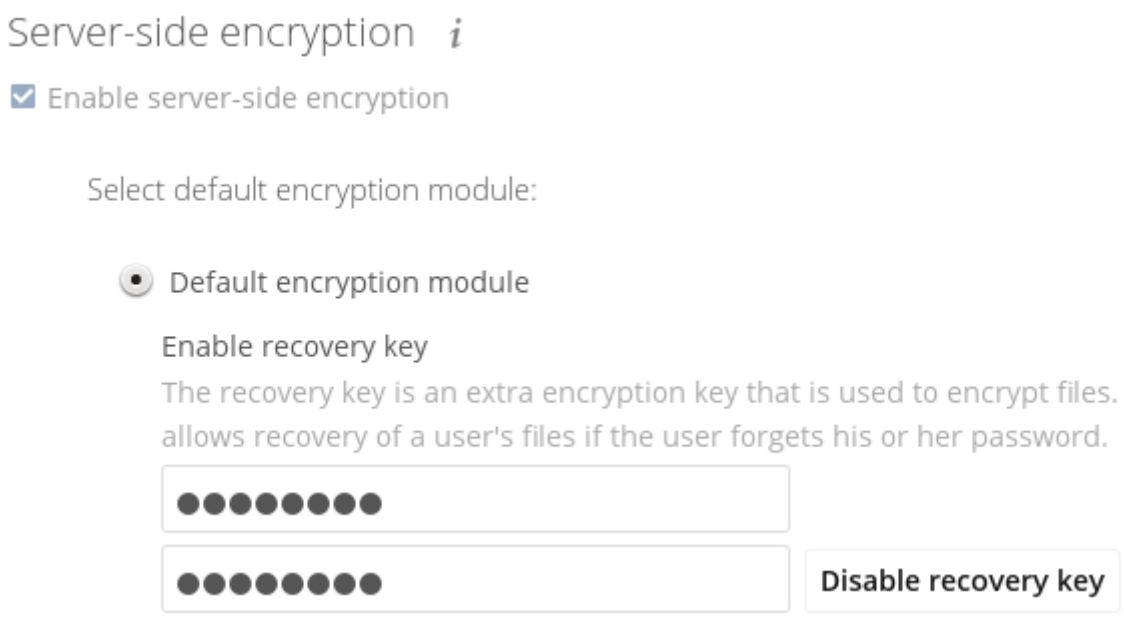
You and your users can encrypt individual external mountpoints. You must have external storage enabled on your Admin page, and enabled for your users.

Encryption settings can be configured in the mount options for an external storage mount, see Mount options (Configuring External Storage (GUI))

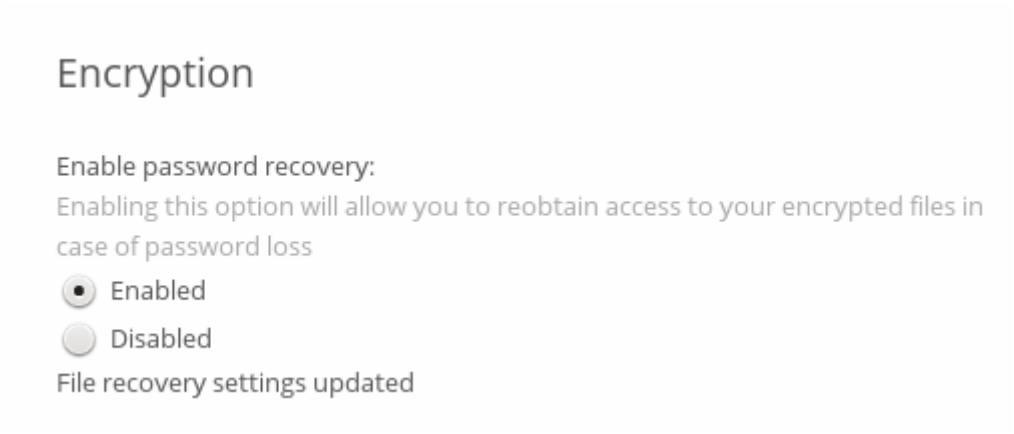
Enabling users file recovery keys

If you lose your Nextcloud password, then you lose access to your encrypted files. If one of your users loses their Nextcloud password their files are unrecoverable. You cannot reset their password in the normal way; you’ll see a yellow banner warning “Please provide an admin recovery password, otherwise all user data will be lost”.

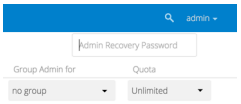
To avoid all this, create a Recovery Key. Go to the Encryption section of your Admin page and set a recovery key password.



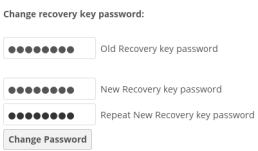
Then your users have the option of enabling password recovery on their Personal pages. If they do not do this, then the Recovery Key won't work for them.



For users who have enabled password recovery, give them a new password and recover access to their encrypted files by supplying the Recovery Key on the Users page.



You may change your Recovery Key password.



occ encryption commands

If you have shell access you may use the `occ` command to perform encryption operations, and you have additional options such as decryption and creating a single master encryption key. See Encryption for detailed instructions on using `occ`.

Get the current status of encryption and the loaded encryption module:

```
occ encryption:status
 - enabled: false
 - defaultModule: OC_DEFAULT_MODULE
```

This is equivalent to checking **Enable server-side encryption** on your Admin page:

```
occ encryption:enable
Encryption enabled

Default module: OC_DEFAULT_MODULE
```

List the available encryption modules:

```
occ encryption:list-modules
 - OC_DEFAULT_MODULE: Default encryption module [default*]
```

Select a different default Encryption module (currently the only available module is OC_DEFAULT_MODULE):

```
occ encryption:set-default-module [Module ID].
```

The [module ID] is taken from the `encryption:list-modules` command.

Encrypt all data files for all users. For performance reasons, when you enable encryption on a Nextcloud server only new and changed files are encrypted. This command gives you the option to encrypt all files.

Run `occ`:

```
occ encryption:encrypt-all

You are about to start to encrypt all files stored in your Nextcloud.
It will depend on the encryption module you use which files get encrypted.
Depending on the number and size of your files this can take some time.
Please make sure that no users access their files during this process!

Do you really want to continue? (y/n)
```

When you type `y` it creates a key pair for each of your users, and then encrypts their files, displaying progress until all user files are encrypted.

Decrypt all user data files, or optionally a single user:

```
occ encryption:decrypt-all [username]
```

View current location of keys:

```
occ encryption:show-key-storage-root
Current key storage root:  default storage location (data/)
```

Move keys to a different folder, either locally or on a different server. The folder must already exist, be owned by root and your HTTP group, and be restricted to root and your HTTP group. Further the folder needs to be located somewhere in your Nextcloud data folder, either physically, or as a mount. This example is for Ubuntu Linux. Note that the new folder is relative to your `occ` directory:

```
cd /your/nextcloud/data
mkdir keys
chown -R root:www-data keys
chmod -R 0770 keys
occ encryption:change-key-storage-root keys
Start to move keys:
   4 [============================]
Key storage root successfully changed to keys
```

Create a new master key. Use this when you have a single-sign on infrastructure. Use this only on fresh installations with no existing data, or on systems where encryption has not already been enabled. It is not possible to disable it:

```
occ encryption:enable-master-key
```

Disabling encryption

You may disable encryption only with `occ`. Make sure you have backups of all encryption keys, including users'. Put your Nextcloud server into maintenance mode, and then disable your encryption module with this command:

```
occ maintenance:mode --on
occ encryption:disable
occ encryption:decrypt-all
```

Take it out of maintenance mode when you are finished:

```
occ maintenance:mode --off
```

Warning

Disabling encryption without decrypting all the files will lead to decryption errors in the future as this state causes unpredictable behaviors.

Note

The `occ encryption:decrypt-all` can take a lot of time. You can run one user at a time like so: `occ encryption:decrypt-all <user-id>`.

Files not encrypted

Only the data in the files in `data/user/files` are encrypted, and not the filenames or folder structures. These files are never encrypted:

- Existing files in the trash bin & Versions. Only new and changed files after encryption is enabled are encrypted.
- Existing files in Versions
- Image thumbnails from the Gallery app
- Previews from the Files app
- The search index from the full text search app
- Third-party app data

There may be other files that are not encrypted; only files that are exposed to third-party storage providers are guaranteed to be encrypted.

LDAP and other external user back-ends

If you use an external user back-end, such as an LDAP or Samba server, and you change a user's password on the back-end, the user will be prompted to change their Nextcloud login to match on their next Nextcloud login. The user will need both their old and new passwords to do this. If you have enabled the Recovery Key then you can change a user's password in the Nextcloud Users panel to match their back-end password, and then, of course, notify the user and give them their new password.

Troubleshooting

Invalid private key for encryption app

This issue is being worked on. In the meantime there is a workaround which unfortunately is only suitable for administrators comfortable with the command line.

Encryption details

This document - provided by SysEleven - describes the server-side encryption scheme implemented by Nextcloud's default encryption module. This includes:

- the encryption and signature of files with a master key.
- the encryption and signature of files with a public sharing key.
- the encryption and signature of files with a recovery key.
- the encryption and signature of files with a user key.

These conventions apply throughout this document:

- Given file paths in this document are relative to the Nextcloud data directory that can be retrieved as `datadirectory` from the `config.php`.
- Placeholders are denoted as `$variable`. The variable has to be replaced with the appropriate information.
- Static strings are denoted as `"some string"`.
- The concatenation of strings is denoted as `$variable."some string"`.

Note

However, files that have been encrypted in Nextcloud versions 15 and lower may have slightly different structures.

Key type: master key

This is the default encryption mode in Nextcloud. With master key encryption enabled there is one central key that is used to secure the files handled by Nextcloud. The master key is protected by a password that can be generated by the server administrator. The advantage of the master key encryption is that the encryption is transparent to the users but has the disadvantage that the server administrator is able to decrypt user files without knowing any user password.

Key type: public sharing key

The public sharing key is used to secure files that have been publicly shared. The public sharing key is protected by a password that can be generated by the server administrator. The advantage of the public sharing key is that it is independent of the selected encryption mode so that Nextcloud is able to provide publicly shared files to outside parties.

Key type: recovery key

The recovery key is used to provide a restore mechanism in cases where the user key encryption is enabled, where the administrator has enabled the recovery key feature and the user has opted into using the recovery key feature. The recovery key can then be used to restore files when users have lost their passwords. The recovery key is protected by a recovery password that the server administrator should store securely. The advantage of the recovery key is that files can be recovered but has the disadvantage that the server administrator is able to decrypt user files without knowing any user password.

Key type: user key

User key encryption needs to be explicitly activated by calling `./occ encryption:disable-master-key`. In older versions of Nextcloud this had been enabled by default. With user key encryption enabled all users have their own user keys that are used to secure the files handled by Nextcloud. The user keys are protected by the user passwords. The advantage is that the server administrator is not able to decrypt user files without knowing any user password - unless the file is publicly shared or a recovery key is defined - but has the disadvantage that files are permanently lost if the users forget their user passwords - unless the files are (publicly) shared or a recovery key is defined.

Note

This method cannot be used with SAML authentication, because Nextcloud does not get a hold of any credentials whatsoever and therefore cannot use any users' passwords for encryption.

File type: public key file

Public key files contain RSA public keys that are used to encrypt/seal the share key files.

File format

Public key files are stored in PEM format.

File locations

The locations of public key files depend on their key type:

- master public key: `"files_encryption/OC_DEFAULT_MODULE/master_".$random.".publicKey"`
- public sharing public key: `"files_encryption/OC_DEFAULT_MODULE/pubShare_".$random.".publicKey"`
- recovery public key: `"files_encryption/OC_DEFAULT_MODULE/recoveryKey_".$random.".publicKey"`
- user public key: `$username."/files_encryption/OC_DEFAULT_MODULE/".$username.".publicKey"`

File type: private key file

Private key files contain RSA private keys that are used to decrypt/unseal the share key files. The RSA private key is encrypted and signed with a password and stored in a format that is specific to the Nextcloud encryption module.

File format

The RSA private key that is represented in PEM format is encrypted and Base64 encoded (denoted as `$encryption`). For the encryption an initialization vector of 16 bytes is selected (denoted as `$iv`). Furthermore a hexadecimally encoded message authentication code of 64 bytes is calculated (denoted as `$signature`). The resulting file contains:

```
"HBEGIN:cipher:AES-256-CTR:keyFormat:hash:HEND".
$encrypted."00iv00".$iv."00sig00".$signature."xxx"
```

File locations

The locations of private key files depend on their key type:

- master private key: `"files_encryption/OC_DEFAULT_MODULE/master_".$random.".privateKey"`
- public sharing private key: `"files_encryption/OC_DEFAULT_MODULE/pubShare_".$random.".privateKey"`
- recovery private key: `"files_encryption/OC_DEFAULT_MODULE/recoveryKey_".$random.".privateKey"`
- user private key: `$username."/files_encryption/OC_DEFAULT_MODULE/".$username.".privateKey"`

File type: share key file

Share key files contain so-called envelope keys that are needed to decrypt the file key files. The envelope keys are created by `openssl_seal()` during the encryption and are needed for `openssl_open()` during the decryption. The envelope keys are encrypted with the public keys of the recipients that are allowed to read the actual files.

File format

The envelope keys are stored in binary format.

File locations

The locations of share key files depend on the type of the encrypted file:

- regular file
  `$username."/files_encryption/keys/files/".$filename."/OC_DEFAULT_MODULE/".$recipient.".shareKey"`
  - version file: *version files use the same location for the share key file as their regular file*

`files_encryption/keys/files_trashbin/files/".$filename.".d".$timestamp."/OC_DEFAULT_MODULE/".$recip`

  - trashed version file: *trashed version files use the same location for the share key file as their trashed file*

File type: file key file

File key files contain symmetric keys used to encrypt the actual files. The file keys consist of 32 random bytes and are encrypted/sealed with the envelope keys stored in the share key files.

File format

The file keys are stored in binary format.

File locations

The locations of the file key files depend on the type of the encrypted file:

- regular file:
  `$username."/files_encryption/keys/files/".$filename."/OC_DEFAULT_MODULE/fileKey"`
- version file: *version files use the same location for the file key file as their regular file*

`me."/files_encryption/keys/files_trashbin/files/".$filename.".d".$delete_timestamp."/OC_DEFAULT_MODU`

- trashed version file: *trashed version files use the same location for the file key file as their trashed file*

File type: file

Files contain the actual file content. The file content is encrypted and signed with a password and stored in a format that is specific to the Nextcloud encryption module.

File format

The file content is split into blocks of 6072 bytes. Each block is encrypted and Base64 encoded (denoted as `$encryption[0..$n]`). For the encryption an initialization vector of 16 bytes is selected for each block (denoted as `$iv[0..$n]`). Furthermore a hexadecimally encoded message authentication code of 64 bytes is calculated of each block (denoted as `$signature[0..$n]`). An encrypted block has a total size of 8192 bytes (8096 bytes for `$encrypted[]`, 6 bytes for `"00iv00"`, 16 bytes for `$iv[]`, 7 bytes for `"00sig00"`, 64 bytes for `$signature[]` and 3 bytes for `"xxx"`). Only the last encrypted block may be shorter. The header of the encrypted file is padded with 8147 bytes of `"-"` (denoted as `$padding`) to a total of 8192 bytes. The resulting file contains:

```
"HBEGIN:cipher:AES-256-CTR:keyFormat:hash:HEND".$padding.
$encrypted[0]."00iv00".$iv[0]."00sig00".$signature[0]."xxx".
$encrypted[1]."00iv00".$iv[1]."00sig00".$signature[1]."xxx".
$encrypted[2]."00iv00".$iv[2]."00sig00".$signature[2]."xxx".
[...]
$encrypted[$n]."00iv00".$iv[$n]."00sig00".$signature[$n]."xxx"
```

File locations

The locations of the files depend on the type of the encrypted file:

- regular file: `$username."/files/".$filename`
- version file: `$username."/files_versions/".$filename.".v".$version_timestamp`
- trashed file: `$username."/files_trashbin/files/".$filename.".d".$delete_timestamp`
- trashed version file: `$username."/files_trashbin/versions/".$filename.".v".$version_timestamp."d".$delete_timestamp`

Key generation: generate the key pair

The key pair has to be generated with the `openssl_pkey_new()` function. Then the private key and public key are extracted from the the key resource with the `openssl_pkey_export()` function.

Key generation: store the public key

The public key is written to the `$username.".publicKey"` file as documented in File type: public key file.

Key generation: store the private key

Derive the encryption key

The salt for the encryption key is derived by creating a raw SHA256 hash of `$uid.$instanceId.$instanceSecret` with the `hash()` function. `$instanceId` can be retrieved as `instanceid` from the `config.php`. `$instanceSecret` can be retrieved as `secret` from the `config.php`.

The encryption key is then derived by creating a raw SHA256-PBKDF2 hash of the password with the salt, 100.000 rounds and (by default) with a target size of 32 bytes (as required for AES-256-CTR) with the `hash_hmac()` function (denoted as `$passphrase`).

The used password depends on the key type:

- master private key: use `secret` from the `config.php`
- public sharing private key: use an empty password
- recovery private key: use the recovery password
- user private key: use the user password

Encrypt the private key

The initialization vector is generated as a random string of 16 bytes with the `random_bytes()` function (denoted as `$iv`). The private key is (by default) AES-256-CTR encrypted with the `$iv` and the `$passphrase` with the `openssl_encrypt()` function and returned as Base64 encoded without zero-padding (denoted as `$encrypted`).

Sign the private key

The message authentication key is derived by creating a raw SHA512 hash of `$passphrase.$version.$position."a"` with the `hash()` function.

- `$version` is always `"0"`.
- `$position` is always `"0"`.

The signature is then derived by creating a hexadecimally encoded SHA256-HMAC of `$encrypted` and the message authentication key with the `hash_hmac()` function (denoted as `$signature`).

Store the private key

The private key is written to the `$username.".privateKey"` file with the derived `$encrypted`, `$iv` and `$signature` as documented in File type: private key file.

Encryption: generate the file key

Generate the file key

The file key is generated as a random string of 32 bytes with the `random_bytes()` function (denoted as `$filekey`).

Read the public key

The public keys of the recipients are read from the `$username.".publicKey"` files as documented in File type: public key file.

Encrypt/seal the file key

The file key is encrypted/sealed with the `openssl_seal()` function with the public keys. This returns the encrypted file key and the encrypted envelope keys for the recipients.

Store the file key

The encrypted file key is stored in the `"fileKey"` file as documented in File type: file key file.

Store the envelope keys

The encrypted envelope keys for the recipients are stored in the `$username.".shareKey"` files as documented in File type: share key file.

Encryption: encrypt the file

Split the file

The file is split into 6072 bytes sized blocks. Only the last encrypted block may be shorter. Each block is referenced by its zero-based index within the file (denoted as `$position`).

Encrypt the blocks

For each block the initialization vector is generated as a random string of 16 bytes with the `random_bytes()` function (denoted as `$iv[$position]`). The block is (by default) AES-256-CTR encrypted with the `$iv[$position]` and the `$filekey` with the `openssl_encrypt()` function and returned as Base64 encoded without zero-padding (denoted as `$encrypted[$position]`).

Sign the blocks

The message authentication key is derived by creating a raw SHA512 hash of `$filekey.$version.$position."a"` with the `hash()` function.

- `$version` is the `encrypted` value that can be retrieved from the `oc_filecache` table in the database and must not be zero. Take into account that a file in the `oc_filecache` table is identified by its `path` value as well as its `storage` value which references the `numeric_id` field in the `oc_storages` table. Including `$version` into the message authentication key prevents blocks from being swapped between different versions of the same file.
- `$position` is the index of the current block starting at `"0"` and is appended with `"end"` for the last block of the file. Including `$position` into the message authentication key prevents blocks from being swapped within the same file. Furthermore, adding `"end"` to the message authentication key of the last block prevents file truncation attacks.

The signature is then derived by creating a hexadecimally encoded SHA256-HMAC of `$encrypted[$position]` and the message authentication key with the `hash_hmac()` function (denoted as `$signature[$position]`).

Store the file

The encrypted file is written to the file with the derived `$encrypted[0..$n]`, `$iv[0..$n]` and `$signature[0..$n]` as documented in File type: file.

Decryption: read the private key

Read the private key file

The private key is read from the `$username."privateKey"` file and the values `$encrypted`, `$iv` and `$signature` are parsed as documented in File type: private key file.

Derive the decryption key

The salt for the decryption key is derived by creating a raw SHA256 hash of `$uid.$instanceId.$instanceSecret` with the `hash()` function. `$instanceId` can be retrieved as `instanceid` from the `config.php`. `$instanceSecret` can be retrieved as `secret` from the `config.php`.

The decryption key is then derived by creating a raw SHA256-PBKDF2 hash of the password with the salt, 100.000 rounds and (by default) with a target size of 32 bytes (as required for AES-256-CTR) with the `hash_hmac()` function (denoted as `$passphrase`).

The used password depends on the key type:

- master private key: use `secret` from the `config.php`
- public sharing private key: use an empty password
- recovery private key: use the recovery password
- user private key: use the user password

Check the signature

The message authentication key is derived by creating a raw SHA512 hash of `$passphrase.$version.$position."a"` with the `hash()` function.

- `$version` is always `"0"`.
- `$position` is always `"0"`.

The signature is then derived by creating a hexadecimally encoded SHA256-HMAC of `$encrypted` and the message authentication key with the `hash_hmac()` function. Only proceed when the derived signature is equal to *$signature* which is checked with the `hash_equals()` function.

Decrypt the private key

The private key is (by default) AES-256-CTR decrypted with the `$iv` and the `$passphrase` with the `openssl_decrypt()` function.

Decryption: read the file key

Read the file key

The encrypted file key is read from the `"fileKey"` file as documented in File type: file key file.

Read the envelope key

The encrypted envelope key for the recipient is read from the `$username.".shareKey"` file as documented in File type: share key file.

Decrypt/unseal the file key

The encrypted file key is decrypted/unsealed with the `openssl_open()` function with the private key and encrypted envelope key for the recipient (denoted as `$filekey`).

Decryption: decrypt the file

Split the file

The encrypted file is split into a 8192 bytes sized header and one or more 8192 bytes sized blocks. Only the last encrypted block may be shorter. Each block is referenced by its zero-based index within the file (denoted as `$position`). The values `$encrypted[0..$n]`, `$iv[0..$n]` and `$signature[0..$n]` are parsed as documented in File type: file.

Check the block signatures

The message authentication key is derived by creating a raw SHA512 hash of `$filekey.$version.$position."a"` with the `hash()` function.

- `$version` is the `encrypted` value that can be retrieved from the `oc_filecache` table in the database and must not be zero. Take into account that a file in the `oc_filecache` table is identified by its `path` value as well as its `storage` value which references the `numeric_id` field in the `oc_storages` table. Including `$version` into the message authentication key prevents blocks from being swapped between different versions of the same file.
- `$position` is the index of the current block starting at `"0"` and is appended with `"end"` for the last block of the file. Including `$position` into the message authentication key prevents blocks from being swapped within the same file. Furthermore, adding `"end"` to the message authentication key of the last block prevents file truncation attacks.

The signature is then derived by creating a hexadecimally encoded SHA256-HMAC of `$encrypted[$position]` and the message authentication key with the `hash_hmac()` function. Only proceed when the derived signature is equal to `$signature[$position]` which is checked with the `hash_equals()` function.

Decrypt the blocks

Each block is (by default) AES-256-CTR decrypted with the `$iv[$position]` and the `$filekey` with the `openssl_decrypt()` function.

Sources

- nextcloud-tools repository on GitHub
- Nextcloud Encryption Configuration documentation
- Nextcloud Help response concering the usage of version information
- Overview of ownCloud Encryption Model
- Sourcecode: Creation of the Message Authentication Code
- Sourcecode: Derivation of the Encryption Key
- Sourcecode: Encryption of the File
- Sourcecode: Encryption/Sealing of the File Key
- Sourcecode: Extraction of the Private and Public Key
- Sourcecode: Generation of the File Key
- Sourcecode: Generation of the Initialization Vector
- Sourcecode: Generation of a Key Pair

Encryption migration

Encryption format

Nextcloud still supports the legacy encryption scheme used for server side encryption where the encrypted files did not contain header information. This may still be used for installations that still have encrypted files from <= ownCloud 6. Files will be updated to the new encryption format once they are written again. However it is recommended to check if you have to still support this scheme.

Starting with version 20 for new installations the legacy encryption will be off by default. However if you are upgrading there is a migration path to check if you can disable legacy encryption.

Checking for old files

On the command line run:

```
occ encryption:scan:legacy-format
```

The command will tell you if you can remove the legacy encryption mode. If so set the *encryption.legacy_format_support* in your config.php to 'false'.

Transactional file locking

Nextcloud's Transactional File Locking mechanism locks files to avoid file corruption during normal operation. It performs these functions:

- Operates at a higher level than the filesystem, so you don't need to use a filesystem that supports locking
- Locks parent directories so they cannot be renamed during any activity on files inside the directories
- Releases locks after file transactions are interrupted, for example when a sync client loses the connection during an upload
- Manages locking and releasing locks correctly on shared files during changes from multiple users
- Manages locks correctly on external storage mounts
- Manages encrypted files correctly

What Transactional File locking is not for: it will not prevent multiple users from editing the same document, or give notice that other users are working on the same document. Multiple users can open and edit a file at the same time and Transactional File locking does not prevent this. Rather, it prevents simultaneous file saving.

File locking is enabled by default, using the database locking backend. This places a significant load on your database. Using `memcache.locking` relieves the database load and improves performance. Admins of Nextcloud servers with heavy workloads should install a memcache. (See Memory caching.)

To use a memcache with Transactional File Locking, you must install the Redis server and corresponding PHP module. After installing Redis you must enter a configuration in your `config.php` file like this example:

```
'filelocking.enabled' => true,
'memcache.locking' => '\OC\Memcache\Redis',
'redis' => array(
     'host' => 'localhost',
     'port' => 6379,
     'timeout' => 0.0,
     'password' => '', // Optional, if not defined no password will be used.
      ),
```

Note

For enhanced security it is recommended to configure Redis to require a password. See http://redis.io/topics/security for more information.

If you want to configure Redis to listen on an Unix socket (which is recommended if Redis is running on the same system as Nextcloud) use this example `config.php` configuration:

```
'filelocking.enabled' => true,
'memcache.locking' => '\OC\Memcache\Redis',
'redis' => array(
     'host' => '/var/run/redis/redis.sock',
     'port' => 0,
     'timeout' => 0.0,
      ),
```

See `config.sample.php` to see configuration examples for Redis, and for all supported memcaches.

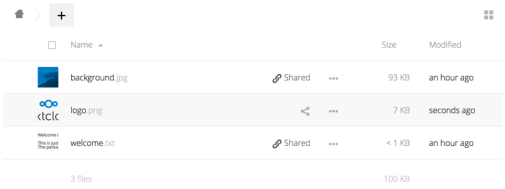
If you are on Ubuntu you can follow this guide for a complete installation from scratch.

Learn more about Redis at Redis. Memcached, the popular distributed memory caching system, is not suitable for the new file locking because it is not designed to store locks, and data can disappear from the cache at any time. Redis is a key-value store, and it guarantees that cached objects are available for as long as they are needed.

Previews configuration

The Nextcloud thumbnail system generates previews of files for all Nextcloud apps that display files, such as Files and Gallery.

The following image shows some examples of previews of various file types.



By default, Nextcloud can generate previews for the following filetypes:

- Images files
- Cover of MP3 files
- Text documents

Note

Technically Nextcloud can also generate the previews of other file types such as PDF, SVG or various office documents. Due to security concerns those providers have been disabled by default and are considered unsupported. While those providers are still available, we discourage enabling them, and they are not documented.

Parameters

Please notice that the Nextcloud preview system comes already with sensible defaults, and therefore it is usually unnecessary to adjust those configuration values.

Disabling previews:

Under certain circumstances, for example if the server has limited resources, you might want to consider disabling the generation of previews. Note that if you do this all previews in all apps are disabled, including the Gallery app, and will display generic icons instead of thumbnails.

Set the configuration option `enable_previews` in config.php to `false`:

```
<?php
  'enable_previews' => false,
```

Maximum preview size:

There are two configuration options to set the maximum size of a preview.

```
<?php
  'preview_max_x' => null,
  'preview_max_y' => null,
```

By default, both options are set to null. ‘Null’ is equal to no limit. Numeric values represent the size in pixels. The following code limits previews to a maximum size of 100×100px:

```
<?php
  'preview_max_x' => 100,
  'preview_max_y' => 100,
```

‘preview_max_x’ represents the x-axis and ‘preview_max_y’ represents the y-axis.

Maximum scale factor:

If a lot of small pictures are stored on the Nextcloud instance and the preview system generates blurry previews, you might want to consider setting a maximum scale factor. By default, pictures are upscaled to 10 times the original size:

```
<?php
  'preview_max_scale_factor' => 10,
```

If you want to disable scaling at all, you can set the config value to ‘1’:

```
<?php
  'preview_max_scale_factor' => 1,
```

If you want to disable the maximum scaling factor, you can set the config value to ‘null’:

```
<?php
  'preview_max_scale_factor' => null,
```

JPEG quality setting:

Default JPEG quality setting for preview images is ‘90’. Change this with:

```
occ config:app:set preview jpeg_quality --value="60"
```

Controlling file versions and aging

The Versions app (files_versions) expires old file versions automatically to ensure that users don’t exceed their storage quotas. This is the default pattern used to delete old versions:

- For the first second we keep one version
- For the first 10 seconds Nextcloud keeps one version every 2 seconds
- For the first minute Nextcloud keeps one version every 10 seconds
- For the first hour Nextcloud keeps one version every minute
- For the first 24 hours Nextcloud keeps one version every hour
- For the first 30 days Nextcloud keeps one version every day
- After the first 30 days Nextcloud keeps one version every week

The versions are adjusted along this pattern every time a new version is created.

The Versions app never uses more than 50% of the user’s currently available free space. If the stored versions exceed this limit, Nextcloud deletes the oldest file versions until it meets the disk space limit again.

You may alter the default pattern in `config.php`. The default setting is `auto`, which sets the default pattern:

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

Additional options are:

- `D, auto`
  Keep versions at least for D days, apply expiration rules to all versions that are older than D days
- `auto, D`
  Delete all versions that are older than D days automatically, delete other versions according to expiration rules
- `D1, D2`
  Keep versions for at least D1 days and delete when they exceed D2 days.
- `disabled`
  Disable the Versions app; no old file versions will be deleted.

Background job

To delete expired versions a background jobs runs every 30 minutes. It's possible to deactivate the background job and setup a (system) cron to expire the versions via occ.

Deactivate background job: `occ config:app:set --value=no files_versions background_job_expire_versions`

Activate background job: `occ config:app:delete files_versions background_job_expire_versions`

Expire versions: `occ versions:expire` or `occ versions:expire --quiet` (without the progress bar)

Deleted Items (trash bin)

If the trash bin app is enabled (default), this setting defines the policy for when files and folders in the trash bin will be permanently deleted.

The app allows for two settings, a minimum time for trash bin retention, and a maximum time for trash bin retention. Minimum time is the number of days a file will be kept, after which it may be deleted. Maximum time is the number of days at which it is guaranteed to be deleted. Both minimum and maximum times can be set together to explicitly define file and folder deletion. For migration purposes, this setting is installed initially set to "auto", which is equivalent to the default setting in Nextcloud.

You may alter the default pattern in `config.php`. The default setting is `auto`, which sets the default pattern:

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

Available values:

- `auto`
  default setting. keeps files and folders in the trash bin for 30 days and automatically deletes anytime after that if space is needed (note: files may not be deleted if space is not needed).
- `D, auto`
  keeps files and folders in the trash bin for D+ days, delete anytime if space needed (note: files may not be deleted if space is not needed)
- `auto, D`
  delete all files in the trash bin that are older than D days automatically, delete other files anytime if space needed
- `D1, D2`
  keep files and folders in the trash bin for at least D1 days and delete when exceeds D2 days (note: files will not be deleted automatically if space is needed)
- `disabled`
  trash bin auto clean disabled, files and folders will be kept forever

Background job

To permanently delete files a background jobs runs every 30 minutes. It's possible to deactivate the background job and setup a (system) cron to expire the versions via occ.

Deactivate background job: `occ config:app:set --value=no files_trashbin background_job_expire_trash`

Activate background job: `occ config:app:delete files_trashbin background_job_expire_trash`

Expire versions: `occ trashbin:expire` or `occ trashbin:expire --quiet` (without the progress bar)

File workflows

Files access control

Nextcloud's File Access Control app enables administrators to create and manage a set of rule groups. Each of the rule groups consists of one or more rules. If all rules of a group hold true, the group matches the request and access is being denied. The rules criteria range from IP address, to user groups, collaborative tags and some more.

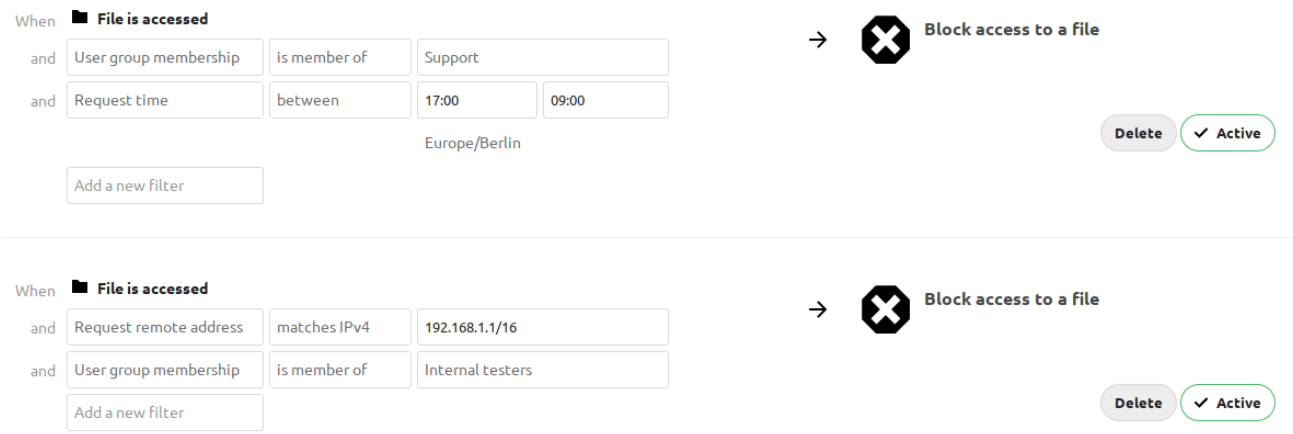
Denied access

If access to a file has been denied for a user, the user can not:

- Create/upload the file
- Modify the files
- Delete the file
- Download the file
- Synchronize the file with clients, such as the Nextcloud desktop and mobile clients

Examples

After installing the File Access Control app as described in Apps management navigate to the configuration and locate the settings for the Flow application.



The first rule group `Support only 9-5` denies any access to files for users of the Support user group, between 5pm and 9am.

The second rule group `Internal testing` prevents users of the Internal testers group to access files from outside of the local network.

Denying access to folders

The easiest way to block access to a folder, is to use a collaborative tag. As mentioned in the Available rules section below, either the file itself or one of the parents needs to have the given tag assigned.

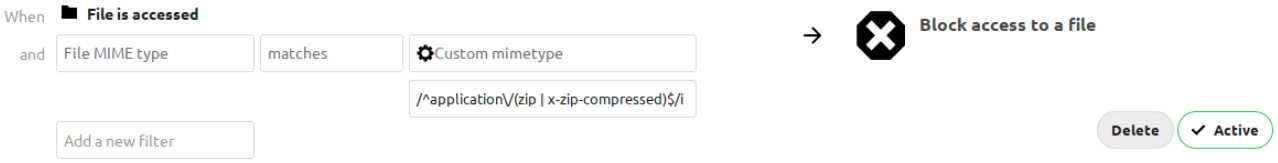
So you just need to assign the tag to the folder or file, and then block the tag with a rule group. The check is independent of the user's permissions for the tag. Therefor restricted and invisible tags are recommended, otherwise a user could remove and reassign the tag.

This example blocks access to any folder with the tag `Confidential`.

Prevent uploading of specific files

It's possible to prevent specific files from being uploaded to Nextcloud. You simply need to define a rule based on the mimetype and our powerful access control engine will block any attempt to upload the file. The safest way to define the rule is to use a regular expression, as it will help you cover all the known media types used for the type of file you're trying to block.

The following example prevents zip files from being uploaded by using the regular expression: `/^application\/(zip|x-zip-compressed)$/i`



Common misconfigurations

Blocking user groups

When trying to deny access to a group of users, make sure that sharing does not allow them to create a way back in. When users are able to create a public link, the users can log themselves out and visit their own public link to access the files. Since at this point they are no user and therefor no member of the blocked group, they will be able to read and change the file.

The recommended work around is to create the same rule again, and deny access for all users that are `not member of` a group, that contains all users of your installation.

External storage

While access to files in external storages is not possible via Nextcloud, users that have direct access to the external storage, can of course change files there directly. Therefor it is recommended to disable the `Allow users to mount external storage` option, when trying to to completely lock out users.

Available rules

All rules can also be inverted (from `is` to `is not`) using the operator option.

- **File collaborative tag:** Either the file itself, or any of the file owner's parent folders needs to be tagged with the tag.

  > **Note**
  >
  > Tags used in access control rules should be restricted tags, otherwise any user can remove the tag to access the file again. The best way to do this is with the Automated tagging of files.

- **File MIME type:** The MIME type of the file, e.g. `text/plain` for a text file or `httpd/unix-directory` for a folder.

> **Note**
>
> see mimetypealiases.dist.json for a full list of possible MIME types.

- **File name:** The name of the file (`is` and `is not` are case-insensitive)
- **File size:** The size of the file (*Only available on upload*)
- **Request remote address:** An IP range (either v4 or v6) for the accessing user
- **Request time:** Time span and timezone when the request happens
- **Request URL:** The URL which requests the file. (*This is the URL the file is served from, not the URL the user is currently looking at.*)
- **Request user agent:** The user agent of the users browser or client. Nextcloud desktop, Android and iOS clients are available as preconfigured options.
- **User group membership:** Whether the user is a member of the given group.

Automated tagging of files

Nextcloud's Files Automated Tagging app allows to assign collaborative tags to files and folders based on rules, similar to Files access control.

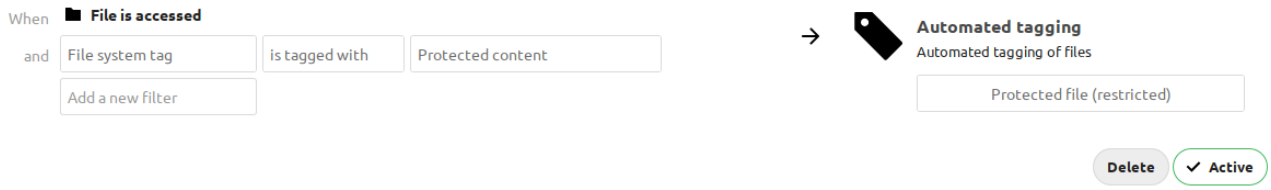
Assigning restricted and invisible tags

The main functionality of this app is to allow users to indirectly assign restricted and invisible tags to files they upload.

This is especially useful for retention and Files access control, so people that got the files shared can not remove the tag to stop the retention or allow access against the owners will.

Example

After installing the Files automated tagging app as described in Apps management navigate to the configuration and locate the Workflow settings.



In the example you can see a simple rule with only one condition. It will tag all files with the restricted tag `Protected file` that are uploaded into a folder that is tagged with `Protect content`. No user can remove the tag `Protected file` and therefor access control and retention both work fine without users being able to work around them.

In this case folder will be also tagged with tag `Protected file`, to avoid this, simply modify the rule to exclude Directory `httpd/unix-directory` from it.

Available rules

The available rules can be seen in the access control section: Available rules.

Executing actions

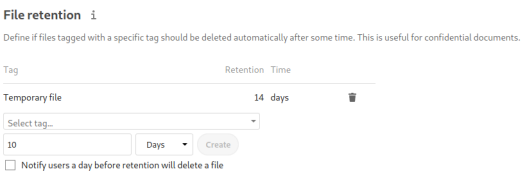
It is possible to execute actions like `convert to PDF` based on assigned tags. Nextcloud GmbH assists customers in this with hands-on help and documentation on our customer portal.

Retention of files

Nextcloud's Files Retention app allows to automatically delete files that are tagged with a collaborative tag and have a certain age.

Example

After installing the Retention app as described in Apps management navigate to the configuration and locate the Workflow settings.



The rule from the example will delete all files tagged with `Temporary file` after 14 days.

While creating a rule you can use the "Notify users a day before retention will delete a file" option to make sure the the users will get a notification before a file gets deleted.

Common misconfigurations

Public collaborative tag

Similar to Files access control retention should use `restricted` or `invisible` tags. Otherwise any user can remove the tag and the file is not removed after the given period. Use Automated tagging of files to assign such tags to newly uploaded files.

File age

Currently retention is based on the creation date of the file. The sync client sends the **original** creation date to the server, while uploading through the web interface will create a new file with a **new** creation date. We hope to be able to add a `upload date` to the filesystem soon, which would make more sense. Until then this potentially unexpected behavior has to be taken into account.

Groupware

Calendar / CalDAV

Events

You can customize the events user interface.

Hide export buttons

By default users can export their calendar data from the editor and the sidebar. Admins can disable this feature:

```
php occ config:app:set calendar hideEventExport --value=yes
```

Invitations

Nextcloud can send invitations for event attendees if this option is activated. Be sure to have configured the email server first so that the invitations go through. See Email.

You must also make sure the “Send invitations to attendees” setting is activated in the admin setting groupware section for the emails to be sent.

Birthday calendar

Contacts that have a birthday date filled are automatically added as events to a special Birthday calendar. If you deactivate this option, all users will no longer have this calendar.

When activating this option, users birthday calendars won’t be available right away because they need to be generated by a background task. See Using the occ command section DAV commands.

Reminder notifications

Nextcloud handles sending notifications for events.

Nextcloud currently handles two types of reminder notifications: Build-in Nextcloud notifications and email notifications. For the emails to be send, you’ll need a configured email server. See Email.

Make sure the “Send notifications for events” and the “Enable notifications for events via push” are activated in the admin setting groupware section for this feature to work.

Background jobs

Running background jobs can be an expensive task when there are a large number of events, reminders, event sharees and attendees. However, this needs to happen often enough so that the notifications are sent on time. To accomplish this you should use a dedicated `occ` command that runs more often than the standard `cron` system:

```
# crontab -u www-data -e
*/5 * * * * php -f /var/www/nextcloud/occ dav:send-event-reminders
```

See Using the occ command section Dav commands.

You’ll also need to change the sending mode from `background-job` to `occ`:

```
php occ config:app:set dav sendEventRemindersMode --value occ
```

If you don’t use this dedicated command, the reminders will just be sent as soon as possible when the background jobs run.

Event alarm types

Nextcloud allows users to set notification and email reminders for events. Admins can enforce one of the two options:

```
occ config:app:set calendar forceEventAlarmType --value=EMAIL
```

Allowed values are `EMAIL` (email) and `DISPLAY` (notification).

Note

This only enforces alarm types for events created with the Nextcloud Calendar. This setting has no influence for other connected applications.

FreeBusy

When logged-in, Nextcloud can return FreeBusy information for all users of the instance, to know when they are available so that you can schedule an event at the right time. If you don’t wish for users to have this capability, you can disable FreeBusy for the whole instance with the following setting:

```
php occ config:app:set dav disableFreeBusy --value yes
```

Subscriptions

Refresh rate

Calendar subscriptions are cached on server and refreshed periodically. The default refresh rate is one week, unless the subscription itself tells otherwise.

To set up a different default refresh rate, change the `calendarSubscriptionRefreshRate` option:

```
php occ config:app:set dav calendarSubscriptionRefreshRate --value "P1D"
```

Where the value is a DateInterval, for instance with the above command all of the Nextcloud instance's calendars would be refreshed every day.

Allow subscriptions on local network

Because of security issues, Nextcloud forbids subscriptions from local network hosts. If you need to allow this, change the following parameter to:

```
php occ config:app:set dav webcalAllowLocalAccess --value yes
```

Trash bin

Nextcloud supports a calendar, events and tasks trash bin.

The default delay before objects are purged from the trash bin is 30 days. A background job runs every 6 hours to clean up expired objects.

To set up a different retention period, change the `calendarRetentionObligation` option:

```
php occ config:app:set dav calendarRetentionObligation --value=2592000
```

Where the value is the number of seconds for the period. Setting the value to `0` disables the trash bin.

Resources and rooms

The Nextcloud CalDAV back end support resources and rooms. Resources and room can be booked for appointments and the system will schedule them so they can only be used once at a time. Those resources and rooms have to be provided by an app that provides a back end for this.

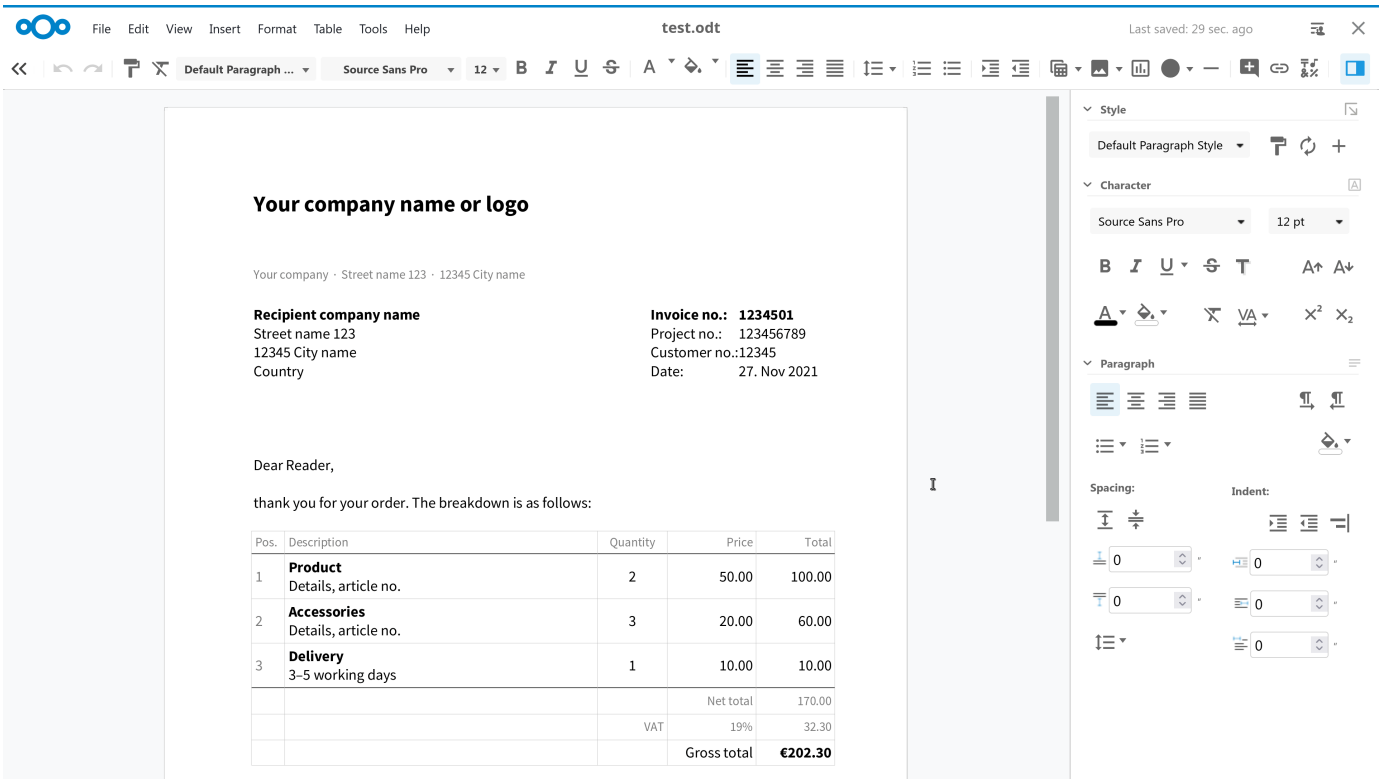
Once a back end app is installed the app typically allows admins or even users to define the resources, but this is subject of the specific implementation.

Nextcloud periodically queries all registered back ends. Therefore new and updated resources and rooms will show with a delay.

Known back ends

- Calendar Resource Management: database back end with CLI configuration for admins

Office



Nextcloud Office supports editing your documents in real time with multiple other editors, showing high fidelity, WYSIWYG rendering and preserving the layout and formatting of your documents.

Users can insert and reply to comments and invite others without a Nextcloud account for anonymous editing of files with a public link shared folder.

Nextcloud Office supports dozens of document formats including DOC, DOCX, PPT, PPTX, XLS, XLSX + ODF, Import/View Visio, Publisher and many more…

Nextcloud Office is based on the Collabora Online Development Edition (CODE) and is available free and under heavy development, adding features and improvements all the time! Enterprise users have access to the more stable, scalable Collabora Online Enterprise based version through a Nextcloud support subscription.

We are able to provide a solution for Online Office for the entire Nextcloud community through our partnership with Collabora with various deployment options. Enterprise users looking for a more reliable solution should contact Nextcloud Sales.

Installation

Nextcloud Office is built on Collabora Online which requires a dedicated service running next to the Nextcloud webserver stack. There are several ways to run the coolwsd service.

- **Nextcloud All In One:** Nextcloud Office comes preinstalled out of the box in the Nextcloud All In One setup and provides easy deployment and maintenance with most features included in this one Nextcloud instance.

For manual installations there are multiple options to get Nextcloud Office deployed:

- **Installation through distribution packages**

  There are packages for all major Linux distributions available which allow deploying a Collabora Online server through installing it through the regular package management. For an example installation guide on Ubuntu, see see: Installation example on Ubuntu 20.04

> Seealso
>
> https://www.collaboraoffice.com/code/linux-packages/
> https://sdk.collaboraonline.com/docs/installation/index.html

- **Installation through Docker**

  Docker images are available for deploying the Collabora Online server in container environments. For a detailed step by step guide, see: Installation example with Docker

  Seealso

  https://sdk.collaboraonline.com/docs/installation/CODE_Docker_image.html

- **Built-in CODE server**

  This app provides a built-in server with all of the document editing features of Collabora Online. Easy to install, for personal use or for small teams. A bit slower than a standalone server and without the advanced scalability features. Installation can be performed by enabling the according Nextcloud app. Further details can be found in the app documentation.

  Note

  This is the default option which works out of the box in most scenarios, however for improved performance it is highly recommended to switch to a dedicated Collabora Online installation using one of the other options.

Note

In most scenarios running a dedicated Collabora Online server will require some sort of reverse proxy to be setup in front of it. For more details see Reverse proxy.

Installation example on Ubuntu 20.04

Import signing keys:

```
cd /usr/share/keyrings && sudo wget https://collaboraoffice.com/downloads/gpg/collaboraonlin
```

Add repository:

```
sudo echo "deb https://www.collaboraoffice.com/repos/CollaboraOnline/CODE-ubuntu2004 ./" > /
```

Install packages

```
sudo apt update && sudo apt install coolwsd code-brand
```

Configuration

Edit /etc/coolwsd/coolwsd.xml. Collabora Online (coolwsd) service runs via systemd. After editing the configuration file, you have to restart the service:

```
sudo systemctl restart coolwsd
```

The default configuration is looking for an SSL certificate and key, which are not present, so probably it's the best to disable SSL, and optionally enable SSL termination, then set up the reverse proxy.

Seealso

Full configuration examples for reverse proxy setup can be found in the Collabora Online documentation: https://sdk.collaboraonline.com/docs/installation/Proxy_settings.html

```
sudo coolconfig set ssl.enable false
sudo coolconfig set ssl.termination true
sudo coolconfig set storage.wopi.host nextcloud.example.com
sudo coolconfig set-admin-password
sudo systemctl restart coolwsd
systemctl status coolwsd
```

Installation example with Docker

We'll describe how to get Nextcloud Office running on your server and how to integrate it into your Nextcloud using the docker image Nextcloud and Collabora built.

To install it the following dependencies are required:

- A host that can run a Docker container
- A subdomain or a second domain that the Collabora Online server can run on
- An Apache server with some enabled modules
- A valid SSL certificate for the domain that Collabora Online should run on
- A valid SSL certificate for your Nextcloud

Install the Collabora Online server

The following steps will download the Collabora Online docker. Make sure to replace "cloud.example.com" with the host that your own Nextcloud runs on. Also make sure to escape all dots with double backslashes (\), since this string will be evaluated as a regular expression (and your bash 'eats' the first backslash.) If you want to use the docker container with more than one Nextcloud, you'll need to use *domain=cloud.nextcloud.com|second.nextcloud.com* instead. (All hosts are separated by |.)

```
docker pull collabora/code
docker run -t -d -p 127.0.0.1:9980:9980 \
    -e 'domain=cloud\\.example\\.com' \
    --restart always \
    --cap-add MKNOD \
    collabora/code
```

That will be enough. Once you have done that the server will listen on "localhost:9980". Now we just need to configure the locally installed Apache reverse proxy.

Install the Apache reverse proxy

On a recent Ubuntu or Debian this should be possible using:

```
apt-get install apache2
a2enmod proxy proxy_wstunnel proxy_http ssl
```

Afterward, configure one VirtualHost properly to proxy the traffic. For security reason we recommend to use a subdomain such as office.example.com instead of running on the same domain. An example config can be found below:

```
########################################
# Reverse proxy for Collabora Online
########################################

AllowEncodedSlashes NoDecode
SSLProxyEngine On
ProxyPreserveHost On
```

```
# cert is issued for collaboraonline.example.com and we proxy to localhost
SSLProxyVerify None
SSLProxyCheckPeerCN Off
SSLProxyCheckPeerName Off

# static html, js, images, etc. served from coolwsd
# browser is the client part of Collabora Online
ProxyPass           /browser https://127.0.0.1:9980/browser retry=0
ProxyPassReverse    /browser https://127.0.0.1:9980/browser

# WOPI discovery URL
ProxyPass           /hosting/discovery https://127.0.0.1:9980/hosting/discovery retry=0
ProxyPassReverse    /hosting/discovery https://127.0.0.1:9980/hosting/discovery

# Capabilities
ProxyPass           /hosting/capabilities https://127.0.0.1:9980/hosting/capabilities retry=
ProxyPassReverse    /hosting/capabilities https://127.0.0.1:9980/hosting/capabilities

# Main websocket
ProxyPassMatch      "/cool/(.*)/ws$"      wss://127.0.0.1:9980/cool/$1/ws nocanon

# Admin Console websocket
ProxyPass           /cool/adminws wss://127.0.0.1:9980/cool/adminws

# Download as, Fullscreen presentation and Image upload operations
ProxyPass           /cool https://127.0.0.1:9980/cool
ProxyPassReverse    /cool https://127.0.0.1:9980/cool
# Compatibility with integrations that use the /lool/convert-to endpoint
ProxyPass           /lool https://127.0.0.1:9980/cool
ProxyPassReverse    /lool https://127.0.0.1:9980/cool
```

After configuring these do restart your apache using `systemctl restart apache2`.

> Seealso
>
> Full configuration examples for reverse proxy setup can be found in the Collabora Online documentation: https://sdk.collaboraonline.com/docs/installation/Proxy_settings.html

Configure the app in Nextcloud

Go to the Apps section and choose “Office & text” Install the “Collabora Online app” Admin -> Office -> Specify the server you have setup before (e.g. “https://office.example.com”) Congratulations, your Nextcloud has Collabora Online Office integrated!

Updating

Occasionally, new versions of this docker image are released with security and feature updates. We will of course let you know when that happens! This is how you upgrade to a new version:

Update the docker image:

```
docker pull collabora/code
```

List running docker containers:

```
docker ps
```

Stop and remove the Collabora Online container with the container id of the running one:

```
docker stop CONTAINER_ID
docker rm CONTAINER_ID
```

Start the new container:

```
docker run -t -d -p 127.0.0.1:9980:9980 -e 'domain=cloud\\.example\\.com' \
    --restart always --cap-add MKNOD collabora/code
```

Reverse proxy

The server part of Nextcloud Office (coolwsd daemon) is listening on port 9980 by default, and clients should be able to communicate with it through port 9980. However on most setups it is common to use a reverse proxy to more easily handle SSL termination and have a unified entrypoint for HTTP requests.

THe following rules should be in place to forward requests to the coolwsd daemon on port 9980:

- /browser
- /hosting/discovery
- /hosting/capabilities
- /cool/adminws
- /cool
- Web socket connections through /cool/(.*)/ws

Seealso

Full configuration examples can be found in the Collabora Online documentation: https://sdk.collaboraonline.com/docs/installation/Proxy_settings.html

Configuration

Nextcloud Office App Settings

Collabora Online Server

URL (and port) of the Collabora Online server that provides the editing functionality as a WOPI client. Collabora Online should use the same protocol (http:// or https://) as the server installation. Naturally, https:// is recommended.

Restrict usage to specific groups By default the app is enabled for all. When this setting is active, only members of specified groups can use Nextcloud Office.

Restrict edit to specific groups

By default all users can edit documents with Nextcloud Office. When this setting is active, only the members of specified groups can edit, others can only view documents.

Use OOXML by default for new files

By default new files created by users are in OpenDocument Format (ODF). When this setting is active, new files will be created in Office Open XML (OOXML) format.

Enable access for external apps

Nextcloud internally passes an access token to Collabora Online that is used later by it to do various operations. By default, it’s not possible to generate this token by 3rd parties; only Nextcloud can generate and pass it to Collabora Online.

In some applications, it might be necessary to generate the token by a 3rd party application. For this, one needs to add the 3rd party application (external apps) in this setting. You need to add an application identifier and a secret token. These credentials then can be used by the 3rd party application to make calls to *wopi/extapp/data/{fileId}* to fetch the access token and URL source for given fileId, both required to open a connection to Collabora Online.

Canonical webroot

Canonical webroot, in case there are multiple, for Collabora Online to use. Provide the one with least restrictions. E.g.: Use non-shibbolized webroot if this instance is accessed by both shibbolized and non-shibbolized webroots. You can ignore this setting if only one webroot is used to access this instance.

Additional configuration options

The coolwsd service allows additional configuration options which can be found in the Collabora Online documentation.

Previews

In order to allow Nextcloud to use the coolwsd conversion API to generate previews, the Nextcloud host IP needs to be added to the allow list:

```
sudo coolconfig set net.post_allow.host 10.0.0.4
```

Custom fonts

When you install coolwsd package, the post-install script will look for additional fonts on your system, and install them in the systemplate. If you install fonts to your system after installing coolwsd, you need to update the systemplate manually.

```
coolconfig update-system-template
```

Seealso

https://sdk.collaboraonline.com/docs/installation/Fonts.html

Migration from Collabora Online

Nextcloud Office is based on Collabora Online so for enabling all Nextcloud Office functionality it would be enough to update to the most recent release. Nextcloud Office is available since CODE 21.11.

Note

This upgrade guide is aimed for upgrading from CODE 6.4 to CODE 21.11.

Update the reverse proxy configuration

Due to naming changes in the Collabora Online releases it may be required to adjust reverse proxy configurations that are already in use for previously existing setups.

- Paths with `lool` have been renamed to `cool`
- Paths with `loleaflet` have been renamed to `browser`

Fully detailed reverse proxy configration guides for various solutions can be found at https://sdk.collaboraonline.com/docs/installation/Proxy_settings.html

Upgrade distribution packages

- The main service has been renamed from `loolwsd` to `coolwsd`
- The service rename also affects the location of the configuration file `/etc/coolwsd/coolwsd.xml`

Required upgrade steps:

- Stop the `loolwsd` service

- Backup `/etc/loolwsd/loolwsd.xml` configuration file.
- Remove `loolwsd` and `collaboraoffice*` packages.
- Change the version number in the repository URL, e.g. from 6.4 to 21.11
- Install the `coolwsd` package
- Adapt the new configuration file in `/etc/coolwsd/coolwsd.xml` to match your previous configuration
- Start and enable the `coolwsd` service

Upgrade the docker image

For upgrading the docker images it is enough to pull the latest CODE image from Docker Hub.

Troubleshooting

In case of connectivity issues, ensure that the following required connections are possible and not blocked by any firewall:

- The users browser can reach both the Nextcloud Server as well as the Collabora Online server through HTTP(S)
- The Nextcloud and the Collabora Online server are using the same protocol
- The Nextcloud server can reach the Collabora Online server through HTTP(S)
- The Collabora Online server can reach the Nextcloud server through HTTP(S)

Both the Nextcloud log as well as the Collabora Online server log may reveal more detailed error messages in case of connection issues.

- **Verify connectivity from the browser:**
  - https://office.example.com/hosting/capabilities
  - https://office.example.com/hosting/discovery
- **Verify connectivity from Nextcloud**
  - `curl https://office.example.com/hosting/capabilities`
  - `curl https://office.example.com/hosting/discovery`
- **Verify connection from the Collabora server**
  - `curl https://nextcloud.example.com/status.php`

Frequently asked questions

Issue: I get connection errors when trying to open documents

Be sure to check the error log from docker through `docker logs container-id`. If the logs note something like: `No acceptable WOPI hosts found matching the target host [YOUR NEXTCLOUD DOMAIN] in config.` Unauthorized WOPI host. Please try again later and report to your administrator if the issue persists. You might have started the docker container with the wrong URL. Be sure to triplecheck that you start it with the URL of your Nextcloud server, not the server where Collabora Online runs on.

Issue: Connection is not allowed errors.

It is possible your firewall is blocking connections. Try to start docker after you started the firewall, it makes changes to your iptables to enable Collabora Online to function.

Issue: We are sorry, this is an unexpected connection error. Please try again. error.

The Collabora Online app doesn't work at the moment, if you enable it only for certain groups. Remove the group filter in the App section.

Issue: Collabora Online doesn't handle my 100 users.

This docker image is designed for home usage. If you need a more scalable solution, consider a support subscription for a reliable, business-ready online office experience.

Issue: Collabora Online doesn’t work with Encryption.

Yes, this is currently unsupported.

Database configuration

Converting database type

You can convert a SQLite database to a better performing MySQL, MariaDB or PostgreSQL database with the Nextcloud command line tool. SQLite is good for testing and simple single-user Nextcloud servers, but it does not scale for multiple-user production users.

Run the conversion

First set up the new database, here called “new_db_name”. In Nextcloud root folder call

```
php occ db:convert-type [options] type username hostname database
```

The Options

- `--port="3306"` the database port (optional)
- `--password="mysql_user_password"` password for the new database. If omitted the tool will ask you (optional)
- `--clear-schema` clear schema (optional)
- `--all-apps` by default, tables for enabled apps are converted, use to convert also tables of deactivated apps (optional)
- `-n, --no-interaction` do not ask any interactive question

Note: The converter searches for apps in your configured app folders and uses the schema definitions in the apps to create the new table. So tables of removed apps will not be converted even with option `--all-apps`

For example

```
php occ db:convert-type --all-apps mysql oc_mysql_user 127.0.0.1 new_db_name
```

To successfully proceed with the conversion, you must type `yes` when prompted with the question `Continue with the conversion?`

On success the converter will automatically configure the new database in your Nextcloud config `config.php`.

Inconvertible tables

If you updated your Nextcloud instance, there might be remnants of old tables which are not used any more. The updater will tell you which ones these are.

```
The following tables will not be converted:
oc_permissions
...
```

You can ignore these tables. Here is a list of known old tables:

- oc_calendar_calendars
- oc_calendar_objects
- oc_calendar_share_calendar
- oc_calendar_share_event
- oc_fscache
- oc_log
- oc_media_albums
- oc_media_artists

- oc_media_sessions
- oc_media_songs
- oc_media_users
- oc_permissions
- oc_privatedata - this table was later added again by the app *privatedata* (https://apps.nextcloud.com/apps/privatedata) and is safe to be removed if that app is not enabled
- oc_queuedtasks
- oc_sharing

Database configuration

Nextcloud requires a database in which administrative data is stored. The following databases are currently supported:

- MySQL / MariaDB
- PostgreSQL
- Oracle

The MySQL or MariaDB databases are the recommended database engines.

Requirements

Choosing to use MySQL / MariaDB, PostgreSQL, or Oracle as your database requires that you install and set up the server software first.

Note

The steps for configuring a third party database are beyond the scope of this document. Please refer to the documentation for your specific database choice for instructions.

Database “READ COMMITTED” transaction isolation level

As discussed above Nextcloud is using the `TRANSACTION_READ_COMMITTED` transaction isolation level. Some database configurations are enforcing other transaction isolation levels. To avoid data loss under high load scenarios (e.g. by using the sync client with many clients/users and many parallel operations) you need to configure the transaction isolation level accordingly. Please refer to the MySQL manual for detailed information.

Parameters

For setting up Nextcloud to use any database, use the instructions in Installation wizard. You should not have to edit the respective values in the config/config.php. However, in special cases (for example, if you want to connect your Nextcloud instance to a database created by a previous installation of Nextcloud), some modification might be required.

Configuring a MySQL or MariaDB database

If you decide to use a MySQL or MariaDB database, ensure the following:

- The transaction isolation level is set to “READ-COMMITTED” in your MariaDB server configuration /etc/mysql/my.cnf to persist even after a restart of your database server.

  Verify the **transaction_isolation** and **binlog_format**:

```
[mysqld]
...
transaction_isolation = READ-COMMITTED
```

```
binlog_format = ROW
...
```

Your /etc/mysql/my.cnf could look like this:

```
[server]
skip_name_resolve = 1
innodb_buffer_pool_size = 128M
innodb_buffer_pool_instances = 1
innodb_flush_log_at_trx_commit = 2
innodb_log_buffer_size = 32M
innodb_max_dirty_pages_pct = 90
query_cache_type = 1
query_cache_limit = 2M
query_cache_min_res_unit = 2k
query_cache_size = 64M
tmp_table_size= 64M
max_heap_table_size= 64M
slow_query_log = 1
slow_query_log_file = /var/log/mysql/slow.log
long_query_time = 1

[client-server]
!includedir /etc/mysql/conf.d/
!includedir /etc/mysql/mariadb.conf.d/

[client]
default-character-set = utf8mb4

[mysqld]
character_set_server = utf8mb4
collation_server = utf8mb4_general_ci
transaction_isolation = READ-COMMITTED
binlog_format = ROW
innodb_large_prefix=on
innodb_file_format=barracuda
innodb_file_per_table=1
```

Please refer to the page in the MySQL manual.

- That you have installed and enabled the pdo_mysql extension in PHP
- That the **mysql.default_socket** points to the correct socket (if the database runs on the same server as Nextcloud).

Note

MariaDB is backwards compatible with MySQL. All instructions work for both. You will not need to replace mysql with anything.

The PHP configuration in /etc/php7/conf.d/mysql.ini could look like this:

```
# configuration for PHP MySQL module
extension=pdo_mysql.so

[mysql]
mysql.allow_local_infile=On
mysql.allow_persistent=On
mysql.cache_size=2000
mysql.max_persistent=-1
mysql.max_links=-1
```

```
mysql.default_port=
mysql.default_socket=/var/lib/mysql/mysql.sock  # Debian squeeze: /var/run/mysqld/mysqld.soc
mysql.default_host=
mysql.default_user=
mysql.default_password=
mysql.connect_timeout=60
mysql.trace_mode=Off
```

Now you need to create a database user and the database itself by using the MySQL command line interface. The database tables will be created by Nextcloud when you login for the first time.

To start the MySQL command line mode use:

```
mysql -uroot -p
```

Then a **mysql>** or **MariaDB [root]>** prompt will appear. Now enter the following lines and confirm them with the enter key:

```
CREATE USER 'username'@'localhost' IDENTIFIED BY 'password';
CREATE DATABASE IF NOT EXISTS nextcloud CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;
GRANT ALL PRIVILEGES on nextcloud.* to 'username'@'localhost';
FLUSH privileges;
```

You can quit the prompt by entering:

```
quit;
```

A Nextcloud instance configured with MySQL would contain the hostname on which the database is running, a valid username and password to access it, and the name of the database. The config/config.php as created by the Installation wizard would therefore contain entries like this:

```
<?php

  "dbtype"        => "mysql",
  "dbname"        => "nextcloud",
  "dbuser"        => "username",
  "dbpassword"    => "password",
  "dbhost"        => "localhost",
  "dbtableprefix" => "oc_",
```

In case of UTF8MB4 you will also find:

```
"mysql.utf8mb4" => true,
```

SSL for MySQL Database

Enabling SSL is only necessary if your database does not reside on the same server as your Nextcloud instance. If you do not connect over localhost and need to allow remote connections then you should enable SSL. This just covers the SSL database configuration on the Nextcloud server. First you need to configure your database server accordingly.

```
'dbdriveroptions' => [
  \PDO::MYSQL_ATTR_SSL_KEY => '/../ssl-key.pem',
  \PDO::MYSQL_ATTR_SSL_CERT => '/../ssl-cert.pem',
  \PDO::MYSQL_ATTR_SSL_CA => '/../ca-cert.pem',
  \PDO::MYSQL_ATTR_SSL_VERIFY_SERVER_CERT => true,
],
```

Adjust the paths to the pem files for your environment.

PostgreSQL database

If you decide to use a PostgreSQL database make sure that you have installed and enabled the PostgreSQL extension in PHP. The PHP configuration in /etc/php7/conf.d/pgsql.ini could look like this:

```
# configuration for PHP PostgreSQL module
extension=pdo_pgsql.so
extension=pgsql.so

[PostgresSQL]
pgsql.allow_persistent = On
pgsql.auto_reset_persistent = Off
pgsql.max_persistent = -1
pgsql.max_links = -1
pgsql.ignore_notice = 0
pgsql.log_notice = 0
```

The default configuration for PostgreSQL (at least in Ubuntu 14.04) is to use the peer authentication method. Check /etc/postgresql/9.3/main/pg_hba.conf to find out which authentication method is used in your setup. To start the postgres command line mode use:

```
sudo -u postgres psql -d template1
```

Then a **template1=#** prompt will appear. Now enter the following lines and confirm them with the enter key:

```
CREATE USER username CREATEDB;
CREATE DATABASE nextcloud OWNER username;
```

You can quit the prompt by entering:

```
\q
```

A Nextcloud instance configured with PostgreSQL would contain the path to the socket on which the database is running as the hostname, the system username the PHP process is using, and an empty password to access it, and the name of the database. The config/config.php as created by the Installation wizard would therefore contain entries like this:

```
<?php

  "dbtype"        => "pgsql",
  "dbname"        => "nextcloud",
  "dbuser"        => "username",
  "dbpassword"    => "",
  "dbhost"        => "/var/run/postgresql",
  "dbtableprefix" => "oc_",
```

Note

The host actually points to the socket that is used to connect to the database. Using localhost here will not work if postgreSQL is configured to use peer authentication. Also note that no password is specified, because this authentication method doesn't use a password.

If you use another authentication method (not peer), you'll need to use the following steps to get the database setup: Now you need to create a database user and the database itself by using the PostgreSQL command line interface. The database tables will be created by Nextcloud when you login for the first time.

To start the postgres command line mode use:

```
psql -hlocalhost -Upostgres
```

Then a **postgres=#** prompt will appear. Now enter the following lines and confirm them with the enter key:

```
CREATE USER username WITH PASSWORD 'password';
CREATE DATABASE nextcloud TEMPLATE template0 ENCODING 'UNICODE';
ALTER DATABASE nextcloud OWNER TO username;
GRANT ALL PRIVILEGES ON DATABASE nextcloud TO username;
```

You can quit the prompt by entering:

```
\q
```

A Nextcloud instance configured with PostgreSQL would contain the hostname on which the database is running, a valid username and password to access it, and the name of the database. The config/config.php as created by the Installation wizard would therefore contain entries like this:

```
<?php

  "dbtype"        => "pgsql",
  "dbname"        => "nextcloud",
  "dbuser"        => "username",
  "dbpassword"    => "password",
  "dbhost"        => "localhost",
  "dbtableprefix" => "oc_",
```

Troubleshooting

How to work around “general error: 2006 MySQL server has gone away”

The database request takes too long and therefore the MySQL server times out. It’s also possible that the server is dropping a packet that is too large. Please refer to the manual of your database for how to raise the configuration options `wait_timeout` and/or `max_allowed_packet`.

Some shared hosters are not allowing the access to these config options. For such systems Nextcloud is providing a `dbdriveroptions` configuration option within your config/config.php where you can pass such options to the database driver. Please refer to Configuration Parameters for an example.

How can I find out if my MySQL/PostgreSQL server is reachable?

To check the server’s network availability, use the ping command on the server’s host name (db.server.com in this example):

```
ping db.server.com

PING db.server.com (ip-address) 56(84) bytes of data.
64 bytes from your-server.local.lan (192.168.1.10): icmp_req=1 ttl=64 time=3.64 ms
64 bytes from your-server.local.lan (192.168.1.10): icmp_req=2 ttl=64 time=0.055 ms
64 bytes from your-server.local.lan (192.168.1.10): icmp_req=3 ttl=64 time=0.062 ms
```

For a more detailed check whether the access to the database server software itself works correctly, see the next question.

How can I find out if a created user can access a database?

The easiest way to test if a database is accessible is by starting the command line interface:

MySQL:

Assuming the database server is installed on the same system you’re running the command from, use:

```
mysql -uUSERNAME -p
```

To access a MySQL installation on a different machine, add the -h option with the respective host name:

```
mysql -uUSERNAME -p -h HOSTNAME

mysql> SHOW VARIABLES LIKE "version";
+---------------+--------+
| Variable_name | Value  |
+---------------+--------+
| version       | 8.0.22 |
+---------------+--------+
```

```
1 row in set (0.00 sec)
mysql> quit
```

PostgreSQL:

Assuming the database server is installed on the same system you're running the command from, use:

```
psql -Uusername -dnextcloud
```

To access a PostgreSQL installation on a different machine, add the -h option with the respective host name:

```
psql -Uusername -dnextcloud -h HOSTNAME

postgres=# SELECT version();
PostgreSQL 8.4.12 on i686-pc-linux-gnu, compiled by GCC gcc (GCC) 4.1.3 20080704 (prerelease
(1 row)
postgres=# \q
```

Useful SQL commands

Show Database Users:

```
MySQL     : SELECT User,Host FROM mysql.user;
PostgreSQL: SELECT * FROM pg_user;
```

Show available Databases:

```
MySQL     : SHOW DATABASES;
PostgreSQL: \l
```

Show Nextcloud Tables in Database:

```
MySQL     : USE nextcloud; SHOW TABLES;
PostgreSQL: \c nextcloud; \d
```

Quit Database:

```
MySQL     : quit
PostgreSQL: \q
```

Enabling MySQL 4-byte support

Note

Be sure to backup your database before performing this database upgrade.

In order to use Emojis (textbased smilies) on your Nextcloud server with a MySQL database, the installation needs to be tweaked a bit.

Note

This manual only covers MySQL 8 or newer and MariaDB 10.3 or newer. If you use an older version, please check an older version of the documentation

1. Make sure the following InnoDB settings are set on your MySQL server:

```
[mysqld]
innodb_file_per_table=1
```

Note:

```
mysql> show variables like 'innodb_file_per_table';
+-----------------------+-------+
| Variable_name         | Value |
+-----------------------+-------+
| innodb_file_per_table | ON    |
+-----------------------+-------+
1 row in set (0.00 sec)
```

2. Open a shell, change dir (adjust `/var/www/nextcloud` to your nextcloud location if needed), and put your nextcloud instance in maintenance mode, if it isn't already:

```
$ cd /var/www/nextcloud
$ sudo -u www-data php occ maintenance:mode --on
```

3. Restart the MySQL server in case you changed the configuration in step 1.
4. Change your databases character set and collation:

```
ALTER DATABASE nextcloud CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;
```

5. Set the `mysql.utf8mb4` config to true in your config.php:

```
$ sudo -u www-data php occ config:system:set mysql.utf8mb4 --type boolean --value="true"
```

6. Convert all existing tables to the new collation by running the repair step:

```
$ sudo -u www-data php occ maintenance:repair
```

Note

This will also change the *ROW_FORMAT* to *COMPRESSED* for your tables, to make sure the used database storage size is not getting out of hand.

7. Disable maintenance mode:

```
$ sudo -u www-data php occ maintenance:mode --off
```

Now you should be able to use Emojis in your file names, calendar events, comments and many more.

Note

Also make sure your backup strategy still work. If you use `mysqldump` make sure to add the `--default-character-set=utf8mb4` option. Otherwise your backups are broken and restoring them will result in ? instead of the emojis, making files inaccessible.

BigInt (64bit) identifiers

Nextcloud uses big integers to store identifiers and auto-increment keys in the database. Because changing columns on huge tables can take quite a while (up to hours or days) depending on the number of files in the Nextcloud instance, this migration on the filecache and activity table has to be triggered manually by a console command.

The command can safely be executed. It will show a success message when there is nothing to do:

```
sudo -u www-data php occ db:convert-filecache-bigint
All tables already up to date!
```

or otherwise ask for confirmation, before performing the heavy actions:

```
sudo -u www-data php occ db:convert-filecache-bigint
This can take up to hours, depending on the number of files in your instance!
Continue with the conversion (y/n)? [n]
```

to suppress the confirmation message append `--no-interaction` to the argument list:

```
sudo -u www-data php occ db:convert-filecache-bigint --no-interaction
```

Note

Similar to a normal update, you should shutdown your Apache or nginx server or enable maintenance mode before running the command to avoid issues with your sync clients.

Splitting databases

Warning

This is still proof-of-concept level. Use with care.

In order to scale at some point it might make sense to split out some tables or apps which allow it as they might be better off with a different replication methods, etc.

A first attempt we do right now with the activity table. In order to make use of this, the app/table needs to match the following criteria:

- No other apps are allowed to have queries directly to the table
- No JOINs are performed between this table and any other tables not on this new separate connection
- The app needs to support a connection parameter prefix

In case of the activity app the prefix is `activity_`. If a database config is not specified it falls back to the normal database configuration option for this value:

- `activity_dbuser` falling back to `dbuser`
- `activity_dbpassword` falling back to `dbpassword`
- `activity_dbname` falling back to `dbname`
- `activity_dbhost` falling back to `dbhost`
- `activity_dbport` falling back to `dbport`
- `activity_dbdriveroptions` falling back to `dbdriveroptions`

Note

It is not possible to use a different database type (SQLite, MySQL, PostrgreSQL, Oracle) for a split database. Also in case of MySQL and MariaDB the utf8mb4 option needs to be the same on both databases.

Initial splitting

For the initial split the affected tables have to be copied over to the new database, in case of the activity app these are:

- `oc_activity`
- `oc_activity_mq`

1. Enable maintenance mode
2. Make sure optional database changes are applied:

1. `occ db:convert-mysql-charset`
2. `occ db:convert-filecache-bigint`
3. `occ db:add-missing-columns`
4. `occ db:add-missing-indices`
5. `occ db:add-missing-primary-keys`

3. Specify the desired configuration values 3. Copy the 2 tables to the new database 4. Disable maintenance mode

Migrations on updates

We will try to avoid migrations on those tables in the future, but it might be necessary at some point. We hope to have a dedicated plan by the time this happens. For now a potential way would be:

1. Enable maintenance mode
2. Update as usual
3. Execute manual queries for schema changes provided by the app authors
4. Execute manual queries for data changes provided by the app authors
5. Disable maintenance mode

Mimetypes management

Mimetype aliases

Nextcloud allows you to create aliases for mimetypes, so that you can display custom icons for files. For example, you might want a nice audio icon for audio files instead of the default file icon.

By default Nextcloud is distributed with `nextcloud/resources/config/mimetypealiases.dist.json`. Do not modify this file, as it will be replaced when Nextcloud is updated. Instead, create your own `nextcloud/config/mimetypealiases.json` file with your custom aliases. Use the same syntax as in `nextcloud/resources/config/mimetypealiases.dist.json`.

Once you have made changes to your `mimetypealiases.json`, use the `occ` command to propagate the changes through the system. This example is for Ubuntu Linux:

```
$ sudo -u www-data php occ maintenance:mimetype:update-js
```

See Using the occ command to learn more about `occ`.

Some common mimetypes that may be useful in creating aliases are:

image
: Generic image

image/vector
: Vector image

audio
: Generic audio file

x-office/document
: Word processed document

x-office/spreadsheet
: Spreadsheet

x-office/presentation
: Presentation

text
: Generic text document

text/code

Source code

Mimetype mapping

Nextcloud allows administrators to specify the mapping of a file extension to a mimetype. For example files ending in `mp3` map to `audio/mpeg`. Which then in turn allows Nextcloud to show the audio icon.

By default Nextcloud comes with `mimetypemapping.dist.json`. This is a simple json array. Administrators should not update this file as it will get replaced on upgrades of Nextcloud. Instead the file `mimetypemapping.json` should be created and modified, this file has precedence over the shipped file.

Icon retrieval

When an icon is retrieved for a mimetype, if the full mimetype cannot be found, the search will fallback to looking for the part before the slash. Given a file with the mimetype 'image/my-custom-image', if no icon exists for the full mimetype, the icon for 'image' will be used instead. This allows specialised mimetypes to fallback to generic icons when the relevant icons are unavailable.

Maintenance

Backup

To backup a Nextcloud installation there are four main things you need to retain:

1. The config folder
2. The data folder
3. The theme folder
4. The database

Maintenance mode

`maintenance:mode` locks the sessions of logged-in users and prevents new logins in order to prevent inconsistencies of your data. You must run `occ` as the HTTP user, like this example on Ubuntu Linux:

```
$ sudo -u www-data php occ maintenance:mode --on
```

You may also put your server into this mode by editing config/config.php. Change `"maintenance" => false` to `"maintenance" => true`:

```
<?php

 "maintenance" => true,
```

Don't forget to change it back to `false` when you are finished.

Backup folders

Simply copy your config, data and theme folders (or even your whole Nextcloud install and data folder) to a place outside of your Nextcloud environment. You could use this command:

```
rsync -Aavx nextcloud/ nextcloud-dirbkp_`date +"%Y%m%d"`/
```

Backup database

Warning

Before restoring a backup see Restoring backup

MySQL/MariaDB

MySQL or MariaDB, which is a drop-in MySQL replacement, is the recommended database engine. To backup MySQL/MariaDB:

```
mysqldump --single-transaction -h [server] -u [username] -p[password] [db_name] > nextcloud-
```

If you use enabled MySQL/MariaDB 4-byte support (Enabling MySQL 4-byte support, needed for emoji), you will need to add `--default-character-set=utf8mb4` like this:

```
mysqldump --single-transaction --default-character-set=utf8mb4 -h [server] -u [username] -p[
```

SQLite

```
sqlite3 data/owncloud.db .dump > nextcloud-sqlbkp_`date +"%Y%m%d"`.bak
```

PostgreSQL

```
PGPASSWORD="password" pg_dump [db_name] -h [server] -U [username] -f nextcloud-sqlbkp_`date
```

Restoring backup

To restore a Nextcloud installation there are four main things you need to restore:

1. The configuration directory
2. The data directory
3. The database
4. The theme directory

> **Note**
>
> You must have both the database and data directory. You cannot complete restoration unless you have both of these.

When you have completed your restoration, also make sure to run the maintenance:data-fingerprint command afterwards, to ensure your sync clients can recover from the restored backup.

Restore folders

> **Note**
>
> This guide assumes that your previous backup is called “nextcloud-dirbkp”

Simply copy your configuration and data folder (or even your whole Nextcloud install and data folder) to your Nextcloud environment. You could use this command:

```
rsync -Aax nextcloud-dirbkp/ nextcloud/
```

Restore database

> **Warning**
>
> Before restoring a backup you need to make sure to delete all existing database tables.

The easiest way to do this is to drop and recreate the database. SQLite does this automatically.

MySQL

MySQL is the recommended database engine. To restore MySQL:

```
mysql -h [server] -u [username] -p[password] -e "DROP DATABASE nextcloud"
mysql -h [server] -u [username] -p[password] -e "CREATE DATABASE nextcloud"
```

If you use UTF8 with multibyte support (e.g. for emojis in filenames), use:

```
mysql -h [server] -u [username] -p[password] -e "DROP DATABASE nextcloud"
mysql -h [server] -u [username] -p[password] -e "CREATE DATABASE nextcloud CHARACTER SET utf
```

PostgreSQL

```
PGPASSWORD="password" psql -h [server] -U [username] -d template1 -c "DROP DATABASE \"nextcl
PGPASSWORD="password" psql -h [server] -U [username] -d template1 -c "CREATE DATABASE \"next
```

Restoring

Note

This guide assumes that your previous backup is called “nextcloud-sqlbkp.bak”

MySQL

MySQL is the recommended database engine. To restore MySQL:

```
mysql -h [server] -u [username] -p[password] [db_name] < nextcloud-sqlbkp.bak
```

SQLite

```
rm data/owncloud.db
sqlite3 data/owncloud.db < nextcloud-sqlbkp.bak
```

PostgreSQL

```
PGPASSWORD="password" psql -h [server] -U [username] -d nextcloud -f nextcloud-sqlbkp.bak
```

How to upgrade

There are three ways to upgrade your Nextcloud server:

- With the Updater.
- Manually upgrading with the Nextcloud `.tar` archive from our Download page.
- Upgrading via the snap packages.
- Manually upgrading is also an option for users on shared hosting; download and unpack the Nextcloud tarball to your PC. Delete your existing Nextcloud files, except `data/` and `config/` files, on your hosting account. Then transfer the new Nextcloud files to your hosting account, again preserving your existing `data/` and `config/` files.

When an update is available for your Nextcloud server, you will see a notification at the top of your Nextcloud Web interface. When you click the notification it brings you here, to this page.

It is best to keep your Nextcloud server upgraded regularly, and to install all point releases and major releases. Major releases are 18, 19 or 20. Point releases are intermediate releases for each major release. For example 18.0.4 and 19.0.2 are point releases.

Nextcloud must be upgraded step by step:

- Before you can upgrade to the next major release, Nextcloud upgrades to the latest point release.
- Then run the upgrade again to upgrade to the next major release's latest point release.
- **You cannot skip major releases.** Please re-run the upgrade until you have reached the highest available (or applicable) release.
- Example: 18.0.5 -> 18.0.11 -> 19.0.5 -> 20.0.2

Upgrading is disruptive. Your Nextcloud server will be put into maintenance mode, so your users will be locked out until the upgrade is completed. Large installations may take several hours to complete the upgrade. Nevertheless usual upgrade times even for bigger installations are in the range of a few minutes.

Warning

Downgrading is not supported and risks corrupting your data! If you want to revert to an older Nextcloud version, make a new, fresh installation and then restore your data from backup. Before doing this, file a support ticket (if you have paid support) or ask for help in the Nextcloud forums to see if your issue can be resolved without downgrading.

Update notifications

Nextcloud has an update notification app, that informs the administrator about the availability of an update. Then you decide which update method to use.

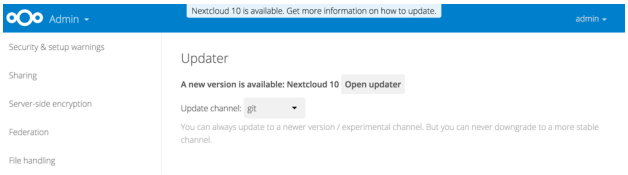


Figure 1: The top banner is the update notification that is shown on every page, and the Updates section can be found in the admin page

From there the web based updater can be used to fetch this new code. There is also an CLI based updater available, that does exactly the same as the web based updater but on the command line.

Prerequisites

You should always maintain regular backups and make a fresh backup before every upgrade.

Then review third-party apps, if you have any, for compatibility with the new Nextcloud release. Any apps that are not developed by Nextcloud show a 3rd party designation. **Install unsupported apps at your own risk**. Then, before the upgrade, all 3rd party apps must be disabled. After the upgrade is complete you may re-enable them.

Maintenance mode

You can put your Nextcloud server into maintenance mode before performing upgrades, or for performing troubleshooting or maintenance. Please see Using the occ command to learn how to put your server into the maintenance mode (`maintenance:mode`) or execute repair commands (`maintenance:repair`) with the `occ` command.

The build-in Updater does this for you before replacing the existing Nextcloud code with the code of the new Nextcloud version.

`maintenance:mode` locks the sessions of logged-in users and prevents new logins. This is the mode to use for upgrades. You must run `occ` as the HTTP user, like this example on Ubuntu Linux:

```
$ sudo -u www-data php occ maintenance:mode --on
```

You may also put your server into this mode by editing config/config.php. Change `"maintenance" => false` to `"maintenance" => true`:

```
<?php

 "maintenance" => true,
```

Then change it back to `false` when you are finished.

Manual steps during upgrade

Some operation can be quite time consuming. Therefore we decided not to add them to the normal upgrade process. We recommend to run them manually after the upgrade was completed. Below you find a list of this commands.

Long running migration steps

From time to time we do some changes to the database layout that take a lot of time, but can be executed while Nextcloud stays online. Thus we moved them into a separate command that an administrator can execute on the CLI without the need to lock the instance into maintenance mode (at least for some of them). The instance will also work without those changes applied, but performance is improved significantly by them. There is also always an hint in the setup checks of the admin settings web interface.

Those include for example:

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

Upgrade via built-in updater

The built-in updater automates many of the steps of upgrading a Nextcloud installation. It is useful for installations that do not have root access, such as shared hosting, for installations with a smaller number of users and data, and it automates updating manual installations.

> **Warning**
>
> **Downgrading** is not supported and risks corrupting your data! If you want to revert to an older Nextcloud version, install it from scratch and then restore your data from backup. Before doing this, file a support ticket if you have paid support or ask for help in the Nextcloud forums to see if your issue can be resolved without downgrading.

You should maintain regular backups (see Backup), and make a backup before every update. The built-in updater does not backup your database or data directory.

What does the updater do?

> **Note**
>
> The updater itself only replaces the existing files with the ones from the version it updates to. The migration steps needs to be executed afterwards. The command line mode provides a way to do this right after the code was successfully replaced.

The built-in updater performs these operations:

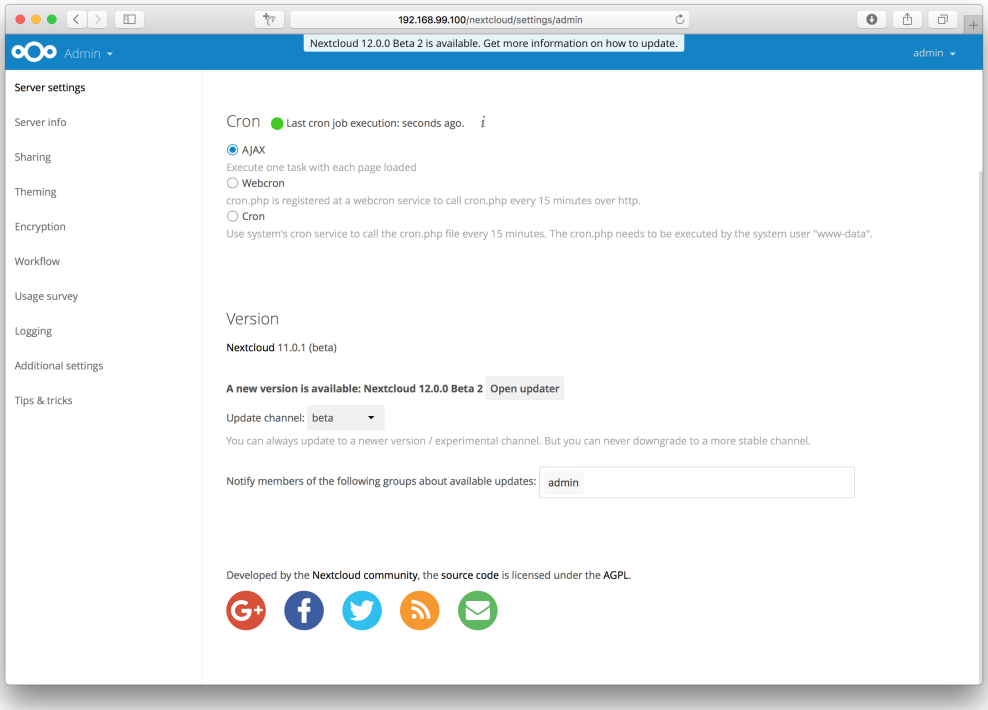
- **Check for expected files:** checks if only the expected files of a Nextcloud installation are present, because it turned out that some files that were left in the Nextcloud directory caused side effects that risked the update procedure.

- **Check for write permissions:** checks if all files that need to be writable during the update procedure are actually writable.
- **Enable maintenance mode:** enables the maintenance mode so that no other actions are executed while running the update of the code.
- **Create backup:** creates a backup of the existing code base in `/updater-INSTANCEID/backups/nextcloud-CURRENTVERSION/` inside of the data directory (this does not contain the `/data` directory nor the database).
- **Downloading:** downloads the code in the version it should update to. This is also shown in the web UI before the update is started. This archive is downloaded to `/updater-INSTANCEID/downloads/`.
- **Extracting:** extracts the archive to the same folder.
- **Replace entry points:** replaces all Nextcloud entry points with dummy files so that when those files are replaced all clients still get the proper maintenance mode response. Examples for those endpoints are `index.php`, `remote.php` or `ocs/v1.php`.
- **Delete old files:** deletes all files except the above mentioned entry points, the data and config dir as well as non-shipped apps and themes. (And the updater itself of course)
- **Move new files in place:** moves the files from the extracted archive in place.
- **Keep maintenance mode active?:** asks you if the maintenance mode should be kept active. This allows the admin to use the web based updater but run the actual migration steps (`occ upgrade`) on the command line. If the maintenance mode is kept active command line access is required. To use the web based upgrade page disable the maintenance mode and click the link to get to the upgrade page. (This step is only available in the web based updater.)
- **Done** the update of the code is done and you either need to go to the linked page or to the command line to finish the upgrade by executing the migration steps.

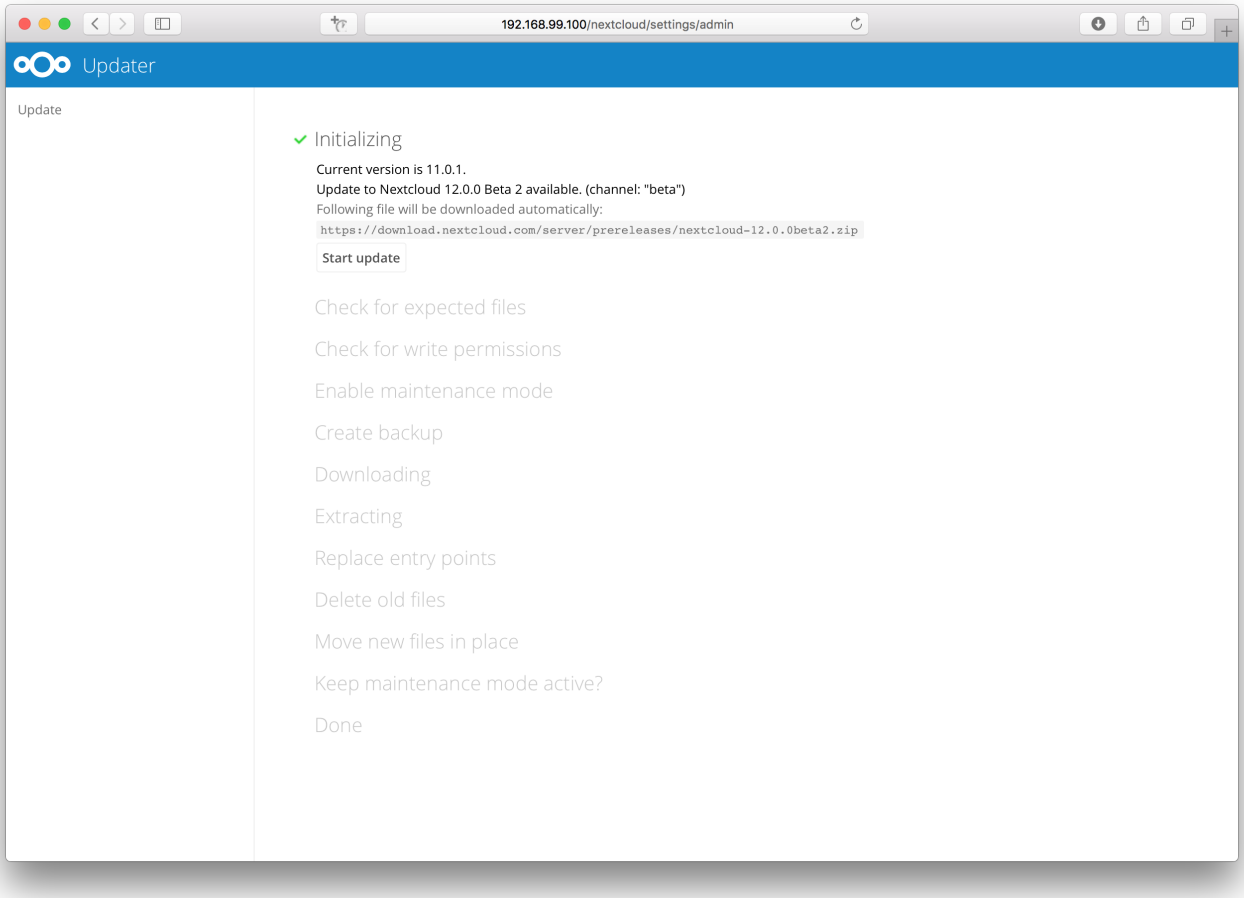
Using the web based updater

Using the built-in updater to update your Nextcloud installation is just a few steps:

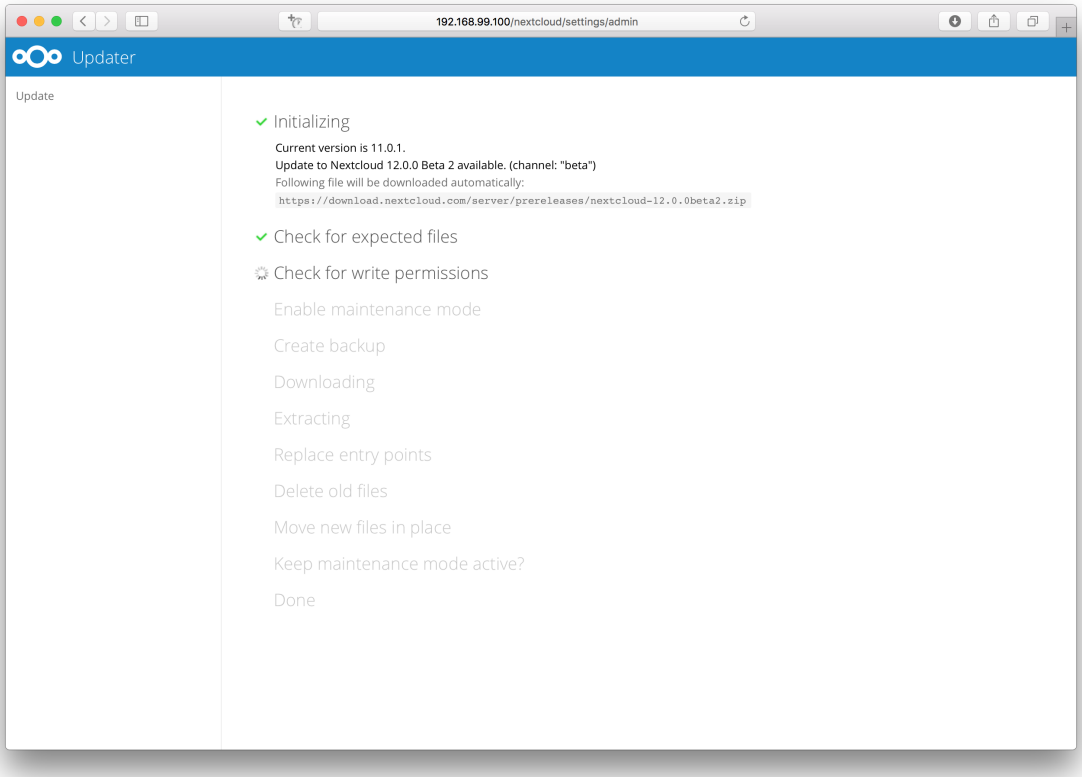
1. You should see a notification at the top of any Nextcloud page when there is a new update available. Go to the admin settings page and scroll to the section “Version”. This section has a button to open the updater. This section as well as the update notification is only available if the update notication app is enabled in the apps management.



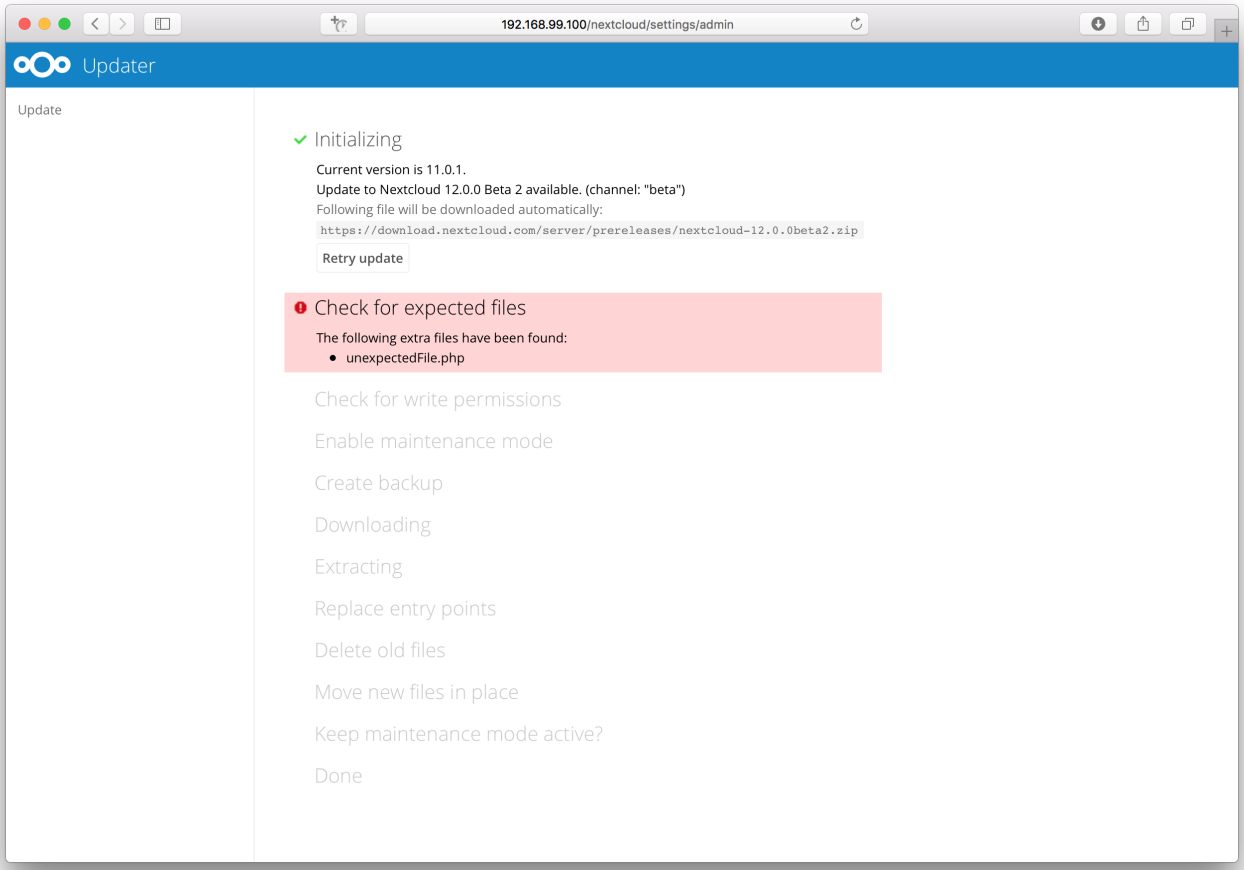
2. Click the button “Open updater”.



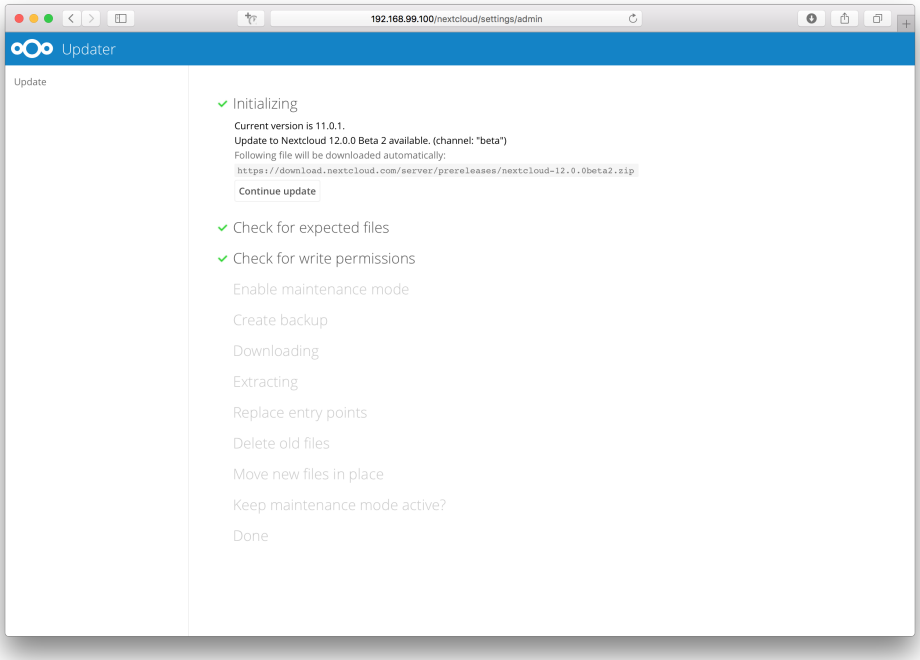
3. Verify the information that is shown and click the button “Start update” to start the update.



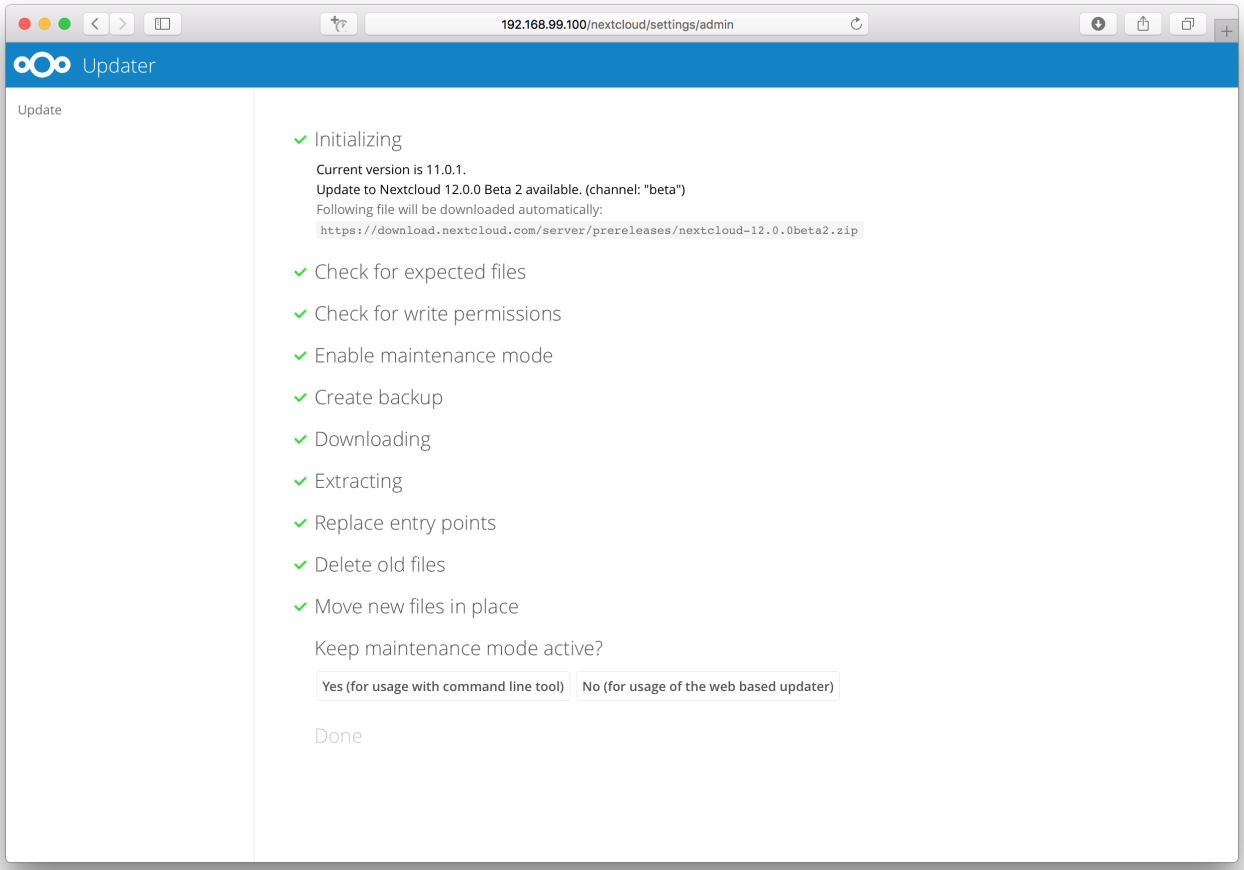
4. In case an error happens or the check failed the updater stops processing and gives feedback. You can now try to solve the problem and click the “Retry update” button. This will continue the update and re-run the failed step. It will not re-run the previous succeeded steps.



5. In case you close the updater, before it finished you can just open the updater page again and proceed at the last succeeded step. Closing the web page will still execute the running step but will not continue with the next one, because this is triggered by the open updater page.



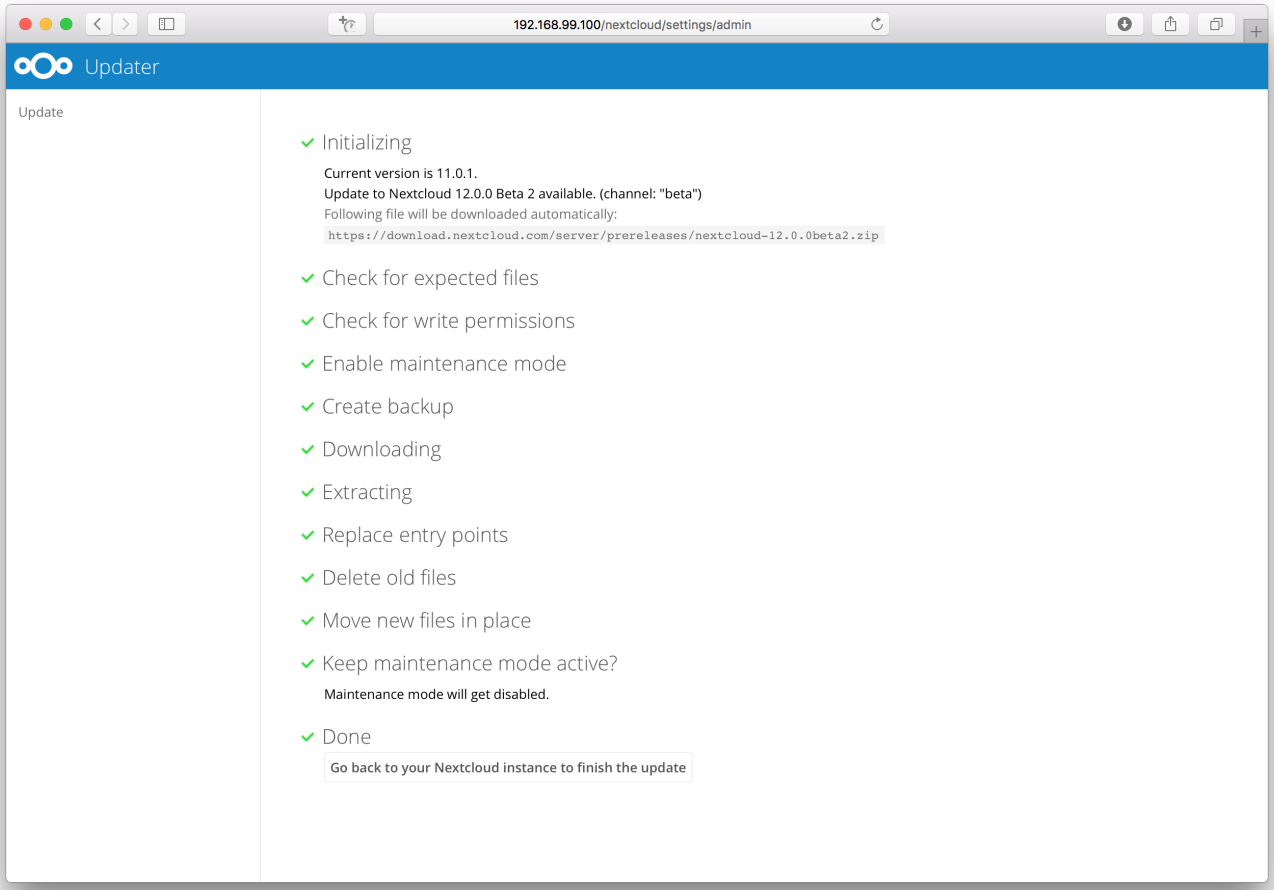
6. Once all steps are executed the updater will ask you a final question: “Keep maintenance mode active?”. This allows you to use either the web based upgrade page or the command line based upgrade procedure (`occ upgrade`). Command line access is required if the maintenance mode is kept active.

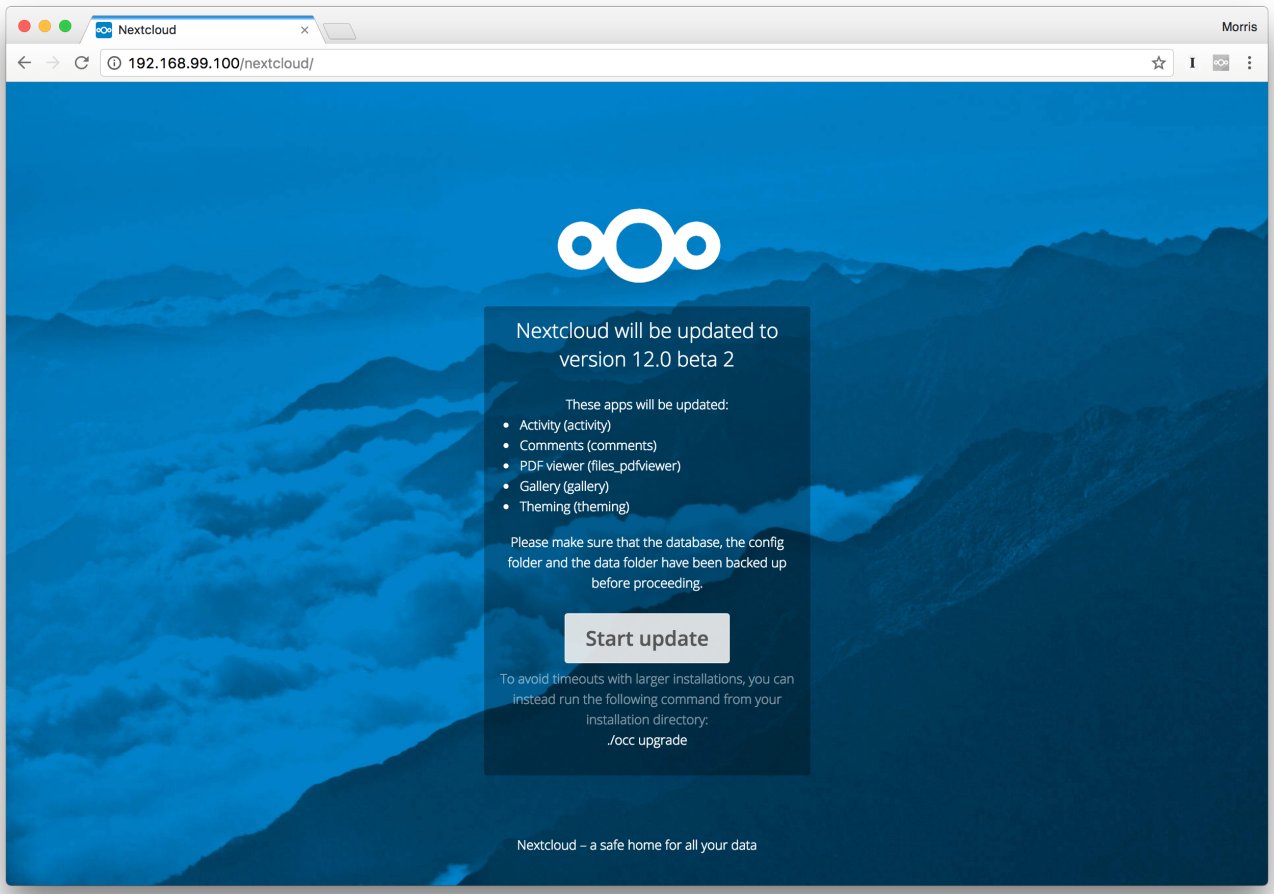


7. Done. You now can continue either to the web based upgrade page or run `occ upgrade`. The two examples “Web based upgrade” and “Command line based upgrade” shows how the screens then look like.

Web based upgrade

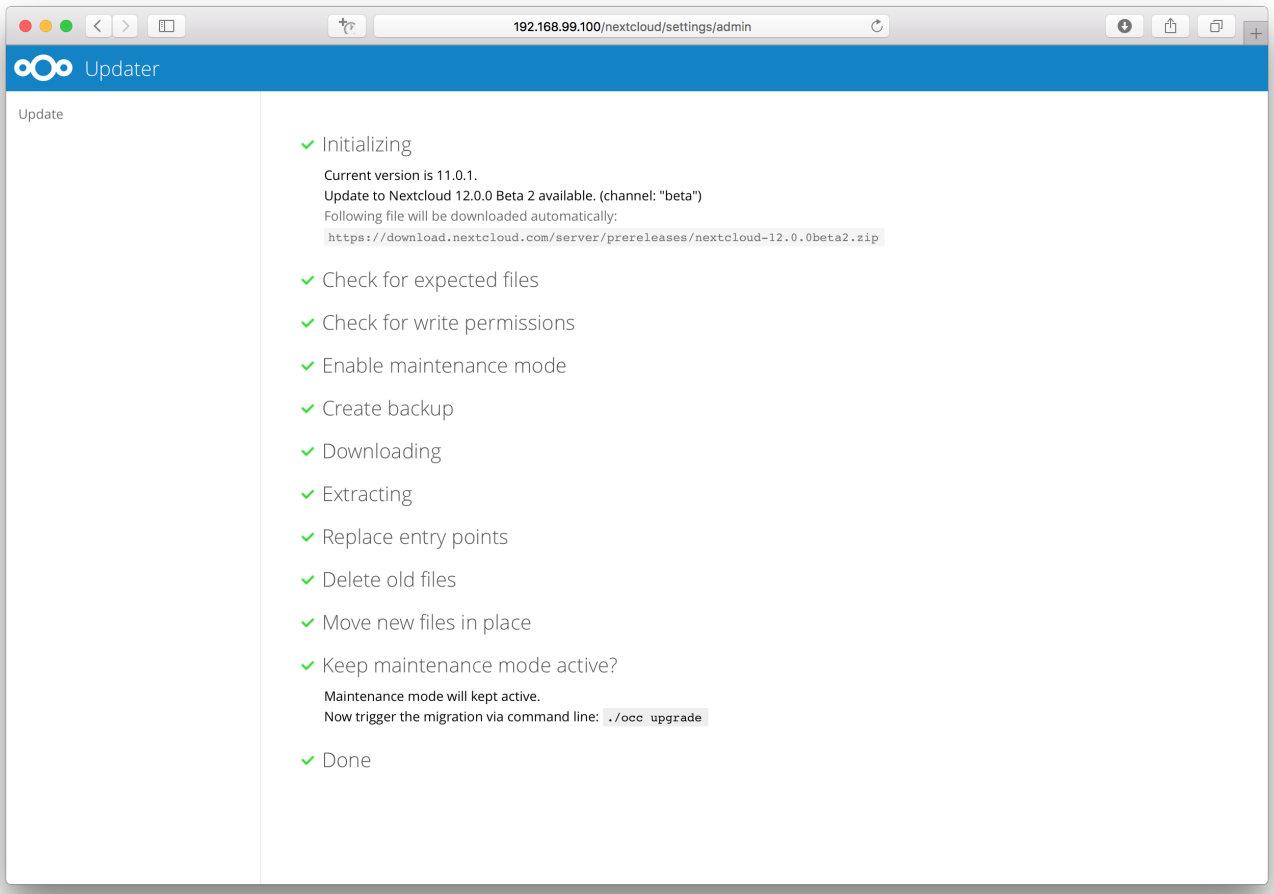
This is how the web based update would continue:


192.168.99.100/nextcloud/settings/admin
Updater
Update
Initializing
Current version is 11.0.1.
Update to Nextcloud 12.0.0 Beta 2 available. (channel: "beta")
Following file will be downloaded automatically:
https://download.nextcloud.com/server/prereleases/nextcloud-12.0.0beta2.zip
Check for expected files
Check for write permissions
Enable maintenance mode
Create backup
Downloading
Extracting
Replace entry points
Delete old files
Move new files in place
Keep maintenance mode active?
Maintenance mode will get disabled.
Done
Go back to your Nextcloud instance to finish the update



Command line based upgrade

This is how the command line based update would continue:




```
$ sudo -u www-data php ./occ upgrade
Nextcloud or one of the apps require upgrade - only a limited number of commands are availab
You may use your browser or the occ upgrade command to do the upgrade
Set log level to debug
Updating database schema
Updated database
Updating <files_pdfviewer> ...
Updated <files_pdfviewer> to 1.1.1
Updating <gallery> ...
Updated <gallery> to 17.0.0
Updating <activity> ...
Updated <activity> to 2.5.2
Updating <comments> ...
Updated <comments> to 1.2.0
Updating <theming> ...
Updated <theming> to 1.3.0
Starting code integrity check...
Finished code integrity check
Update successful
Maintenance mode is kept active
Reset log level
```

Using the command line based updater

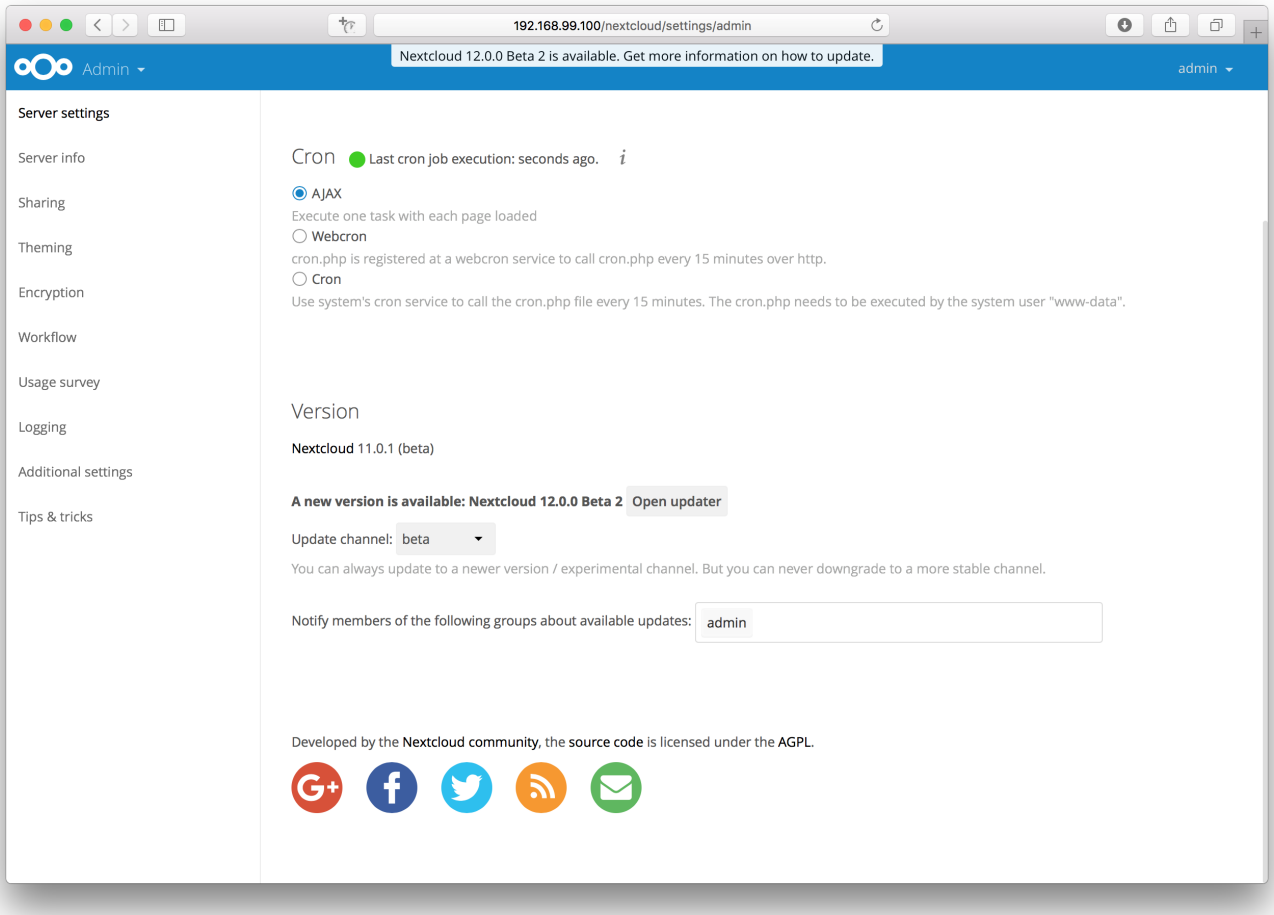
The command line based updater works in the exact same way the web based updater works. The steps and checks are the very same.

Warning

APCu is disabled by default on CLI which could cause issues with nextcloud's command line based updater. Please make sure you set the `apc.enable_cli` to `1` on your `php.ini` config file or append `--define apc.enable_cli=1` to the command line based updater call (like `sudo -u www-data php --define apc.enable_cli=1 /var/www/nextcloud/updater/updater.phar`).

The steps are basically the same as for the web based updater:

1. You should see a notification at the top of any Nextcloud page when there is a new update available. Go to the admin settings page and scroll to the section "Version". This section has a button to open the updater. This section as well as the update notification is only available if the update notication app is enabled in the apps management.



2. Instead of clicking that button you can now invoke the command line based updater by going into the *updater/* directory in the Nextcloud directory and executing the *updater.phar* as the web server user. (i.e. sudo -u www-data php /var/www/nextcloud/updater/updater.phar)

```
Nextcloud Updater - version: 1.0.3

Current version is 11.0.1.

Update to Nextcloud 12.0.0 Beta 2 available. (channel: "beta")
Following file will be downloaded automatically: https://download.nextcloud.com/server/prereleases/nextcloud-12.0.0beta2.zip

Steps that will be executed:
[ ] Check for expected files
[ ] Check for write permissions
[ ] Enable maintenance mode
[ ] Create backup
[ ] Downloading
[ ] Extracting
[ ] Replace entry points
[ ] Delete old files
[ ] Move new files in place
[ ] Done

Start update? [y/N]
```

3. Verify the information that is shown and enter “Y” to start the update

```
Nextcloud Updater - version: 1.0.3

Current version is 11.0.1.

Update to Nextcloud 12.0.0 Beta 2 available. (channel: "beta")
Following file will be downloaded automatically: https://download.nextcloud.com/server/prereleases/nextcloud-12.0.0beta2.zip

Steps that will be executed:
[ ] Check for expected files
[ ] Check for write permissions
[ ] Enable maintenance mode
[ ] Create backup
[ ] Downloading
[ ] Extracting
[ ] Replace entry points
[ ] Delete old files
[ ] Move new files in place
[ ] Done

Start update? [y/N] y

Info: Pressing Ctrl-C will finish the currently running step and then stops the updater.

[✔] Check for expected files
[ ] Check for write permissions ...
```

```
Nextcloud Updater - version: 1.0.3

Current version is 11.0.1.

Update to Nextcloud 12.0.0 Beta 2 available. (channel: "beta")
Following file will be downloaded automatically: https://download.nextcloud.com/server/prereleases/nextcloud-12.0.0beta2.zip

Steps that will be executed:
[ ] Check for expected files
[ ] Check for write permissions
[ ] Enable maintenance mode
[ ] Create backup
[ ] Downloading
[ ] Extracting
[ ] Replace entry points
[ ] Delete old files
[ ] Move new files in place
[ ] Done

Start update? [y/N] y

Info: Pressing Ctrl-C will finish the currently running step and then stops the updater.

[✘] Check for expected files failed
The following extra files have been found:
    unexpectedFile.php

Update failed. To resume or retry just execute the updater again.
```

4. In case an error happens or the check failed the updater stops processing and gives feedback. You can now try to solve the problem and re-run the updater command. This will continue the update and re-run the failed step. It will not re-run the previous succeeded steps

```
Nextcloud Updater - version: 1.0.3

Current version is 11.0.1.

Update to Nextcloud 12.0.0 Beta 2 available. (channel: "beta")
Following file will be downloaded automatically: https://download.nextcloud.com/server/prereleases/nextcloud-12.0.0beta2.zip

Steps that will be executed:
[✔] Check for expected files
[ ] Check for write permissions
[ ] Enable maintenance mode
[ ] Create backup
[ ] Downloading
[ ] Extracting
[ ] Replace entry points
[ ] Delete old files
[ ] Move new files in place
[ ] Done

Continue update? [y/N]
```

6. Once all steps are executed the updater will ask you a final question: “Should the “occ upgrade” command be executed?”. This allows you to directly execute the command line based upgrade procedure (`occ upgrade`). If you select “No” then it will finish with *Please now execute “ /occ upgrade” to finish the upgrade.*.

```
Info: Pressing Ctrl-C will finish the currently running step and then stops the updater.

[✔] Check for expected files
[✔] Check for write permissions
[✔] Enable maintenance mode
[✔] Create backup
[✔] Downloading
[✔] Extracting
[✔] Replace entry points
[✔] Delete old files
[✔] Move new files in place
[✔] Done

Update of code successful.

Should the "occ upgrade" command be executed? [Y/n]
```

7. Once the `occ upgrade` is done you get asked if the maintenance mode should be kept active.

```
Update of code successful.

Should the "occ upgrade" command be executed? [Y/n] y
Nextcloud or one of the apps require upgrade - only a limited number of commands are available
You may use your browser or the occ upgrade command to do the upgrade
Set log level to debug
Updating database schema
Updated database
Updating <federatedfilesharing> ...
Updated <federatedfilesharing> to 1.2.0
Updating <files_pdfviewer> ...
Updated <files_pdfviewer> to 1.1.1
Updated <systemtags> to 1.2.0
Updating <theming> ...
Updated <theming> to 1.3.0
Starting code integrity check...
Finished code integrity check
Update successful
Maintenance mode is kept active
Reset log level

Keep maintenance mode active? [y/N]
```

Batch mode for command line based updater

It is possible to run the command line based updater in a non-interactive mode. The updater then doesn't ask any interactive questions. It is assumed that if an update is available it should be installed and the `occ upgrade` command is executed as well. After finishing the maintenance mode will be turned off except an error occured during the `occ upgrade` or the replacement of the code.

To execute this, run the command with the `--no-interaction` option. (i.e. `sudo -u www-data php /var/www/nextcloud/updater/updater.phar --no-interaction`)

```
Nextcloud Updater - version: 1.0.3

Current version is 11.0.3.

Update to Nextcloud 12.0.0 Beta 2 available. (channel: "beta")
Following file will be downloaded automatically: https://download.nextcloud.com/server/prereleases/nextcloud-12.0.0beta2.zip

Updater run in non-interactive mode.

Start update

Info: Pressing Ctrl-C will finish the currently running step and then stops the updater.

[✔] Check for expected files
[✔] Check for write permissions
[✔] Enable maintenance mode
[✔] Create backup
[✔] Downloading
[✔] Extracting
[✔] Replace entry points
[✔] Delete old files
[✔] Move new files in place
[✔] Done

Update of code successful.
Updater run in non-interactive mode - will start "occ upgrade" now.

Nextcloud or one of the apps require upgrade - only a limited number of commands are available
You may use your browser or the occ upgrade command to do the upgrade
Set log level to debug
Updating database schema
Updated database
Updating <federatedfilesharing> ...
Updated <federatedfilesharing> to 1.2.0
Updating <files_pdfviewer> ...
Updated <files_pdfviewer> to 1.1.1
Updating <theming> ...
Updated <theming> to 1.3.0
Starting code integrity check...
Finished code integrity check
Update successful
Maintenance mode is kept active
Reset log level

Updater run in non-interactive mode - will disable maintenance mode now.

Maintenance mode is disabled
```

Upgrade manually

Always start by making a fresh backup and disabling all 3rd party apps.

1. Back up your existing Nextcloud Server database, data directory, and `config.php` file. (See Backup, for restore information see Restoring backup)
2. Download and unpack the latest Nextcloud Server release (Archive file) from nextcloud.com/install/ into an empty directory outside of your current installation.

> **Note**
>
> To unpack your new tarball, run: unzip nextcloud-[version].zip or tar -xjf nextcloud-[version].tar.bz2

3. Stop your Web server.
4. In case you are running a cron-job for nextcloud's house-keeping disable it by commenting the entry in the crontab file

   crontab -u www-data -e

   (Put a # at the beginning of the corresponding line.)
5. Rename your current Nextcloud directory, for example `nextcloud-old`.
6. Unpacking the new archive creates a new `nextcloud` directory populated with your new server files. Move this directory and its contents to the original location of your old server. For example `/var/www/`, so that once again you have `/var/www/nextcloud`.
7. Copy the `config/config.php` file from your old Nextcloud directory to your new Nextcloud directory.
8. If you keep your `data/` directory in your `nextcloud/` directory, copy it from your old version of Nextcloud to your new `nextcloud/`. If you keep it outside of `nextcloud/` then you don't have to do anything with it, because its location is configured in your original `config.php`, and none of the upgrade steps touch it.
9. If you are using 3rd party application, it may not always be available in your upgraded/new Nextcloud instance. To check this, compare a list of the apps in the new `nextcloud/apps/` folder to a list of the of the apps in your backed-up/old `nextcloud/apps/` folder. If you find 3rd party apps in the old folder that needs to be in the new/upgraded instance, simply copy them over and ensure the permissions are set up as shown below.
10. If you are using 3rd party theme make sure to copy it from your `themes/` directory to your new one. It is possible you will have to make some modifications to it after the upgrade.
11. Adjust file ownership and permissions:

    ```
    chown -R www-data:www-data nextcloud
    find nextcloud/ -type d -exec chmod 750 {} \;
    find nextcloud/ -type f -exec chmod 640 {} \;
    ```
12. Restart your Web server.
13. Now launch the upgrade from the command line using `occ`, like this example on Ubuntu Linux:

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

    (!) this MUST be executed from within your nextcloud installation directory
14. The upgrade operation takes a few minutes to a few hours, depending on the size of your installation. When it is finished you will see a success message, or an error message that will tell where it went wrong.
15. Reenable the nextcloud cron-job. (See step 4 above.)

    crontab -u www-data -e

    (Delete the # at the beginning of the corresponding line in the crontab file.)

Login and take a look at the bottom of your Admin page to verify the version number. Check your other settings to make sure they're correct. Go to the Apps page and review the core apps to make sure the right ones are enabled. Re-enable your third-party apps.

Previous Nextcloud releases

You'll find previous Nextcloud releases in the Nextcloud Server Changelog.

Troubleshooting

Occasionally, *files do not show up after a upgrade*. A rescan of the files can help:

```
sudo -u www-data php console.php files:scan --all
```

See the nextcloud.com support page for further resources.

Sometimes, Nextcloud can get *stuck in a upgrade* if the web based upgrade process is used. This is usually due to the process taking too long and encountering a PHP time-out. Stop the upgrade process this way:

```
sudo -u www-data php occ maintenance:mode --off
```

Then start the manual process:

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

If this does not work properly, try the repair function:

```
sudo -u www-data php occ maintenance:repair
```

Upgrade via packages

Upgrade quickstart

One effective, if unofficial method for keeping Nextcloud current on Linux servers is by configuring your system to use Nextcloud via a self contained “Snap” package, A technology allowing users to always have the latest version of an “app”.

That version from Canonical is quite restrictive. It is not aimed at developers or advanced users who would want to tune their configuration by installing extra features. It is aimed at end-users who want a no-brainer solution. Install it, use it. No need to worry about updating Nextcloud any more.

It will work for as long as Canonical pushes releases, just like with any other Linux package maintained independently of Nextcloud.

Installation

Ubuntu $ sudo snap install nextcloud

All other distros Go to https://docs.snapcraft.io/installing-snapd/6735 Type the command to install snapd Install Nextcloud $ sudo snap install nextcloud

1st login

After a successful install, assuming you and the device on which it was installed are on the same network, you should be able to reach the Nextcloud installation by visiting .local in your browser. If your hostname is localhost or localhost.localdomain, like on an Ubuntu Base device (IoT), nextcloud.local will be used instead.

You will be asked to create a password for “admin” and your favourite cloud will be ready

Note

Do not use on IoT devices yet. You probably don’t need these instructions anyway if you’re using Snappy Base 16.04 as it’s currently unreleased.

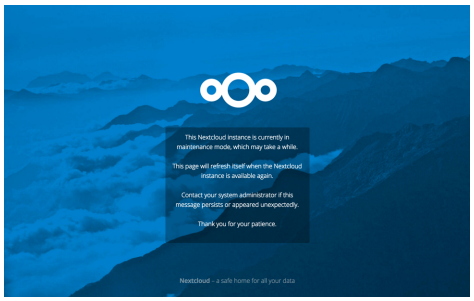
- Make a fresh backup.
- Upgrade your Nextcloud snap: sudo snap refresh nextcloud
- Run occ upgrade.
- Take your Nextcloud server out of maintenance mode.
- Re-enable third-party apps.

Upgrade tips

Upgrading Nextcloud from a Snap is just like upgrading any snap package. For example:

```
sudo snap refresh nextcloud
```

Your Snap package manager only upgrades the current Nextcloud Snap. Then your Nextcloud server is immediately put into maintenance mode. You may not see this until you refresh your Nextcloud page.



Then use `occ` to complete the upgrade. You must run `occ` as your HTTP user. This example is for Debian/Ubuntu:

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

This example is for CentOS/RHEL/Fedora:

```
sudo -u apache php occ upgrade
```

Upgrading across skipped releases

It is best to update your Nextcloud installation with every new point release, and to never skip any major releases. While this requirement is being worked on, for the moment If you have skipped any major releases you can bring your Nextcloud current with these steps:

If you are using a Snap package: sudo snap refresh nextcloud

If you did **not** install via a Snap package:

1. Upgrade your current version to the latest point release
2. Upgrade your current version to the next major release
3. Run upgrade routine
4. Repeat from step 2 until you reach the last available major release

You'll find previous Nextcloud releases in the Nextcloud Server Changelog.

If upgrading via your Snap package manager fails, then you must perform a Upgrade manually.

Migrating to a different server

If the need arises Nextcloud can be migrated to a different server. A typical use case would be a hardware change or a migration from the virtual Appliance to a physical server. All migrations have to be performed with Nextcloud offline and no accesses being made. Online migration is supported by Nextcloud only when implementing industry standard clustering and HA solutions before Nextcloud is installed for the first time.

To start let us be specific about the use case. A configured Nextcloud instance runs reliably on one machine. For some reason (e.g. more powerful machine is available but a move to a clustered environment not yet needed) the instance needs to be moved to a new machine. Depending on the size of the Nextcloud instance the migration might take several hours. As a prerequisite it is assumed that the end users reach the Nextcloud instance via a virtual hostname (a `CNAME` record in DNS) which can be pointed at the new location. It is also assumed that the authentication method (e.g. LDAP) remains the same after the migration.

Warning

At NO TIME any changes to the **ORIGINAL** system are required **EXCEPT** putting Nextcloud into maintenance mode.

This ensures, should anything unforseen happen you can go back to your existing installation and provide your users with a running Nextcloud while debugging the problem.

1. Set up the new machine with the desired OS, install and configure the Web server as well as PHP for Nextcloud (e.g. permissions or file upload size limits) and make sure the PHP version matches Nextcloud supported configuration and all relevant PHP extensions are installed. Also set up the database and make sure it is a Nextcloud supported configuration. If your original machine was installed recently just copying that base configuration is a safe best practice.
2. On the original machine then stop Nextcloud. First activate the maintenance mode. After waiting for 6-7 minutes for all sync clients to register the server as in maintenance mode stop the application and/or Web server that serves Nextcloud.
3. Create a dump from the database and copy it to the new machine. There import it in the database (See Backup and Restoring backup).
4. Copy all files from your Nextcloud instance, the Nextcloud program files, the data files, the log files and the configuration files, to the new machine (See Backup and Restoring backup). The data files should keep their original timestamp (can be done by using `rsync` with `-t` option) otherwise the clients will re-download all the files after the migration. Depending on the original installation method and the OS the files are located in different locations. On the new system make sure to pick the appropriate locations. If you change any paths, make sure to adapt the paths in the Nextcloud config.php file. Note: This step might take several hours, depending on your installation.
5. Check the config.php file of the **ORIGINAL** system to see if it has the `data-fingerprint` set to a non-empty value. If this is the case, make

   sure to also run the `maintenance:data-fingerprint` command on the **NEW** system, similarly to how it is required when performing a backup restoration (See Restoring backup for details).
6. While still having Nextcloud in maintenance mode (confirm!) and **BEFORE** changing the `CNAME` record in the DNS start up the database, Web server / application server on the new machine and point your web browser to the migrated Nextcloud instance. Confirm that you see the maintenance mode notice, that a logfile entry is written by both the Web server and Nextcloud and that no error messages occur. Then take Nextcloud out of maintenance mode and repeat. Log in as admin and confirm normal function of Nextcloud.
7. Change the `CNAME` entry in the DNS to point your users to the new location.

Migrating from ownCloud

Note

Especially when migrating from ownCloud to Nextcloud you should create a backup of the config, database and the data directory, in case something goes wrong.

Currently migrating from ownCloud is like performing a manual update. So it is quite easy, to migrate from one ownCloud version to at least one Nextcloud version. However this does only work with versions that are close enough database and code-wise. See the table below for a version map, where migrating is easily possible:

ownCloud	Nextcloud
10.0.5 or later	20.0.x (but at least 20.0.5)
10.0.1 - 10.0.5	12.0.x (but at least 12.0.1)
10.0.0	12.0.0
9.1.x	10.0.x
9.0.x	10.0.x
9.0.x	9.0.x

Note

While we understand, that you want to migrate as soon as possible, we also don't want to put your data at risk. Since we never know what ownCloud changes in a future release, we only allow migrations from versions that were released before our last maintenance release, so we can at least perform some basic migration tests, before you migrate your production instance.

1. First download the correct version of Nextcloud from our older releases page,
2. Make sure to have do a backup before migrating.
3. Follow the upgrade instructions described in the Upgrade manually manual.
4. When migrating to Nextcloud 20.0 or later, you will also need to run the following commands after `occ upgrade`:
   - `occ db:convert-filecache-bigint`
   - `occ db:add-missing-columns`
   - `occ db:add-missing-indices`
   - `occ db:add-missing-primary-keys`
5. If system cron was used, please verify if crontab entry was using the command `occ system:cron`. If yes, please adjust it to use the `php` command instead according to the background jobs configuration documentation
6. Use the Nextcloud built-in updater to update your instance to the newest version.
7. Make sure to also verify the "Security & setup warnings" in the "Overview" section on the settings page.

Issues and troubleshooting

General troubleshooting

If you have trouble installing, configuring or maintaining Nextcloud, please refer to our community support channels:

- **The Nextcloud Forums**

  The Nextcloud forums have a FAQ page where each topic corresponds to typical mistakes or frequently occurring issues

Please understand that all these channels essentially consist of users like you helping each other out. Consider helping others out where you can, to contribute back for the help you get. This is the only way to keep a community like Nextcloud healthy and sustainable!

If you are using Nextcloud in a business or otherwise large scale deployment, note that Nextcloud GmbH offers commercial support options.

Bugs

If you think you have found a bug in Nextcloud, please:

- Search for a solution (see the options above)
- Double-check your configuration

If you can't find a solution, please use our bugtracker. You can generate a configuration report with the occ config command, with passwords automatically obscured.

General troubleshooting

Check the Nextcloud System requirements, especially supported browser versions.

When you see warnings about `code integrity`, refer to Code signing.

Disable 3rdparty / non-shipped apps

It might be possible that 3rd party / non-shipped apps are causing various different issues. Always disable 3rd party apps before upgrades, and for troubleshooting. Please refer to the Apps commands on how to disable an app from command line.

Nextcloud logfiles

In a standard Nextcloud installation the log level is set to `Normal`. To find any issues you need to raise the log level to `All` in your `config.php` file, or to **Everything** on your Nextcloud Admin page. Please see Logging for more information on these log levels.

Some logging - for example JavaScript console logging - needs debugging enabled. Edit config/config.php and change `'debug' => false,` to `'debug' => true,` Be sure to change it back when you are finished.

For JavaScript issues you will also need to view the javascript console. All major browsers have developer tools for viewing the console, and you usually access them by pressing F12.

Note

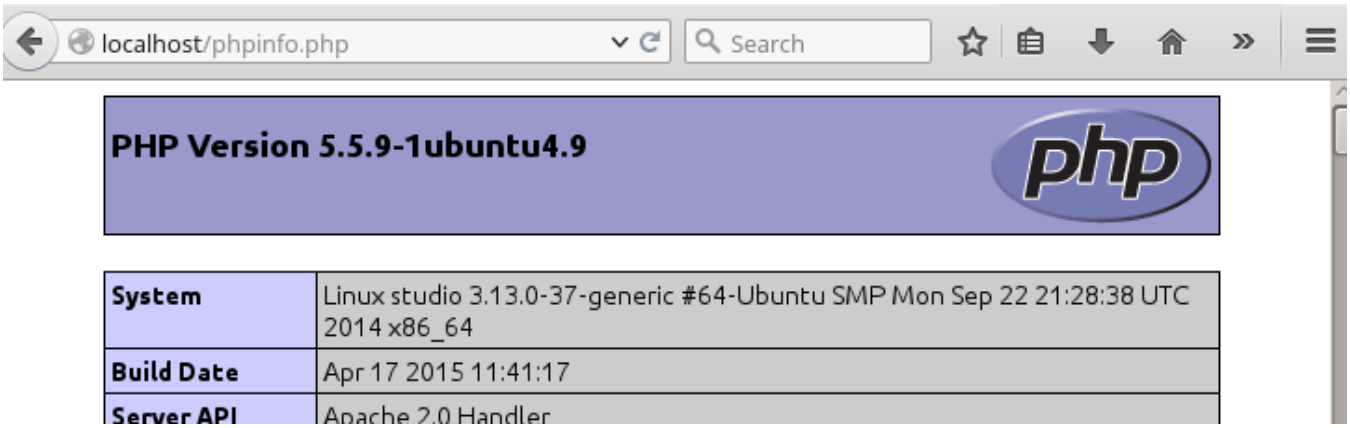
The logfile of Nextcloud is located in the data directory `nextcloud/data/nextcloud.log`.

PHP version and information

You will need to know your PHP version and configurations. To do this, create a plain-text file named **phpinfo.php** and place it in your Web root, for example `/var/www/html/phpinfo.php`. (Your Web root may be in a different location; your Linux distribution documentation will tell you where.) This file contains just this line:

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

Open this file in a Web browser by pointing your browser to `localhost/phpinfo.php`:



Your PHP version is at the top, and the rest of the page contains abundant system information such as active modules, active `.ini` files, and much more. When you are finished reviewing your information you must delete `phpinfo.php`, or move it outside of your Web directory, because it is a security risk to expose such sensitive data.

Debugging sync issues

Warning

The data directory on the server is exclusive to Nextcloud and must not be modified manually.

Disregarding this can lead to unwanted behaviors like:

- Problems with sync clients

- Undetected changes due to caching in the database

If you need to directly upload files from the same server please use a WebDAV command line client like `cadaver` to upload files to the WebDAV interface at:

`https://example.com/nextcloud/remote.php/dav`

Common problems / error messages

Some common problems / error messages found in your logfiles as described above:

- `SQLSTATE[HY000] [1040] Too many connections` -> You need to increase the connection limit of your database, please refer to the manual of your database for more information.
- `SQLSTATE[HY000]: General error: 5 database is locked` -> You're using `SQLite` which can't handle a lot of parallel requests. Please consider converting to another database like described in Converting database type.
- `SQLSTATE[HY000]: General error: 2006 MySQL server has gone away` -> Please refer to Troubleshooting for more information.
- `SQLSTATE[HY000] [2002] No such file or directory` -> There is a problem accessing your SQLite database file in your data directory (`data/nextcloud.db`). Please check the permissions of this folder/file or if it exists at all. If you're using MySQL please start your database.
- `Connection closed / Operation cancelled` -> This could be caused by wrong `KeepAlive` settings within your Apache config. Make sure that `KeepAlive` is set to `On` and also try to raise the limits of `KeepAliveTimeout` and `MaxKeepAliveRequests`.
- `No basic authentication headers were found` -> This error is shown in your `data/nextcloud.log` file. Some Apache modules like `mod_fastcgi`, `mod_fcgid` or `mod_proxy_fcgi` are not passing the needed authentication headers to PHP and so the login to Nextcloud via WebDAV, CalDAV and CardDAV clients is failing.

Troubleshooting Web server and PHP problems

Logfiles

When having issues the first step is to check the logfiles provided by PHP, the Web server and Nextcloud itself.

> **Note**
>
> In the following the paths to the logfiles of a default Debian installation running Apache2 with mod_php is assumed. On other Web servers, Linux distros or operating systems they can differ.

- The logfile of Apache2 is located in `/var/log/apache2/error.log`.
- The logfile of PHP can be configured in your `/etc/php/7.4/apache2/php.ini`. You need to set the directive `log_errors` to `On` and choose the path to store the logfile in the `error_log` directive. After those changes you need to restart your Web server.
- The logfile of Nextcloud is located in the data directory `/var/www/nextcloud/data/nextcloud.log`.

Web server and PHP modules

> **Note**
>
> Lighttpd is not supported with Nextcloud, and some Nextcloud features may not work at all on Lighttpd.

There are some Web server or PHP modules which are known to cause various problems like broken up-/downloads. The following shows a draft overview of these modules:

1. Apache
   - mod_pagespeed
   - mod_evasive
   - mod_security
   - mod_reqtimeout
   - mod_deflate
   - libapache2-mod-php*filter (use libapache2-mod-php7.4 instead)
   - mod_spdy together with libapache2-mod-php5 / mod_php (use fcgi or php-fpm instead)
   - mod_dav
   - mod_xsendfile / X-Sendfile (causing broken downloads if not configured correctly)
2. NginX
   - ngx_pagespeed
   - HttpDavModule
   - X-Sendfile (causing broken downloads if not configured correctly)
3. PHP
   - eAccelerator

Troubleshooting WebDAV

Nextcloud uses SabreDAV, and the SabreDAV documentation is comprehensive and helpful.

See:

- SabreDAV FAQ
- Web servers (Lists lighttpd as not recommended)
- Working with large files (Shows a PHP bug in older SabreDAV versions and information for mod_security problems)
- 0 byte files (Reasons for empty files on the server)
- Clients (A comprehensive list of WebDAV clients, and possible problems with each one)
- Finder, OS X's built-in WebDAV client (Describes problems with Finder on various Web servers)

There is also a well maintained FAQ thread available at the ownCloud Forums which contains various additional information about WebDAV problems.

Service discovery

Some clients - especially on iOS/macOS - have problems finding the proper sync URL, even when explicitly configured to use it.

If you want to use CalDAV or CardDAV clients or other clients that require service discovery together with Nextcloud it is important to have a correct working setup of the following URLs:

```
https://example.com/.well-known/carddav
https://example.com/.well-known/caldav
```

Those need to be redirecting your clients to the correct endpoints. If Nextcloud is running at the document root of your Web server the correct URL is `https://example.com/remote.php/dav` for CardDAV and CalDAV and if running in a subfolder like `nextcloud`, then `https://example.com/nextcloud/remote.php/dav`.

For the first case the .htaccess file shipped with Nextcloud should do this work for you when you're running Apache. You need to make sure that your Web server is using this file. Additionally, you need the mod_rewrite Apache module installed to process these redirects. When running Nginx please refer to NGINX configuration.

If your Nextcloud instance is installed in a subfolder called `nextcloud` and you're running Apache create or edit the .htaccess file within the document root of your Web server and add the following lines:

```
<IfModule mod_rewrite.c>
  RewriteEngine on
  RewriteRule ^\.well-known/carddav /nextcloud/remote.php/dav [R=301,L]
  RewriteRule ^\.well-known/caldav /nextcloud/remote.php/dav [R=301,L]
  RewriteRule ^\.well-known/webfinger /nextcloud/index.php/.well-known/webfinger [R=301,L]
  RewriteRule ^\.well-known/nodeinfo /nextcloud/index.php/.well-known/nodeinfo [R=301,L]
</IfModule>
```

Make sure to change /nextcloud to the actual subfolder your Nextcloud instance is running in.

If you are running NGINX, make sure `location = /.well-known/carddav {` and `location = /.well-known/caldav {` are properly configured as described in NGINX configuration, adapt to use a subfolder if necessary.

Now change the URL in the client settings to just use:

`https://example.com`

instead of e.g.

`https://example.com/nextcloud/remote.php/dav/principals/username`.

There are also several techniques to remedy this, which are described extensively at the Sabre DAV website.

Troubleshooting sharing

Users' Federated Cloud IDs not updated after a domain name change

1. run Database query

```
DELETE FROM oc_cards_properties WHERE name = 'CLOUD' AND addressbookid = (select id f
rom oc_addressbooks where principaluri = 'principals/system/system' AND uri = 'system
');
```

2. run occ commands

```
occ dav:sync-system-addressbook
occ federation:sync-addressbooks
```

Troubleshooting contacts & calendar

Unable to update contacts or events

If you get an error like:

`PATCH https://example.com/remote.php/dav HTTP/1.0 501 Not Implemented`

it is likely caused by one of the following reasons:

Using Pound reverse-proxy/load balancer

As of writing this Pound doesn't support the HTTP/1.1 verb. Pound is easily patched to support HTTP/1.1.

Misconfigured Web server

Your Web server is misconfigured and blocks the needed DAV methods. Please refer to Troubleshooting WebDAV above for troubleshooting steps.

Troubleshooting data-directory

If you have a fresh install, consider reinstalling with your preferred directory location.

Unofficially moving the data directory can be done as follows:

1. Make sure no cron jobs are running
2. Stop apache

3. Move /data to the new location
4. Change the config.php entry
5. Edit the database: In oc_storages change the path on the local::/old-data-dir/ entry
6. Ensure permissions are still correct
7. Restart apache

For a safe moving of data directory, supported by Nextcloud, recommended actions are:

1. Make sure no cron jobs are running
2. Stop apache
3. Move /data to the new location
4. Create a symlink from the original location to the new location
5. Ensure permissions are still correct
6. Restart apache

Troubleshooting encryption

Problems when downloading or decrypting files

In some rare cases it can happen that encrypted files cannot be downloaded and return a “500 Internal Server Error”. If the Nextcloud log contains an error about “Bad Signature”, then the following command can be used to repair affected files:

```
occ encryption:fix-encrypted-version userId --path=/path/to/broken/file.txt
```

Replace “userId” and the path accordingly. The command will do a test decryption for all files and automatically repair the ones with a signature error.

Other issues

Some services like *Cloudflare* can cause issues by minimizing JavaScript and loading it only when needed. When having issues like a not working login button or creating new users make sure to disable such services first.

Patching Nextcloud

Applying a patch

Patching server

1. Navigate into your Nextcloud server’s root directory (contains the `status.php` file)
2. Now apply the patch with the following command:

```
patch -p 1 < /path/to/the/file.patch
```

> **Note**
>
> There can be errors about not found files, especially when you take a patch from GitHub there might be development or test files included in the patch. when the files are in build/ or a tests/ subdirectory it is mostly being

Patching apps

1. Navigate to the root of this app (mostly `apps/[APPID]/`), if you can not find the app there use the `sudo -u www-data php occ app:getpath APPID` command to find the path.
2. Now apply the patch with the same command as in Patching server

Reverting a patch

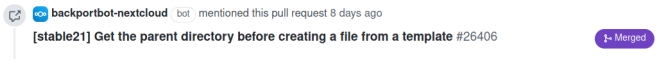
1. Navigate to the directory where you applied the patch.
2. Now revert the patch with the `-R` option:

```
patch -R -p 1 < /path/to/the/file.patch
```

Getting a patch from a GitHub pull request

If you found a related pull request on GitHub that solves your issue, or you want to help developers and verify a fix works, you can get a patch for the pull request.

1. Using https://github.com/nextcloud/server/pull/26396 as an example.
2. Append `.patch` to the URL: https://github.com/nextcloud/server/pull/26396.patch
3. Download the patch to your server and follow the Applying a patch steps.
4. In case you are on an older version, you might first need to go the the correct version of the patch.



5. You can find it by looking for a link by the `backportbot-nextcloud` or a developer will leave a manual comment about the backport to an older Nextcloud version. For the example above you the pull request for Nextcloud 21 is at https://github.com/nextcloud/server/pull/26406 and the patch at https://github.com/nextcloud/server/pull/26406.patch

Code signing

Nextcloud supports code signing for the core releases, and for Nextcloud applications. Code signing gives our users an additional layer of security by ensuring that nobody other than authorized persons can push updates.

It also ensures that all upgrades have been executed properly, so that no files are left behind, and all old files are properly replaced. In the past, invalid updates were a significant source of errors when updating Nextcloud.

FAQ

Why did Nextcloud add code signing?

By supporting Code Signing we add another layer of security by ensuring that nobody other than authorized persons can push updates for applications, and ensuring proper upgrades.

Do we lock down Nextcloud?

The Nextcloud project is open source and always will be. We do not want to make it more difficult for our users to run Nextcloud. Any code signing errors on upgrades will not prevent Nextcloud from running, but will display a warning on the Admin page. For applications that are not tagged “Official” the code signing process is optional.

Not open source anymore?

The Nextcloud project is open source and always will be. The code signing process is optional, though highly recommended. The code check for the core parts of Nextcloud is enabled when the Nextcloud release version branch has been set to stable.

For custom distributions of Nextcloud it is recommended to change the release version branch in version.php to something else than “stable”.

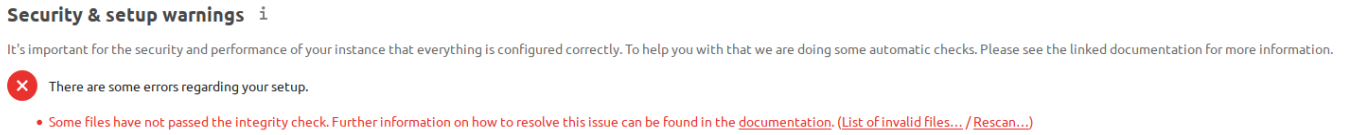
Is code signing mandatory for apps?

Code signing is required for all applications on apps.nextcloud.com.

Fixing invalid code integrity messages

A code integrity error message (“Some files have not passed the integrity check…”) appears on your Nextcloud admin page under “Overview”, which provides the following options:

1. Link to this documentation entry.
2. Show a list of invalid files.
3. Trigger a rescan.



To debug issues caused by the code integrity check click on “List of invalid files…”, and you will be shown a text document listing the different issues. The content of the file will look similar to the following example:

```
Technical information
=====================
The following list covers which files have failed the integrity check. Please read
the previous linked documentation to learn more about the errors and how to fix
them.

Results
=======
- core
      - INVALID_HASH
              - /index.php
              - /version.php
      - EXTRA_FILE
              - /test.php
- calendar
      - EXCEPTION
              - OC\IntegrityCheck\Exceptions\InvalidSignatureException
              - Signature data not found.

Raw output
==========
Array
(
    [core] => Array
        (
            [INVALID_HASH] => Array
                (
                    [/index.php] => Array
                        (
                            [expected] =>
                            f1c5e2630d784bc9cb02d5a28f55d6f24d06dae2a0fee685f3
                            c2521b050955d9d452769f61454c9ddfa9c308146ade10546c
                            fa829794448eaffbc9a04a29d216
                            [current] =>
                            ce08bf30bcbb879a18b49239a9bec6b8702f52452f88a9d321
                            42cad8d2494d5735e6bfa0d8642b2762c62ca5be49f9bf4ec2
                            31d4a230559d4f3e2c471d3ea094
                        )

                    [/version.php] => Array
```

```
                        (
                            [expected] =>
                            c5a03bacae8dedf8b239997901ba1fffd2fe51271d13a00cc4
                            b34b09cca5176397a89fc27381cbb1f72855fa18b69b6f87d7
                            d5685c3b45aee373b09be54742ea
                            [current] =>
                            88a3a92c11db91dec1ac3be0e1c87f862c95ba6ffaaaa3f2c3
                            b8f682187c66f07af3a3b557a868342ef4a271218fe1c1e300
                            c478e6c156c5955ed53c40d06585
                        )

                )

            [EXTRA_FILE] => Array
                (
                    [/test.php] => Array
                        (
                            [expected] =>
                            [current] =>
                            09563164f9904a837f9ca0b5f626db56c838e5098e0ccc1d8b
                            935f68fa03a25c5ec6f6b2d9e44a868e8b85764dafd1605522
                            b4af8db0ae269d73432e9a01e63a
                        )

                )

        )

    [calendar] => Array
        (
            [EXCEPTION] => Array
                (
                    [class] => OC\IntegrityCheck\Exceptions\InvalidSignature
                    Exception
                    [message] => Signature data not found.
                )

        )

)
```

In above error output it can be seen that:

1. In the Nextcloud core (that is, the Nextcloud server itself) the files “index.php” and “version.php” do have the wrong version.
2. In the Nextcloud core the unrequired extra file “/test.php” has been found.
3. It was not possible to verify the signature of the calendar application.

The solution is to upload the correct “index.php” and “version.php” files, and delete the “test.php” file. For the calendar exception contact the developer of the application. For other means on how to receive support please take a look at https://nextcloud.com/support/. After fixing these problems verify by clicking “Rescan…”.

Note

When using a FTP client to upload those files make sure it is using the `Binary` transfer mode instead of the `ASCII` transfer mode.

Rescans

Rescans are triggered at installation, and by updates. You may run scans manually with the `occ` command. The first command scans the Nextcloud server files, and the second command scans the named app. There is not yet a command to manually scan all apps:

```
occ integrity:check-core
occ integrity:check-app $appid
```

See Using the occ command to learn more about using `occ`.

Errors

> **Warning**
>
> Please don't modify the mentioned `signature.json` itself.

The following errors can be encountered when trying to verify a code signature.

- `INVALID_HASH`
  - The file has a different hash than specified within `signature.json`. This usually happens when the file has been modified after writing the signature data.
- `MISSING_FILE`
  - The file cannot be found but has been specified within `signature.json`. Either a required file has been left out, or `signature.json` needs to be edited.
- `EXTRA_FILE`
  - The file does not exist in `signature.json`. This usually happens when a file has been removed and `signature.json` has not been updated. It also happens if you have placed additional files in your Nextcloud installation folder.
- `EXCEPTION`
  - Another exception has prevented the code verification. There are currently these following exceptions:
    - `Signature data not found.`
      - The app has mandatory code signing enforced but no `signature.json` file has been found in its `appinfo` folder.
    - `Certificate is not valid.`
      - The certificate has not been issued by the official Nextcloud Code Signing Root Authority.
    - `Certificate is not valid for required scope. (Requested: %s, current: %s)`
      - The certificate is not valid for the defined application. Certificates are only valid for the defined app identifier and cannot be used for others.
    - `Signature could not get verified.`
      - There was a problem with verifying the signature of `signature.json`.

GDPR-compliance

Cookies

Nextcloud only stores cookies needed for Nextcloud to work properly. All cookies comes from your Nextcloud server directly, no 3rd-party cookies will be sent to your system. Regarding GDPR, only data which contain personal data are relevant.

Cookies stored by Nextcloud

Cookie	Data Stored	Lifetime
Session cookie	• session ID • secret token (used to decrypt the session on the server)	24 minutes
Same-site cookies	no user-related data are stored, all same-site cookies are the same for all users on all Nextcloud instances	forever
Remember-me cookie	• user id • original session id • remember token	15 days (can be configured)

The same-site cookies are used to determine how a request reaches the Nextcloud server. We use them to prevent CSRF attacks. No identifable information is stored in those. The rest of the cookies are strictly used to identify the user to the system.