# Nextcloud Server Administration Manual

***Release 11 alpha***

**The Nextcloud developers**

July 19, 2017

CONTENTS

CHAPTER

ONE

# 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, Mac OS X and Linux, and mobile clients for the Android and Apple iOS operating systems.

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

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.

See `../release_notes` for more information on the different Nextcloud editions.

## Videos and Blogs

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

Visit Nextcloud Planet for news and developer blogs.

## 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/ownCloud Desktop Client
- Nextcloud Android App
- Nextcloud iOS App

CHAPTER

TWO

# INSTALLATION

## System Requirements

### Memory

Memory requirements for running an Nextcloud server are greatly variable, depending on the numbers of users and files, and volume of server activity. Nextcloud needs a minimum of 128MB RAM, and we recommend a minimum of 512MB.

### Recommended Setup for Running Nextcloud

For best performance, stability, support, and full functionality we recommend:

- Red Hat Enterprise Linux 7 / Ubuntu 16.04 LTS
- MySQL/MariaDB
- PHP 7.0 +
- Apache 2.4 with mod_php

### Supported Platforms

- Server: Linux (Debian 7, SUSE Linux Enterprise Server 11 SP3 & 12, Red Hat Enterprise Linux/CentOS 6.5 and 7 (7 is 64-bit only), Ubuntu 14.04 LTS, 16.04 LTS)
- Web server: Apache 2 (mod_php, php-fpm) or Nginx (php-fpm)
- Databases: MySQL/MariaDB 5.5+; PostgreSQL; Oracle 11g (currently only possible if you *contact us <https://nextcloud.com/enterprise>* as part of a subscription)
- PHP 5.6 + required
- Hypervisors: Hyper-V, VMware ESX, Xen, KVM
- Desktop: Windows XP SP3 (EoL Q2 2015), Windows 7+, Mac OS X 10.7+ (64-bit only), Linux (CentOS 6.5, 7 (7 is 64-bit only), Ubuntu 12.04 LTS, 14.04 LTS, 14.10, Fedora 20, 21, openSUSE 12.3, 13, Debian 7 & 8).
- Mobile apps: iOS 7+, Android 4+
- Web browser: IE11+, Microsoft Edge, Firefox 14+, Chrome 18+, Safari 7+

See Manual Installation on Linux for minimum software versions for installing Nextcloud.

### Database Requirements for MySQL / MariaDB

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

- Disabled or BINLOG_FORMAT = MIXED configured Binary Logging (See: *MySQL / MariaDB with Binary Logging Enabled*)
- InnoDB storage engine (MyISAM is not supported)
- "READ COMMITED" transaction isolation level (See: *Database "READ COMMITED" transaction isolation level*)

#### Emoji (UTF8 4-byte) support with MySQL / MariaDB

If you want to use UTF8 4-byte characters such as Emojis on your server, the database needs to be created with character set `utf8mb4` and collate `utf8mb4_general_ci`, e.g.:

```
CREATE DATABASE nextcloud CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;
```

Additionally the following InnoDB settings need to be set:

```
[mysqld]
innodb_large_prefix=on
innodb_file_format=barracuda
innodb_file_per_table=true
```

See Enabling MySQL 4-byte support for more information.

## Deployment Recommendations

What is the best way to install and maintain Nextcloud? The answer to that is *"it depends"* because every Nextcloud customer has their own particular needs and IT infrastructure. Nextcloud and the LAMP stack are highly-configurable, so we will present three typical scenarios and make best-practice recommendations for both software and hardware.

### General Recommendations

**Note:** Whatever the size of your organization, always keep one thing in mind: the amount of data stored in Nextcloud will only grow. Plan ahead.

Consider setting up a scale-out deployment, or using Federated Cloud Sharing to keep individual Nextcloud instances to a manageable size.

- Operating system: Linux (Ubuntu 16.04 or Red Hat Enterprise Linux 7 is recommended).
- Web server: Apache 2.4.
- Database: MySQL/MariaDB.
- PHP 5.6+. PHP 5.6 is the minimum supported version. We recommend to deploy on PHP 7 if possible. This version is known to offer significan performance advantages. `mod_php` is the recommended Apache module due to vendor support and ease of configuration. `php-fpm` with Apache Event MPM (or nginx) is an alternative with potentially better scalability in high load and limited RAM environments. For the best results we recommend working with the Nextcloud GmbH enterprise support team for large deployments.

## Small Workgroups or Departments

- **Number of users** Up to 150 users.
- **Storage size** 100 GB to 10TB.
- **High availability level** Zero-downtime backups via Btrfs snapshots, component failure leads to interruption of service. Alternate backup scheme on other filesystems: nightly backups with service interruption.

### Recommended System Requirements

One machine running the application server, Web server, database server and local storage.

Authentication via an existing LDAP or Active Directory server.


- **Components** One server with at least 2 CPU cores, 16GB RAM, local storage as needed.
- **Operating system** Enterprise grade Linux distribution with full support from OS vendor. Red Hat Enterprise Linux or Ubuntu 16.04 are recommended.
- **SSL Configuration** The SSL termination is done in Apache. A standard SSL certificate is needed, installed according to the Apache documentation.
- **Load Balancer** None.
- **Database** MySQL, MariaDB or PostgreSQL. We currently recommend MySQL / MariaDB, as our customers have had good experiences when moving to a Galera cluster to scale the DB.
- **Backup** Install Nextcloud, Nextcloud data directory and database on Btrfs filesystem. Make regular snapshots at desired intervals for zero downtime backups. Mount DB partitions with the "nodatacow" option to prevent fragmentation.

  Alternatively, make nightly backups with service interruption:

  – Shut down Apache.

  – Create database dump.

  – Push data directory to backup.

  – Push database dump to backup.

  – Start Apache.

  Then optionally rsync to a backup storage or tape backup. (See the Maintenance section of the Administration manual for tips on backups and restores.)
- **Authentication** User authentication via one or several LDAP or Active Directory servers. (See User Authentication with LDAP for information on configuring Nextcloud to use LDAP and AD.)
- **Session Management** Local session management on the application server. PHP sessions are stored in a tmpfs mounted at the operating system-specific session storage location. You can find out where that is by running `grep -R 'session.save_path' /etc/php5` and then add it to the `/etc/fstab` file, for example: `echo "tmpfs /var/lib/php5/pool-www tmpfs defaults,noatime,mode=1777 0 0" >> /etc/fstab`.

- **Memory Caching** A memcache speeds up server performance, and Nextcloud supports four memcaches; refer to Configuring Memory Caching for information on selecting and configuring a memcache.
- **Storage** Local storage.

## Mid-sized Enterprises

- **Number of users** 150 to 1,000 users.
- **Storage size** Up to 200TB.
- **High availability level** Every component is fully redundant and can fail without service interruption. Backups without service interruption

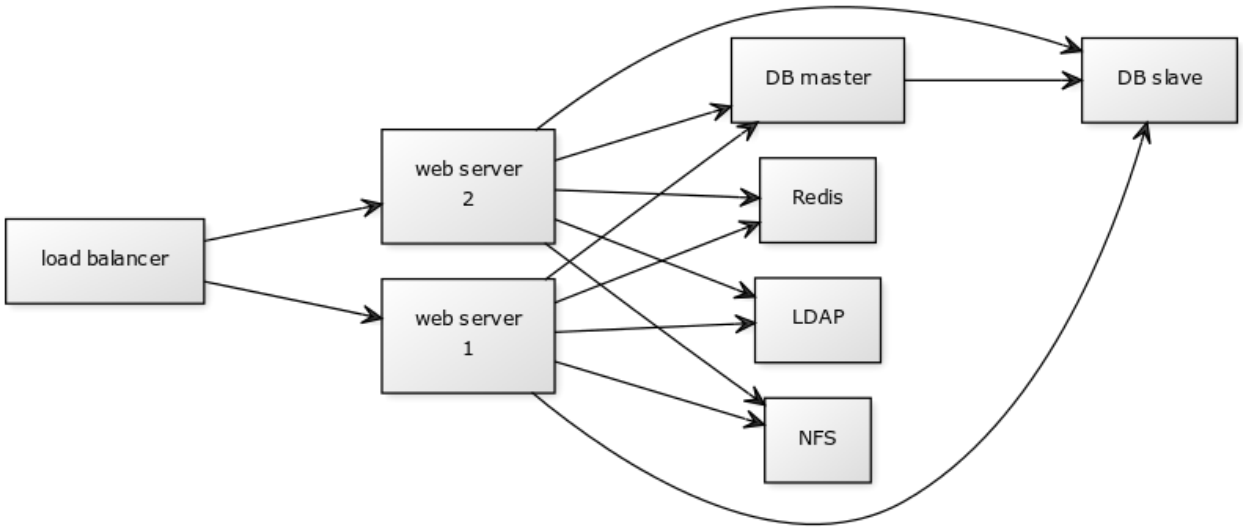
### Recommended System Requirements

2 to 4 application servers.

A cluster of two database servers.

Storage on an NFS server.

Authentication via an existing LDAP or Active Directory server.




- **Components**
  - 2 to 4 application servers with 4 sockets and 32GB RAM.
  - 2 DB servers with 4 sockets and 64GB RAM.
  - 1 HAproxy load balancer with 2 sockets and 16GB RAM.
  - NFS storage server as needed.
- **Operating system** Enterprise grade Linux distribution with full support from OS vendor. Red Hat Enterprise Linux or Ubuntu 16.04 are recommended.
- **SSL Configuration** The SSL termination is done in the HAProxy load balancer. A standard SSL certificate is needed, installed according to the HAProxy documentation.

- **Load Balancer** HAProxy running on a dedicated server in front of the application servers. Sticky session needs to be used because of local session management on the application servers.
- **Database** MySQL/MariaDB Galera cluster with master-slave replication. The slave is only used as failover in case the master is down. This could be extended with a load balancer infront to distribute writes to the master and reads to the slave as well. (see "Database load balancer" below)
- **Backup** Minimum daily backup without downtime. All MySQL/MariaDB statements should be replicated to a backup MySQL/MariaDB slave instance.
  - Create a snapshot on the NFS storage server.
  - At the same time stop the MySQL replication.
  - Create a MySQL dump of the backup slave.
  - Push the NFS snapshot to the backup.
  - Push the MySQL dump to the backup.
  - Delete the NFS snapshot.
  - Restart MySQL replication.
- **Authentication** User authentication via one or several LDAP or Active Directory servers. (See User Authentication with LDAP for information on configuring Nextcloud to use LDAP and AD.)
- **LDAP** Read-only slaves should be deployed on every application server for optimal scalability
- **Session Management** Session management on the application server. PHP sessions are stored in a tmpfs mounted at the operating system-specific session storage location. You can find out where that is by running `grep -R 'session.save_path' /etc/php5` and then add it to the `/etc/fstab` file, for example: `echo "tmpfs /var/lib/php5/pool-www tmpfs defaults,noatime,mode=1777 0 0" >> /etc/fstab`.
- **Memory Caching** A memcache speeds up server performance, and Nextcloud supports four memcaches; refer to Configuring Memory Caching for information on selecting and configuring a memcache.
- **Storage** Use an off-the-shelf NFS solution, such as IBM Elastic Storage or RedHat Ceph.

## Large Enterprises and Service Providers

- **Number of users** 5,000 to >100,000 users.
- **Storage size** Up to 1 petabyte.
- **High availabily level** Every component is fully redundant and can fail without service interruption. Backups without service interruption

### Recommended System Requirements

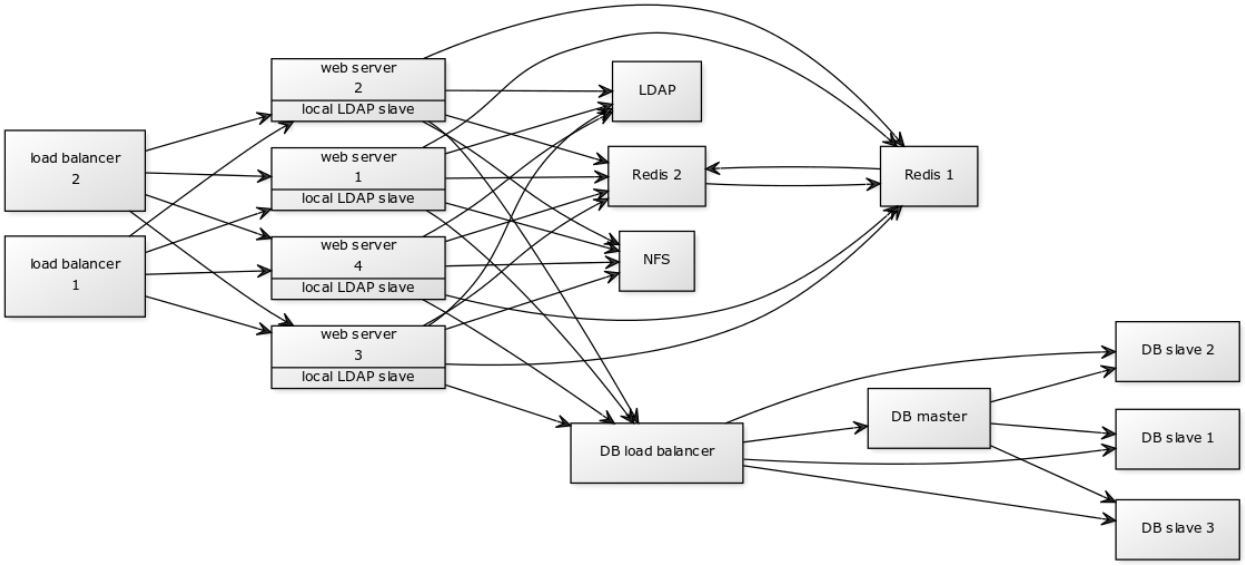
4 to 20 application/Web servers.

A cluster of two or more database servers which are behind a load balancer to send all writes to the master and reads to the slaves. (see "Database load balancer" below)

Storage is an NFS server, or an object store that is S3 compatible.

Cloud federation for a distributed setup over several data centers.

Authentication via an existing LDAP or Active Directory server, or SAML.




- **Components**
  - 4 to 20 application servers with 4 sockets and 64GB RAM.
  - 4 DB servers with 4 sockets and 128GB RAM plus a DB load balancer (see "Database load balancer" below)
  - 2 load balancer - either HAProxy with keepalived (heartbeat) and a shared virutal IP address as a software solution or a hardware load balancer. For the HAProxy we recommend at least 2 sockets and 16GB RAM each.
  - NFS storage server as needed.
- **Operating system** Enterprise grade Linux distribution with full support from OS vendor. Red Hat Enterprise Linux or Ubuntu 16.04 are recommended.
- **SSL Configuration** The SSL termination is done in the load balancer. A standard SSL certificate is needed, installed according to the load balancer documentation.
- **Load Balancer** A redundant load-balancer with heartbeat, for example HAProxy. This runs two load balancers in front of the application servers.
- **Database** MySQL/MariaDB Galera Cluster with master - slave replication (master & 3 slaves). The load balancer infront distributes writes to the master and reads to the slaves. (see "Database load balancer" below)
- **Backup** Minimum daily backup without downtime. All MySQL/MariaDB statements should be replicated to a backup MySQL/MariaDB slave instance.
  - Create a snapshot on the NFS storage server.
  - At the same time stop the MySQL replication.
  - Create a MySQL dump of the backup slave.
  - Push the NFS snapshot to the backup.
  - Push the MySQL dump to the backup.
  - Delete the NFS snapshot.
  - Restart MySQL replication.

- **Authentication** User authentication via one or several LDAP or Active Directory servers, or SAML/Shibboleth. (See User Authentication with LDAP.)
- **LDAP** Read-only slaves should be deployed on every application server for optimal scalability.
- **Session Management** Redis should be used for the session management storage.
- **Caching** Redis for distributed in-memory caching (see Configuring Memory Caching).
- **Storage** An off-the-shelf NFS solution should be used. Examples are IBM Elastic Storage or RedHAT Ceph. Optionally, an S3 compatible object store can also be used.

## Hardware Considerations

- Solid-state drives (SSDs) for I/O.
- Separate hard disks for storage and database, SSDs for databases.
- Multiple network interfaces to distribute server synchronisation and backend traffic across multiple subnets.

### Single Machine / Scale-Up Deployment

The single-machine deployment is widely used in the community.

Pros:

- Easy setup: no session storage daemon, use tmpfs and memory caching to enhance performance, local storage.
- No network latency to consider.
- To scale buy a bigger CPU, more memory, larger hard drive, or additional hard drives.

Cons:

- Fewer high availability options.
- The amount of data in Nextcloud tends to continually grow. Eventually a single machine will not scale; I/O performance decreases and becomes a bottleneck with multiple up- and downloads, even with solid-state drives.

### Scale-Out Deployment

Provider setup:

- DNS round robin to HAProxy servers (2-n, SSL offloading, cache static resources)
- Least load to Apache servers (2-n)
- Memcached/Redis for shared session storage (2-n)
- Database cluster with single Master, multiple slaves and proxy to split requests accordingly (2-n) - MaxScale is a possible proxy solutions to load balance the writes to the master and reads to the slaves (see “Database load balancer” below)
- GPFS or Ceph via phprados (2-n, 3 to be safe, Ceph 10+ nodes to see speed benefits under load)

Pros:

- Components can be scaled as needed.
- High availability.
- Test migrations easier.

Cons:

- More complicated to setup.
- Network becomes the bottleneck (10GB Ethernet recommended).
- Currently DB filecache table will grow rapidly, making migrations painful in case the table is altered.

#### What About Nginx / PHP-FPM?

Could be used instead of HAproxy as the load balancer. But on uploads stores the whole file on disk before handing it over to PHP-FPM.

#### A Single Master DB is Single Point of Failure, Does Not Scale

When master fails another slave can become master.

A multi-master setup with Galera cluster is not supported, because we require `READ-COMMITTED` as transaction isolation level. Galera doesn't support this with a master-master replication which will lead to deadlocks during uploads of multiple files into one directory for example.

#### Database load balancer

When Galera cluster is used as DB cluster solution, we recommend to use MaxScale as load balancer infront of the cluster to distribute writes to the master node and reads to the slaves.

### Software Considerations

#### Operating System

We are dependent on distributions that offer an easy way to install the various components in up-to-date versions. We recommend Red Hat Enterprise Linux 7 or Ubuntu 16.04 - for both commercial support can be purchased. Debian and Ubuntu are free of cost, and include newer software packages. CentOS is the community-supported free-of-cost Red Hat Enterprise Linux clone.

#### Web server

Taking Apache and Nginx as the contenders, Apache with mod_php is currently the best option, as Nginx does not support all features necessary for enterprise deployments. Mod_php is recommended instead of PHP_FPM, because in scale-out deployments separate PHP pools are simply not necessary.

#### Relational Database

More often than not the customer already has an opinion on what database to use. In general, the recommendation is to use what their database administrator is most familiar with. Taking into account what we are seeing at customer deployments, we recommend MySQL/MariaDB in a master - slave deployment with a MySQL proxy in front of them to send updates to master, and selects to the slave(s). (see "Database load balancer" above)

The second best option is PostgreSQL (alter table does not lock table, which makes migration less painful) although we have yet to find a customer who uses a master-slave setup.

What about the other DBMS?

- Sqlite is adequate for simple testing, and for low-load single-user deployments. It is not adequate for production systems.
- Microsoft SQL Server is not a supported option.
- For Oracle DB support please contact the Nextcloud team to get more information on this.

### File Storage

While many customers are starting with NFS, sooner or later that requires scale-out storage. Currently the options are GPFS or GlusterFS, or an object store protocol like S3 or Swift. S3 also allows access to Ceph Storage.

### Session Storage

- Redis: provides persistence, nice graphical inspection tools available, supports Nextcloud high-level file locking.
- If Shibboleth is a requirement you must use Memcached, and it can also be used to scale-out shibd session storage (see Memcache StorageService).

### References

Database High Availability

Performance enhancements for Apache and PHP

How to Set Up a Redis Server as a Session Handler for PHP on Ubuntu 14.04

## Manual Installation on Linux

If there are no packages for your Linux distribution, or 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 14.04 LTS Server with Apache and MariaDB, using the Nextcloud .tar archive.

**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.

## Prerequisites

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 (>= 5.6, 7.0 or 7.1)
- PHP module ctype
- PHP module dom
- PHP module GD
- PHP module iconv
- PHP module JSON
- PHP module libxml (Linux package libxml2 must be >=2.7.0)
- PHP module mb multibyte
- PHP module posix
- PHP module SimpleXML
- PHP module XMLWriter
- PHP module zip
- PHP module zlib

Database connectors (pick the one for your database:)

- PHP module sqlite (>= 3, usually not recommended for performance reasons)
- PHP module pdo_mysql (MySQL/MariaDB)
- PHP module pgsql (requires PostgreSQL >= 9.0)

*Recommended* packages:

- PHP module curl (highly recommended, some functionality, e.g. HTTP user authentication, depends on this)
- PHP module fileinfo (highly recommended, enhances file analysis performance)
- PHP module bz2 (recommended, required for extraction of apps)
- PHP module intl (increases language translation performance and fixes sorting of non-ASCII characters)
- PHP module mcrypt (increases file encryption performance)
- PHP module openssl (required for accessing HTTPS resources)

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)

Recommended for specific apps (*optional*):

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

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 Configuring 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`)

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.)

## Example Installation on Ubuntu 16.04 LTS Server

On a machine running a pristine Ubuntu 16.04 LTS server, install the required and recommended modules for a typical Nextcloud installation, using Apache and MariaDB, by issuing the following commands in a terminal:

```
apt-get install apache2 mariadb-server libapache2-mod-php7.0
apt-get install php7.0-gd php7.0-json php7.0-mysql php7.0-curl php7.0-mbstring
apt-get install php7.0-intl php7.0-mcrypt php-imagick php7.0-xml php7.0-zip
```

- This installs the packages for the Nextcloud core system. `libapache2-mod-php7.0` provides the following PHP extensions: `bcmath bz2 calendar Core ctype date dba dom ereg exif fileinfo filter ftp gettext hash iconv libxml mhash openssl pcre Phar posix Reflection session shmop SimpleXML soap sockets SPL standard sysvmsg sysvsem sysvshm tokenizer wddx xmlreader xmlwriter zlib`. If you are planning on running additional apps, keep in mind that they might require additional packages. See *Prerequisites* for details.
- At the installation of the MySQL/MariaDB server, you will be prompted to create a root password. Be sure to remember your password as you will need it during Nextcloud database setup.

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 -xjf 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.

## BINLOG_FORMAT = STATEMENT

If your Nextcloud installation fails and you see this in your Nextcloud log:

```
An unhandled exception has been thrown: exception 'PDOException' with message
'SQLSTATE[HY000]: General error: 1665 Cannot execute statement: impossible to
write to binary log since BINLOG_FORMAT = STATEMENT and at least one table
uses a storage engine limited to row-based logging. InnoDB is limited to
row-logging when transaction isolation level is READ COMMITTED or READ
UNCOMMITTED.'
```

See *MySQL / MariaDB with Binary Logging Enabled*.

## Apache Web Server Configuration

On Debian, Ubuntu, and their derivatives, Apache installs with a useful configuration so all you have to do is create a `/etc/apache2/sites-available/nextcloud.conf` file with these lines in it, replacing the **Directory** and other filepaths with your own filepaths:

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

<Directory /var/www/nextcloud/>
  Options +FollowSymlinks
  AllowOverride All

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

```
 SetEnv HOME /var/www/nextcloud
 SetEnv HTTP_HOME /var/www/nextcloud

</Directory>
```

Then create a symlink to `/etc/apache2/sites-enabled`:

```
ln -s /etc/apache2/sites-available/nextcloud.conf /etc/apache2/sites-enabled/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 are created automatically when `.htaccess` is writable by the HTTP user, `mod_env` and `mod_rewrite` are installed, and `'overwrite.cli.url'` in your `config.php` is set to any non-null value.

## 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/php5/apache2/php.ini
or
  /etc/php5/fpm/php.ini
or ...
```

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

```
/etc/php5/cli/php.ini
```

## php-fpm Configuration Notes

**Security: Use at least PHP >= 5.6.6**

Due to a bug with security implications in older PHP releases with the handling of XML data you are highly encouraged to run at least PHP 5.6.6 when in a threaded environment.

**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:

| Ubuntu/Mint | CentOS/Red Hat/Fedora |
|---|---|
| `/etc/php5/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.

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.

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 `php5-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 Example Configurations

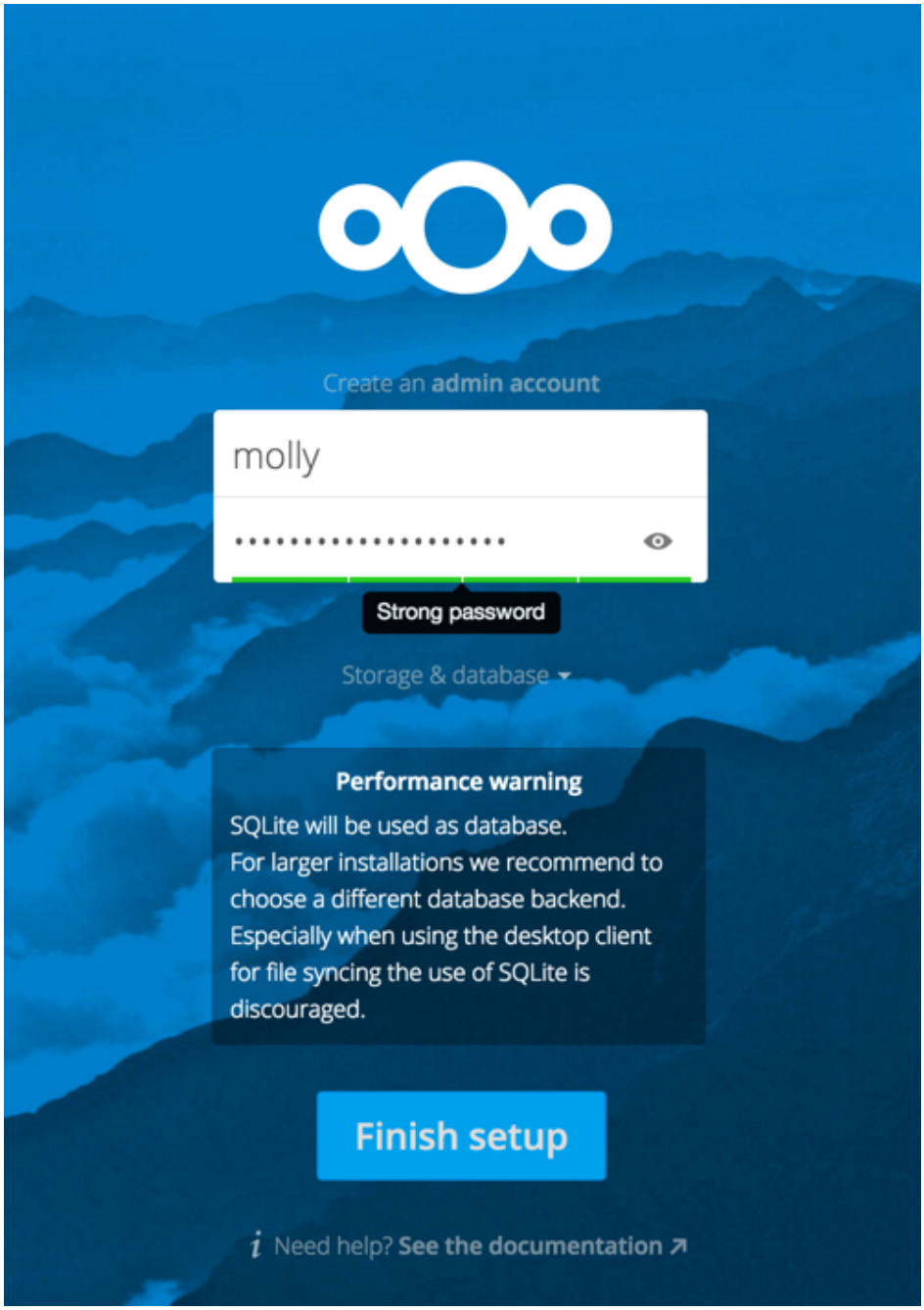
Other HTTP servers (Nextcloud)

# 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**.



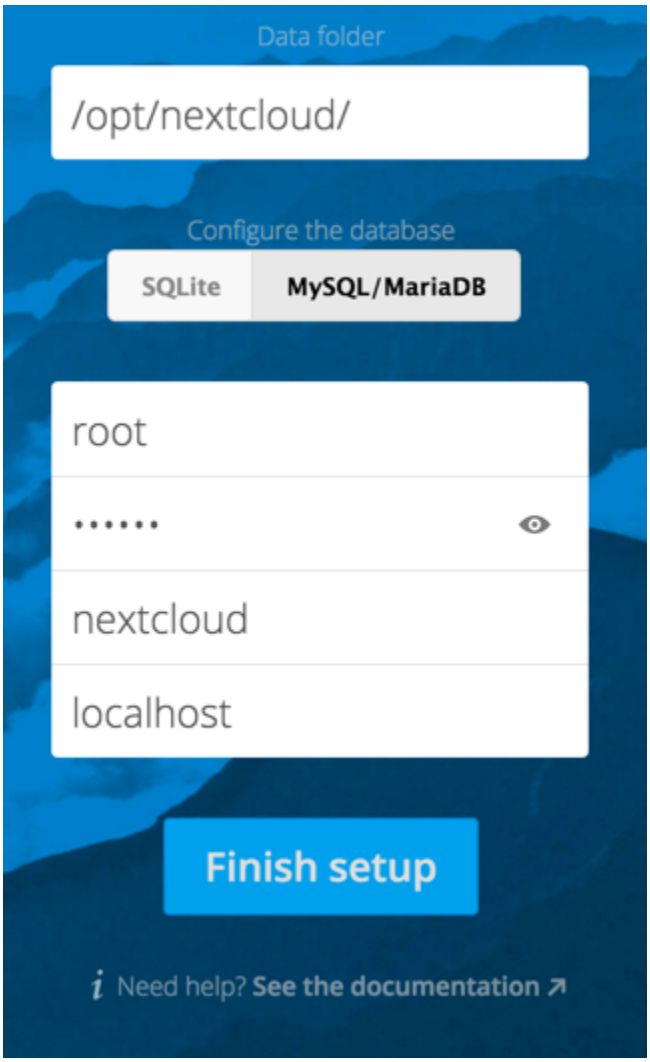
You're finished and can start using your new Nextcloud server.

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 `/var/oc_data`. This directory must already exist, and must be owned by your HTTP user.

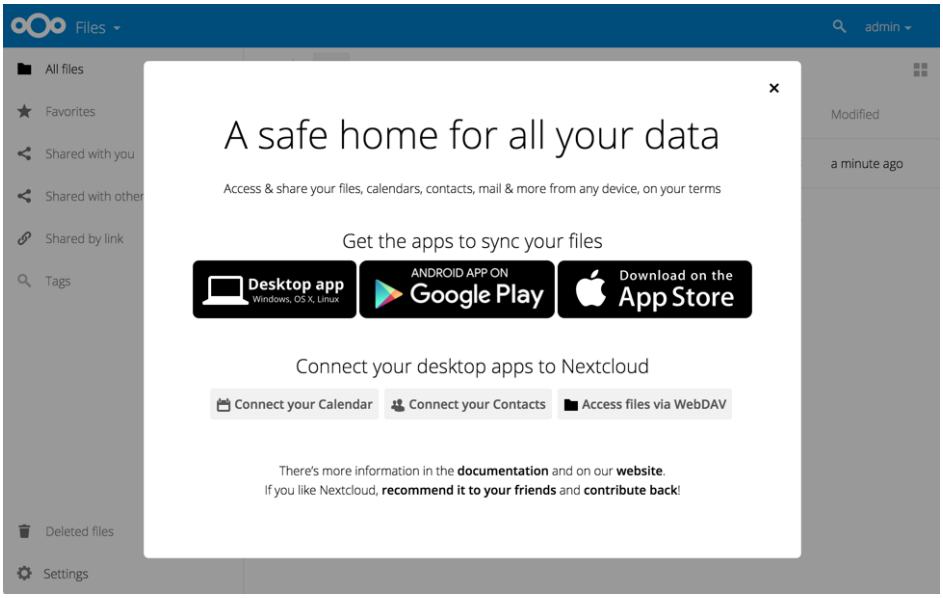
## Database Choice

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 Manual Installation on Linux for a detailed listing of required and optional PHP modules). You will need the root database login, or any administrator login that has permissions to create and modify databases, and then enter any name you want for your Nextcloud database.

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.



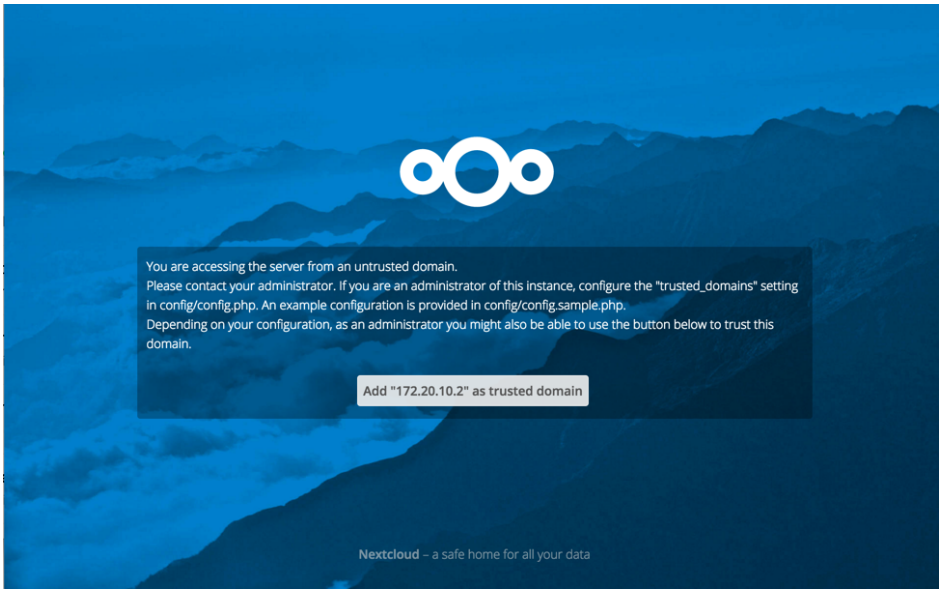

Now we will look at some important post-installation steps.

## 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',
),
```

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 Manual 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/enterprise as pa
```

See *Command Line Installation* for more information.

### BINLOG_FORMAT = STATEMENT

If your Nextcloud installation fails and you see this in your Nextcloud log:

```
An unhandled exception has been thrown: exception 'PDOException' with message
'SQLSTATE[HY000]: General error: 1665 Cannot execute statement: impossible to
write to binary log since BINLOG_FORMAT = STATEMENT and at least one table
uses a storage engine limited to row-based logging. InnoDB is limited to
row-logging when transaction isolation level is READ COMMITTED or READ
UNCOMMITTED.'
```

See *MySQL / MariaDB with Binary Logging Enabled*.

## Installing and Managing Apps

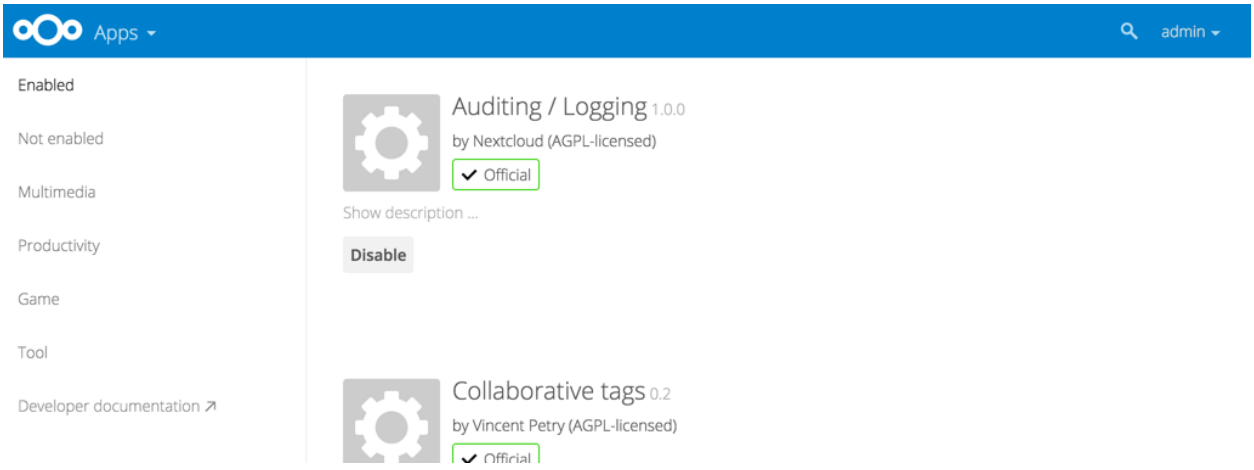
After installing Nextcloud, you may provide added functionality by installing applications.

### Supported Apps

See Supported Apps for a list of supported apps.

### Viewing Enabled Apps

During the Nextcloud installation, some apps are enabled by default. To see which apps are enabled go to your Apps page.




You will see which apps are enabled, not enabled, and recommended. You'll also see additional filters, such as Multimedia, Productivity, and Tool for finding more apps quickly.

## Managing Apps

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 on your Nextcloud Admin page, Personal page, or in `config.php`.

## Adding Third Party Apps

Some apps are developed and supported by Nextcloud directly. These have an **Official** tag. Apps with the **Approved** tag are community-developed and supported; they are maintained by trusted developers, and are under active development. Only **Official** and **Approved** apps are linked on the Apps page by default.

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.

Click the gear icon on the lower left to browse experimental apps in the ownCloud Apps Store. Install experimental apps at your own risk.

Sometimes the installation of a third-party app fails silently, possibly because `'appcodechecker' => true,` is enabled in `config.php`. When `appcodechecker` is enabled it checks if third-party apps are using the private API, rather than the public API. If they are then they will not be installed.

---

**Note:** If you would like to create or add your own Nextcloud app, please refer to the developer manual.

---

## 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.

---

```
<?php

  "apps_paths" => array (
      0 => array (
              "path"     => OC::$SERVERROOT."/apps",
              "url"      => "/apps",
              "writable" => false,
      ),
      1 => array (
              "path"     => OC::$SERVERROOT."/apps2",
              "url"      => "/apps2",
              "writable" => true,
      ),
  ),
```

### Using Your Own Appstore

You can enable the installation of apps from your own apps store. This requires that you can write to at least one of the configured apps directories.

To enable installation from your own apps store:

1. Set the **appstoreenabled** parameter to "true".

   This parameter is used to enable your 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 the Nextcloud apps store. The appstore server must use OCS (Open Collaboration Services).

```
<?php

  "appstoreenabled" => true,
  "appstoreurl" => "https://api.nextcloud.com/v1",
```

## Supported Apps

### AGPL Apps

- Activity
- Admin Audit Log
- AntiVirus
- Calendar
- Collaborative Tags
- Comments
- Contacts
- Encryption
- External Sites
- External Storage
- Federated File Sharing (allows file sharing across Nextcloud instances)
- Federation (allows usernname auto-complete across Nextcloud instances)
- Files (cannot be disabled)
- Files Access Control
- Files Automated Tagging
- Files PDF Viewer
- Files Sharing
- Files Text Editor
- Files Trashbin
- Files Versions

- Files Video Player
- First Run Wizard
- Gallery
- Notifications
- Object Storage (Swift)
- Password Policy
- Provisioning API
- Template Editor (for notification emails)
- Theming
- Update Notifications
- User External
- User LDAP
- User Shibboleth/SAML
- WebDAV Endpoint (handles old and new webdav endpoints)
- Workflow Engine (cannot be disabled)

## Installing PHP 5.4 on RHEL 6 and CentOS 6

Red Hat Enterprise Linux and CentOS 6 still ship with PHP 5.3. Nextcloud requires PHP 5.4 or better. There are several third-party repositories that supply PHP 5.4, but you must use the Software Collections (SCL) repository to be in compliance with your RHEL support contract, and not any other third-party repository.

### RHEL 6

Follow these steps to install PHP 5.4 from SCL. First you must use your Subscription Manager to enable SCL:

```
subscription-manager repos --enable rhel-server-rhscl-6-eus-rpms
```

Then install PHP 5.4 and these modules:

```
yum install php54 php54-php php54-php-gd php54-php-mbstring
```

You must also install the updated database module for your database. This example installs the new PHP 5.4 module for MySQL/MariaDB:

```
yum install php54-php-mysqlnd
```

Disable loading the old PHP 5.3 Apache module:

```
mv /etc/httpd/conf.d/php.conf /etc/httpd/conf.d/php53.off
```

You should now have a /etc/httpd/conf.d/php54-php.conf file, which loads the correct PHP 5.4 module for Apache.

Then restart Apache:

```
service httpd restart
```

Verify with *PHP Version and Information* that your Apache server is using PHP 5.4 and loading the correct modules.

### CentOS 6

First install the SCL repo:

```
yum install centos-release-SCL
```

Then install PHP 5.4 and these modules:

```
yum install php54 php54-php php54-php-gd php54-php-mbstring
```

You must also install the updated database module. This installs the new PHP 5.4 module for MySQL/MariaDB:

```
yum install php54-php-mysqlnd
```

Disable loading the old PHP 5.3 Apache module:

```
mv /etc/httpd/conf.d/php.conf /etc/httpd/conf.d/php53.off
```

You should now have a /etc/httpd/conf.d/php54-php.conf file, which loads the correct PHP 5.4 module for Apache.

Finally, restart Apache:

```
service httpd restart
```

Verify with *PHP Version and Information* that your Apache server is using PHP 5.4 and loading the correct modules.

## Installing PHP 5.5 on RHEL 7 and CentOS 7

PHP 5.4 has been end-of-life since September 2015 and is no longer supported by the PHP team. RHEL 7 still ships with PHP 5.4, and Red Hat supports it. Nextcloud also supports PHP 5.4, so upgrading is not required. However, it is highly recommended to upgrade to PHP 5.5+ for best security and performance.

**Before upgrading, evaluate all of your PHP apps for compatibility with PHP 5.5.**

### RHEL 7 Upgrade to PHP 5.5

To upgrade to PHP 5.5, you must use the Software Collections (SCL) repository to be in compliance with your RHEL support contract, and not any other third-party repository. Follow these steps to install PHP 5.5 from SCL. First you must use your Subscription Manager to enable SCL:

```
subscription-manager repos --enable rhel-server-rhscl-7-eus-rpms
```

Then install PHP 5.5 and these modules:

```
yum install php55 php55-php php55-php-gd php55-php-mbstring
```

You must also install the updated database module for your database. This installs the new PHP 5.5 module for MySQL/MariaDB:

```
yum install php55-php-mysqlnd
```

If you are using the Nextcloud LDAP app, you need this module:

```
yum install php55-php-ldap
```

Disable loading the old PHP Apache modules by changing their names:

```
mv /etc/httpd/conf.d/php.conf /etc/httpd/conf.d/php54.off
mv /etc/httpd/conf.modules.d/10-php.conf /etc/httpd/conf.modules.d/10-php54.off
```

Copy the PHP 5.5 Apache modules into place:

```
cp /opt/rh/httpd24/root/etc/httpd/conf.d/php55-php.conf /etc/httpd/conf.d/
cp /opt/rh/httpd24/root/etc/httpd/conf.modules.d/10-php55-php.conf /etc/httpd/conf.modules.d/
cp /opt/rh/httpd24/root/etc/httpd/modules/libphp55-php5.so /etc/httpd/modules/
```

Then restart Apache:

```
service httpd restart
```

Verify with `phpinfo` that your Apache server is using PHP 5.5 and loading the correct modules; see *PHP Version and Information* to learn how to use phpinfo.

## CentOS 7 Upgrade to PHP 5.5

To upgrade to PHP 5.5, use the Red Hat Software Collections (SCL) repository.

**Before upgrading, evaluate all of your PHP apps for compatibility with PHP 5.5.**

Follow these steps to install PHP 5.5 from SCL. First install the SCL repository:

```
yum install centos-release-scl
```

Then install PHP 5.5 and these modules:

```
yum install php55 php55-php php55-php-gd php55-php-mbstring
```

You must also install the updated database module for your database. This installs the new PHP 5.5 module for MySQL/MariaDB:

```
yum install php55-php-mysqlnd
```

If you are using the Nextcloud LDAP app, you need this module:

```
yum install php55-php-ldap
```

Disable loading the old PHP Apache modules by changing their names:

```
mv /etc/httpd/conf.d/php.conf /etc/httpd/conf.d/php54.off
mv /etc/httpd/conf.modules.d/10-php.conf /etc/httpd/conf.modules.d/10-php54.off
```

Copy the PHP 5.5 Apache modules into place:

```
cp /opt/rh/httpd24/root/etc/httpd/conf.d/php55-php.conf /etc/httpd/conf.d/
cp /opt/rh/httpd24/root/etc/httpd/conf.modules.d/10-php55-php.conf /etc/httpd/conf.modules.d/
cp /opt/rh/httpd24/root/etc/httpd/modules/libphp55-php5.so /etc/httpd/modules/
```

Then restart Apache:

```
service httpd restart
```

Verify with `phpinfo` that your Apache server is using PHP 5.5 and loading the correct modules; see *PHP Version and Information* to learn how to use `phpinfo`.

# 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'

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'

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 Example Configurations

This page covers example Nginx configurations to use with running an Nextcloud server. This page is community-maintained. (Thank you, contributors!)

- You need to insert the following code into **your Nginx configuration file.**
- The configuration assumes that Nextcloud is installed in `/var/www/nextcloud` and that it is accessed via `http(s)://cloud.example.com`.
- 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).
- `add_header` statements are only taken from the current level and are not cascaded from or to a different level. All necessary `add_header` statements must be defined in each level needed. For better readability it is possible to move *common* add header statements into a separate file and include that file wherever necessary. However, each `add_header` statement must be written in a single line to prevent connection problems with sync clients.

### Example Configurations

- Nginx Configuration

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.

- Remove the server block containing the redirect
- Change **listen 443 ssl** to **listen 80;**
- Remove **ssl_certificate** and **ssl_certificate_key**.
- Remove **fastcgi_params HTTPS on;**

#### 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.

### Performance Tuning

- nginx (<1.9.5) <ngx_http_spdy_module
- nginx (+1.9.5) <ngx_http_http2_module

To use http_v2 for nginx you have to check two things:

1.) be aware that this module is not built in by default due to a dependency to the openssl version used on your system. It will be enabled with the `--with-http_v2_module` configuration parameter during compilation. The dependency should be checked automatically. You can check the presence of http_v2 with `nginx -V 2>&1 | grep http_v2 -o`. An example of how to compile nginx can be found in section "Configure nginx with the `nginx-cache-purge` module" below.

2.) When you have used SPDY before, the nginx config has to be changed from `listen 443 ssl spdy;` to `listen 443 ssl http2;`

### nginx: caching Nextcloud gallery thumbnails

One of the optimizations for Nextcloud when using nginx as the Web server is to combine FastCGI caching with "Cache Purge", a 3rdparty nginx module that adds the ability to purge content from *FastCGI*, *proxy*, *SCGI* and *uWSGI* caches. This mechanism speeds up thumbnail presentation as it shifts requests to nginx and minimizes php invocations which otherwise would take place for every thumbnail presented every time.

The following procedure is based on an Ubuntu 14.04 system. You may need to adapt it according your OS type and release.

---

**Note:** Unlike Apache, nginx does not dynamically load modules. All modules needed must be compiled into nginx. This is one of the reasons for nginx´s performance. It is expected to have an already running nginx installation with a working configuration set up as described in the Nextcloud documentation.

---

### nginx module check

As a first step, it is necessary to check if your nginx installation has the `nginx cache purge` module compiled in:

```
nginx -V 2>&1 | grep ngx_cache_purge -o
```

If your output contains `ngx_cache_purge`, you can continue with the configuration, otherwise you need to manually compile nginx with the module needed.

### Compile nginx with the `nginx-cache-purge` module

1. **Preparation:**

```
cd /opt
wget http://nginx.org/keys/nginx_signing.key
sudo apt-key add nginx_signing.key
sudo vi /etc/apt/sources.list.d/nginx.list
```

Add the following lines (if different, replace `{trusty}` by your distribution name):

```
deb http://nginx.org/packages/mainline/ubuntu/ trusty nginx
deb -src http://nginx.org/packages/mainline/ubuntu/ trusty nginx
```

Then run `sudo apt-get update`

---

**Note:** If you're not overly cautious and wish to install the latest and greatest nginx packages and features, you may have to install nginx from its mainline repository. From the nginx homepage: "In general, you should deploy nginx from its mainline branch at all times." If you would like to use standard nginx from the latest mainline branch but without compiling in any additional modules, just run `sudo apt-get install nginx`.

---

2. **Download the nginx source from the ppa repository**

```
cd /opt
sudo apt-get build-dep nginx
sudo apt-get source nginx
```

3. **Download module(s) to be compiled in and configure compiler arguments**

```
ls -la
```

Please replace `{release}` with the release downloaded:

```
cd /opt/nginx-{release}/debian
```

If folder "modules" is not present, do:

```
sudo mkdir modules
cd modules
sudo git clone https://github.com/FRiCKLE/ngx_cache_purge.git
sudo vi /opt/nginx-{release}/debian/rules
```

If not present, add the following line at the top under:

```
#export DH_VERBOSE=1:
MODULESDIR = $(CURDIR)/debian/modules
```

And at the end of every `configure` command add:

```
--add-module=$(MODULESDIR)/ngx_cache_purge
```

Don't forget to escape preceding lines with a backslash \. The parameters may now look like:

```
--with-cc-opt="$(CFLAGS)" \
--with-ld-opt="$(LDFLAGS)" \
--with-ipv6 \
--add-module=$(MODULESDIR)/ngx_cache_purge
```

4. **Compile and install nginx**

```
cd /opt/nginx-{release}
sudo dpkg-buildpackage -uc -b
ls -la /opt
sudo dpkg --install /opt/nginx_{release}~{distribution}_amd64.deb
```

5. **Check if the compilation and installation of the ngx_cache_purge module was successful**

```
nginx -V 2>&1 | grep ngx_cache_purge -o
```

It should now show: `ngx_cache_purge`

Show nginx version including all features compiled and installed:

```
nginx -V 2>&1 | sed s/" --"/"\n\t--"/g
```

6. **Mark nginx to be blocked from further updates via apt-get**

```
sudo dpkg --get-selections | grep nginx
```

For every nginx component listed run `sudo apt-mark hold <component>`

7. **Regular checks for nginx updates**

Do a regular visit on the nginx news page and proceed in case of updates with items 2 to 5.

### Configure nginx with the `nginx-cache-purge` module

1. **Preparation** Create a directory where nginx will save the cached thumbnails. Use any path that fits to your environment. Replace `{path}` in this example with your path created:

```
sudo mkdir -p /usr/local/tmp/cache
```

2. **Configuration**

```
sudo vi /etc/nginx/sites-enabled/{your-nextcloud-nginx-config-file}
```

Add at the *beginning*, but *outside* the `server{}` block:

```
# cache_purge
fastcgi_cache_path {path} levels=1:2 keys_zone=NEXTCLOUD:100m inactive=60m;
map $request_uri $skip_cache {
     default 1;
     ~*/thumbnail.php 0;
     ~*/apps/galleryplus/ 0;
     ~*/apps/gallery/ 0;
}
```

**Note:** Please adopt or delete any regex line in the `map` block according your needs and the Nextcloud version used. As an alternative to mapping, you can use as many `if` statements in your server block as necessary:

```
set $skip_cache 1;
if ($request_uri ~* "thumbnail.php")       { set $skip_cache 0; }
if ($request_uri ~* "/apps/galleryplus/") { set $skip_cache 0; }
if ($request_uri ~* "/apps/gallery/")      { set $skip_cache 0; }
```

Add *inside* the `server{}` block, as an example of a configuration:

```
# cache_purge (with $http_cookies we have unique keys for the user)
fastcgi_cache_key $http_cookie$request_method$host$request_uri;
fastcgi_cache_use_stale error timeout invalid_header http_500;
fastcgi_ignore_headers Cache-Control Expires Set-Cookie;

location ~ \.php(?:$/) {
      fastcgi_split_path_info ^(.+\.php)(/.*)$;

      include fastcgi_params;
      fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
      fastcgi_param PATH_INFO $fastcgi_path_info;
      fastcgi_param HTTPS on;
      fastcgi_pass php-handler;
      fastcgi_request_buffering off; #Available since nginx 1.7.11

      # cache_purge
      fastcgi_cache_bypass $skip_cache;
      fastcgi_no_cache $skip_cache;
      fastcgi_cache NEXTCLOUD;
      fastcgi_cache_valid  60m;
      fastcgi_cache_methods GET HEAD;
      }
```

**Note:** Note regarding the `fastcgi_pass` parameter: Use whatever fits your configuration. In the example above, an `upstream` was defined in an nginx global configuration file. This may look like:

```
upstream php-handler {
    server unix:/var/run/php5-fpm.sock;
    # or
    # server 127.0.0.1:9000;
}
```

3. **Test the configuration**

```
sudo nginx -s reload
```

- Open your browser and clear your cache.
- Logon to your Nextcloud instance, open the gallery app, move through your folders and watch while the thumbnails are generated for the first time.
- You may also watch with eg. `htop` your system load while the thumbnails are processed.
- Go to another app or logout and relogon.
- Open the gallery app again and browse to the folders you accessed before. Your thumbnails should appear more or less immediately.

- `htop` will not show up additional load while processing, compared to the high load before.

## Nginx Configuration

The following configuration should be used when Nextcloud is placed in the webroot of your Nginx installation. 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`.

Thanks to @josh4trunks for providing / creating these configuration examples.

### Nextcloud in the webroot of nginx

The following config should be used when Nextcloud is placed in the webroot of your nginx installation.

```
upstream php-handler {
    server 127.0.0.1:9000;
    #server unix:/var/run/php5-fpm.sock;
}

server {
    listen 80;
    server_name cloud.example.com;
    # enforce https
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl;
    server_name cloud.example.com;

    ssl_certificate /etc/ssl/nginx/cloud.example.com.crt;
    ssl_certificate_key /etc/ssl/nginx/cloud.example.com.key;

    # Add headers to serve security related headers
    # Before enabling Strict-Transport-Security headers please read into this
    # topic first.
    # add_header Strict-Transport-Security "max-age=15768000;
    # includeSubDomains; preload;";
    #
    # 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 X-Content-Type-Options nosniff;
    add_header X-Frame-Options "SAMEORIGIN";
    add_header X-XSS-Protection "1; mode=block";
    add_header X-Robots-Tag none;
    add_header X-Download-Options noopen;
    add_header X-Permitted-Cross-Domain-Policies none;

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

    location = /robots.txt {
```

```
        allow all;
        log_not_found off;
        access_log off;
    }

    # The following 2 rules are only needed for the user_webfinger app.
    # Uncomment it if you're planning to use this app.
    #rewrite ^/.well-known/host-meta /public.php?service=host-meta last;
    #rewrite ^/.well-known/host-meta.json /public.php?service=host-meta-json
    # last;

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

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

    # Disable gzip to avoid the removal of the ETag header
    gzip off;

    # Uncomment if your server is build with the ngx_pagespeed module
    # This module is currently not supported.
    #pagespeed off;

    location / {
        rewrite ^ /index.php$uri;
    }

    location ~ ^/(?:build|tests|config|lib|3rdparty|templates|data)/ {
        deny all;
    }
    location ~ ^/(?:\.|autotest|occ|issue|indie|db_|console) {
        deny all;
    }

    location ~ ^/(?:index|remote|public|cron|core/ajax/update|status|ocs/v[12]|updater/.+|ocs-provide
        fastcgi_split_path_info ^(.+\.php)(/.*)$;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param PATH_INFO $fastcgi_path_info;
        fastcgi_param HTTPS on;
        #Avoid sending the security headers twice
        fastcgi_param modHeadersAvailable true;
        fastcgi_param front_controller_active true;
        fastcgi_pass php-handler;
        fastcgi_intercept_errors on;
        fastcgi_request_buffering off;
    }

    location ~ ^/(?:updater|ocs-provider)(?:$|/) {
        try_files $uri/ =404;
        index index.php;
    }
```

```
    # Adding the cache control header for js and css files
    # Make sure it is BELOW the PHP block
    location ~* \.(?:css|js|woff|svg|gif)$ {
        try_files $uri /index.php$uri$is_args$args;
        add_header Cache-Control "public, max-age=7200";
        # Add headers to serve security related headers (It is intended to
        # have those duplicated to the ones above)
        # Before enabling Strict-Transport-Security headers please read into
        # this topic first.
        # add_header Strict-Transport-Security "max-age=15768000;
        #  includeSubDomains; preload;";
        #
        # 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 X-Content-Type-Options nosniff;
        add_header X-Frame-Options "SAMEORIGIN";
        add_header X-XSS-Protection "1; mode=block";
        add_header X-Robots-Tag none;
        add_header X-Download-Options noopen;
        add_header X-Permitted-Cross-Domain-Policies none;
        # Optional: Don't log access to assets
        access_log off;
    }

    location ~* \.(?:png|html|ttf|ico|jpg|jpeg)$ {
        try_files $uri /index.php$uri$is_args$args;
        # Optional: Don't log access to other assets
        access_log off;
    }
}
```

## Nextcloud in a subdir of nginx

The following config should be used when Nextcloud is placed within a subdir of your nginx installation.

```
upstream php-handler {
    server 127.0.0.1:9000;
    #server unix:/var/run/php5-fpm.sock;
}

server {
    listen 80;
    server_name cloud.example.com;
    # enforce https
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl;
    server_name cloud.example.com;

    ssl_certificate /etc/ssl/nginx/cloud.example.com.crt;
    ssl_certificate_key /etc/ssl/nginx/cloud.example.com.key;
```

```
    # Add headers to serve security related headers
    # Before enabling Strict-Transport-Security headers please read into this
    # topic first.
    #add_header Strict-Transport-Security "max-age=15768000;
    # includeSubDomains; preload;";
    add_header X-Content-Type-Options nosniff;
    add_header X-Frame-Options "SAMEORIGIN";
    add_header X-XSS-Protection "1; mode=block";
    add_header X-Robots-Tag none;
    add_header X-Download-Options noopen;
    add_header X-Permitted-Cross-Domain-Policies none;

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

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

    # The following 2 rules are only needed for the user_webfinger app.
    # Uncomment it if you're planning to use this app.
    # rewrite ^/.well-known/host-meta /nextcloud/public.php?service=host-meta
    # last;
    #rewrite ^/.well-known/host-meta.json
    # /nextcloud/public.php?service=host-meta-json last;

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

    location /.well-known/acme-challenge { }

    location ^~ /nextcloud {

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

        # Disable gzip to avoid the removal of the ETag header
        gzip off;

        # Uncomment if your server is build with the ngx_pagespeed module
        # This module is currently not supported.
        #pagespeed off;

        location /nextcloud {
            rewrite ^ /nextcloud/index.php$uri;
        }

        location ~ ^/nextcloud/(?:build|tests|config|lib|3rdparty|templates|data)/ {
            deny all;
        }
        location ~ ^/nextcloud/(?:\.|autotest|occ|issue|indie|db_|console) {
```

```
            deny all;
        }

        location ~ ^/nextcloud/(?:index|remote|public|cron|core/ajax/update|status|ocs/v[12]|updater/
            fastcgi_split_path_info ^(.+\.php)(/.*)$;
            include fastcgi_params;
            fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
            fastcgi_param PATH_INFO $fastcgi_path_info;
            fastcgi_param HTTPS on;
            #Avoid sending the security headers twice
            fastcgi_param modHeadersAvailable true;
            fastcgi_param front_controller_active true;
            fastcgi_pass php-handler;
            fastcgi_intercept_errors on;
            fastcgi_request_buffering off;
        }

        location ~ ^/nextcloud/(?:updater|ocs-provider)(?:$|/) {
            try_files $uri/ =404;
            index index.php;
        }

        # Adding the cache control header for js and css files
        # Make sure it is BELOW the PHP block
        location ~* \.(?:css|js|woff|svg|gif)$ {
            try_files $uri /nextcloud/index.php$uri$is_args$args;
            add_header Cache-Control "public, max-age=7200";
            # Add headers to serve security related headers  (It is intended
            # to have those duplicated to the ones above)
            # Before enabling Strict-Transport-Security headers please read
            # into this topic first.
            # add_header Strict-Transport-Security "max-age=15768000;
            # includeSubDomains; preload;";
            add_header X-Content-Type-Options nosniff;
            add_header X-Frame-Options "SAMEORIGIN";
            add_header X-XSS-Protection "1; mode=block";
            add_header X-Robots-Tag none;
            add_header X-Download-Options noopen;
            add_header X-Permitted-Cross-Domain-Policies none;
            # Optional: Don't log access to assets
            access_log off;
        }

        location ~* \.(?:png|html|ttf|ico|jpg|jpeg)$ {
            try_files $uri /nextcloud/index.php$uri$is_args$args;
            # Optional: Don't log access to other assets
            access_log off;
        }
    }
}
```

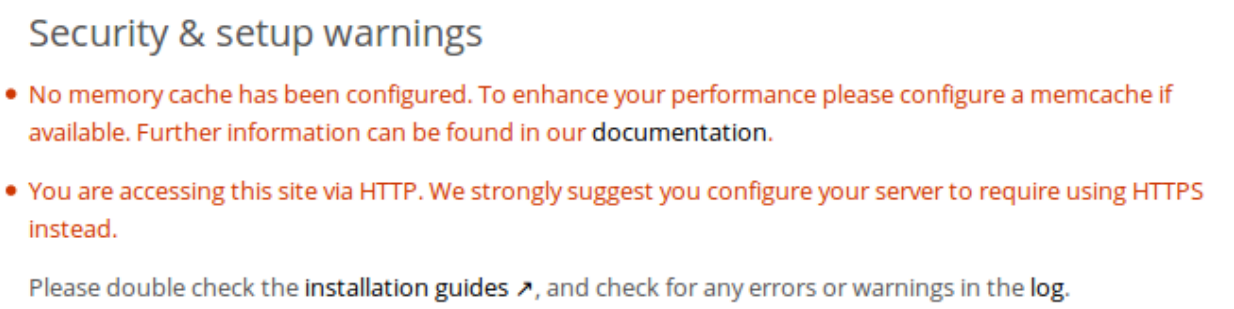
CHAPTER

THREE

# SERVER 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 Configuring 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 Example Configurations

### 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 misbehaviour 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 COMMITED" 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*
- *Encryption*
- *Federation Sync*
- *File Operations*
- *Files External*
- *Integrity Check*
- *l10n, Create Javascript Translation Files for Apps*
- *LDAP Commands*
- *Logging Commands*
- *Maintenance Commands*
- *Security*
- *Shibboleth Modes (Enterprise Edition only)*
- *Trashbin*
- *User Commands*
- *Versions*
- *Command Line Installation*
- *Command Line Upgrade*
- *Two-factor Authentication*
- *Disable Users*

## 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-PHP56 installed, the command looks like this:

```
sudo -u apache /opt/rh/php56/root/usr/bin/php /var/www/html/nextcloud/occ
```

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

```
sudo -u www-data php occ
Nextcloud version 9.0.0

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 9.0.0
```

Query your Nextcloud server status:

```
sudo -u www-data php occ status
  - installed: true
  - version: 9.0.0.19
  - versionstring: 9.0.0
  - 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":"9.0.0.19","versionstring":"9.0.0","edition":""}
```

or `json_pretty`:

```
sudo -u www-data php occ status --output=json_pretty
{
   "installed": true,
   "version": "9.0.0.19",
   "versionstring": "9.0.0",
   "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:** This 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 can **not** use this.

Since Nextcloud 11 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 `--programm 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:check-code   check code to be compliant
 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
```

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
```

`app:check-code` has multiple checks: it checks if an app uses Nextcloud's public API (`OCP`) or private API (`OC_`), and it also checks for deprecated methods and the validity of the `info.xml` file. By default all checks are enabled. The Activity app is an example of a correctly-formatted app:

```
sudo -u www-data php occ app:check-code notifications
App is compliant - awesome job!
```

If your app has issues, you'll see output like this:

```
sudo -u www-data php occ app:check-code foo_app
Analysing /var/www/nextcloud/apps/files/foo_app.php
4 errors
   line   45: OCP\Response - Static method of deprecated class must not be
   called
   line   46: OCP\Response - Static method of deprecated class must not be
   called
   line   47: OCP\Response - Static method of deprecated class must not be
   called
   line   49: OC_Util - Static method of private class must not be called
```

You can get the full filepath to an app:

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

## 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 Defining 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 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
9.0.0.19

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 order to set (and also get) the value of one key, you can specify multiple `config` names separated by spaces:

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

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
```

### 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 addressbooks, calendars, and to migrate addressbooks from 8.2 when you upgrade to 9.0:

```
dav
 dav:create-addressbook        Create a dav addressbook
 dav:create-calendar           Create a dav calendar
 dav:migrate-addressbooks      Migrate addressbooks from the contacts
                               app to core
 dav:migrate-calendars         Migrate calendars from the calendar app to
                               core
 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 on her Calendar and Contacts pages.

In 9.0, the CalDAV server has been integrated into core. Your existing calendars and contacts should migrate automatically when you upgrade. If something goes wrong you can try a manual migration. First delete any partially-migrated calendars or addressbooks. Then run this command to migrate user's contacts:

```
sudo -u www-data php occ dav:migrate-addressbooks [user]
```

Run this command to migrate calendars:

```
sudo -u www-data php occ dav:migrate-calendars [user]
```

See ownCloud 9.0 - calendar migration analysis for help with troubleshooting and reporting problems.

`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

## 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:migrate                   initial migration to encryption 2.0
 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 *single-user 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 *single-user mode* to prevent any user activity until decryption is completed.

Use `encryption:disable` to disable your encryption module. You must first put your Nextcloud server into *single-user 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.

`encryption:migrate` migrates encryption keys after a major Nextcloud version upgrade. You may optionally specify individual users in a space-delimited list.

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:scan                 rescan filesystem
 files:transfer-ownership   All files and folders are moved to another
                            user - shares are moved as well.
```

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 and are exclusive - only one must be specified.

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

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>
```

## 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 mount
```

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 an official tag MUST be code signed with Nextcloud. Unsigned official 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 --certifica
```

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. .. TODO ON RELEASE: Update version number above on release

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

See Code Signing to learn more.

## I10n, 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 no `configID`, like this example:

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

This is a holdover from the early days, when there was no option to create additional configurations. The second, 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:manage     manage logging configuration
 log:owncloud   manipulate Nextcloud logging backend
```

Run `log:owncloud` to see your current logging status:

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

Use the `--enable` option to turn on logging. Use `--file` to set a different log file path. Set your rotation by log file size in bytes with `--rotate-size`; 0 disables rotation.

`log:manage` sets your logging backend, log level, and timezone. The defaults are `owncloud`, `Warning`, and `UTC`. Available options are:

- –backend [owncloud, syslog, errorlog]
- –level [debug, info, warning, error]

## 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: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:singleuser               set single user mode
```

`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
```

Putting your Nextcloud server into single-user mode allows admins to log in and work, but not ordinary users. This is useful for performing maintenance and troubleshooting on a running server:

```
sudo -u www-data php occ maintenance:singleuser --on
Single user mode enabled
```

Turn it off when you're finished:

```
sudo -u www-data php occ maintenance:singleuser --off
Single user mode disabled
```

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.

## 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:import /path/to/certificate
```

Remove a certificate:

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

## Shibboleth Modes (Enterprise Edition only)

**Note:** This command is only available when the "Shibboleth user backend" app (`user_shibboleth`) is enabled.

`shibboleth:mode` sets your Shibboleth mode to `notactive`, `autoprovision`, or `ssoonly`:

```
shibboleth:mode [mode]
```

## Trashbin

**Note:** This command is only available when the "Deleted files" app (`files_trashbin`) is enabled.

The `trashbin:cleanup` command removes the deleted files of the specified users in a space-delimited list, or all users if none are specified.

```
trashbin
 trashbin:cleanup   Remove deleted files
```

This example removes the deleted files of all users:

```
sudo -u www-data php occ trashbin:cleanup
Remove all deleted files
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: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 her 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: 09.01.2015 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  |
+------------------+----+
```

## 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
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 9.0.0

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
 app:check-code        check code to be compliant
 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
 --database-table-prefix  Prefix for all tables (default: oc_)
 --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 [--no-app-disable]

Options:
--no-app-disable       skips the disable of third party apps
--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 available
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 an user losing access to the second factor (e.g. lost phone with two-factor SMS verification), the admin can temporarily disable the two-factor check for that user via the occ command:

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

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

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

### 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.

## Configuring the 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 Configuration is mandatory.

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

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

# Configuring Single-Sign-On

Using the SSO & SAML app of your Nextcloud you can make it easily possible to integrate your existing Single-Sign-On solution with Nextcloud. In addition, you can use the Nextcloud LDAP user provider to keep the convenience for users. (e.g. when sharing)

The following providers are supported and tested at the moment:

- **SAML 2.0**
  - OneLogin
  - Shibboleth
  - Active Directory Federation Services (ADFS)
- **Authentication via Environment Variable**
  - Kerberos (mod_auth_kerb)
  - Any other provider that authenticates using the environment variable

While theoretically any other authentication provider implementing either one of those standards is compatible, we like to note that they are not part of any internal test matrix.

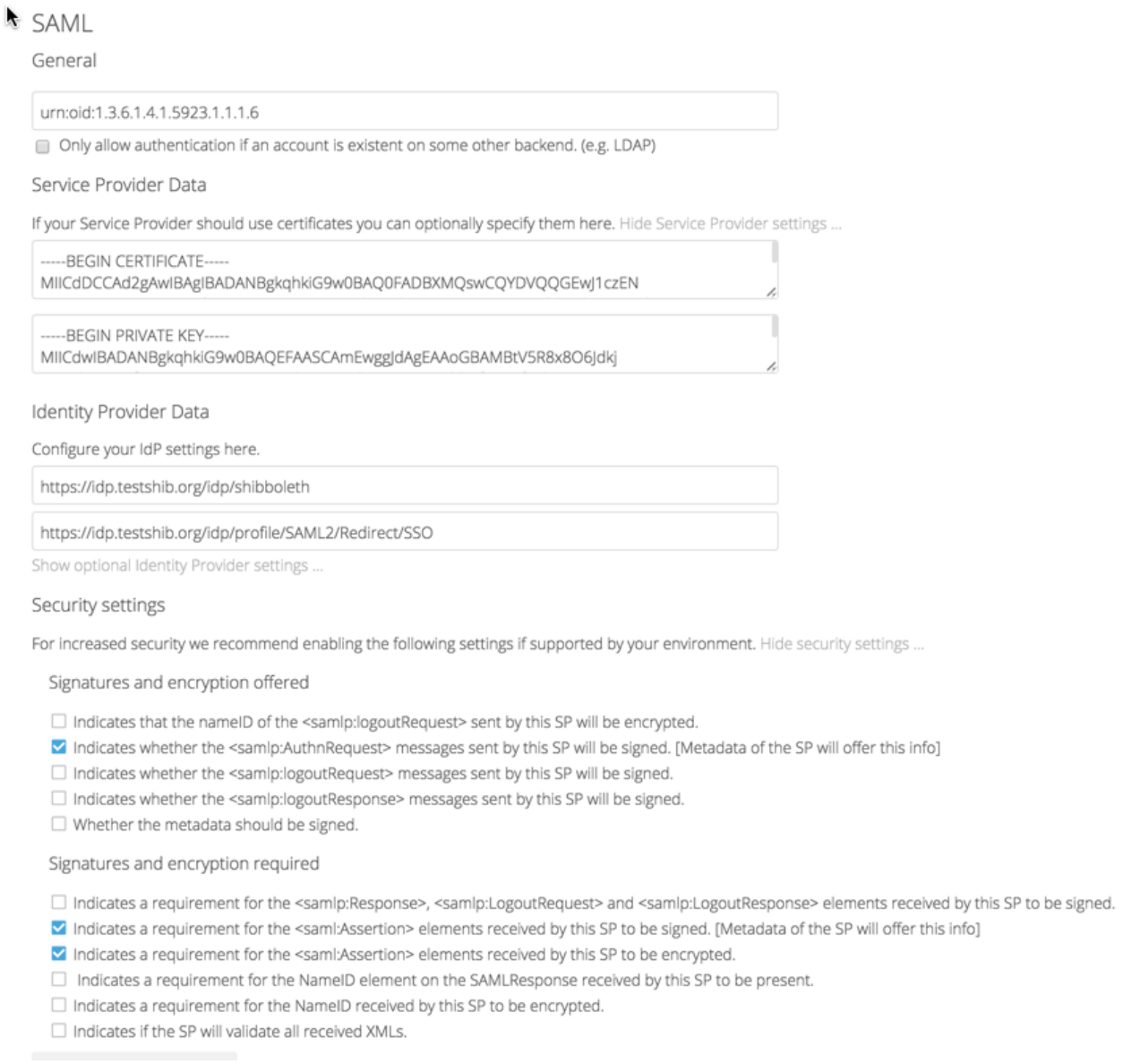
## Enabling the SSO & SAML app

**Warning:** Make sure to configure an administrative user that can access the instance via SSO. Logging-in with your regular Nextcloud account won't be possible anymore.

The "SSO & SAML" App is shipped and disabled by default. To enable the app enabled simply go to your Nextcloud Apps page to enable it. It can then be found in the "SSO & SAML authentication" section of your Nextcloud.

## Configuring SAML 2.0

To configure using SAML choose the "SAML authentication" in the setup wizard of the application. Then configure the application as required by your Service Provider.




## Configuring environment based authentication

It is possible to authenticate against Nextcloud using an environment variable. This is for example relevant in case you use an service provider incompatible with SAML such as Kerberos or don't want to configure SAML in the software yourself.

To enable that choose the "Environment variable" authentication provider in the application and then specify the environment variable. (e.g. *REMOTE_USER* for Kerberos)

Once done you also need to protect the login route properly. On an Apache server with mod_auth_kerb the following configuration would protect the login route:

```
<Location "/index.php/login">
    AuthType Kerberos
```

```
    AuthName "Kerberos Login"
    KrbServiceName HTTP
    KrbMethodNegotiate On
    KrbMethodK5Passwd Off
    KrbSaveCredentials Off
    KrbVerifyKDC On
    KrbAuthRealms NEXTCLOUD-AD.LOCAL
    Krb5KeyTab /etc/apache2/webpage.HTTP.keytab
    Require valid-user
</Location>
```

**Warning:** If this authentication approach is used clients do require an application specific password for authentication. A better integration into our desktop and mobile clients is considered for the future though.

## Configuring 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.
- **Memcached** Distributed cache for multi-server Nextcloud installations.
- **Redis, PHP module 2.2.6 and up required.** 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 Config.php 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*.

### APCu

PHP 5.5 and up include the Zend OPcache in core, and on most Linux distributions it is enabled by default. However, it does not bundle a data cache. 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 `php5-apcu` or `php7.0-apcu`. On Ubuntu 14.04 LTS, the APCu version (4.0.2) is too old to use with Nextcloud (requires 4.0.6+). You may install 4.0.7 from Ubuntu backports with this command:

```
apt-get install php5-apcu/trusty-backports
```

Then restart your Web server.

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.

## 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 `php5-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' => array(
     array('localhost', 11211),
     array('server1.example.com', 11211),
     array('server2.example.com', 11211),
     ),
```

## Redis

Redis is an excellent modern memcache to use for both distributed caching, and as a local cache 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 `php5-redis` or `php7.0-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 local server cache:

```
'memcache.local' => '\OC\Memcache\Redis',
'redis' => array(
     'host' => 'localhost',
     '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\Redis',
'redis' => array(
     'host' => '/var/run/redis/redis.sock',
     'port' => 0,
     'dbindex' => 0,
     'password' => 'secret',
     'timeout' => 1.5,
      ),
```

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

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

## 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 Config.php Parameters) to select a different location.

## Recommendations Based on Type of Deployment

### Small/Private Home Server

Only use APCu:

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

### Small Organization, Single-server Setup

Use APCu for local caching, Redis for file locking:

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

```
      'port' => 6379,
       ),
```

### Large Organization, Clustered Setup

Use Redis for everything except local memcache:

```
'memcache.distributed' => '\OC\Memcache\Redis',
'memcache.locking' => '\OC\Memcache\Redis',
'memcache.local' => '\OC\Memcache\APCu',
'redis' => array(
     'host' => 'localhost',
     '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 `php5-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.

The Redis PHP module must be at least version 2.2.6. Please note that the Redis PHP module versions 2.2.x will only work for PHP 5.6.x.

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 `php5-redis` version, or the version of your installed package:

```
apt-cache policy php5-redis
```

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

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

# Defining 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 an Nextcloud internal process that runs such background jobs on demand.

Nextcloud plug-in applications 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

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.

---

**Note:** Especially when using the Activity App or external storages, where new files are added, updated or deleted one of the two methods below should be preferred.

---

### Webcron

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
```

### 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 15 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
*/15  *  *  *  * 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
*/15  *  *  *  * 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:** 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

[Install]
WantedBy=basic.target
```

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.

**nextcloudcron.timer** should look like this:

```
[Unit]
Description=Run Nextcloud cron.php every 15 minutes

[Timer]
OnBootSec=5min
OnUnitActiveSec=15min
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 15 minute timer after the service-unit was last activated.

Now all that is left is to start and enable the timer by running these commands:

```
systemctl start nextcloudcron.timer
systemctl enable nextcloudcron.timer
```

**Note:** Select the option `Cron` in the admin menu for background jobs. if left on `AJAX` it would execute the AJAX job on every page load.

## Config.php 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!**

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' =>
  array (
    'demo.example.org',
    'otherdomain.example.org',
  ),
```

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

```
'datadirectory' => '/var/www/nextcloud/data',
```

Where user files are stored; this defaults to `data/` in the Nextcloud directory. The SQLite database is also stored here, when you use SQLite.

```
'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' => 'sqlite',
```

Identifies the database used with this installation. See also config option `supportedDatabases`

**Available:**

- sqlite (SQLite3)
- mysql (MySQL/MariaDB)
- pgsql (PostgreSQL)
- oci (Oracle)

```
'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 `localhost:/path/to/socket`.

```
'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.

```
'installed' => false,
```

Indicates whether the Nextcloud instance was installed successfully; `true` indicates a successful installation, and `false` indicates an unsuccessful installation.

## 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' => '2c57784763467863O6303',
  '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.

```
'defaultapp' => '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 Files app.

```
'knowledgebaseenabled' => true,
```

`true` enables the Help menu item in the user menu (top right of the Nextcloud Web interface). `false` removes the Help item.

```
'enable_avatars' => true,
```

`true` enables avatars, or user profile photos. These appear on the User page, on user's Personal pages and are used by some apps (contacts, mail, etc). `false` disables them.

```
'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, which is set when the user clicks the `remember` checkbox on the login screen. The default is 15 days, expressed in seconds.

```
'session_lifetime' => 60 * 60 * 24,
```

The lifetime of a session after inactivity; the default is 24 hours, expressed in seconds.

```
'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.

```
'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.

```
'auth.bruteforce.protection.enabled' => true,
```

Whether the bruteforce protection shipped with Nextcloud should be enabled or not.

Disabling this is discouraged for security reasons.

```
'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.

```
'user_backends' => array(
        array(
                'class' => 'OC_User_IMAP',
                'arguments' => array('{imap.gmail.com:993/imap/ssl}INBOX')
        )
),
```

The `user_backends` app (which needs to be enabled first) allows you to configure alternate authentication backends. Supported backends are: IMAP (OC_User_IMAP), SMB (OC_User_SMB), and FTP (OC_User_FTP).

```
'lost_password_link' => 'https://example.org/link/to/password/reset',
```

If your user backend does not allow to reset the password (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.

## 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.

```
'mail_smtpdebug' => false,
```

Enable SMTP class debugging.

```
'mail_smtpmode' => 'sendmail',
```

Which mode to use for sending mail: `sendmail`, `smtp`, `qmail` or `php`.

If you are using local or remote SMTP, set this to `smtp`.

If you are using PHP mail you must have an installed and working email system on the server. The program used to send email is defined in the `php.ini` file.

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.

```
'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`.

```
'mail_smtpport' => 25,
```

This depends on `mail_smtpmode`. Specify the port for sending mail.

```
'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.

```
'mail_smtpsecure' => '',
```

This depends on `mail_smtpmode`. Specify when you are using `ssl` or `tls`, or leave empty for no encryption.

```
'mail_smtpauth' => false,
```

This depends on `mail_smtpmode`. Change this to `true` if your mail server requires authentication.

```
'mail_smtpauthtype' => 'LOGIN',
```

This depends on `mail_smtpmode`. If SMTP authentication is required, choose the authentication type as `LOGIN` (default) or `PLAIN`.

```
'mail_smtpname' => '',
```

This depends on `mail_smtpauth`. Specify the username for authenticating to the SMTP server.

```
'mail_smtppassword' => '',
```

This depends on `mail_smtpauth`. Specify the password for authenticating to the SMTP server.

## 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]$`

```
'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`

```
'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 above rule is not valid in every case, 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* and when following conditions are met Nextcloud uses URLs without index.php in it:

- *mod_rewrite* is installed
- *mod_env* is installed

```
'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`.

```
'proxyuserpwd' => '',
```

The optional authentication for the proxy to use to connect to the internet.

The format is: `username:password`.

## 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
- **`disabled`** trash bin auto clean disabled, files and folders will be kept forever

## 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

## 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.

```
'updatechecker' => true,
```

Check if Nextcloud is up-to-date and shows a notification if a new version is available.

```
'updater.server.url' => 'https://updates.nextcloud.com/updater_server/',
```

URL that Nextcloud should use to look for updates

```
'updater.release.channel' => 'stable',
```

The channel that Nextcloud should use to look for updates

**Supported values:**

- `daily`
- `beta`
- `stable`
- `production`

```
'has_internet_connection' => true,
```

Is Nextcloud connected to the Internet or running in a closed network?

```
'check_for_working_webdav' => true,
```

Allows Nextcloud to verify a working WebDAV connection. This is done by attempting to make a WebDAV request from PHP.

```
'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/

```
'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.

```
'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.

## Logging

```
'log_type' => 'file',
```

By default the Nextcloud logs are sent to the `nextcloud.log` file in the default Nextcloud data directory.

If syslogging is desired, set this parameter to `syslog`. Setting this parameter to `errorlog` will use the PHP error_log function for logging.

```
'logfile' => '/var/log/nextcloud.log',
```

Log file path for the Nextcloud logging type.

Defaults to `[datadirectory]/nextcloud.log`

```
'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.

```
'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`.

The default value is `Nextcloud`.

```
'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 http://php.net/manual/en/function.date.php

```
'logtimezone' => 'Europe/Berlin',
```

The default timezone for logfiles is UTC. You may change this; see http://php.net/manual/en/timezones.php

```
'log_query' => false,
```

Append all database queries and parameters to the log file. Use this only for debugging, as your logfile will become huge.

```
'cron_log' => true,
```

Log successful cron runs.

```
'log_rotate_size' => false,
```

Enables log rotation and limits the total size of logfiles. The default is 0, or 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.

## 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',
```

This section is for configuring the download links for Nextcloud clients, as seen in the first-run wizard and on Personal pages.

## 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.

```
'apps_paths' => array(
        array(
                '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.

## 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

```
'preview_max_x' => 2048,
```

The maximum width, in pixels, of a preview. A value of `null` means there is no limit.

```
'preview_max_y' => 2048,
```

The maximum height, in pixels, of a preview. A value of `null` means there is no limit.

```
'preview_max_scale_factor' => 10,
```

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. A value of `1` or `null` disables scaling.

```
'preview_max_filesize_image' => 50,
```

max file size for generating image previews with imagegd (default behaviour) If the image is bigger, it'll try other preview generators, but will most likely show the default mimetype icon

Value represents the maximum filesize in megabytes Default is 50 Set to -1 for no limit

```
'preview_libreoffice_path' => '/usr/bin/libreoffice',
```

custom path for LibreOffice/OpenOffice binary

```
'preview_office_cl_parameters' =>
        ' --headless --nologo --nofirststartwizard --invisible --norestore '.
        '--convert-to pdf --outdir ',
```

Use this if LibreOffice/OpenOffice requires additional arguments.

```
'enabledPreviewProviders' => array(
        'OC\Preview\PNG',
        'OC\Preview\JPEG',
        'OC\Preview\GIF',
        'OC\Preview\BMP',
```

```
        'OC\Preview\XBitmap',
        'OC\Preview\MP3',
        'OC\Preview\TXT',
        'OC\Preview\MarkDown'
),
```

Only register providers that have been explicitly enabled

The following providers are enabled by default:

- OC\Preview\PNG
- OC\Preview\JPEG
- OC\Preview\GIF
- OC\Preview\BMP
- OC\Preview\XBitmap
- OC\Preview\MarkDown
- OC\Preview\MP3
- OC\Preview\TXT

The following providers are disabled by default due to performance or privacy concerns:

- OC\Preview\Illustrator
- OC\Preview\Movie
- OC\Preview\MSOffice2003
- OC\Preview\MSOffice2007
- OC\Preview\MSOfficeDoc
- OC\Preview\OpenDocument
- OC\Preview\PDF
- OC\Preview\Photoshop
- OC\Preview\Postscript
- OC\Preview\StarOffice
- OC\Preview\SVG
- OC\Preview\TIFF
- OC\Preview\Font

**Note:** Troubleshooting steps for the MS Word previews are available at the `../configuration_files/collaborative_documents_configuration` section of the Administrators Manual.

The following providers are not available in Microsoft Windows:

- OC\Preview\Movie
- OC\Preview\MSOfficeDoc
- OC\Preview\MSOffice2003
- OC\Preview\MSOffice2007

- OC\Preview\OpenDocument
- OC\Preview\StarOffice

## 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`

## 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.

```
'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 comments.

## 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.

```
'singleuser' => false,
```

When set to `true`, the Nextcloud instance will be unavailable for all users who are not in the `admin` group.

## SSL

```
'openssl' => array(
        'config' => '/absolute/location/of/openssl.cnf',
),
```

Extra SSL options to be used for configuration.

## Memory caching backend configuration

Available cache backends:

- `\OC\Memcache\APC` Alternative PHP Cache backend
- `\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
- `\OC\Memcache\XCache` XCache 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

```
'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

```
'redis' => array(
        'host' => 'localhost', // can also be a unix domain socket: '/tmp/redis.sock'
        'port' => 6379,
        'timeout' => 0.0,
        'password' => '', // Optional, if not defined no password will be used.
        'dbindex' => 0, // Optional, if undefined SELECT will not run and will use Redis Server's de
),
```

Connection details for redis to use for memory caching.

For enhanced security it is recommended to configure Redis to require a password. See http://redis.io/topics/security for more information.

```
'memcached_servers' => array(
        // hostname, port and optional weight. Also see:
        // http://www.php.net/manual/en/memcached.addservers.php
        // http://www.php.net/manual/en/memcached.addserver.php
        array('localhost', 11211),
        //array('other.host.local', 11211),
),
```

Server details for one or more memcached servers to use for memory caching.

```
'memcached_options' => array(
        // 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, see http://apprize.info/php/scaling/15.html

```
'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.

```
'cache_chunk_gc_ttl' => 86400, // 60*60*24 = 1 day
```

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.

## Using Object Store with Nextcloud

```
'objectstore' => [
        'class' => 'OC\\Files\\ObjectStore\\Swift',
        'arguments' => [
                // trystack will user 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/

## 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 be used that – for instance – uses the filesystem instead of the database to keep the share information.

```
'sharing.maxAutocompleteResults' => 0,
```

Define max number of results returned by the user search for auto-completion Default is unlimited (value set to 0).

```
'sharing.minSearchStringLength' => 0,
```

Define the minimum length of the search string before we start auto-completion Default is no limit (value set to 0)

## All other configuration options

```
'dbdriveroptions' => array(
        PDO::MYSQL_ATTR_SSL_CA => '/file/path/to/ca_cert.pem',
        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.

```
'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,
```

If this setting is set to true MySQL can handle 4 byte characters instead of 3 byte characters

MySQL requires a special setup for longer indexes (> 767 bytes) which are needed:

[mysqld] innodb_large_prefix=true innodb_file_format=barracuda innodb_file_per_table=true

**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/

WARNING: EXPERIMENTAL

```
'supportedDatabases' => array(
        'sqlite',
        'mysql',
        'pgsql',
        'oci',
),
```

Database types that are supported for installation.

**Available:**

- sqlite (SQLite3)
- mysql (MySQL)
- pgsql (PostgreSQL)
- oci (Oracle)

```
'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 storages which do not support streaming are in use.

The Web server user must have write access to this directory.

```
'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' => array('.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.

```
'share_folder' => '/',
```

Define a default folder for shared files and folders other than root.

```
'theme' => '',
```

If you are applying a theme to Nextcloud, enter the name of the theme here.

The default location for themes is `nextcloud/themes/`.

```
'cipher' => 'AES-256-CFB',
```

The default cipher for encrypting files. Currently AES-128-CFB and AES-256-CFB are supported.

```
'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.

```
'quota_include_external_storage' => false,
```

EXPERIMENTAL: option whether to include external storage in quota calculation, defaults to false.

```
'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 storages.

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.

```
'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.

```
'mount_file' => '/var/www/nextcloud/data/mount.json',
```

Where `mount.json` file should be stored, 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.

```
'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' => array('203.0.113.45', '198.51.100.128'),
```

List of trusted proxy servers

If you configure these also consider setting *forwarded_for_headers* which otherwise defaults to *HTTP_X_FORWARDED_FOR* (the *X-Forwarded-For* header).

```
'forwarded_for_headers' => array('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_FORWARED_FOR' if unset

```
'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. Default is `10`. Set to `-1` for no limit.

```
'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.

```
'filelocking.ttl' => 3600,
```

Set the time-to-live for locks in secconds.

Any lock older than this will be automatically cleaned up.

If not set this defaults to either 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*.

```
'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

```
'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.

```
'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.

## App config options

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`

```
'wnd.logging.enable' => true,
```

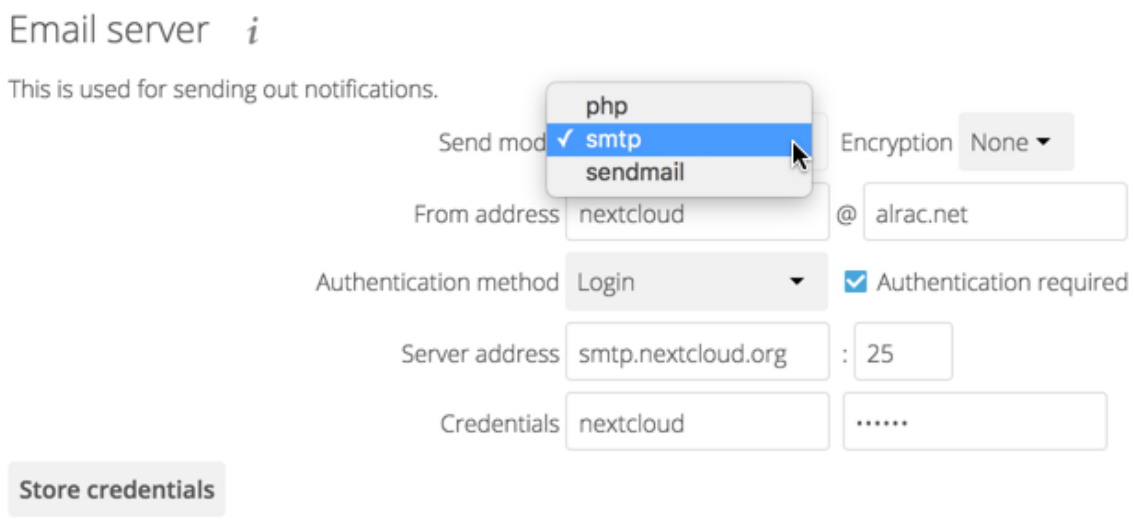
This enables debug logs for the windows_network_drive app.

## Email Configuration

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.

Nextcloud 7 introduces a new feature, the graphical Email Configuration Wizard.




With the new 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, PHP, and Sendmail. Use the SMTP configurator for a remote server, and PHP or Sendmail when your mail server is on the same machine as Nextcloud.

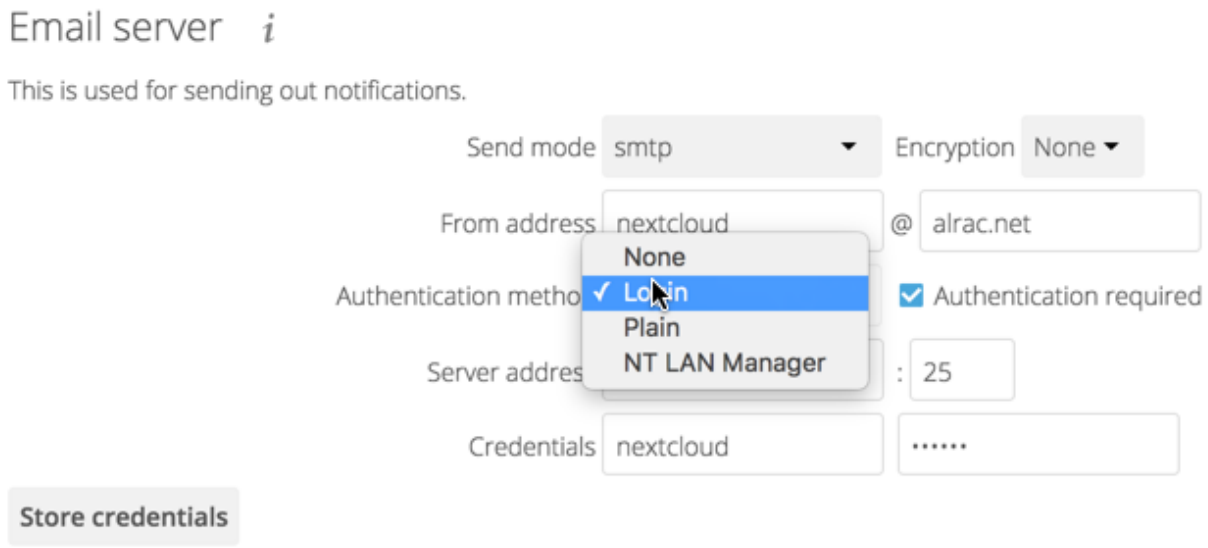
**Note:** The Sendmail option refers to the Sendmail SMTP server, and any drop-in Sendmail replacement such as Postfix, Exim, or Courier. All of these include a `sendmail` binary, and are freely-interchangeable.

### Configuring an SMTP Server

You need the following information from your mailserver administrator to connect Nextcloud to a remote SMTP server:

- Encryption type: None, SSL, or TLS
- 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
- Login credentials, if required



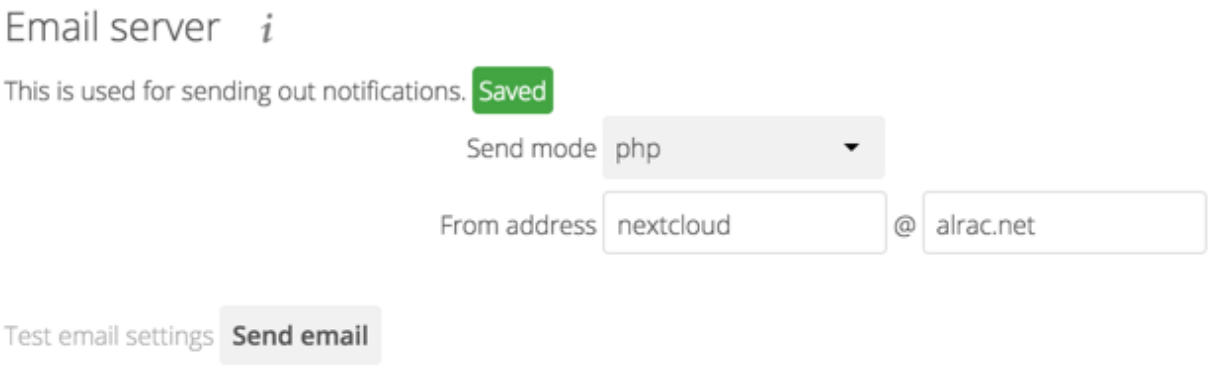

Your changes are saved immediately, and you can click the Send Email button to test your configuration. This sends a test message to the email address you configured on your Personal page. The test message says:

```
If you received this email, the settings seem to be correct.

--
Nextcloud
a safe home for all your data
```

## Configuring PHP and Sendmail

Configuring PHP or Sendmail requires only that you select one of them, and then enter your desired return address.




How do you decide which one to use? PHP mode uses your local `sendmail` binary. Use this if you want to use `php.ini` to control some of your mail server functions, such as setting paths, headers, or passing extra command

options to the `sendmail` binary. These vary according to which server you are using, so consult your server's documentation to see what your options are.

In most cases the `smtp` option is best, because it removes the extra step of passing through PHP, and you can control all of your mail server options in one place, in your mail server configuration.

## Using Email Templates

Another useful new feature is editable email templates. Now you can edit Nextcloud's email templates on your Admin page. These are your available templates:

- Sharing email (HTML) – HTML version of emails notifying users of new file shares
- Sharing email (plain text fallback) – Plain text email notifying users of new file shares
- Lost password mail – Password reset email for users who lose their passwords.
- Activity notification mail – Notification of activities that users have enabled in the Notifications section of their Personal pages.

In addition to providing the email templates, this feature enables you to apply any preconfigured themes to the email.

To modify an email template to users:

1. Access the Admin page.
2. Scroll to the Mail templates section.
3. Select a template from the drop-down menu.
4. Make any desired modifications to the template.

The templates are written in PHP and HTML, and are already loaded with the relevant variables such as username, share links, and filenames. You can, if you are careful, edit these even without knowing PHP or HTML; don't touch any of the code, but you can edit the text portions of the messages. For example, this the lost password mail template:

```
<?php

   echo str_replace('{link}', $_['link'], $l->t('Use the following link to
   reset your password: {link}'));
```

You could change the text portion of the template, `Use the following link to reset your password:` to say something else, such as `Click the following link to reset your password. If you did not ask for a password reset, ignore this message.`

Again, be very careful to change nothing but the message text, because the tiniest coding error will break the template.

---

**Note:** You can edit the templates directly in the template text box, or you can copy and paste them to a text editor for modification and then copy and paste them back to the template text box for use when you are done.

---

## 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, PHP, 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.

```
<?php

  "mail_smtpmode"     => "smtp",
  "mail_smtphost"     => "smtp.server.dom:425",
```

or

```
<?php

  "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:

```
<?php

  "mail_smtptimeout"  => 30,
```

If the SMTP server accepts insecure connections, the default setting can be used:

```
<?php

"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:

```
<?php

  "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:

```
<?php

  "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.

```
<?php

  "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**.

```
<?php

  "mail_smtpauth"     => true,
  "mail_smtpauthtype" => "LOGIN",
  "mail_smtpname"     => "username",
  "mail_smtppassword" => "password",
```

### PHP mail

If you want to use PHP mail it is necessary to have an installed and working email system on your server. Which program in detail is used to send email is defined by the configuration settings in the **php.ini** file. (On *nix systems this will most likely be Sendmail.) Nextcloud should be able to send email out of the box.

```
<?php

  "mail_smtpmode"     => "php",
  "mail_smtphost"     => "127.0.0.1",
  "mail_smtpport"     => 25,
  "mail_smtptimeout"  => 10,
  "mail_smtpsecure"   => "",
  "mail_smtpauth"     => false,
  "mail_smtpauthtype" => "LOGIN",
  "mail_smtpname"     => "",
  "mail_smtppassword" => "",
```

### 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.

```
<?php

  "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 sendmail binary (**/var/qmail/bin/sendmail**) will then be used to send email. Nextcloud should be able to send email out of the box.

```
<?php

  "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`.

```
<?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:

```
<?php

  "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.
```

### 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:

```
<?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.

## Linking External Sites

You can embed external Web sites inside your Nextcloud pages with the External Sites app, as this screenshot shows.

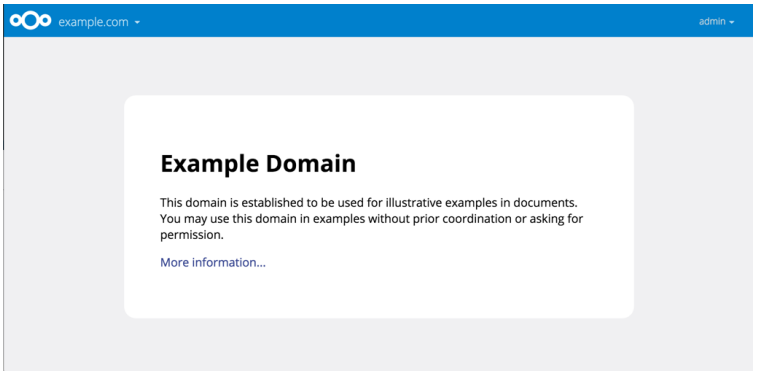



Fig. 3.1: *Click to enlarge*

This is useful for quick access to important Web 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 is included in all versions of Nextcloud. Go to **Apps > Not Enabled** to enable it. Then go to your Nextcloud Admin page to create your links, which are saved automatically. There is a dropdown menu to select an icon, but there is only one default icon so you don't have to select one. Hover your cursor to the right of your links to make the trashcan icon appear when you want to remove them.

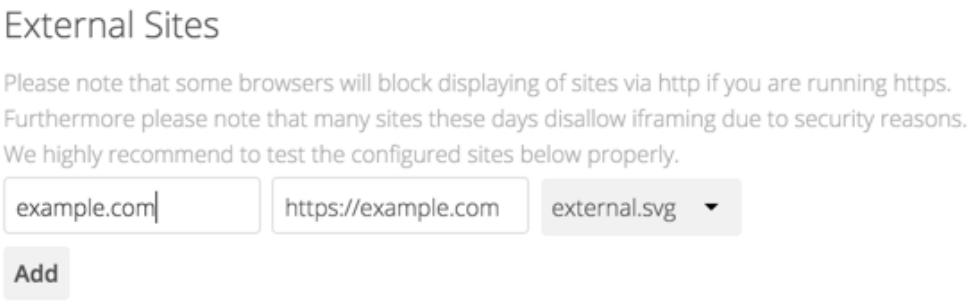
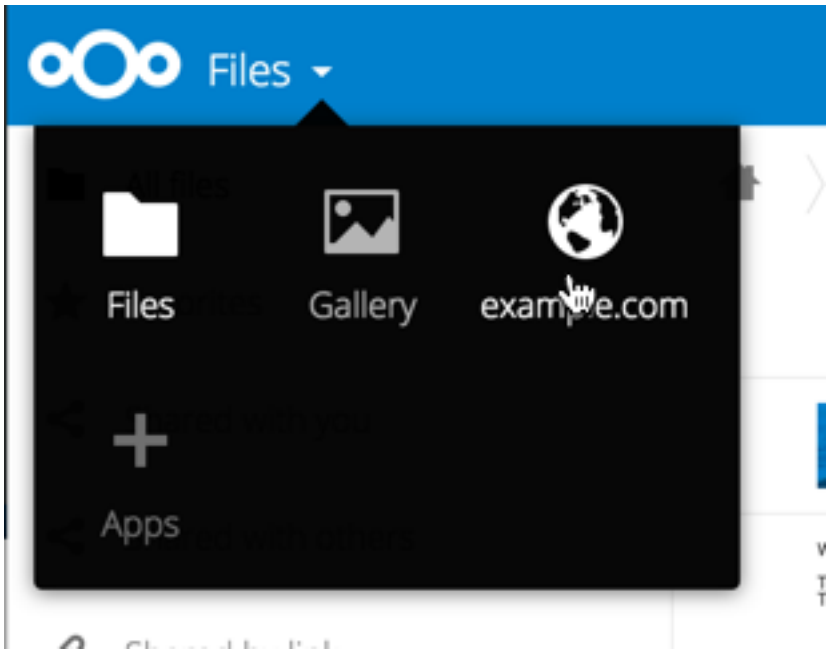


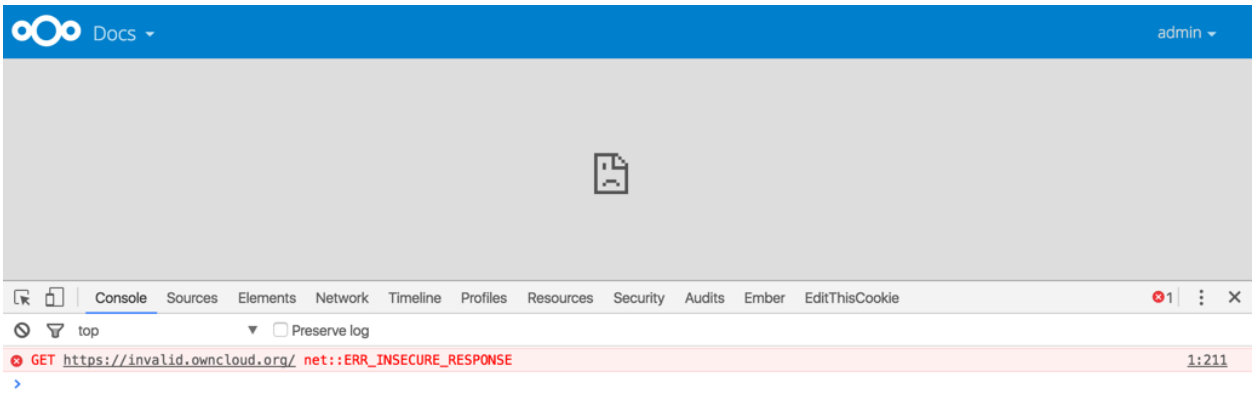

Fig. 3.2: *Click to enlarge*

The links appear in the Nextcloud dropdown menu on the top left after refreshing your page, and have globe icons.

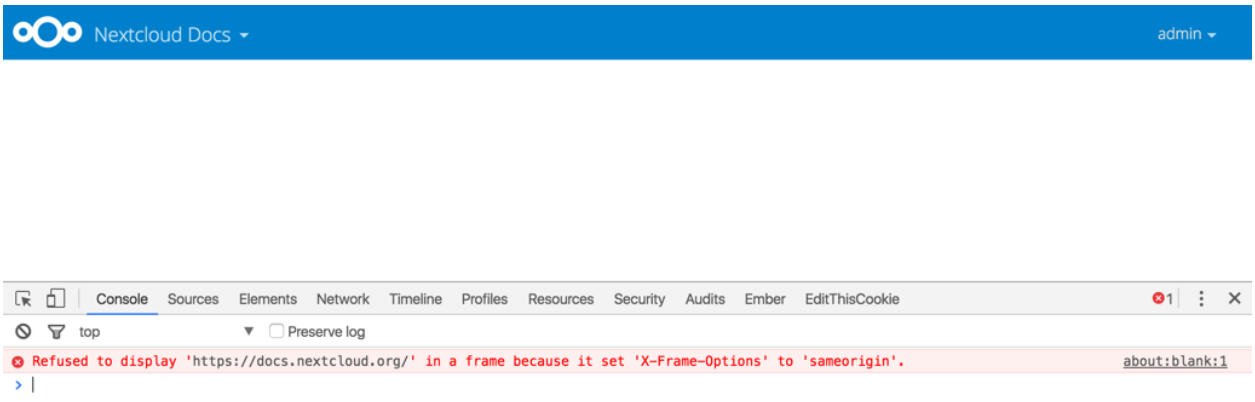



Your links may or may not work correctly due to the various ways that Web browsers and Web sites handle HTTP and HTTPS URLs, and because the External Sites app embeds external links in IFrames. Modern Web browsers try very hard to protect Web surfers from dangerous links, and safety apps like Privacy 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 isn't much you can do about these issues, but if you're curious you can see what is happening.

## Custom Client Download Repositories

You may configure the URLs to your own download repositories for your Nextcloud desktop clients and mobile apps in `config/config.php`. This example shows the default download locations:

```
<?php

  "customclient_desktop" => "https://nextcloud.com/install/",
  "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",
```

Simply replace the URLs with the links to your own preferred download repos.

You may test alternate URLs without editing `config/config.php` by setting a test URL as an environment variable:

```
export OCC_UPDATE_URL=https://test.example.com
```

When you're finished testing you can disable the environment variable:

```
unset OCC_UPDATE_URL
```

## Knowledge Base Configuration

The usage of Nextcloud is more or less self explaining but nevertheless a user might run into a problem where he needs to consult the documentation or knowledge base. To ease access to the Nextcloud documentation and knowledge base, a help menu item is shown in the settings menu by default.

### Parameters

If you want to disable the Nextcloud help menu item you can use the **knowledgebaseenabled** parameter inside the `config/config.php`.

```
<?php

  "knowledgebaseenabled" => true,
```

**Note:** Disabling the help menu item might increase the number of support requests you have to answer in the future

## Language Configuration

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 use the **default_language** parameter.

Please keep in mind, that this will not effect a users language preference, which has been configured under "personal -> language" once he has logged in.

Please check Transifex language codes for the list of valid language codes.

### Parameters

```
<?php

  "default_language" => "en",
```

This parameters can be set in the `config/config.php`

## Logging Configuration

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.

### Parameters

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, or on the Admin page of your Nextcloud Web GUI.

#### ownCloud

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" => "owncloud",
"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",
"logfile" => "",
"loglevel" => "3",
```

## 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.

---

## 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 an 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, and the free Qualys SSL Labs Tests gives good guidance on whether your SSL server is correctly configured.

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.

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.

## Reverse Proxy Configuration

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 IP address 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`. The **overwritehost** parameter is used to set the hostname of the proxy. You can also specify a port. The **overwriteprotocol** parameter is used to set the protocol of the proxy. You can choose between the two options **http** and **https**. The **overwritewebroot** parameter is used to set the absolute web path of the proxy to the Nextcloud folder. When you want to keep the automatic detection of one of the three parameters you can leave the value empty or don't set it. The **overwritecondaddr** parameter is used to 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.

## 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.

# Using Third Party PHP Components

Nextcloud uses some third party PHP components to provide some of its functionality. These components are part of the software package and are contained in the **/3rdparty** folder.

## Managing Third Party Parameters

When using third party components, keep the following parameters in mind:

- **3rdpartyroot** – Specifies the location of the 3rd-party folder. To change the default location of this folder, you can use this parameter to define the absolute file system path to the folder location.
- **3rdpartyurl** – Specifies the http web path to the 3rdpartyroot folder, starting at the Nextcloud web root.

An example of what these parameters might look like is as follows:

```
<?php

"3rdpartyroot" => OC::$SERVERROOT."/3rdparty",
"3rdpartyurl"  => "/3rdparty",
```

# Automatic Configuration 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 `../nextcloud/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 `../nextcloud/config/autoconfig.php` is automatically removed after the initial configuration has been applied.

## 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 = array(
  "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 = array(
  "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" => "",
);
```

**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.

### 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.

# Server Tuning

## Using cron to perform background jobs

See Defining Background Jobs for a description and the benefits.

## 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 Configuring 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.

### 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.

### SSL / Encryption App

SSL (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.

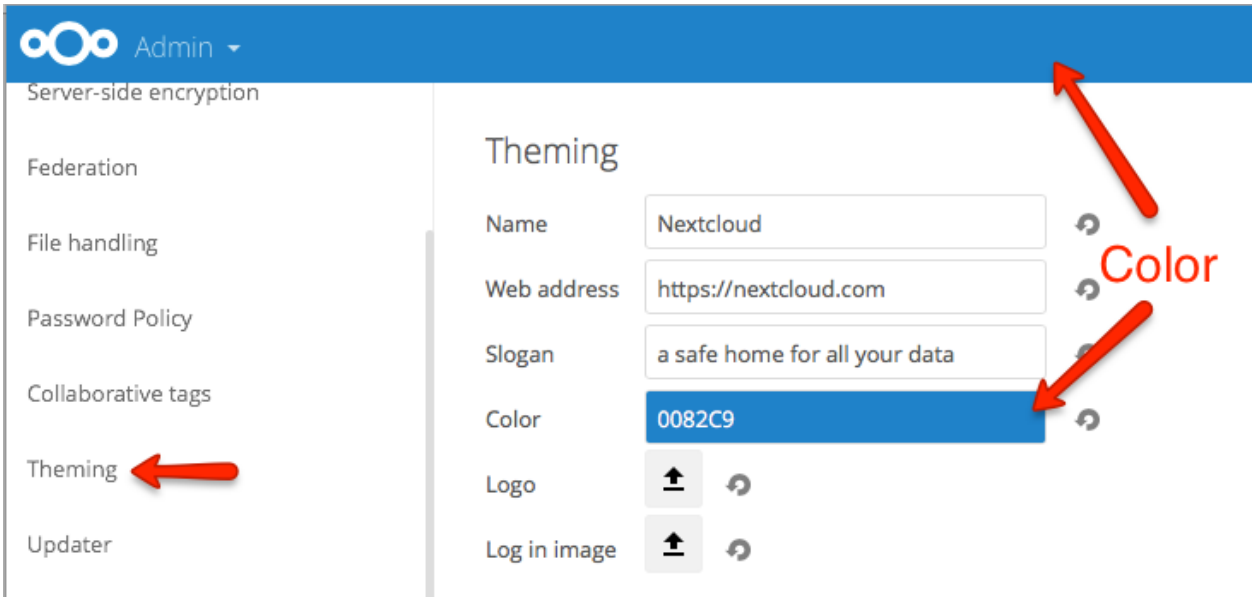
### Enable HTTP2 for faster loading

HTTP2 has huge speed improvements over HTTP with multiple request. Most browsers already support HTTP2 over SSL (HTTPS). So refer to your server manual for guides on how to use HTTP2.

## Theming

In the administrative settings you can modify the appearance of Nextcloud:

- Name
- Web Address
- 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.
- Log in image: The background image of the log in page




Log in page


## Theming of icons

Nextcloud will automatically generate favicons and home screen icons depending on the current app and theming color.

This requires the following additional dependencies:

- PHP module imagick
- SVG support for imagick (e.g. *libmagickcore5-extra*)

CHAPTER

FOUR

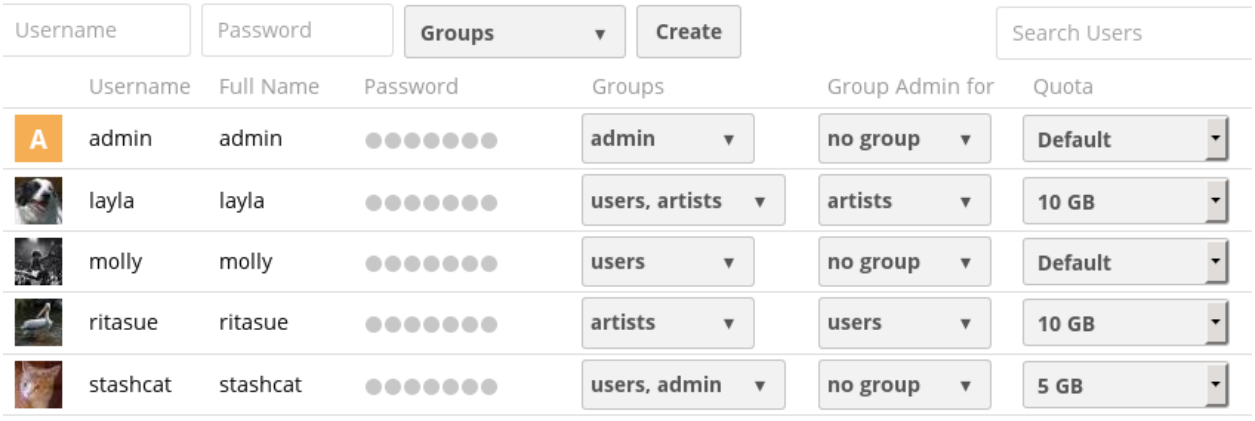
# USER MANAGEMENT

## User Management

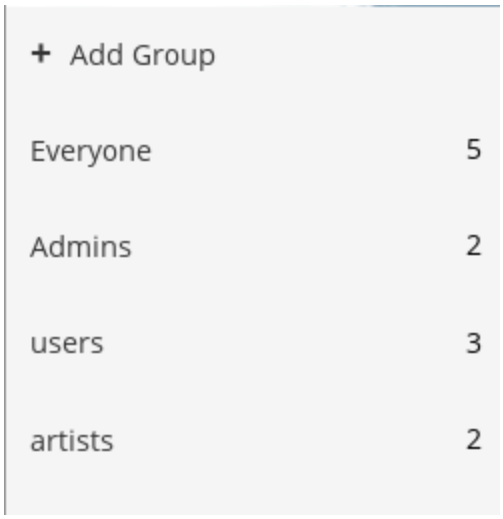
On the User management page of your Nextcloud Web UI you can:

- 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
- Delete them with a single click

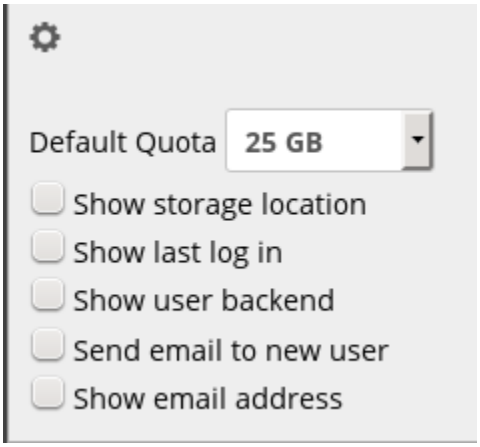
The default view displays basic information about your users.




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 an 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.

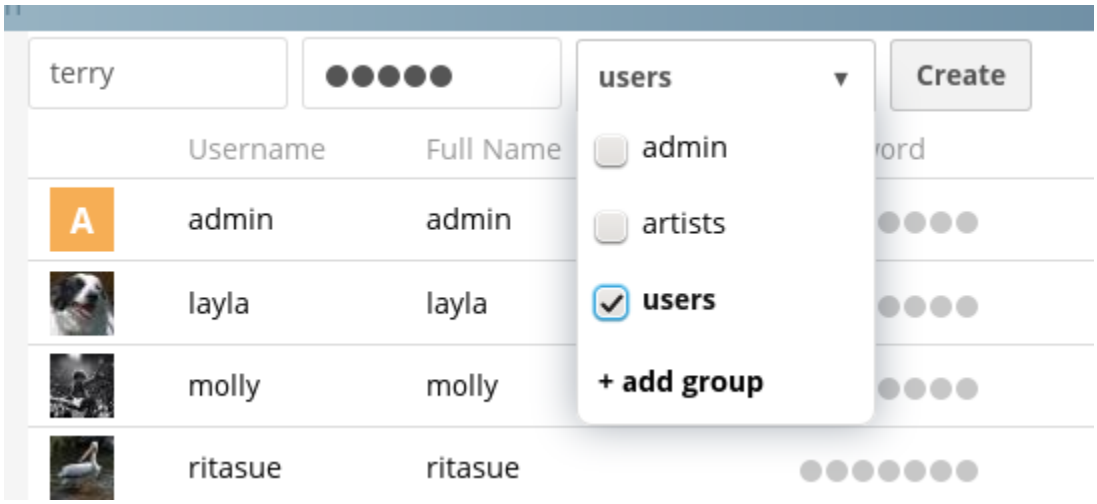
***Group Admin*** Group admins are granted administrative privileges on specific groups, and can add and remove users from their groups.

***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 (.) 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 Configuration).

## 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.

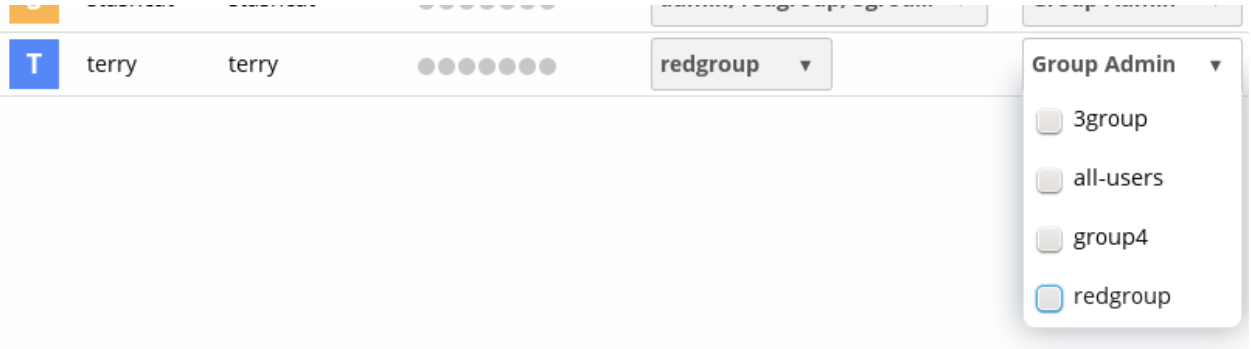
To set or change a user's display name:

- 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,
```

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.

### Deleting users

Deleting a user is easy: hover your cursor over their name on the **Users** page until a trashcan icon appears at the far right. Click the trashcan, and they're gone. You'll see an undo button at the top of the page, which remains until you refresh the page. 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. `occ` is in the `nextcloud` directory, for example `/var/www/nextcloud/occ`. `occ` has a command for resetting all user passwords, `user:resetpassword`. It is best to run `occ` as the HTTP user, as in this example on Ubuntu Linux:

```
$ 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.

You can find your HTTP user in your HTTP configuration file. These are the default Apache HTTP user:group on Linux distros:

- Centos, Red Hat, Fedora: apache:apache
- Debian, Ubuntu, Linux Mint: www-data:www-data
- openSUSE: wwwrun:www

See Using the occ Command to learn more about using the `occ` command.

## 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:

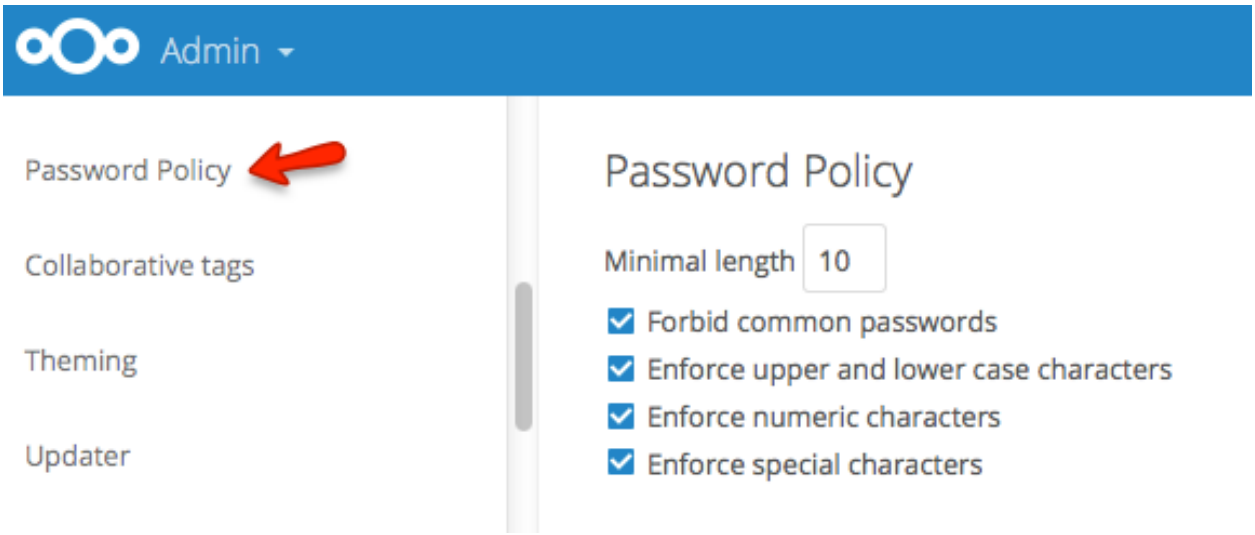
```
'lost_password_link' => 'https://example.org/link/to/password/reset',
```

## User Password Policy App

A password policy is a set of rules designed to enhance computer security by encouraging users to employ strong passwords and use them properly.

You can configure

- a minimal length of a password. Default is 10 characters.
- to forbid common passwords like 'california' or 'enterprise'.
- enforce upper and lower case characters
- Enforce numeric characters
- Enforce special characters like ! or :

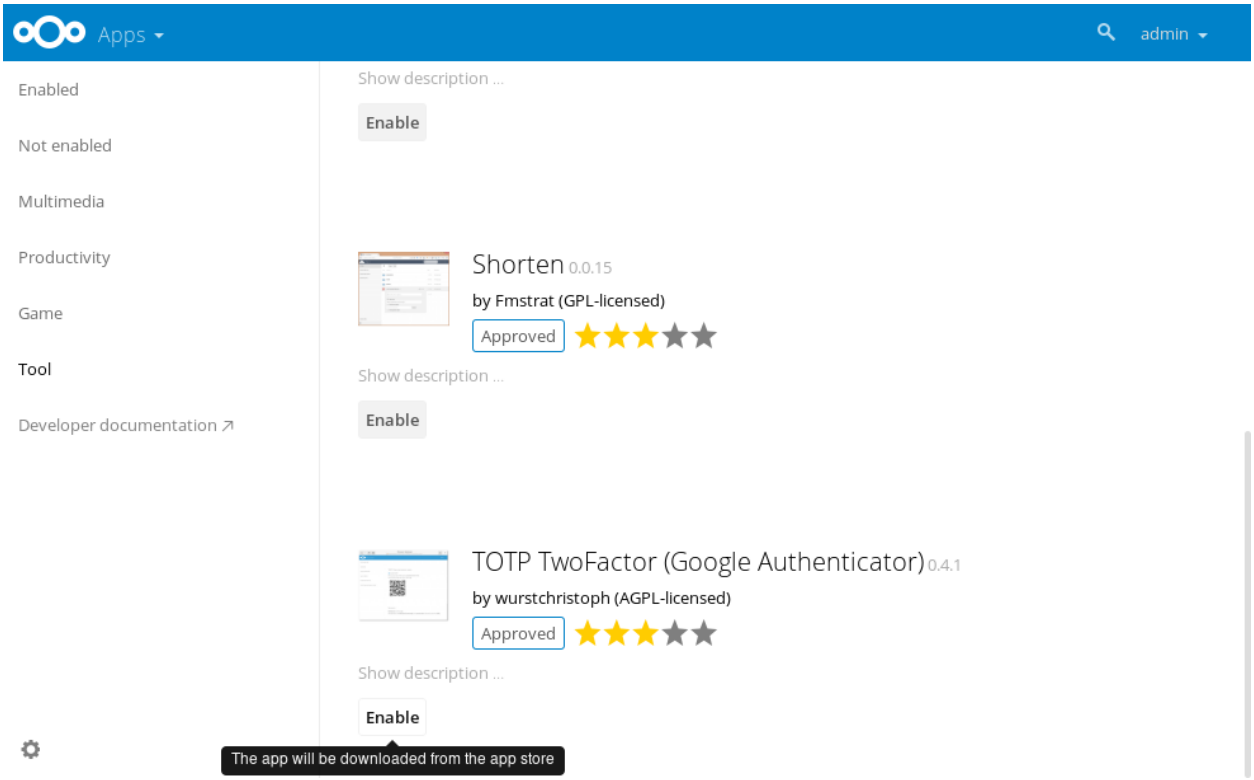



## Two Factor Authentication

Starting with Nextcloud 10, it is possible to use two factor authentication (2FA) with Nextcloud. It is a plugin based system requiring a 2FA app. Several 2FA apps are already available including TOTP, SMS 2-factor and U2F. Developers can built new two-factor provider apps. .. TODO ON RELEASE: Update version number above on release

### 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. .. TODO ON RELEASE: Update version number above on release

## User Authentication with IMAP, SMB, and FTP

You may configure additional user backends in Nextcloud's configuration `config/config.php` using the following syntax:

```
<?php

"user_backends" => array (
    0 => array (
            "class"     => ...,
            "arguments" => array (
                                0 => ...
                                ),
    ),
),
```

**Note:** A non-blocking or correctly configured SELinux setup is needed for these backends to work. Please refer to the *SELinux Configuration*.

Currently the "External user support" (user_external) app, which you need to enable first (See Installing and Managing Apps) provides the following user backends:

### IMAP

Provides authentication against IMAP servers

- **Class:** OC_User_IMAP
- **Arguments:** a mailbox string as defined in the PHP documentation
- **Dependency:** php-imap (See Manual Installation on Linux)
- **Example:**

```
<?php

"user_backends" => array (
    0 => array (
            "class"     => "OC_User_IMAP",
            "arguments" => array (
                                0 => '{imap.gmail.com:993/imap/ssl}'
                                ),
    ),
),
```

### SMB

Provides authentication against Samba servers

- **Class:** OC_User_SMB
- **Arguments:** the samba server to authenticate against
- **Dependency:** PHP smbclient module or smbclient (see SMB/CIFS)
- **Example:**

```
<?php

"user_backends" => array (
    0 => array (
            "class"     => "OC_User_SMB",
            "arguments" => array (
                                0 => 'localhost'
                                ),
    ),
),
```

### FTP

Provides authentication against FTP servers

- **Class:** OC_User_FTP
- **Arguments:** the FTP server to authenticate against
- **Dependency:** php-ftp (See Manual Installation on Linux)
- **Example:**

```
<?php

"user_backends" => array (
    0 => array (
            "class"     => "OC_User_FTP",
            "arguments" => array (
                               0 => 'localhost'
                               ),
    ),
),
```

## 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 `php5-ldap` on Debian/Ubuntu, and `php-ldap` on CentOS/Red Hat/Fedora. PHP 5.6+ is required in Nextcloud.

---

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)

**Warning:** The LDAP app is not compatible with the `User backend using remote HTTP servers` app. You cannot use both of them at the same time.

---

**Note:** A non-blocking or correctly configured SELinux setup is needed for the LDAP backend to work. Please refer to the *SELinux Configuration*.

---

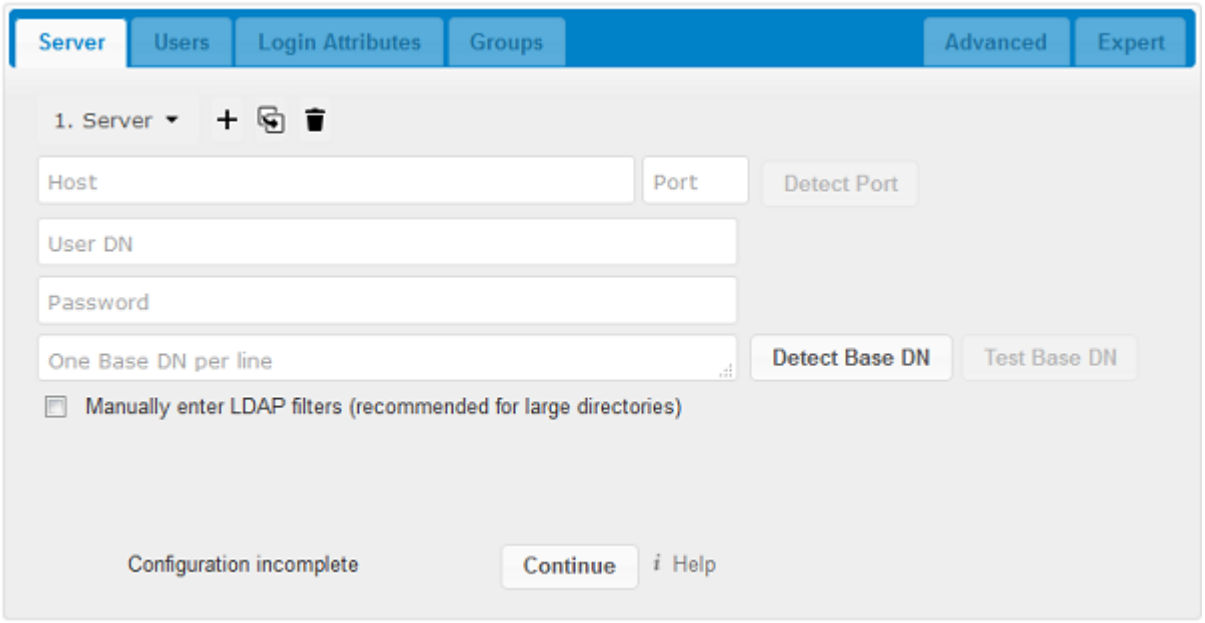
### Configuration

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. 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.

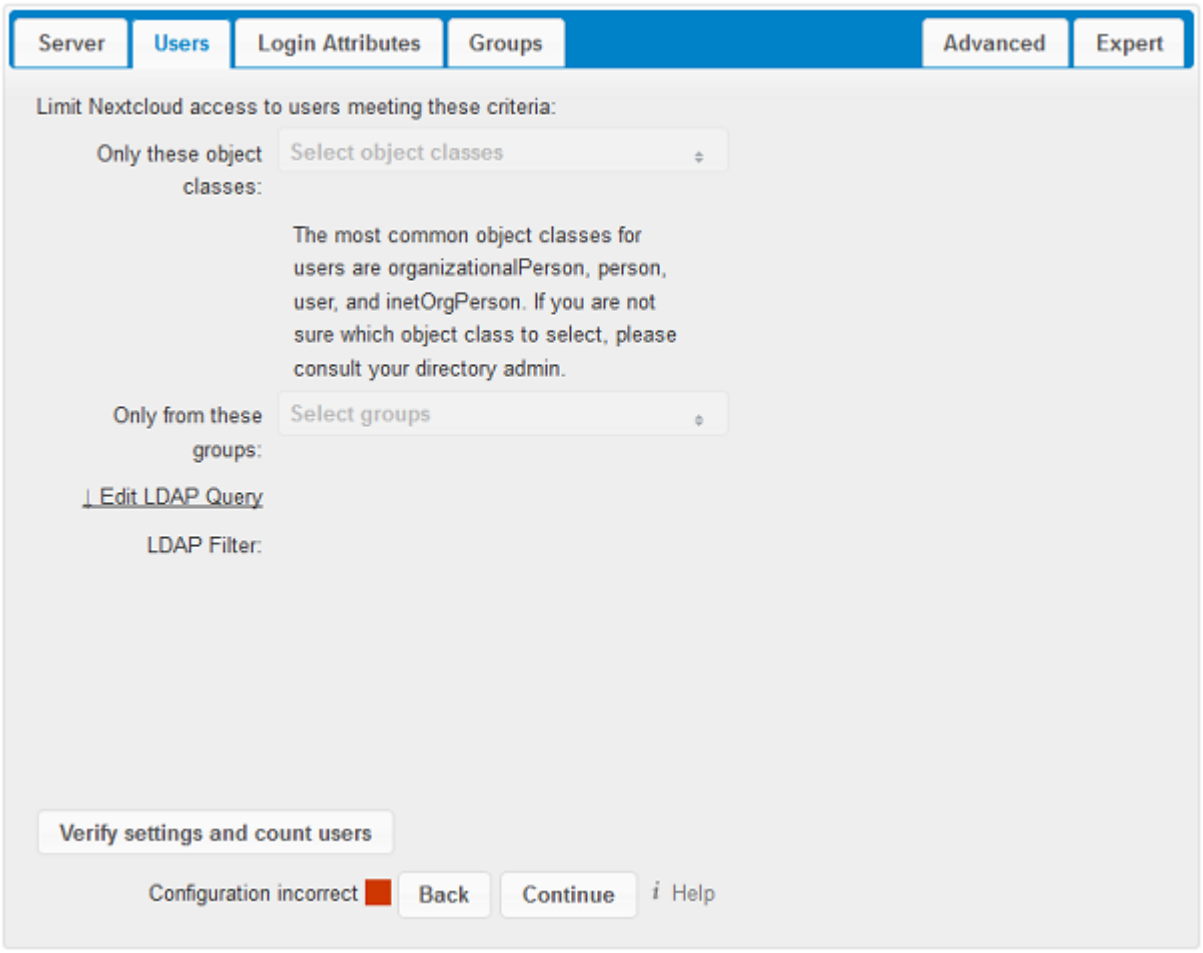
Nextcloud attempts to determine the Base DN according to the provided User DN or the provided Host, and you must enter it manually if Nextcloud does not detect it.

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:

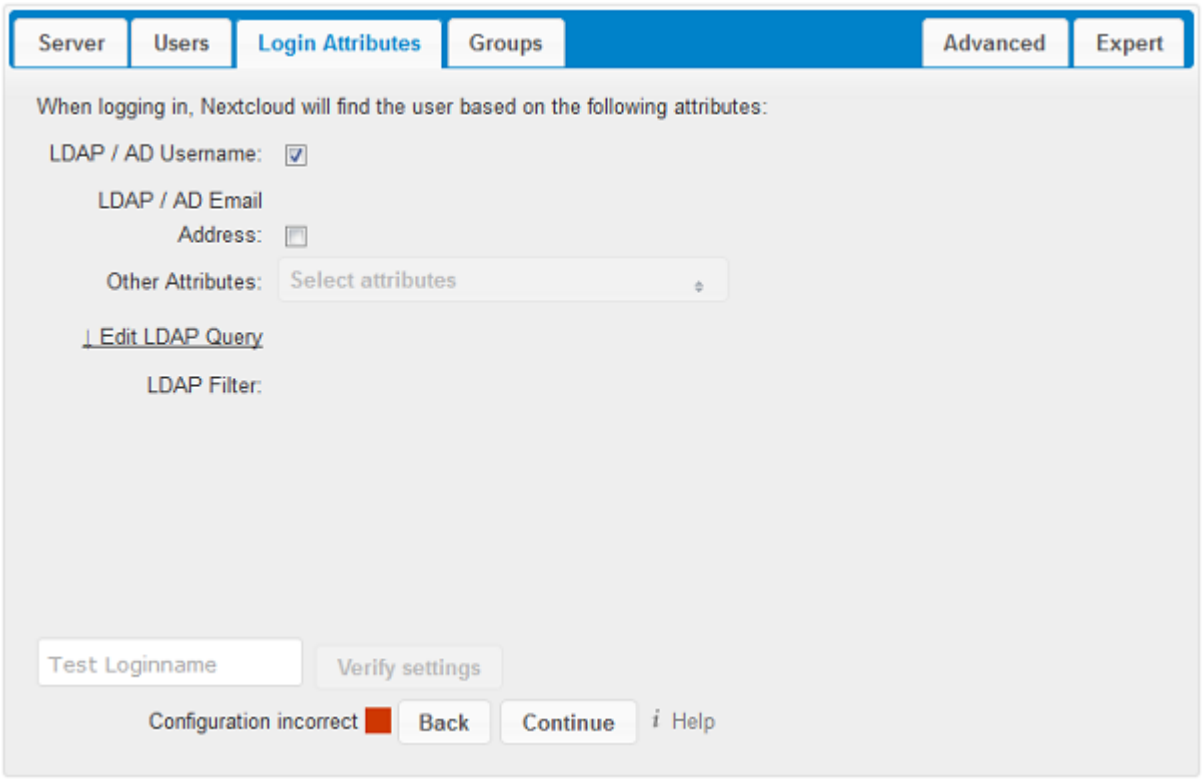
```
(&(objectClass=inetOrgPerson)(memberOf=cn=nextcloudusers,ou=groups,
dc=example,dc=com))
```

**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:

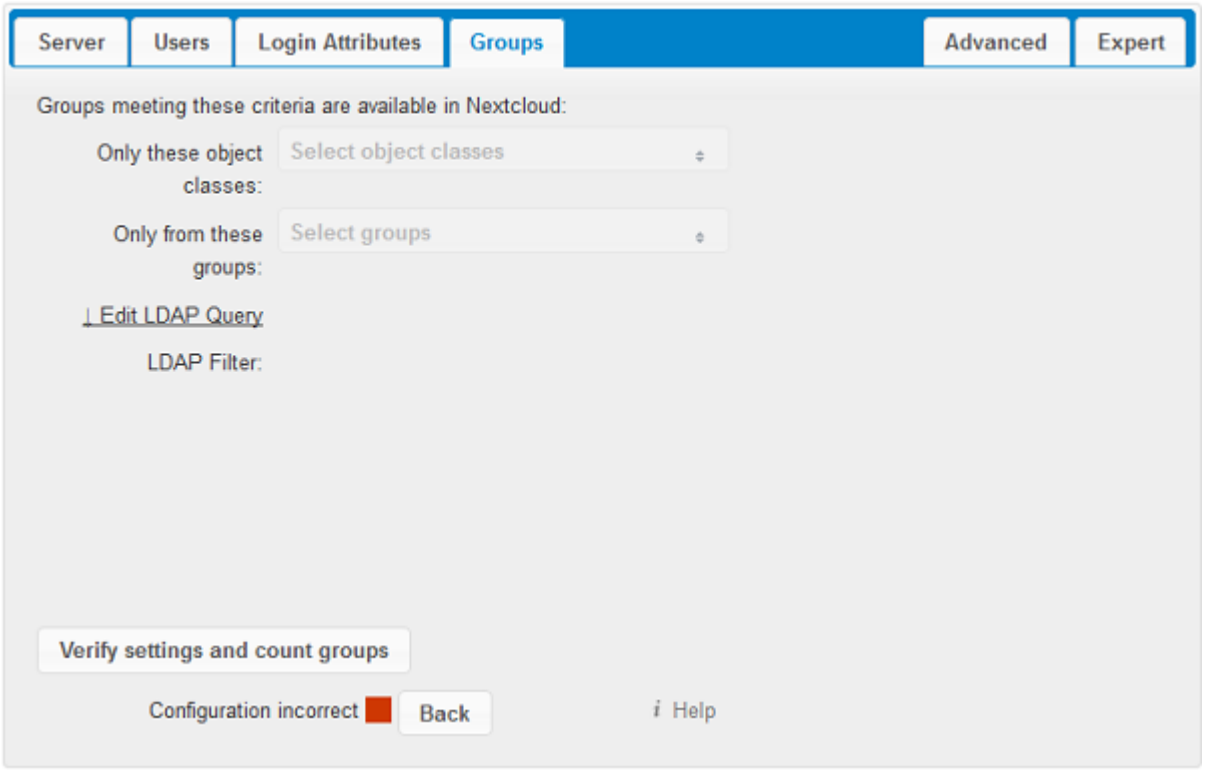
```
(&(objectClass=inetOrgPerson)(memberOf=cn=nextcloudusers,ou=groups,
dc=example,dc=com)(uid=%uid)
```

- 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.

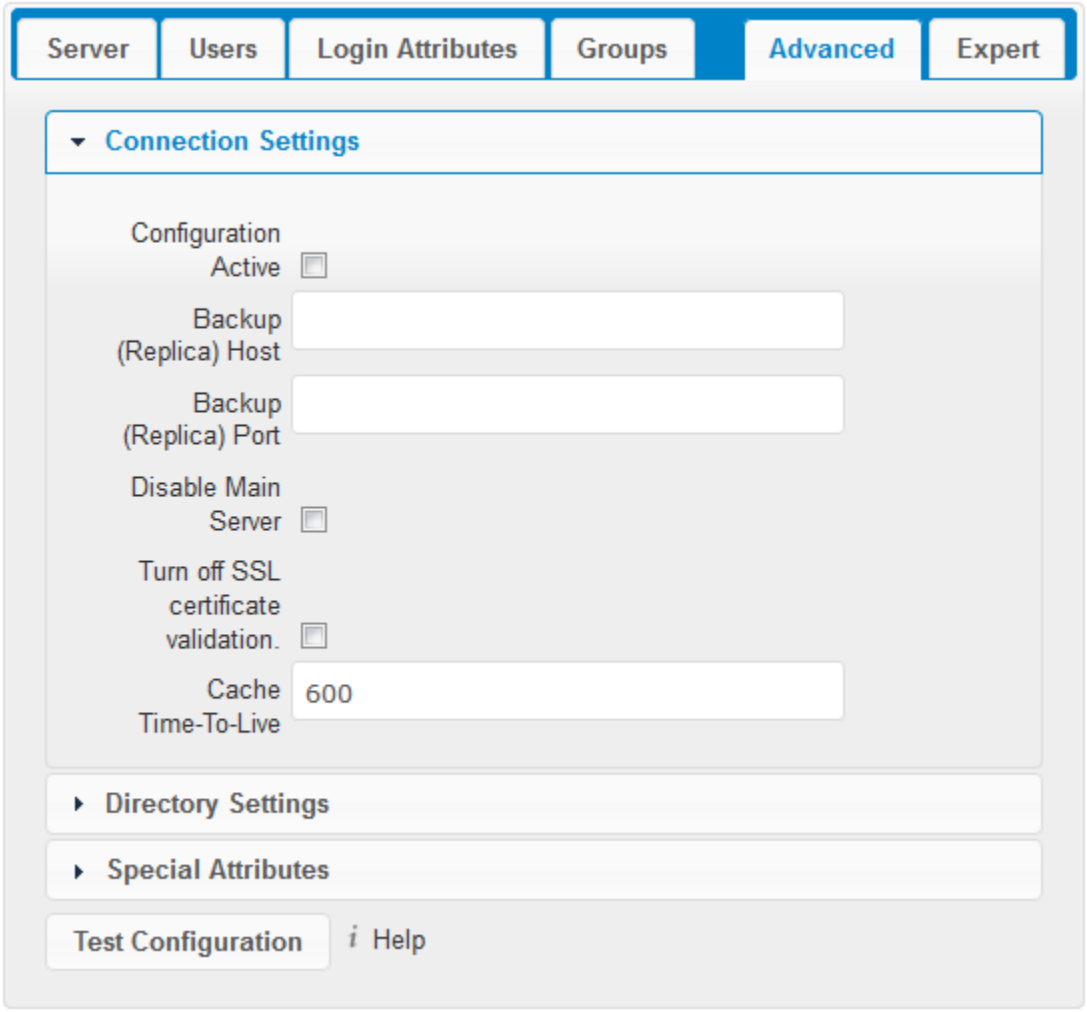
## Advanced Settings

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!

**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.

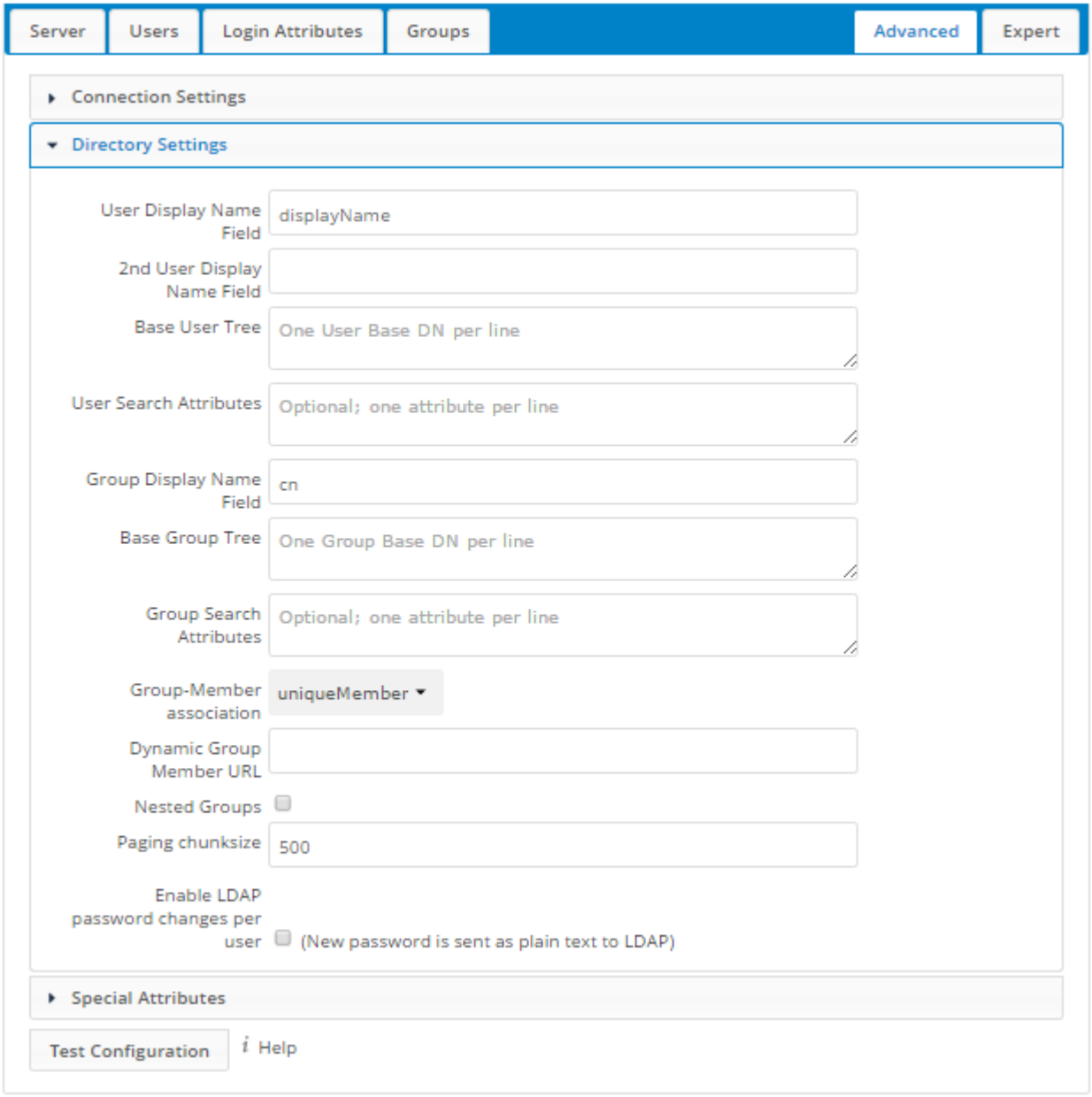
Note that almost every PHP request requires a new connection to the LDAP server. If you require fresh PHP requests we recommend defining a minimum lifetime of 15s or so, rather than completely eliminating the cache.

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.

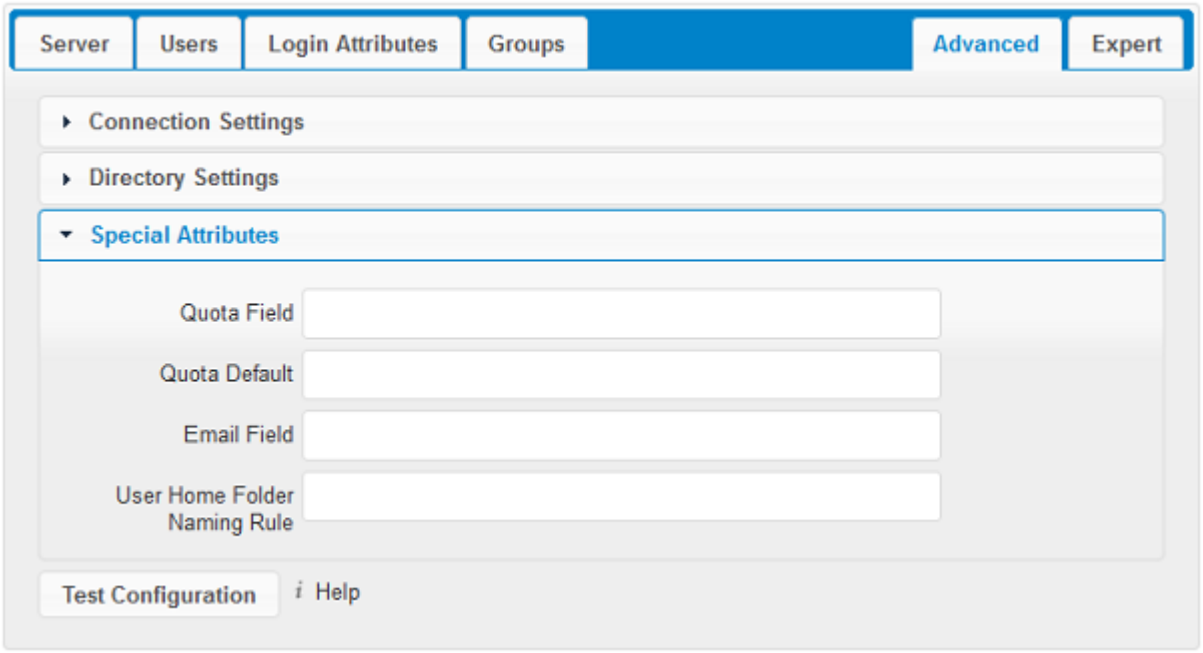
- Example: *uniquemember*

**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.
- 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.

### 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.

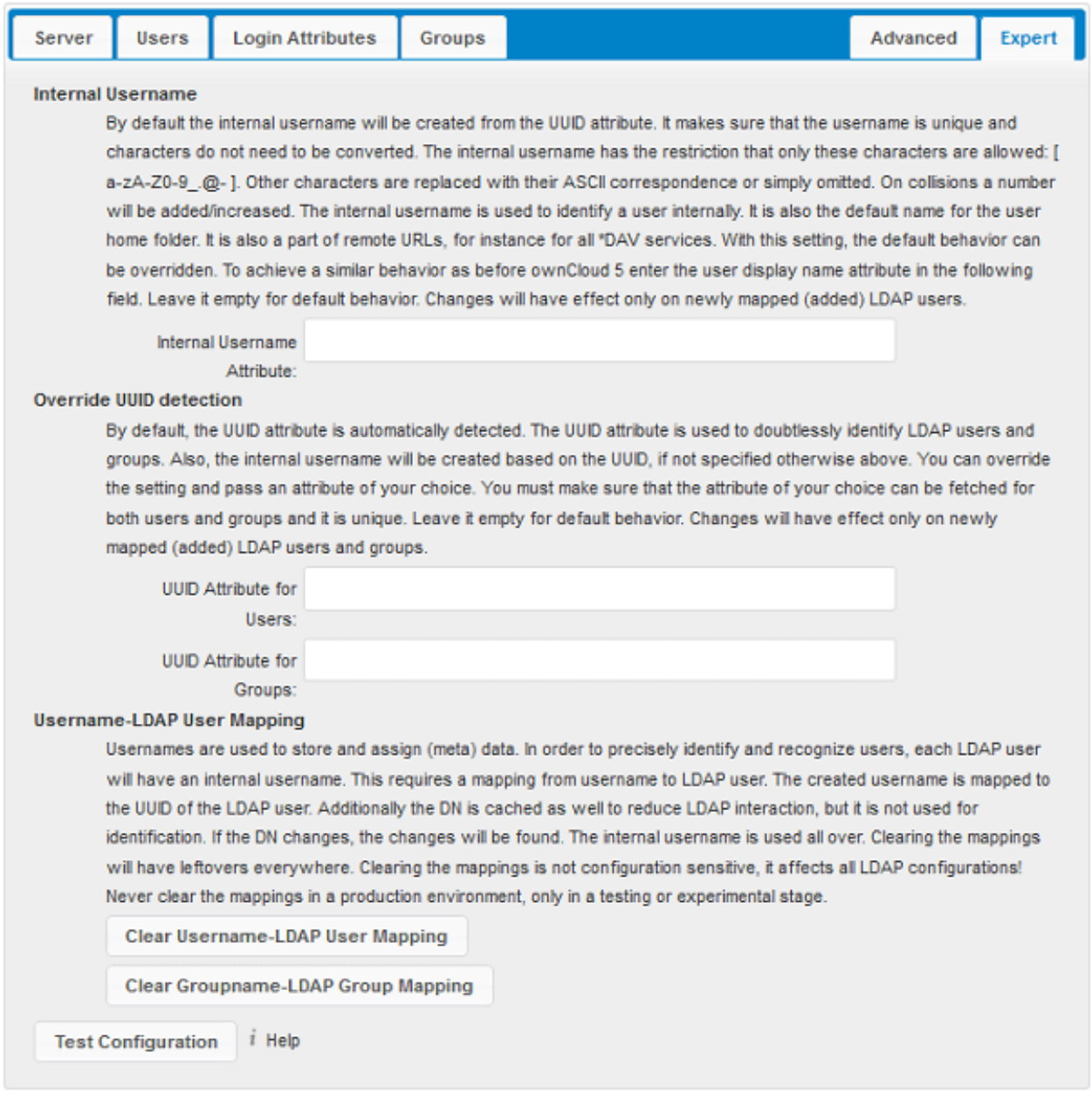
- Example: *cn*

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 behaviour. Changes will affect only newly mapped LDAP users.

- 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 behaviour. 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!

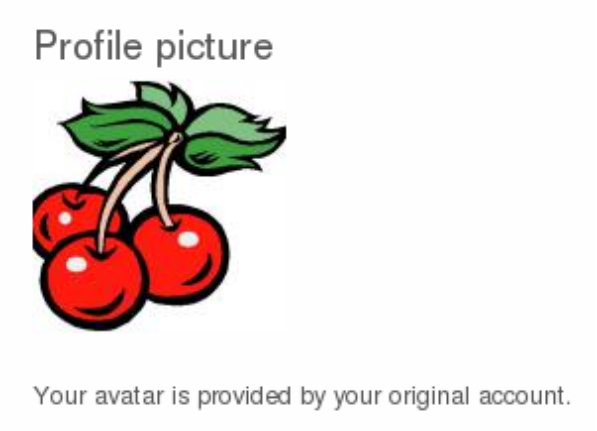
## Testing the configuration

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.

## 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.

## 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.

### 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.

#### "Sizelimit exceeded" message in logs

> ldap_search(): Partial search results returned: Sizelimit exceeded at apps/user_ldap/lib/LDAP.php#256

This error message means one of the following:

1. Pagination of the results is used for communicating with the LDAP server (pagination is by default enabled in OpenLDAP and AD), but there are more results to return than what the pagination limit is set to. If there are no users missing in you setup then you can ignore this error message for now.
2. No pagination is used and this indicates that there are more results on the LDAP server than what is returned. You should then enabled pagination on your LDAP server to import all available users.

## 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.

As of Nextcloud 5 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.

### Caching

The LDAP information is cached in Nextcloud memory cache, and you must install and configure the memory cache (see Configuring 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 an oC-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.

## 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.

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=com|
| aaliyah_kunze   | aaliyah kunze   | aaliyah_kunze    | uid=aaliyah_kunze,ou=people,dc=com   |
+-----------------+-----------------+------------------+-----------------------------------+
```

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.

## LDAP Configuration API

Any used method requires the a header "OCS-APIREQUEST" set to "true". And any method takes an optional "format" parameter, which accepts "xml" (default) or "json".

### Methods

#### 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

- POST `https://admin:secret@example.com/ocs/v2.php/apps/user_ldap/api/v1/config` -H "OCS-APIREQUEST: true"
- 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

- DELETE `https://admin:secret@example.com/ocs/v2.php/apps/user_ldap/api/v1/config/s02 -H "OCS-APIREQUEST: true"`
- 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

- GET `https://admin:secret@example.com/ocs/v2.php/apps/user_ldap/api/v1/config/s02?showPa -H "OCS-APIREQUEST: true"`
- 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

- PUT `https://admin:secret@example.com/ocs/v2.php/apps/user_ldap/api/v1/config/s01 -H "OCS-APIREQUEST: true" -d "configData[ldapHost]=ldap%3A%2F%2Fldap.server.tld &configData[ldapPort]=389"`
- fetches 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 |
| ldapGidNumber | rw | no | group ID attribute, needed for primary groups on OpenLDAP (and con |
| 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, u |
| 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 |
| 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 |
| ldapExpertUUIDUserAttr | rw | no | override the LDAP servers UUID attribute to identify LDAP user recor |
| ldapExpertUUIDGroupAttr | rw | no | override the LDAP servers UUID attribute to identify LDAP group rec |
| lastJpegPhotoLookup | r | no | set in runtime |
| ldapNestedGroups | rw | no | Whether LDAP supports nested groups |

Table 4.1 – continued from previous page

| Key | Mode | Required | Description |
|---|---|---|---|
| ldapPagingSize | rw | no | Number of results to return per page |
| turnOnPasswordChange | rw | no | Whether users are allowed to change passwords (hashing must happen |
| 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 **nextcloud_base_url/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, other require to set the header explicitly)

## Instruction Set For Users

### users / adduser

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 required password for the new user

Status codes:

- 100 - successful
- 101 - invalid input data
- 102 - username already exists
- 103 - unknown error occurred whilst adding the user

#### Example

- POST `http://admin:secret@example.com/ocs/v1.php/cloud/users -d userid="Frank" -d password="frankspassword"`
- Creates the user `Frank` with password `frankspassword`

#### XML Output

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>100</statuscode>
  <message/>
 </meta>
 <data/>
</ocs>
```

### users / getusers

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

- 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>
```

### users / getuser

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

- 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>
    <email>frank@example.org</email>
    <quota>0</quota>
    <enabled>true</enabled>
  </data>
</ocs>
```

### users / edituser

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/users/{userid}**

- HTTP method: PUT
- PUT argument: key, the field to edit (email, quota, display, password)
- PUT argument: value, the new value for the field

Status codes:

- 100 - successful
- 101 - user not found
- 102 - invalid input data

#### Examples

- PUT `http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank -d key="email" -d value="franksnewemail@example.org"`
- Updates the email address for the user `Frank`

- PUT `http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank -d key="quota" -d value="100MB"`
- Updates the quota for the user `Frank`

#### XML Output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>100</statuscode>
    <status>ok</status>
  </meta>
  <data/>
</ocs>
```

### users / disableuser

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

- PUT `http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/disable -d key="disable" -d value="true"`
- Disables the user `Frank`

#### XML Output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <status>ok</status>
    <statuscode>100</statuscode>
    <message/>
  </meta>
  <data/>
</ocs>
```

### users / enableuser

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

- PUT `http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/enable -d key="enable" -d value="true"`
- Enables the user `Frank`

#### XML Output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <status>ok</status>
    <statuscode>100</statuscode>
    <message/>
  </meta>
  <data/>
</ocs>
```

### users / deleteuser

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

- 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>
```

### users / getgroups

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

- 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>
```

### users / addtogroup

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

- POST `http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/groups -d groupid="newgroup"`
- 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>
```

### users / removefromgroup

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

- DELETE `http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/groups -d groupid="newgroup"`
- 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>
```

### users / createsubadmin

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

- POST `https://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/subadmins -d groupid="group"`
- 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>
```

### users / removesubadmin

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

- DELETE `https://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/subadmins -d groupid="oldgroup"`
- 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>
```

### users / getsubadmingroups

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

- 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>
```

## Instruction Set For Groups

### groups / getgroups

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

- 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>
```

### groups / addgroup

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

- 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>
```

### groups / getgroup

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

- POST `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>
```

### groups / getsubadmins

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

- 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>
```

### groups / deletegroup

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

- 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

### apps / getapps

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

- 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>
```

### apps / getappinfo

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

- 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>
```

### apps / enable

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

- 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>
```

### apps / disable

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

- 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>
```

CHAPTER

FIVE

# FILE SHARING AND MANAGEMENT

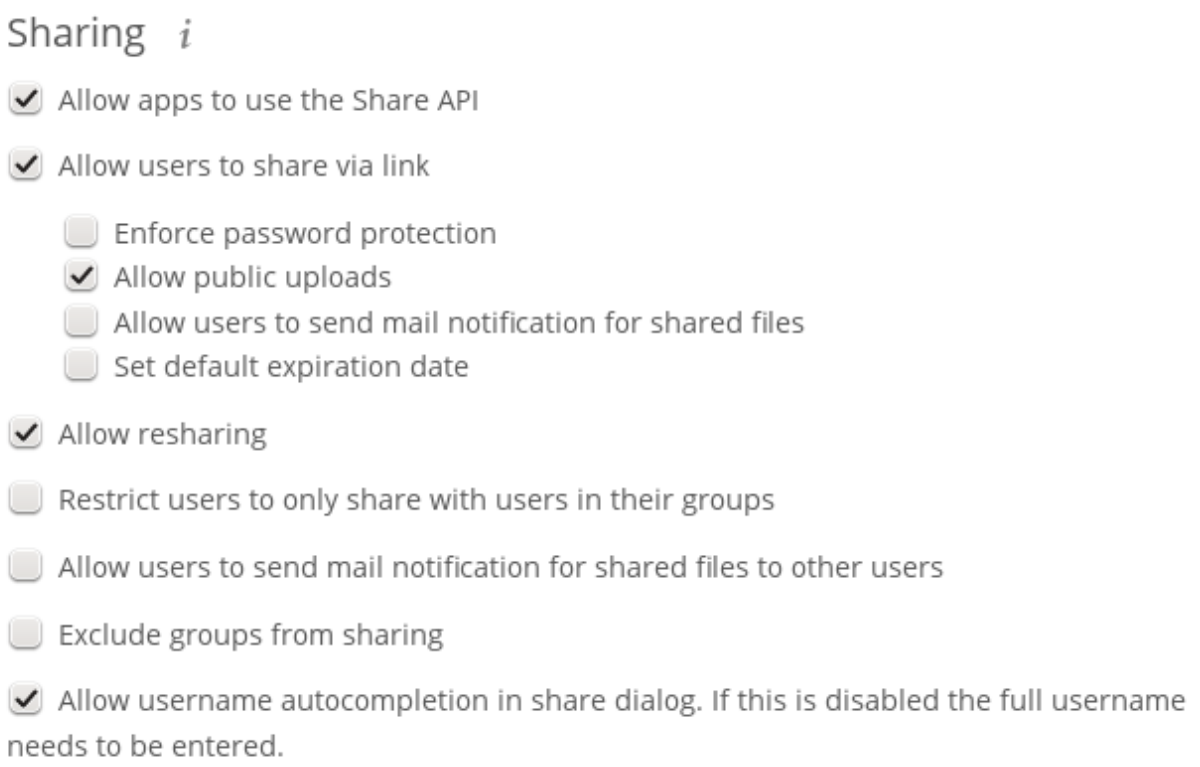
## File Sharing

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:

- Allow users to share files
- Allow users to create public shares
- Require a password on public shares
- Allow public uploads to public shares
- Require an expiration date on public share links
- Allow resharing
- Restrict sharing to group members only
- Allow email notifications of new public shares
- Exclude groups from creating shares

**Note:** Nextcloud includes a Share Link Password Policy app; see *Share Link Password Policy*.

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 `Allow users to share via link` to enable creating public shares for people who are not Nextcloud users via hyperlink.
- 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 `Allow public uploads` to allow anyone to upload files to public shares.
- Check `Allow users to send mail notification for shared files` to enable sending notifications from Nextcloud. (Your Nextcloud server must be configured to send mail)
- Check `Set default expiration date` to set a default expiration date on public shares.
- Check `Allow resharing` to enable users to re-share files shared with them.
- 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 `Allow users to send mail notification for shared files` enables users to send an email notification to every Nextcloud user that the file is shared with.
- 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.

---

**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.

---

## 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.)

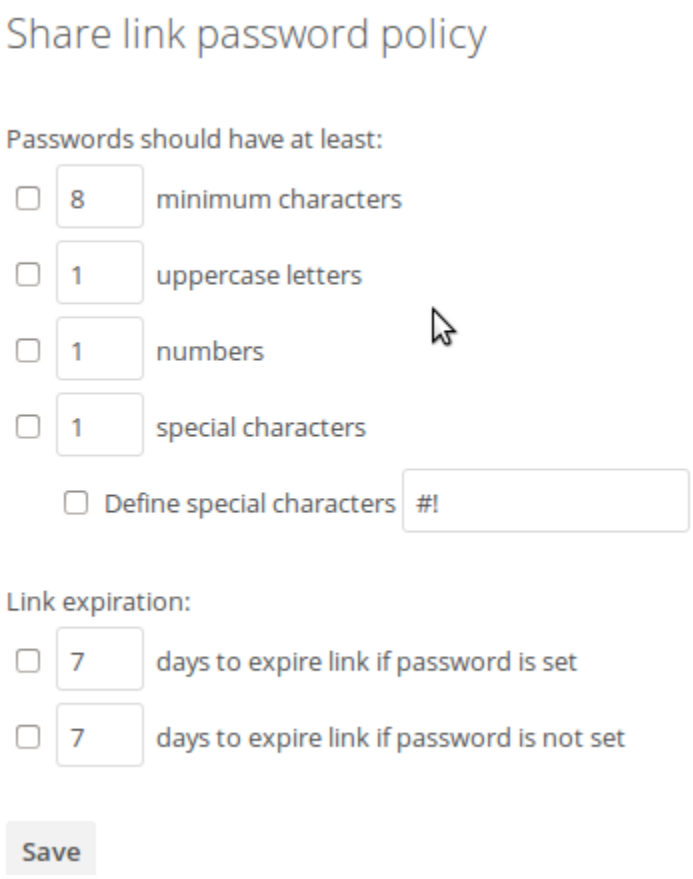
## Creating Persistent File Shares

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.

## Share Link Password Policy

Nextcloud users have the option of enabling the Share Link Password Policy app. This allows you to enforce password length, required characters, define special characters, and expiration dates on share links.




Note that you cannot use Emojis as special characters with MySQL, as it supports UTF8 characters only of 1-3 bytes, and emojis require 4 bytes.

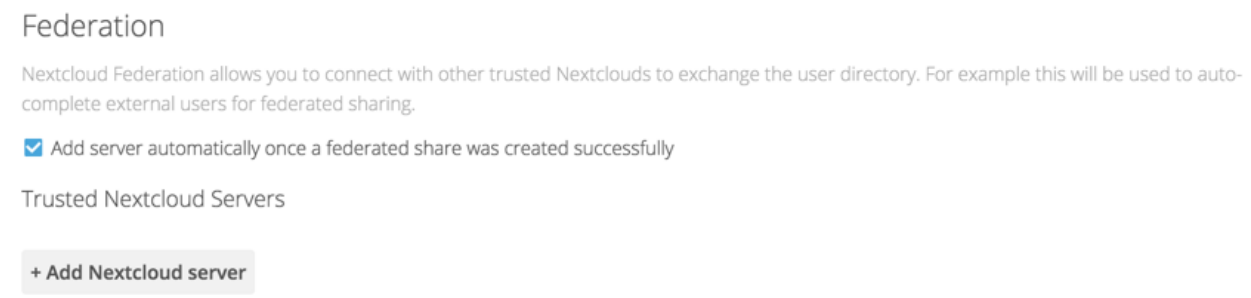
## Configuring Federation Sharing

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.

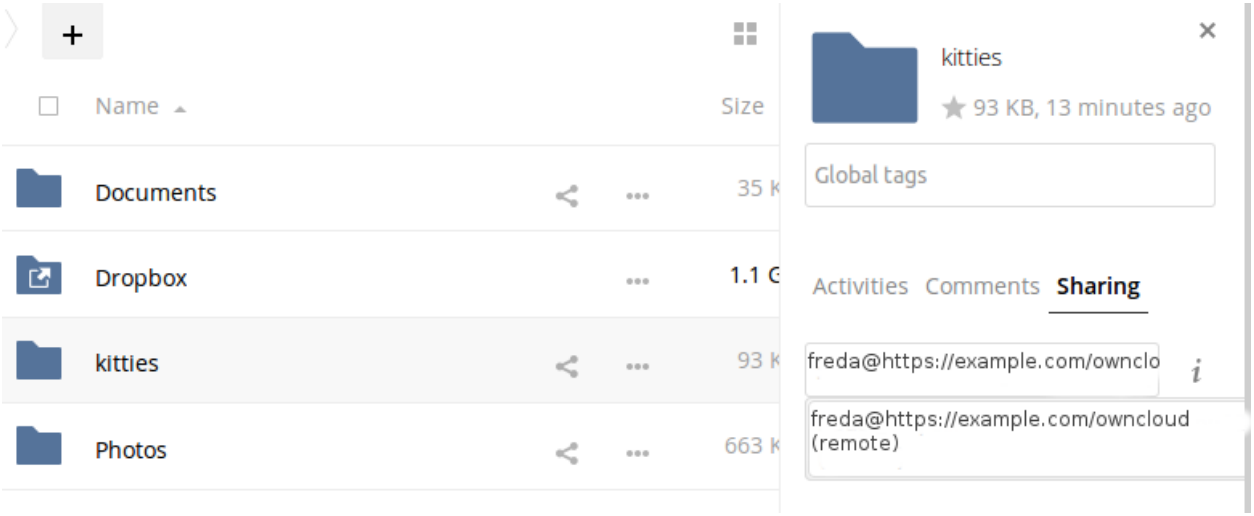
### Creating a new Federation Share

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.

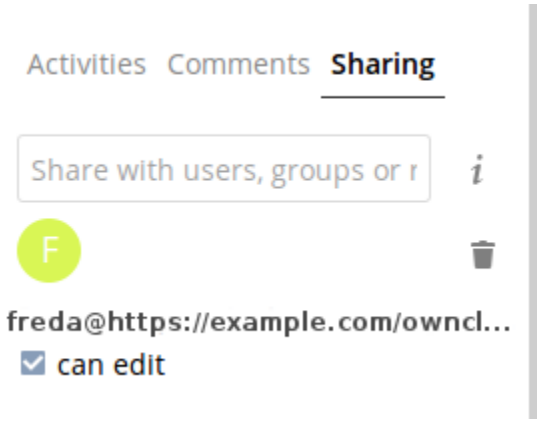
1. Enable the Federation app.
2. Go to your Nextcloud Admin page and scroll to the Sharing section. Verify that **Allow users on this server to send shares to other servers** and **Allow users on this server to receive shares from other servers** are enabled.
3. Now go to the Federation section. By default, **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**.

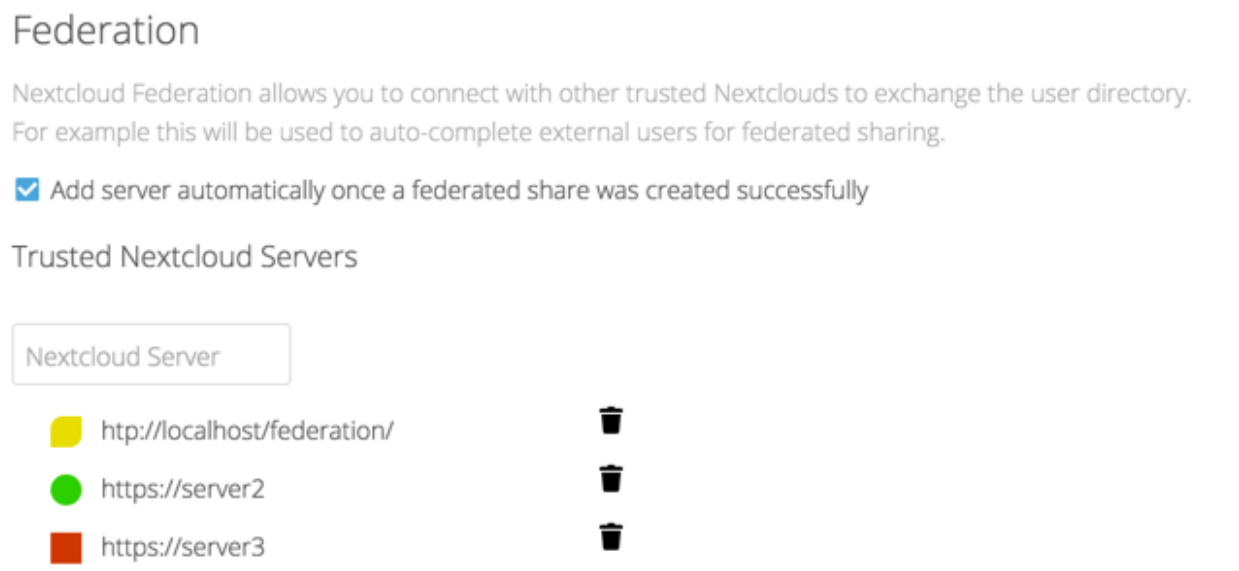



You may disconnect the share at any time by clicking the trash can icon.

## Configuring Trusted Nextcloud Servers

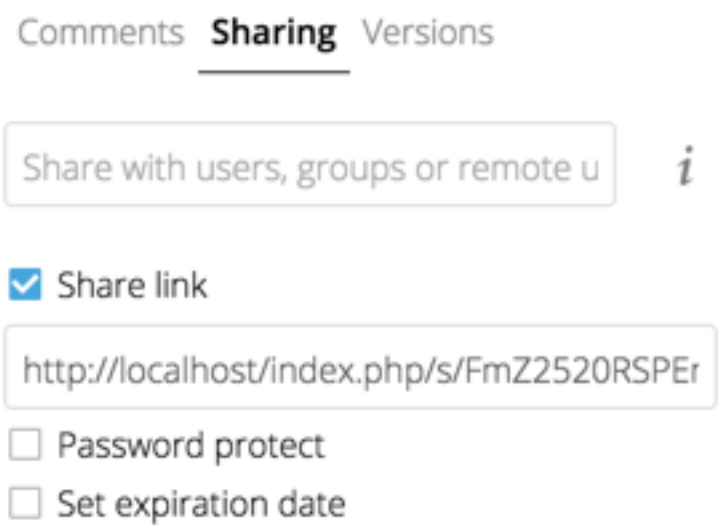
You may create a list of trusted Nextcloud servers for Federation sharing. This allows your linked Nextcloud servers to share user directories, and to auto-fill user names in share dialogs. If **Add server automatically once a federated share was created successfully** is enabled on your Admin page, 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.

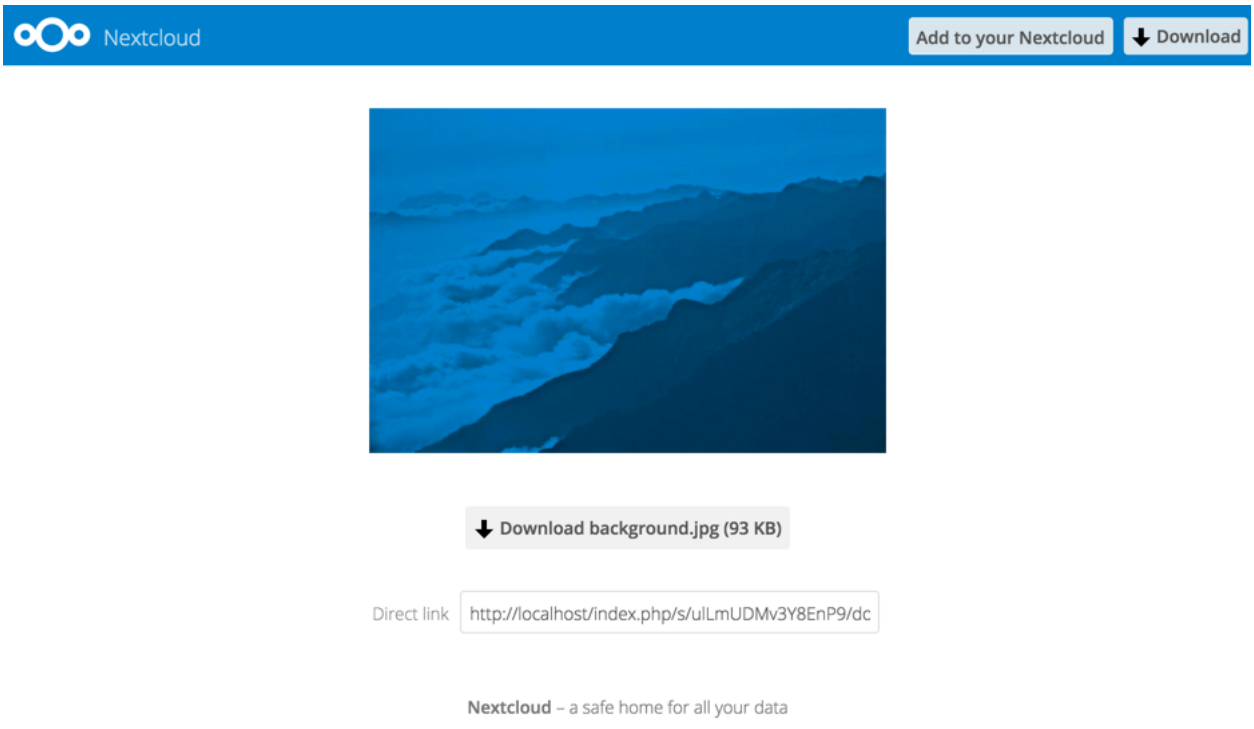



## Creating Federation Shares via Public Link Share

Check the `Share Link` checkbox 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.

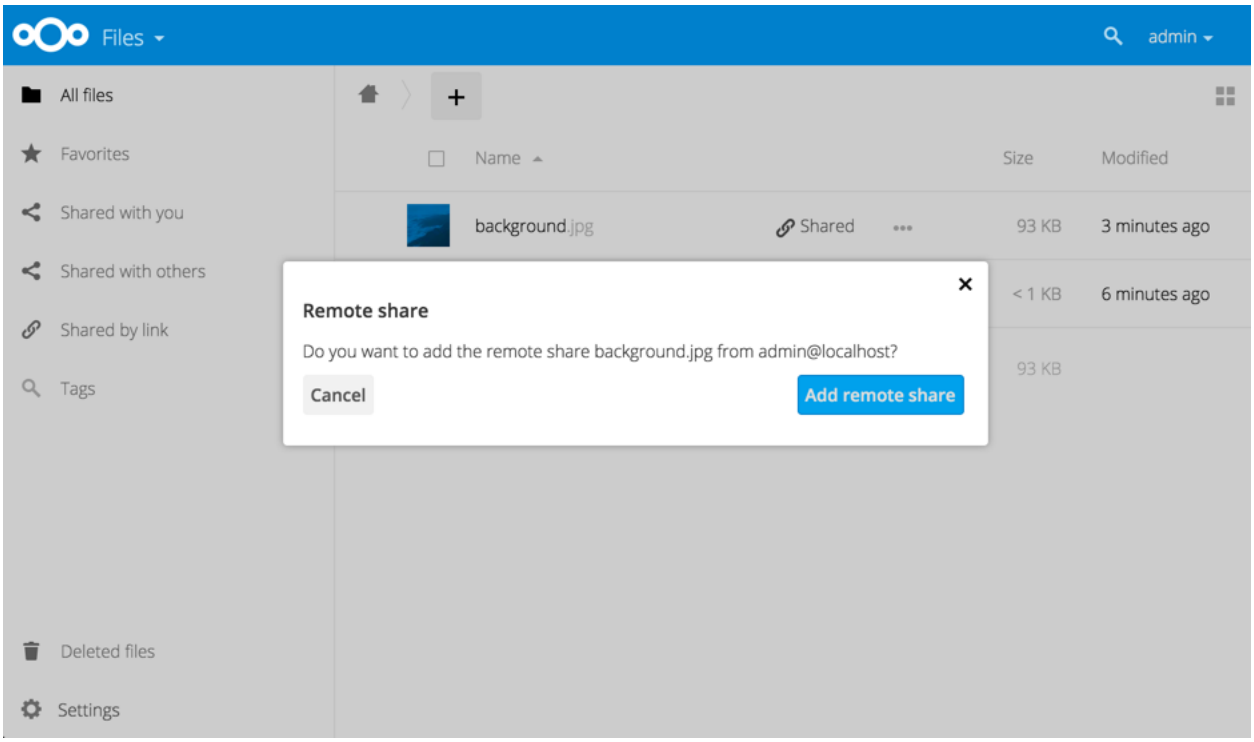



You may optionally set a password and expiration date on it. When your recipient receives your email they must click the link, or copy it to a Web browser. They will see a page displaying a thumbnail of the file, with a button to **Add to your Nextcloud**.




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 **Add remote share** 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.

### System Configuration

- Make sure that the latest version of PHP (at least 5.6.6) 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
```

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

### 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.

---

### nginx

- client_max_body_size
- fastcgi_read_timeout
- client_body_temp_path

Since nginx 1.7.11 a new config option **'fastcgi_request_buffering <https://nginx.org/en/docs/http/ngx_http_fastcgi_module.html#f** is availabe. 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.

For more info how to configure nginx to raise the upload limits see also this wiki entry.

---

**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`

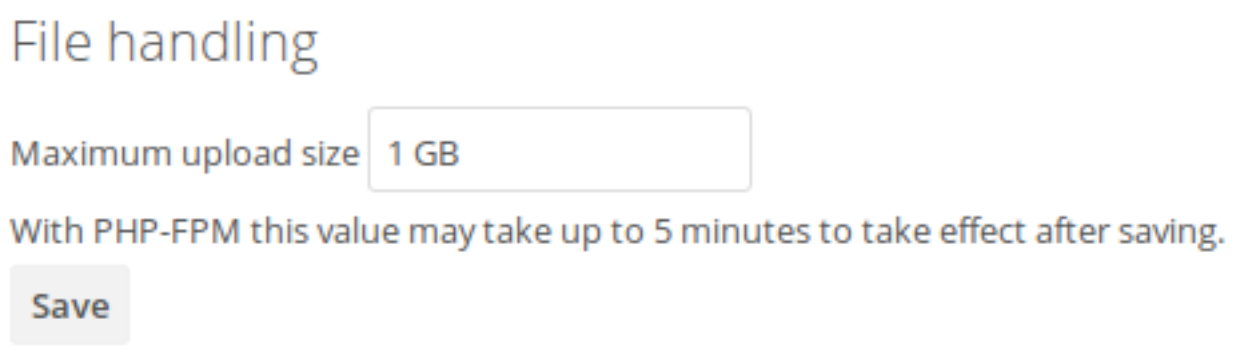
## Configuring Nextcloud

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 Config.php Parameters).

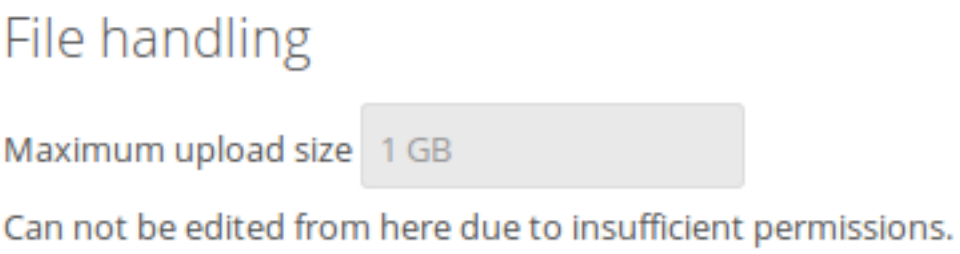
If you have configured the `session_lifetime` setting in your `config.php` (See Config.php Parameters) file then make sure it is not too low. This setting needs to be configured to at least the time (in seconds) that the longest upload will take. If unsure remove this completely from your configuration to reset it to the default shown in the `config.sample.php`.

### Configuring upload limits within the GUI

If all prerequisites described in this documentation are in place an admin can change the upload limits on demand by using the `File handling` input box within the administrative backend of Nextcloud.



Depending on your environment you might get an insufficient permissions message shown for this input box.



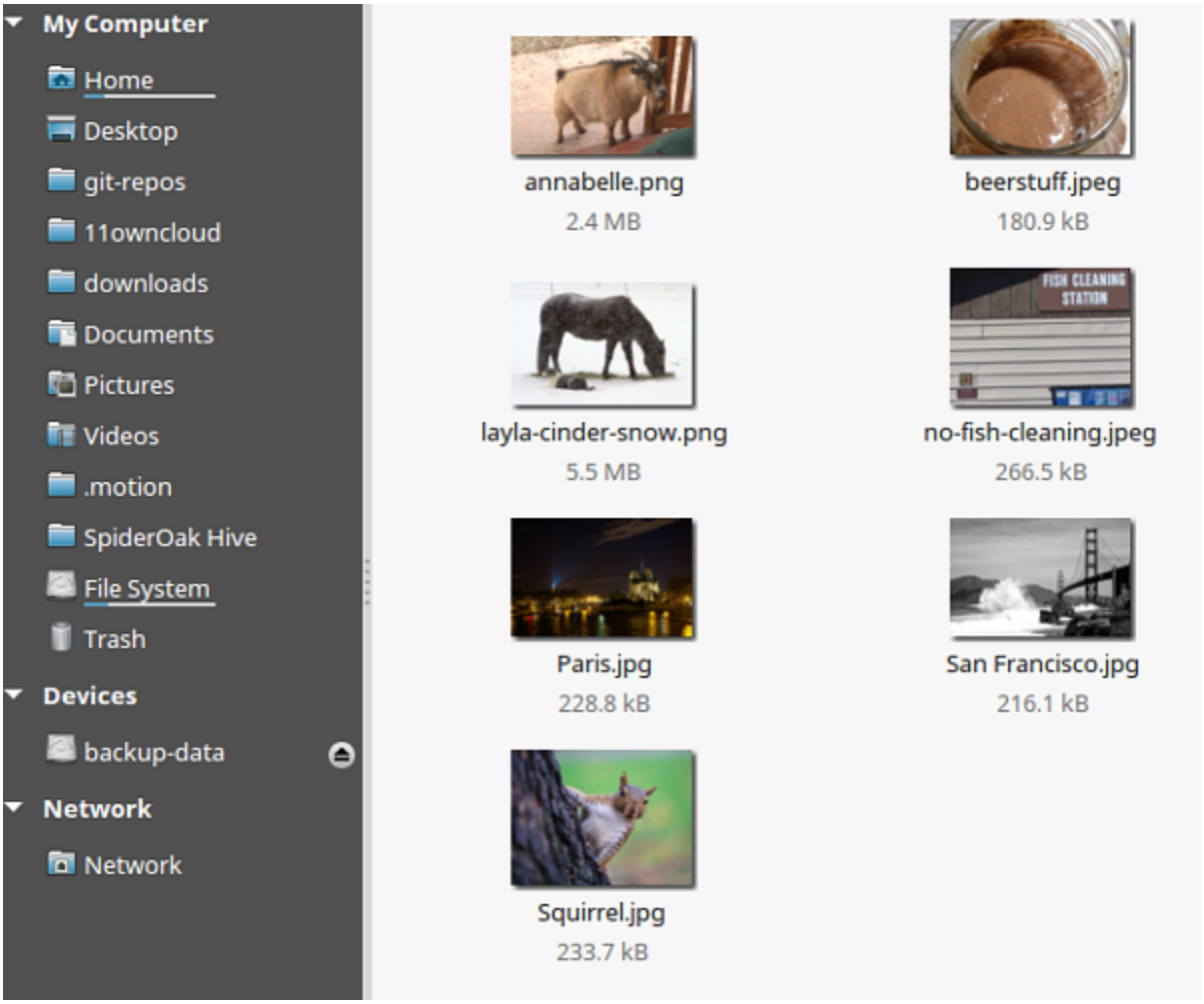

To be able to use this input box you need to make sure that:

- your Web server is able to use the `.htaccess` file shipped by Nextcloud (Apache only)
- the user your Web server is running as has write permissions to the files `.htaccess` and `.user.ini`

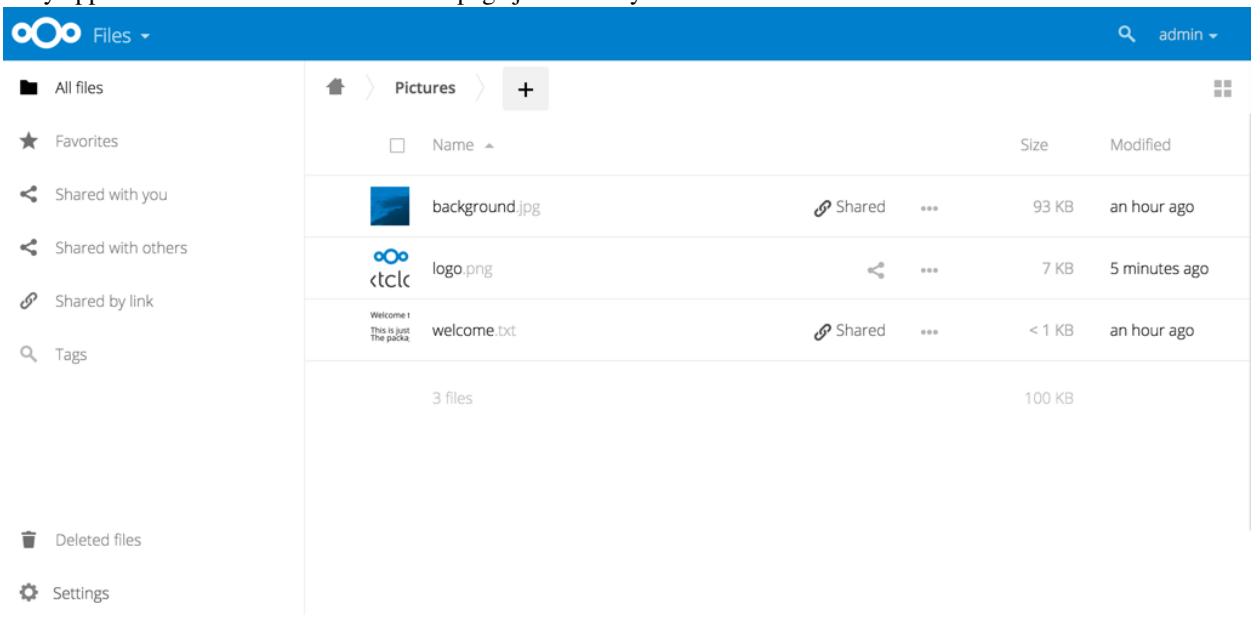
## Providing Default Files

You may distribute a set of default files and folders to all users by placing them in the `nextcloud/core/skeleton` directory on your Nextcloud server. These files appear 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.



### Additional Configuration

The configuration option `skeletondirectory` available in your `config.php` (See Config.php Parameters) allows you to configure 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.

## 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.

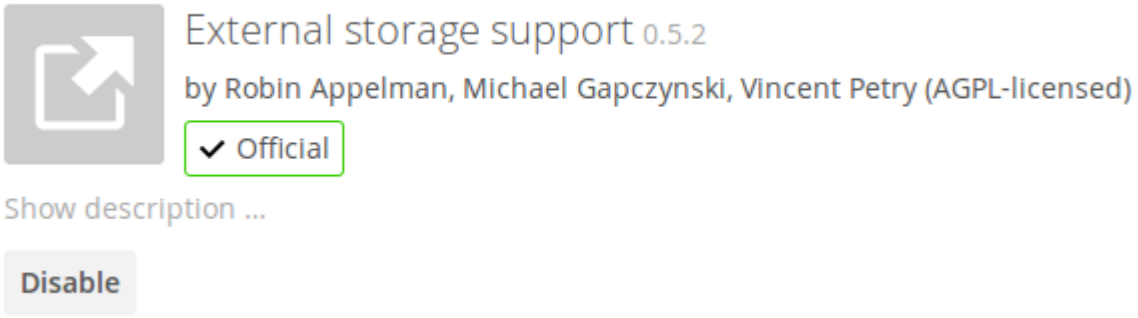
Nextcloud 9.0 introduces a new set of *occ commands for managing external storage*.

Also new in 9.0 is an option for the Nextcloud admin to enable or disable sharing on individual external mountpoints (see *Mount Options*). Sharing on such mountpoints is disabled by default.

### Enabling External Storage Support

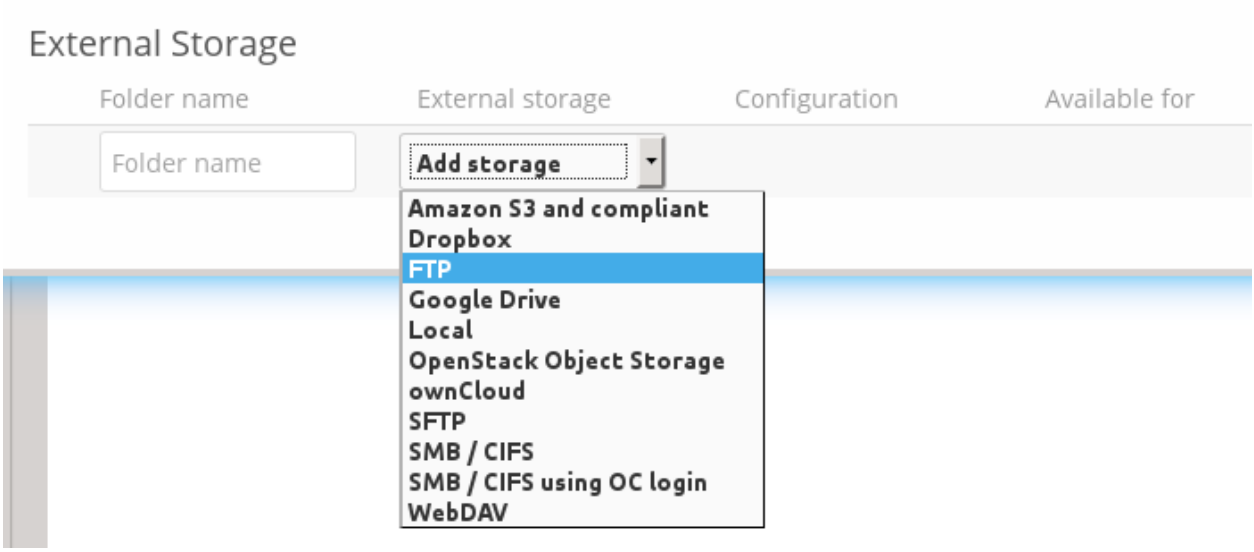
> **Warning:** Enabling this app will disable the **Stay logged in** checkbox on the login page.

The External storage support application is enabled on your Apps page.



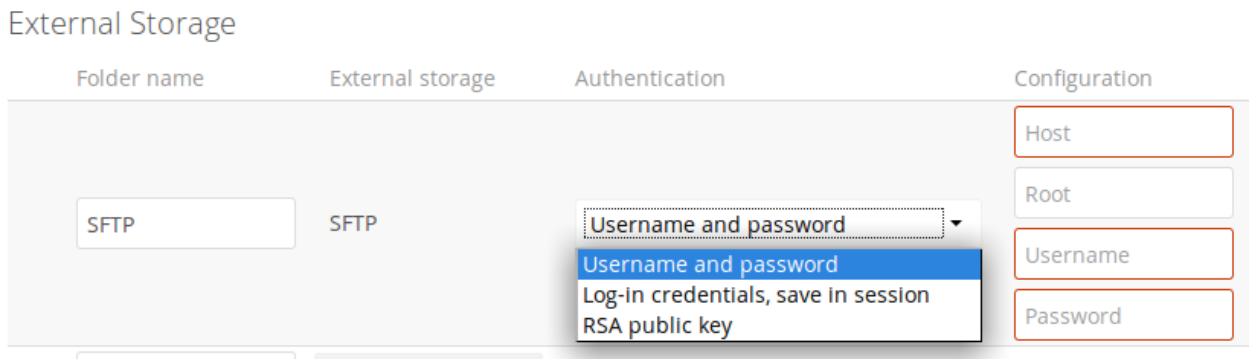

### Storage Configuration

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**.

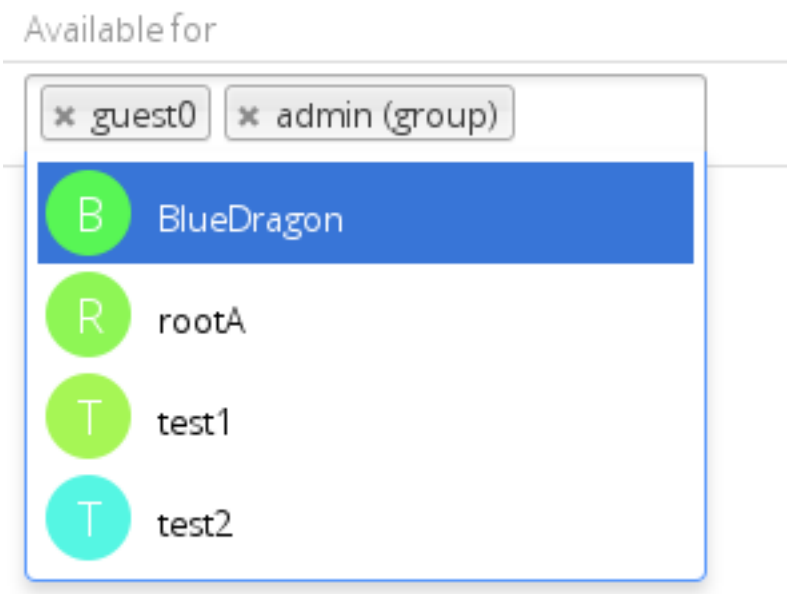



Required fields are marked with a red border. When all required fields are filled, the storage is automatically saved. A green dot next to the storage row indicates the storage is ready for use. A red or yellow icon indicates that Nextcloud could not connect to the external storage, so you need to re-check 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.

## 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.



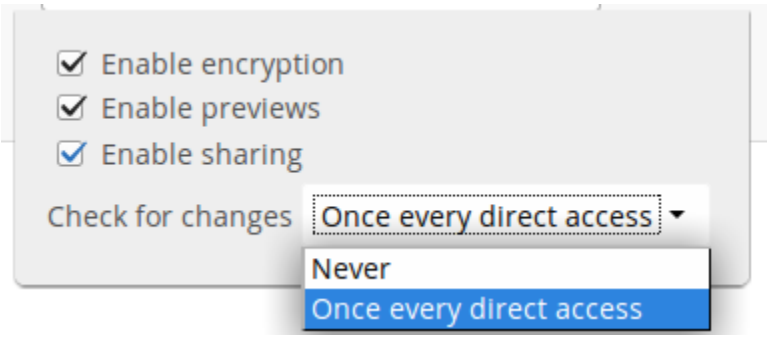

## Mount Options

Hover your cursor to the right of any storage configuration to expose the settings button 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)

The **Encryption** checkbox is visible only when the Encryption app is enabled.

**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. Other apps may provide their own backends, which are not listed here.

### Amazon S3

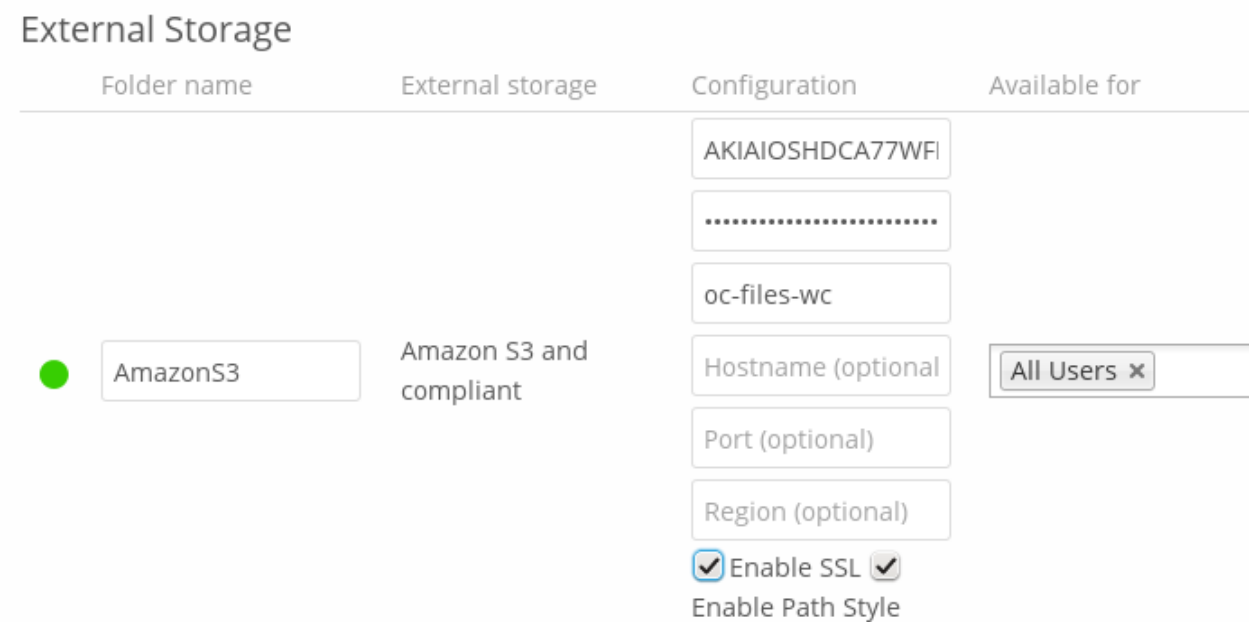
To connect your Amazon S3 buckets to Nextcloud, you will need:

- 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.

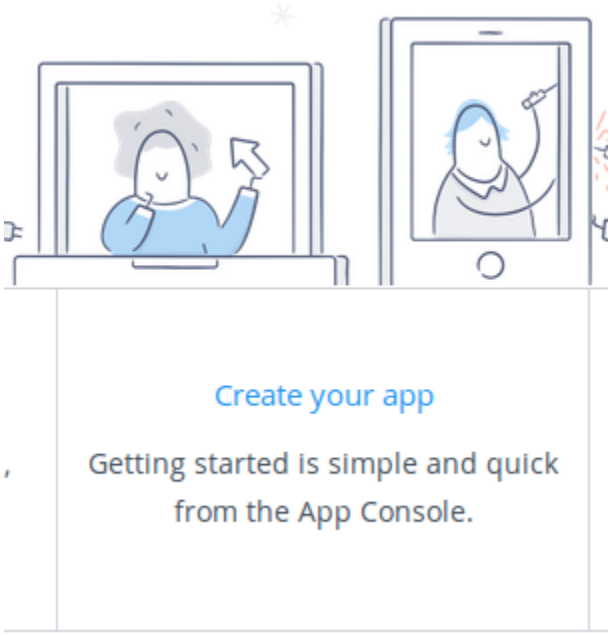
See Configuring External Storage (GUI) for additional mount options and information.

See External Storage Authentication mechanisms for more information on authentication schemes.

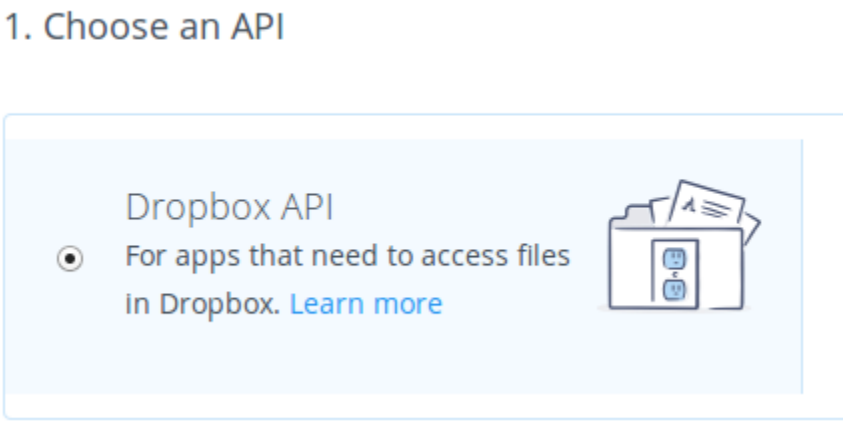
### Dropbox

While Dropbox supports the newer OAuth 2.0, Nextcloud uses OAuth 1.0, so you can safely ignore any references to OAuth 2.0 in the Dropbox configuration.

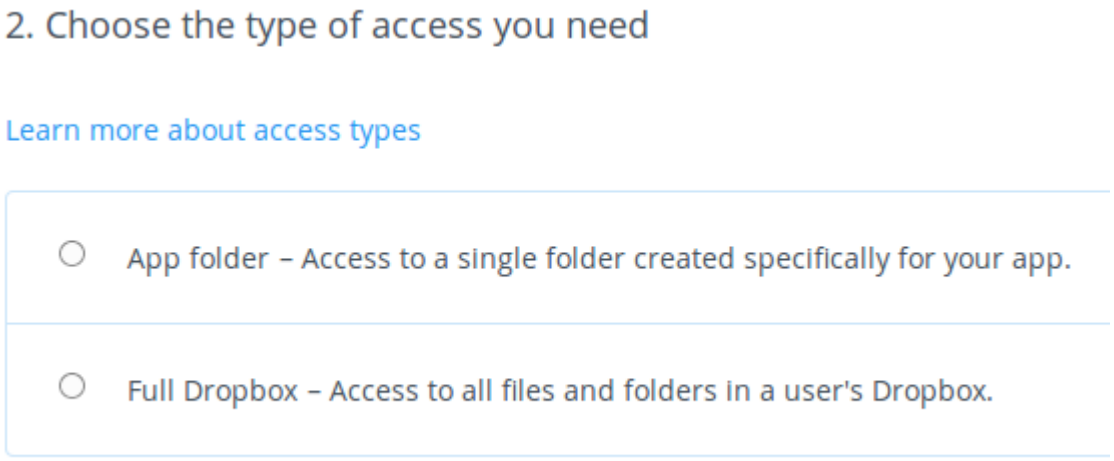
Connecting Dropbox is a little more work because you have to create a Dropbox app. Log into the Dropbox Developers page and click **Create Your App**:



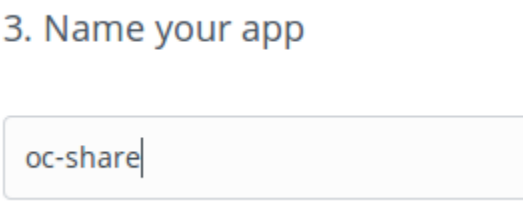

Next, for **Choose an API** check **Dropbox API**.




The next option is choosing which folders to share, or to share everything in your Dropbox.

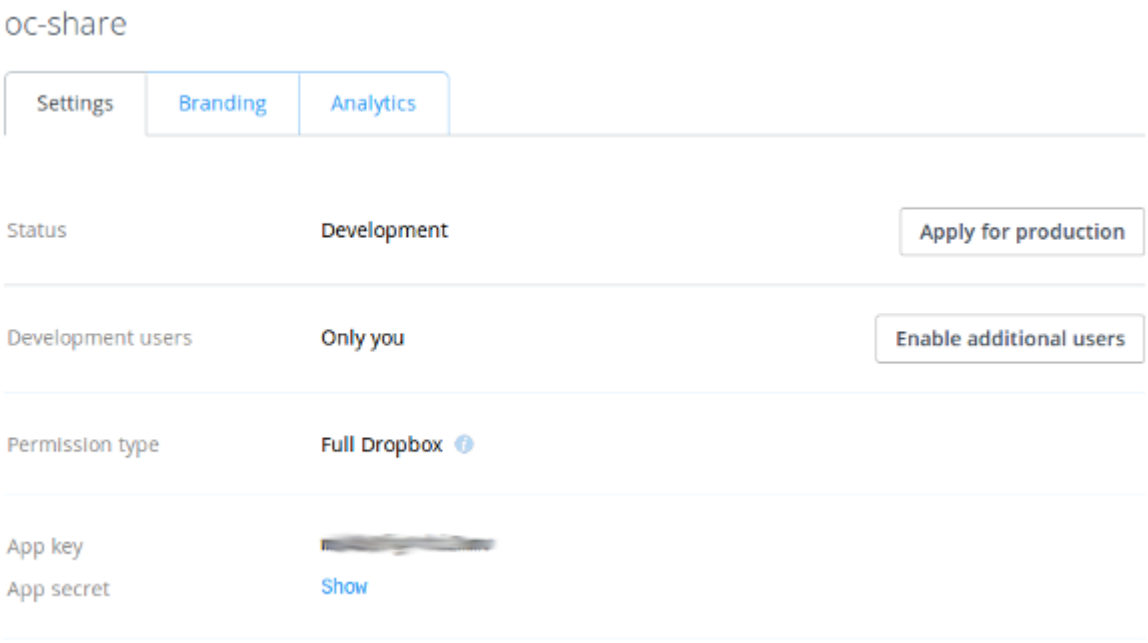



Then enter your app name. This is anything you want it to be.




Then click the **Create App** button.

Now you are on your app page, which displays its settings and more options. Do not click **Development (Apply for production)** because that is for apps that you want to release publicly.

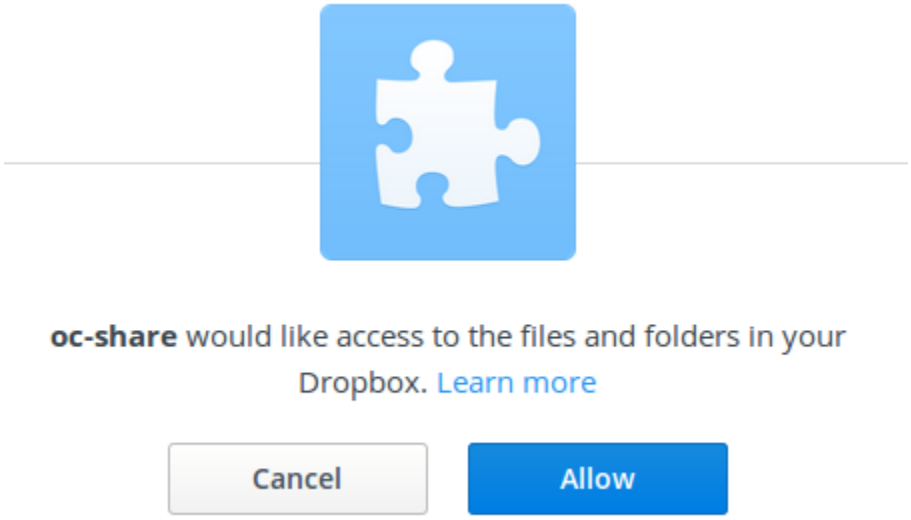



Click **Enable additional users** to allow multiple Nextcloud users to access your new Dropbox share.

Now go to your Nextcloud Admin page. Your Nextcloud configuration requires only the local mount name, the **App Key** and the **App Secret**, and which users or groups have access to the share. Remember the little gear icon at the far right for additional options.

After entering your local mount name, **App Key** and **App Secret**, click **Grant access**.


If you are not already logged into Dropbox, you will be prompted to login and authorize access. This happens only once, when you are first creating the new share. Click **Allow**, and you're done.




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.

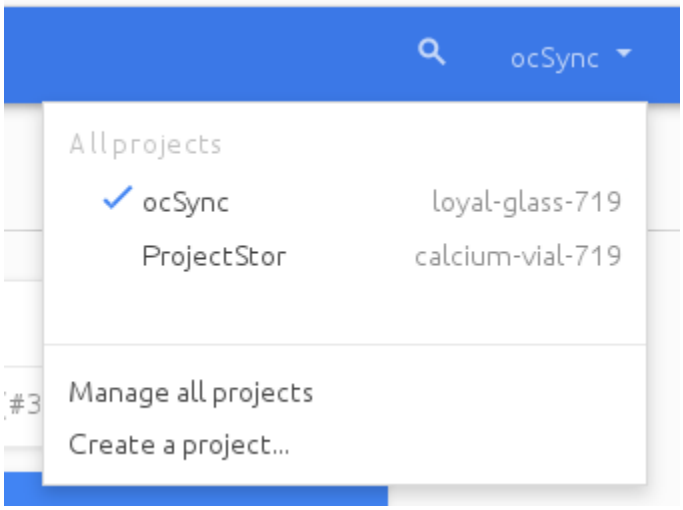
FTP uses the password authentication scheme; see External Storage Authentication mechanisms for more information on authentication schemes.

### Google Drive

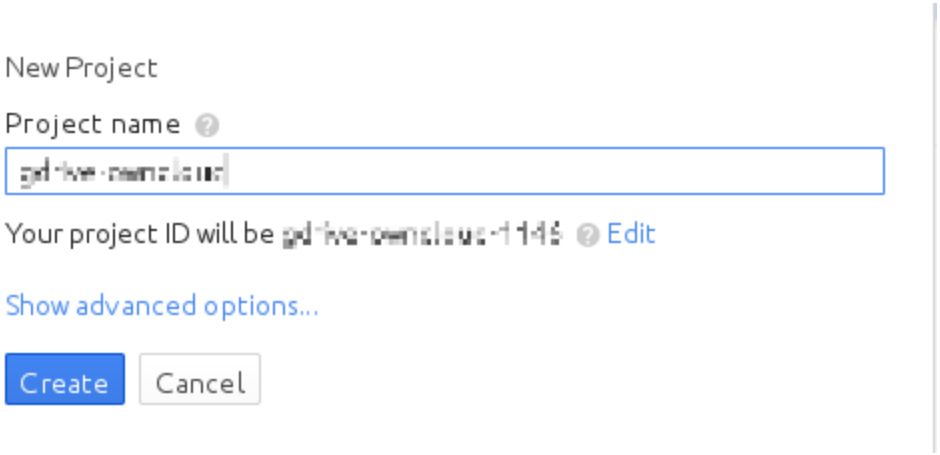
Nextcloud uses OAuth 2.0 to connect to Google Drive. This requires configuration through Google to get an app ID and app secret, as Nextcloud registers itself as an app.

All applications that access a Google API must be registered through the Google Cloud Console. Follow along carefully because the Google interface is a bit of a maze and it's easy to get lost.

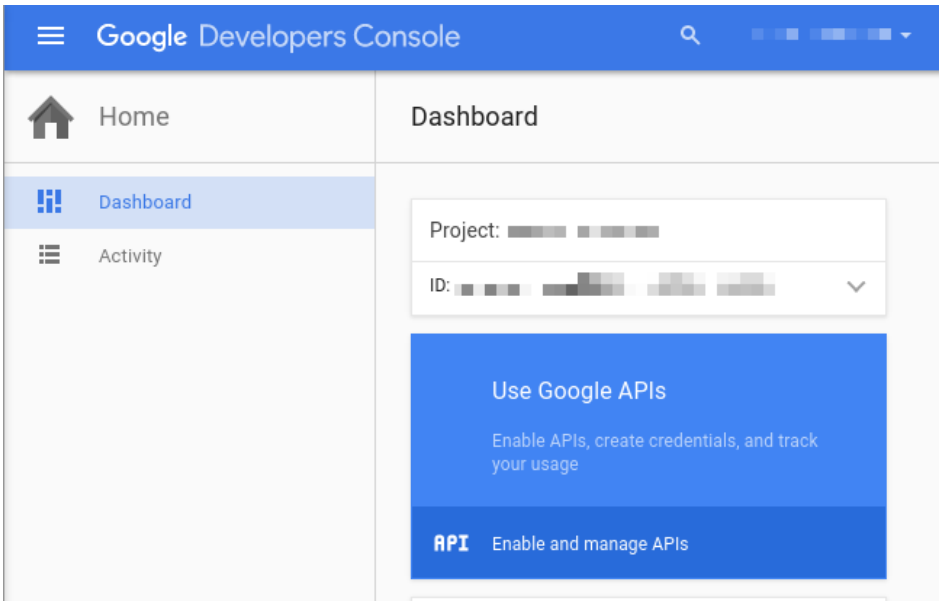
If you already have a Google account, such as Groups, Drive, or Mail, you can use your existing login to log into the Google Cloud Console. After logging in click the **Create Project** button.



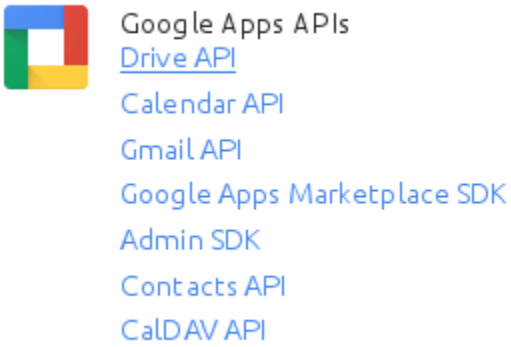

Give your project a name, and either accept the default **Project ID** or create your own, then click the **Create** button.



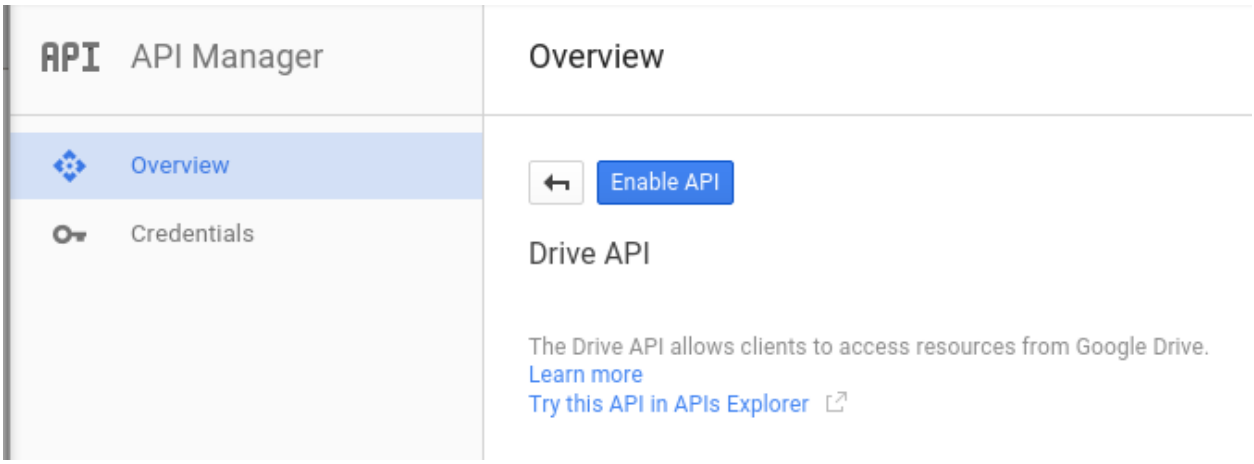
You'll be returned to your dashboard.



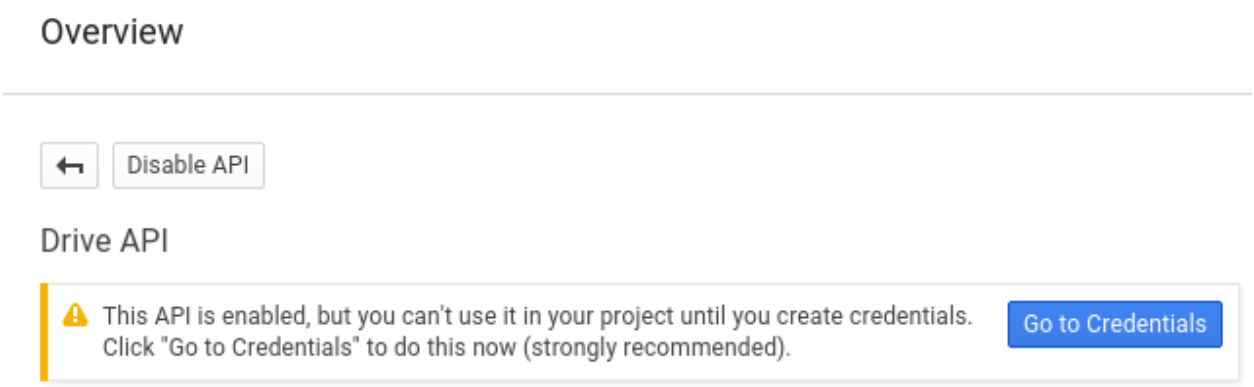

Google helpfully highlights your next step in blue, the **Use Google APIs** box. Make sure that your new project is selected, click on **Use Google APIs** , and it takes you to Google's APIs screen. There are many Google APIs; look for the **Google Apps APIs** and click **Drive API.**



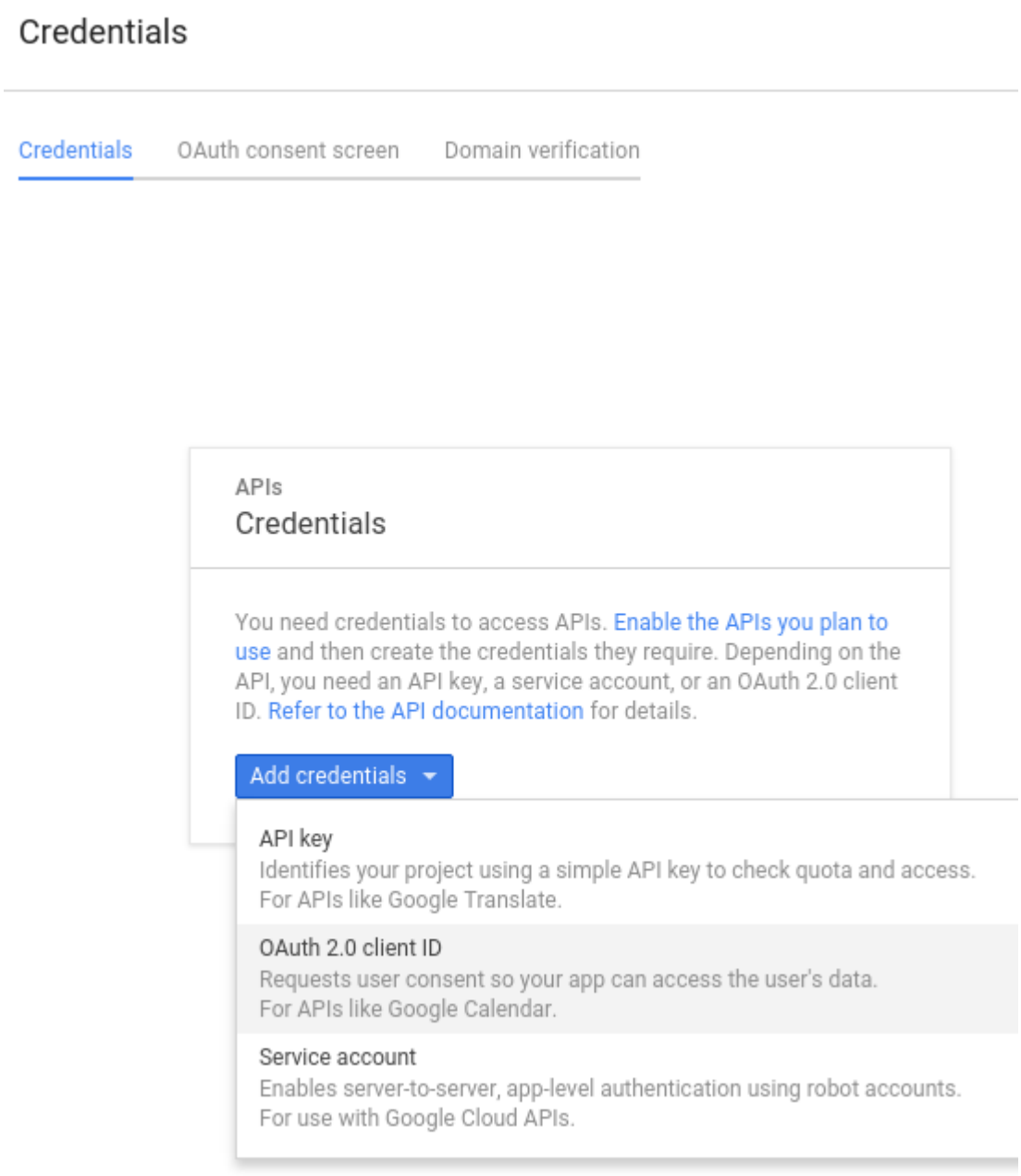

**Drive API** takes you to the API Manager overview. Click the blue **Enable API** button.



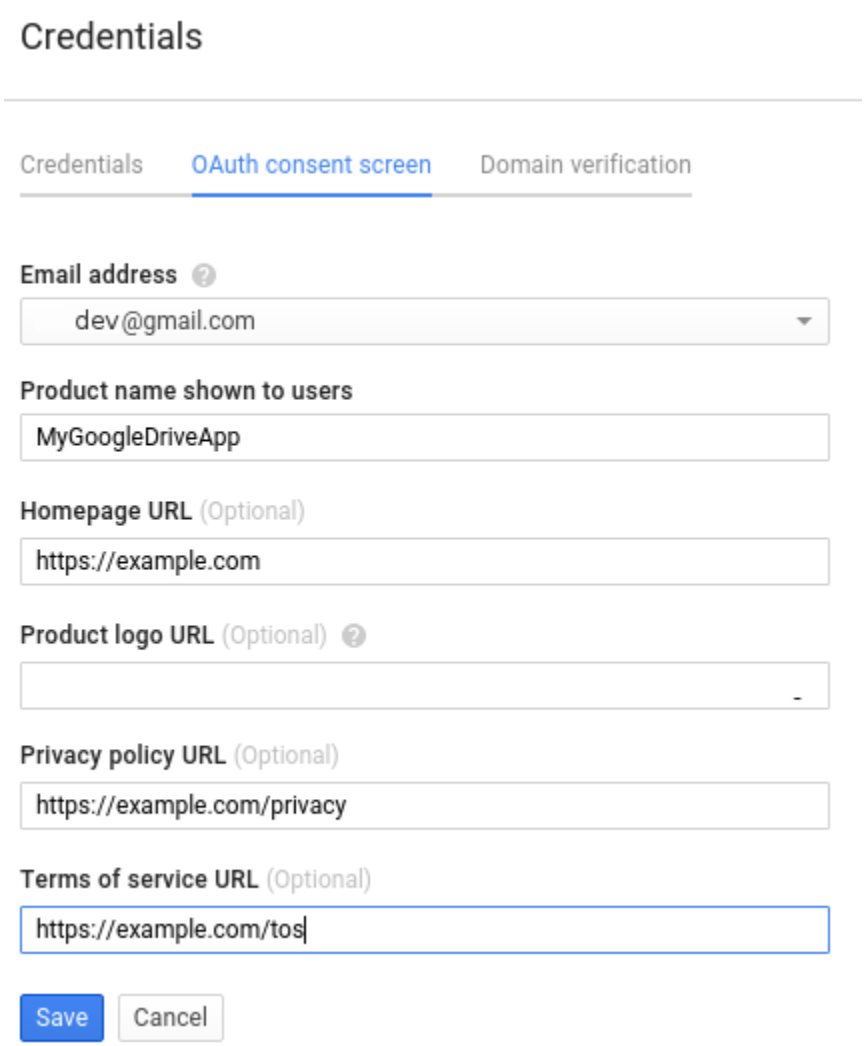

Now you must create your credentials, so click on **Go to credentials**.




For some reason Google warns us again that we need to create credentials. We will use 0Auth 2.0.



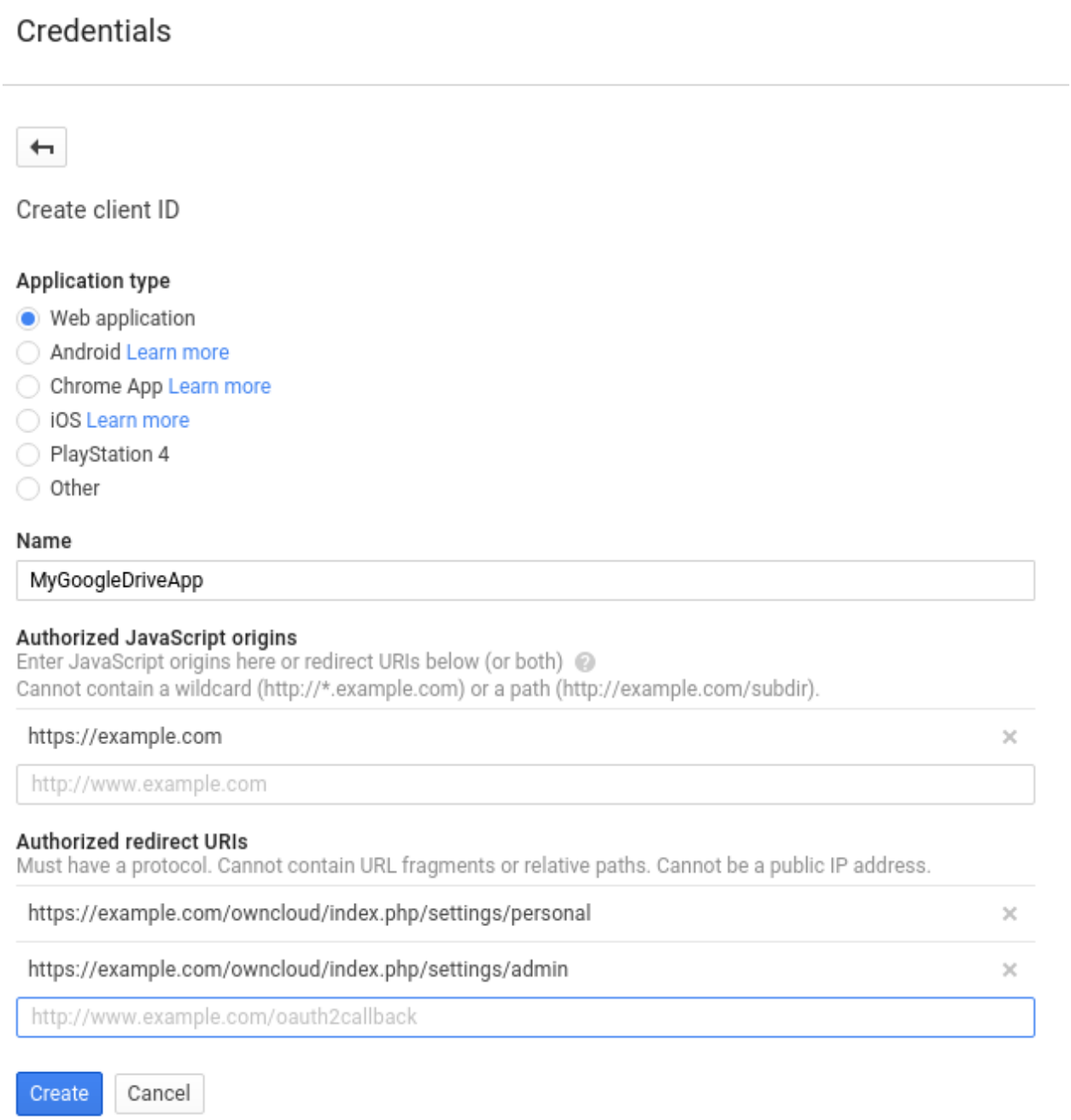

Now we have to create a consent screen. This is the information in the screen Google shows you when you connect your new Google app to Nextcloud the first time. Click **Configure consent screen**. Then fill in the required form fields. Your logo must be hosted, as you cannot upload it, so enter its URL. When you're finished click **Save**.



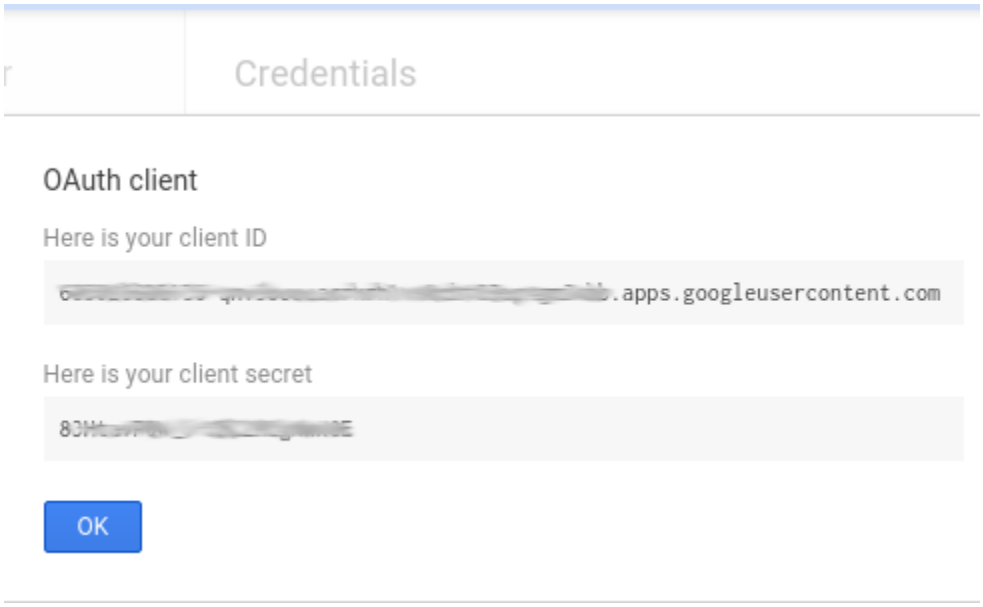

The next screen that opens is **Create Client ID**. Check **Web Application**, then enter your app name. **Authorized JavaScript Origins** is your root domain, for example `https://example.com`, without a trailing slash. You need two **Authorized Redirect URIs**, and they must be in this form:

```
https://example.com/nextcloud/index.php/settings/personal/
https://example.com/nextcloud/index.php/personal/
https://example.com/nextcloud/index.php/settings/admin/externalstorages
https://example.com/nextcloud/settings/admin/externalstorages
```

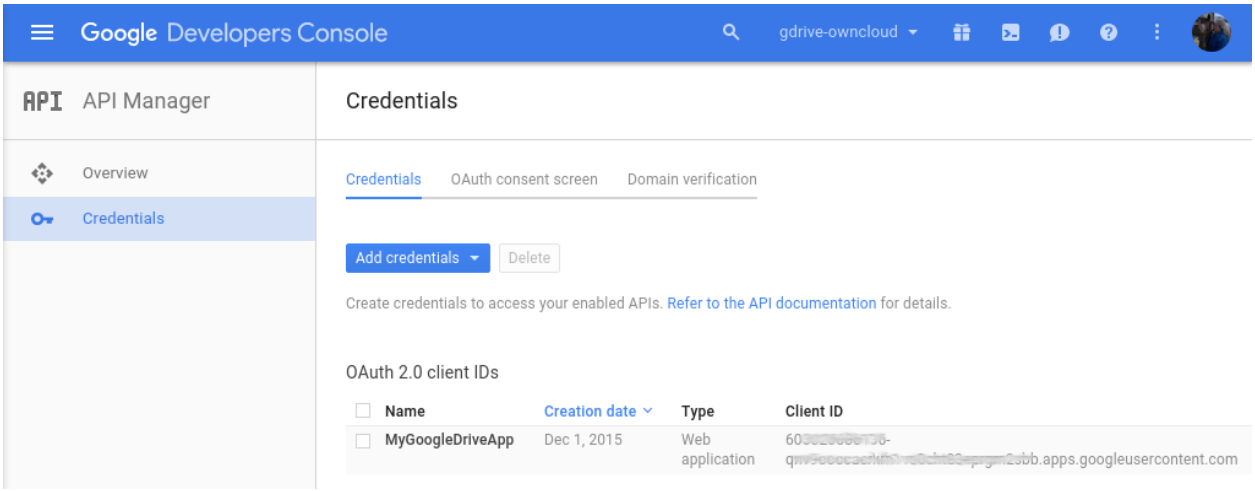
Replace `https://example.com/nextcloud/` with your own Nextcloud server URL, then click **Create**.




Now Google reveals to you your **Client ID** and **Client Secret**. Click **OK**.

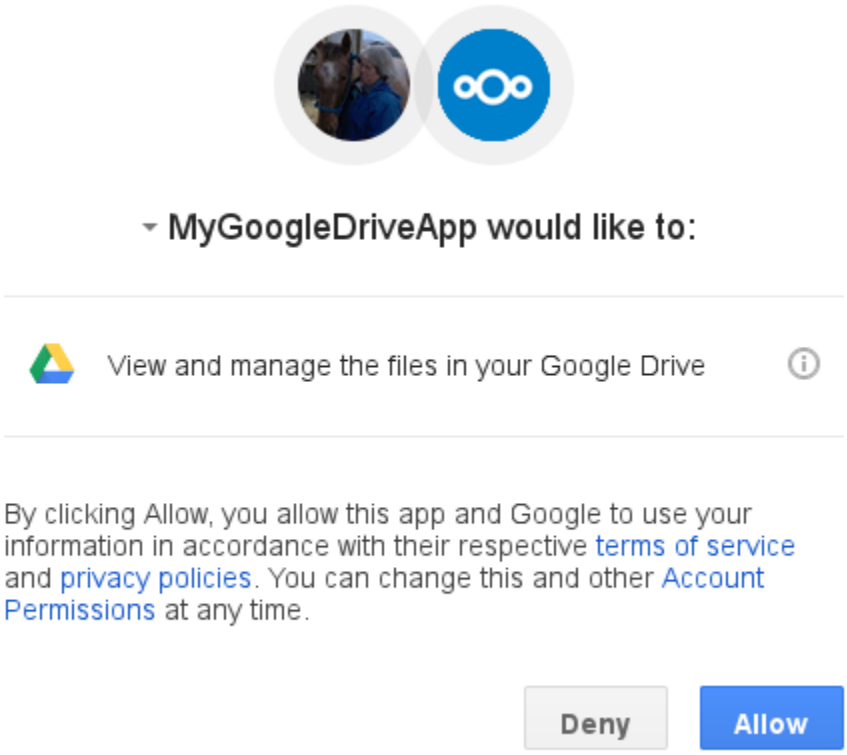



You can see these anytime in your Google console; just click on your app name to see complete information.

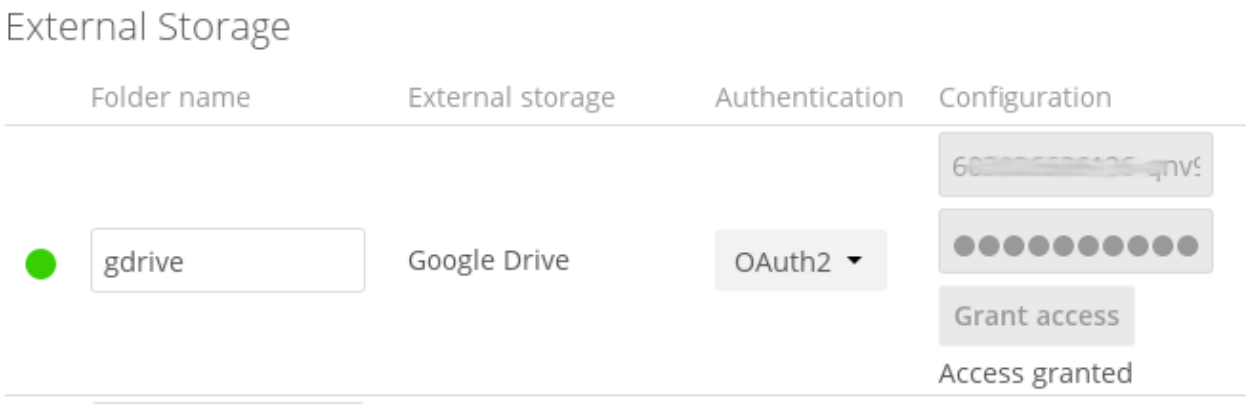



Now you have everything you need to mount your Google Drive in Nextcloud.

Go to the External Storage section of your Admin page, create your new folder name, enter the Client ID and Client Secret, and click **Grant Access**. Your consent page appears when Nextcloud makes a successful connection. Click **Allow**.




When you see the green light confirming a successful connection you're finished.




See Configuring External Storage (GUI) for additional mount options and information.

See External Storage Authentication mechanisms for more information on authentication schemes. 603026686136-qnv9ooocacrkrh1vs0cht83eprgm2sbb.apps.googleusercontent.com

### 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.

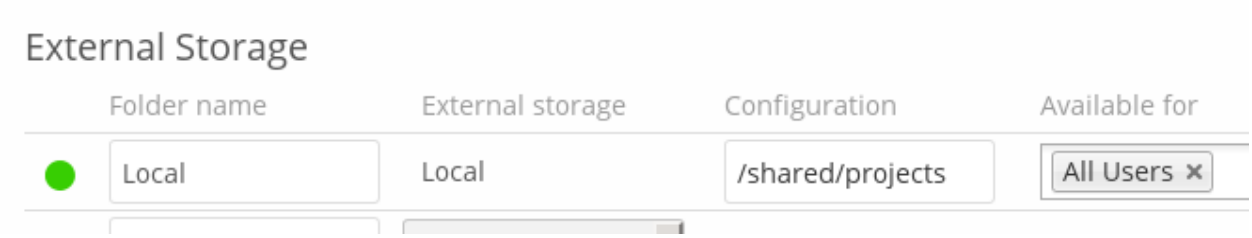
Use this to mount any directory on your Nextcloud server that is outside of your Nextcloud `data/` directory. This directory must be readable and writable by your HTTP server user. These ownership and permission examples are on Ubuntu Linux:

```
sudo -u www-data chown -R www-data:www-data /localdir
sudo -u www-data chmod -R 0750 /localdir
```

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

An Nextcloud storage is a specialized WebDAV storage, with optimizations for Nextcloud-Nextcloud communication. See the WebDAV documentation to learn how to configure an 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.

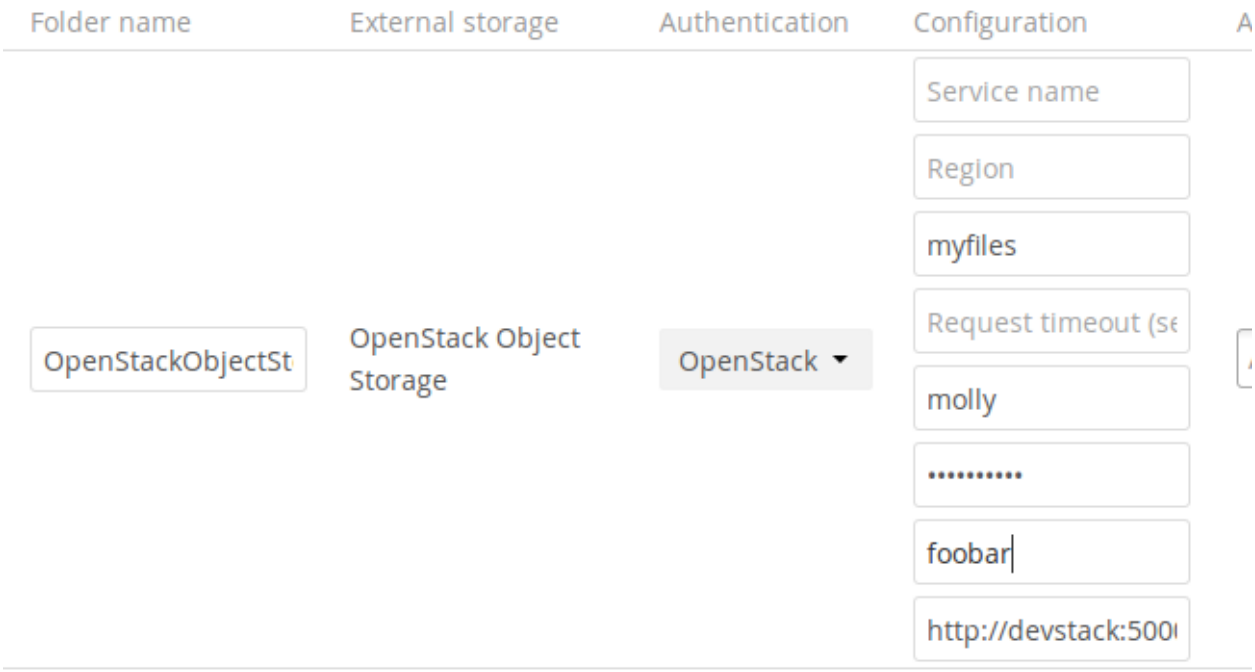
### OpenStack Object Storage

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.)

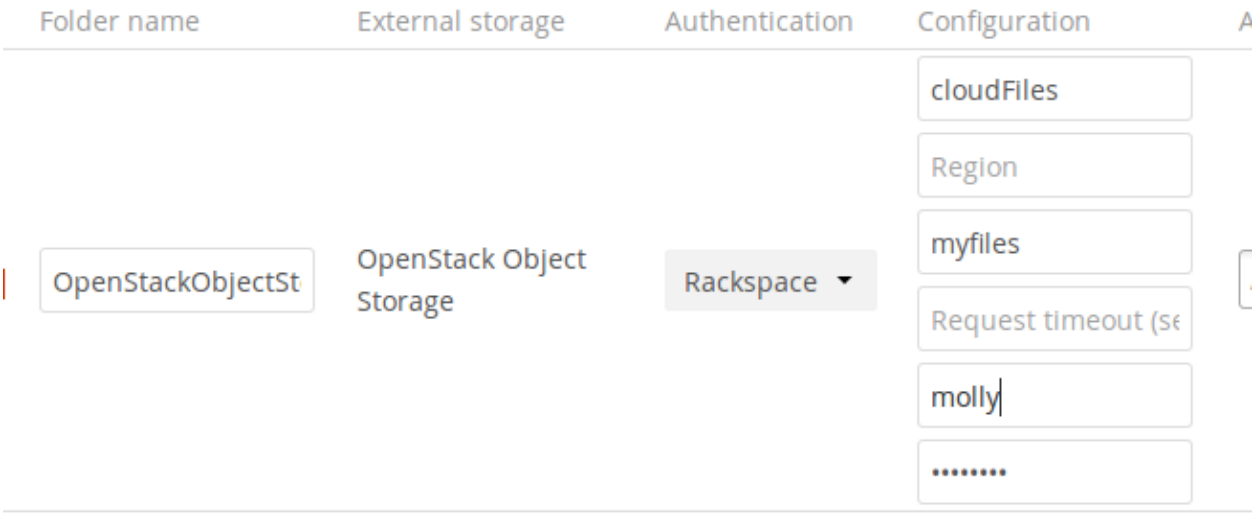
- **Identity Endpoint URL**, the URL to log in to your OpenStack account.




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.

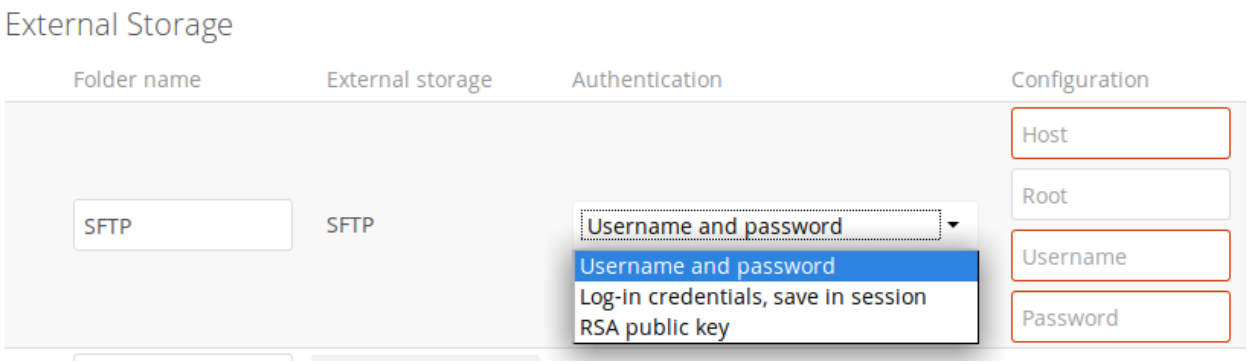
See External Storage Authentication mechanisms for more information on authentication schemes.

### SFTP

Nextcloud's SFTP (FTP over an SSH tunne) 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.)

---

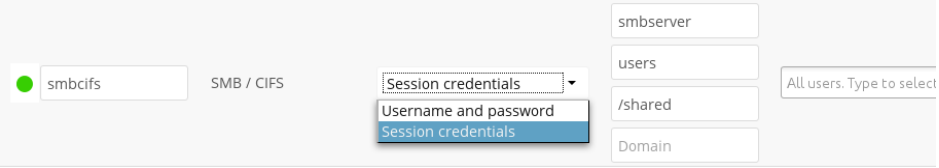
You need the following information:

- Folder name for your local mountpoint.
- Host: The URL of the Samba server.
- Username: The username or domain/username 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

Starting with Nextcloud 10, Nextcloud can use smb update notifications to listen to 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** In some cases (such as when using login credentials) it's not possible to read the smb credentials from the storage configuration, in those cases you can provide the username and password using the `--username` and `--password` arguments.

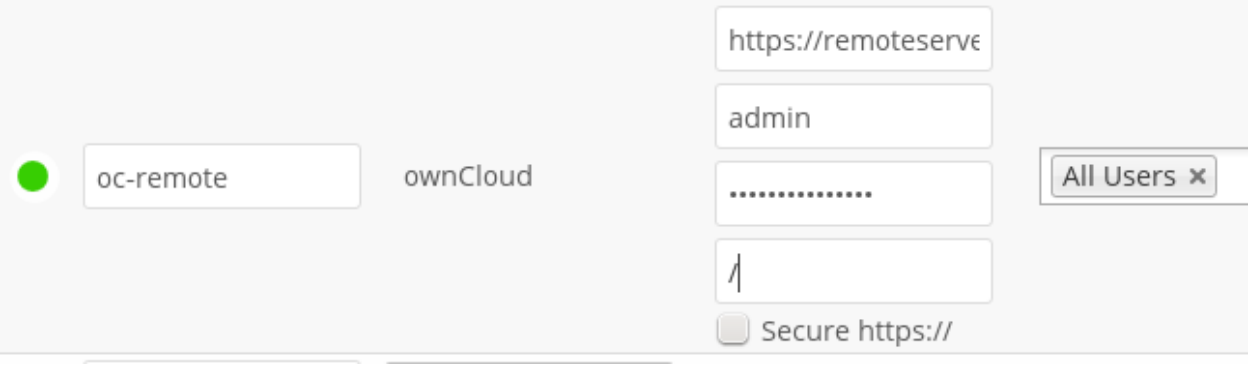
### WebDAV

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.

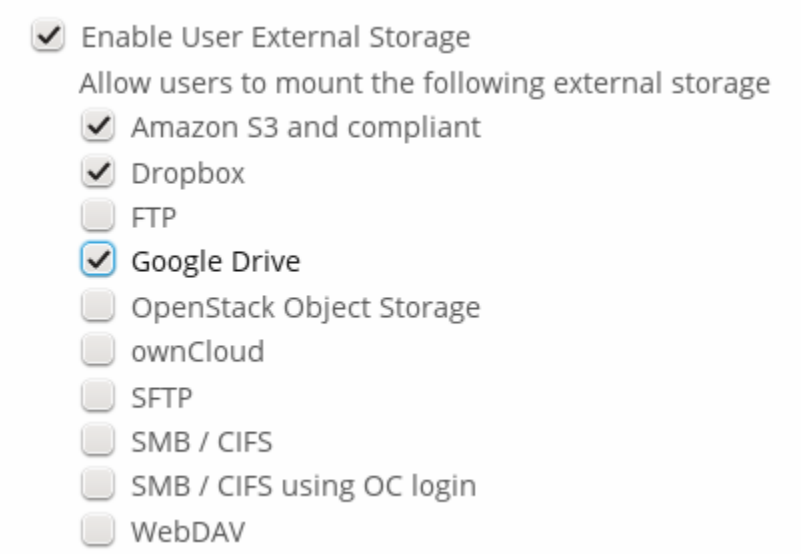
See Configuring External Storage (GUI) for additional mount options and information.

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 Defining 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.

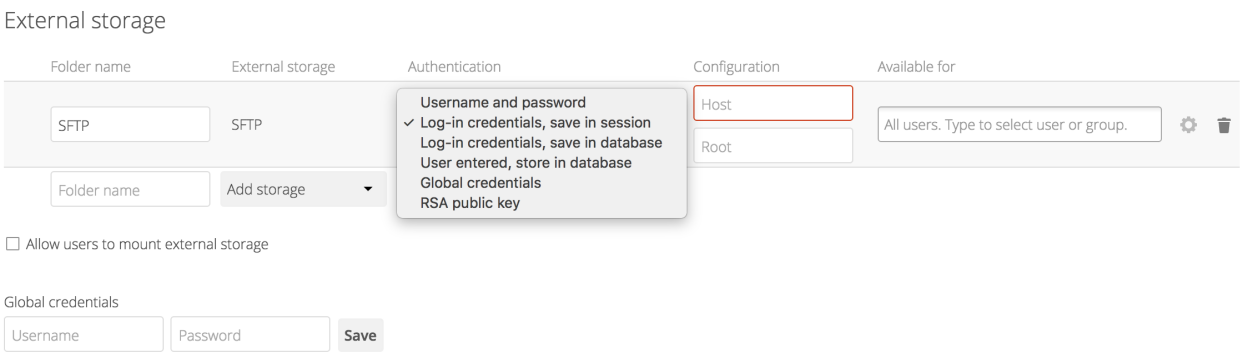
You might need to setup a cron job that runs `sudo -u www-data php occ files:scan --all` (or replace "–all" with the user name, see also Using the occ Command) to trigger a rescan of the user's files periodically (for example every 15 minutes), which includes the mounted external storage.

## Configuring External Storage (Configuration File)

Starting with Nextcloud 9.0, the `data/mount.json` file for configuring external storages has been removed, and replaced with a set of *occ commands*.

## 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 multiple authentication mechanisms. Different mechanisms require different configuration parameters, depending on their behaviour.




## 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. The drawbacks are that sharing is disabled when this mechanism is in use, as Nextcloud has no access to the storage credentials, and background file scanning does not work.

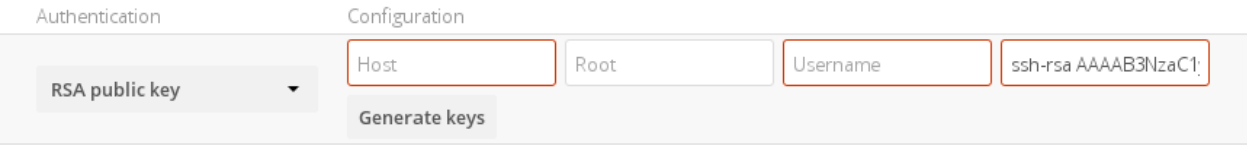
The **Log-in credentials, save in database** mechanism uses the Nextcloud login credentials of the user to connect to the storage. These are stored in the database encrypted with the shared secret. This allows to share files from within this mount point.

The **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.

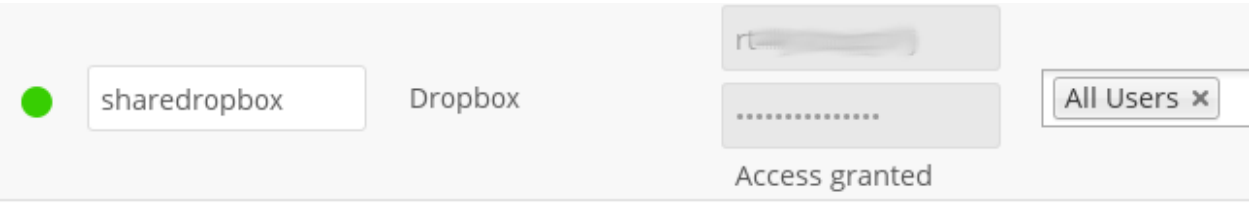
## Public-key Mechanisms

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.




### OAuth

OAuth 1.0 and OAuth 2.0 are both implemented, but currently limited to the Dropbox and Google Drive backends respectively. These mechanisms require additional configuration at the service provider, where an app ID and app secret are provided and then entered into Nextcloud. Then Nextcloud can perform an authentication request, establishing the storage connection.




## Primary Storage

It's possible to use an object store as primary storage, this replaces the default way of storing files in `nextcloud/data` (note that the data directory might still be used for other 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.

### Configuring

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' => array(
        'class' => 'Object\\Storage\\Backend\\Class',
        'arguments' => array(
                ...
        ),
),
```

### Openstack Swift

The 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`

```
'objectstore' => array(
        'class' => 'OC\\Files\\ObjectStore\\Swift',
        'arguments' => array(
                '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'
        ),
),
```

## Amazon S3

The S3 backend mounts a bucket on an Amazon S3 Storage or compatible server into the virtual filesystem. The class to be used is `\\OC\\Files\\ObjectStore\\S3`

```
'objectstore' => array(
        'class' => 'OC\\Files\\ObjectStore\\S3',
        'arguments' => array(
                '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
        ),
),
```

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 servers such as Ceph Object Gateway, which in turn usually don't require the region to be set.

`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 it's data over multiple buckets for scalability purpose.

To setup multiple buckets, `config.php` needs to be configured using `'objectstore_multibucket'`

```
'objectstore_multibucket' => array(
        'class' => 'Object\\Storage\\Backend\\Class',
        'arguments' => array(
                // optional, defaults to 64
                'num_buckets' => 64,
                // will be prefixed by an integer in the range from 0 to (num_nuckets-1)
                'bucket' => 'nextcloud_',
                ...
        ),
),
```

Nextcloud will map every user to a range of buckets and save all files for that user in it's respective bucket.

---

**Note:** Changing the number of buckets for an existing Nextcloud instance is supported but the mapping from users to buckets is persistent so only newly created users will be mapped to the updated range of buckets.

---

## 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.

In Nextcloud 9.0 the 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:** Starting with Nextcloud 9.0 we support 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

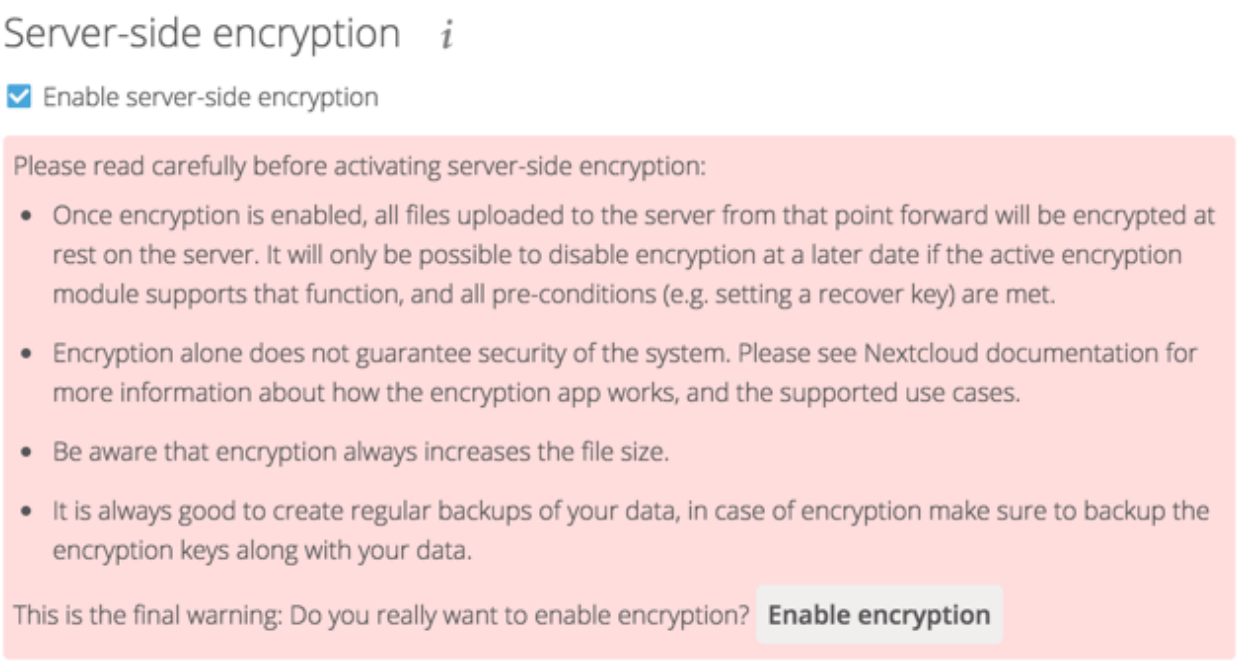
Plan very carefully before enabling encryption because it is not reversible via the Nextcloud Web interface. If you lose your encryption keys your files are not recoverable. Always have backups of your encryption keys stored in a safe location, and consider enabling all recovery options.

You have more options via the `occ` command (see *occ Encryption Commands*)

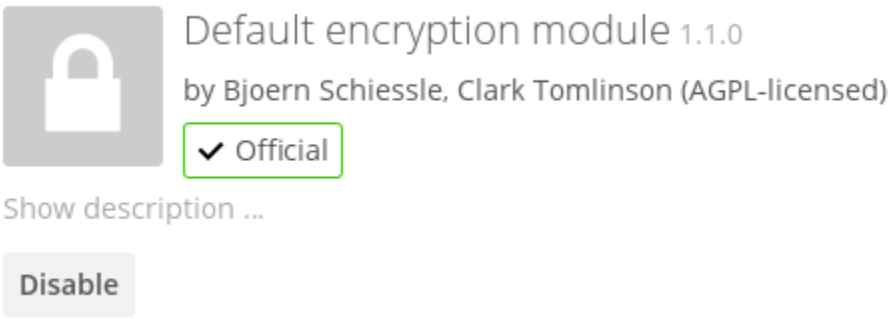
## Enabling Encryption

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.

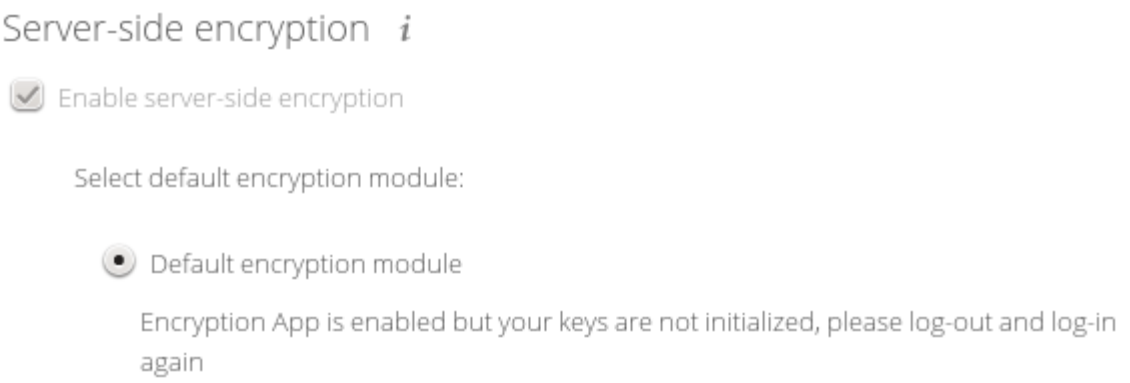
First go to the **Server-side encryption** section of your Admin page and check **Enable server-side encryption**. You have one last chance to change your mind.



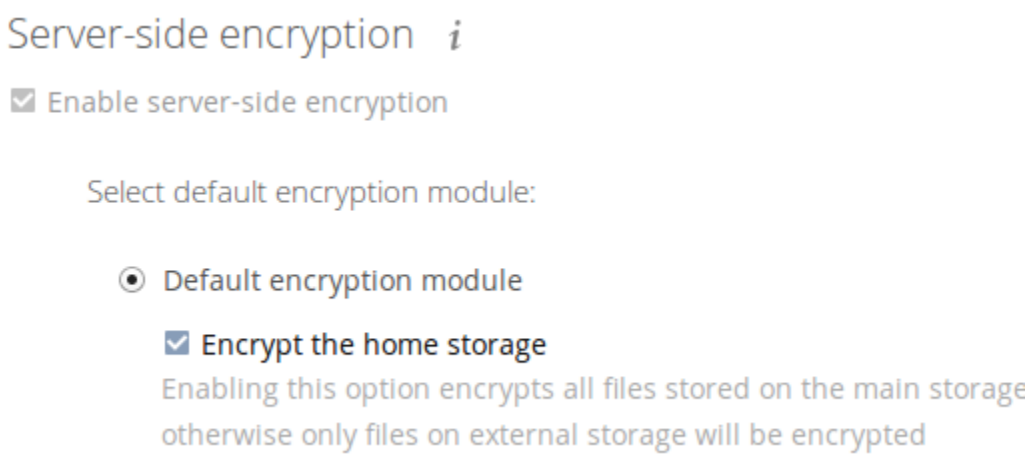

After clicking the **Enable Encryption** button you see the message "No encryption module loaded, please load a encryption module in the app menu", so go to your Apps page to enable the Nextcloud Default Encryption Module.




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.



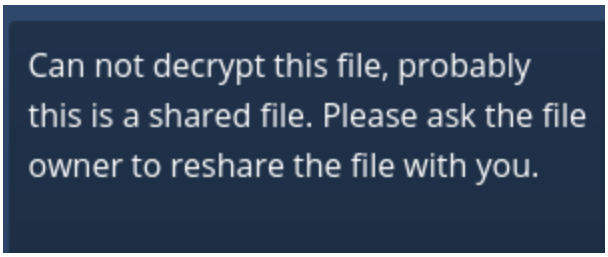

When you log back in, there is a checkbox for enabling encryption on your home storage. This is checked by default. Un-check to avoid encrypting your home storage.




## 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

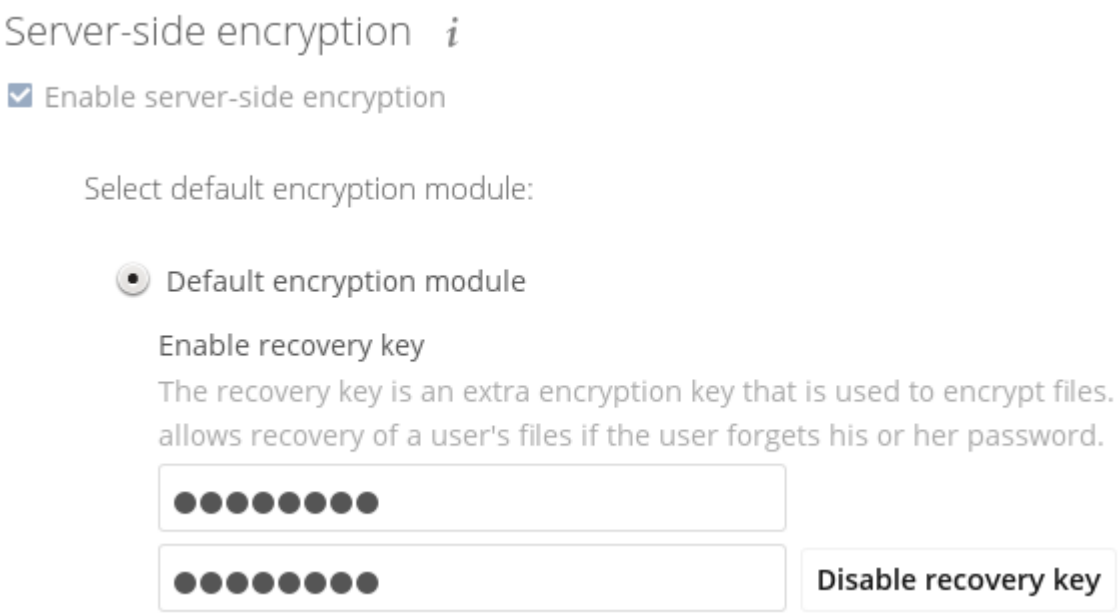
You and your users can encrypt individual external mountpoints. You must have external storage enabled on your Admin page, and enabled for your users.

Encryption settings can be configured in the mount options for an external storage mount, see *Mount Options* (Configuring External Storage (GUI))

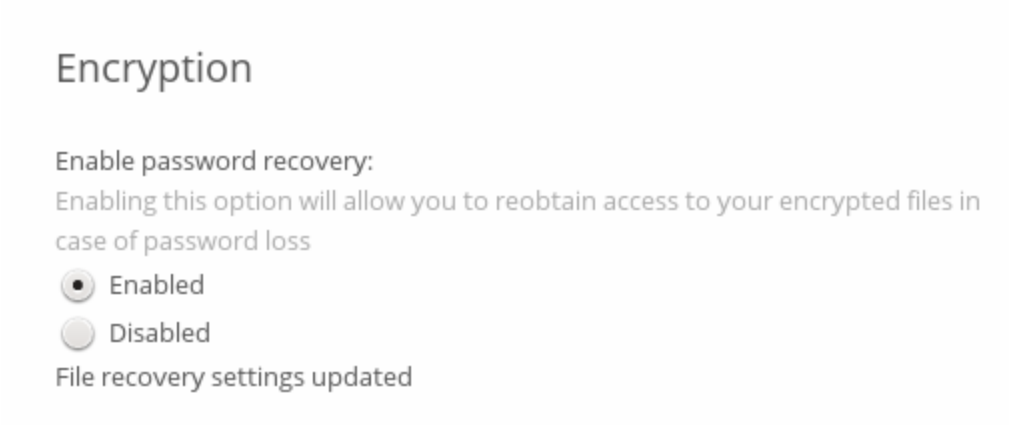
## Enabling Users File Recovery Keys

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".

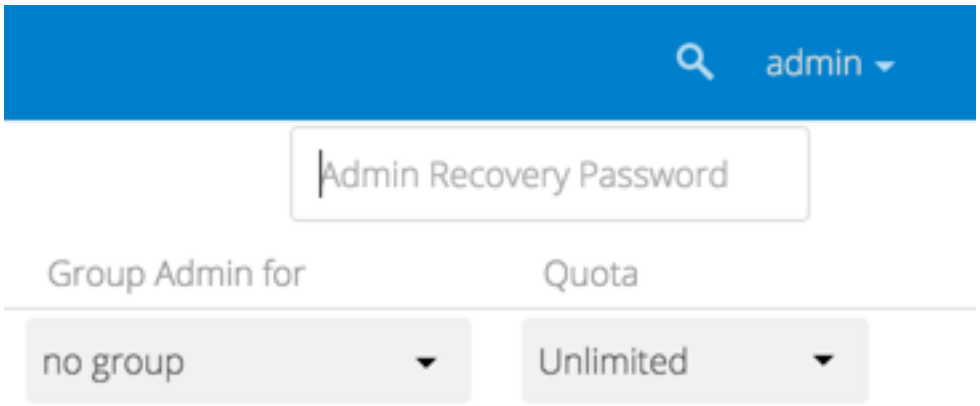
To avoid all this, create a Recovery Key. Go to the Encryption section of your Admin page and set a recovery key password.



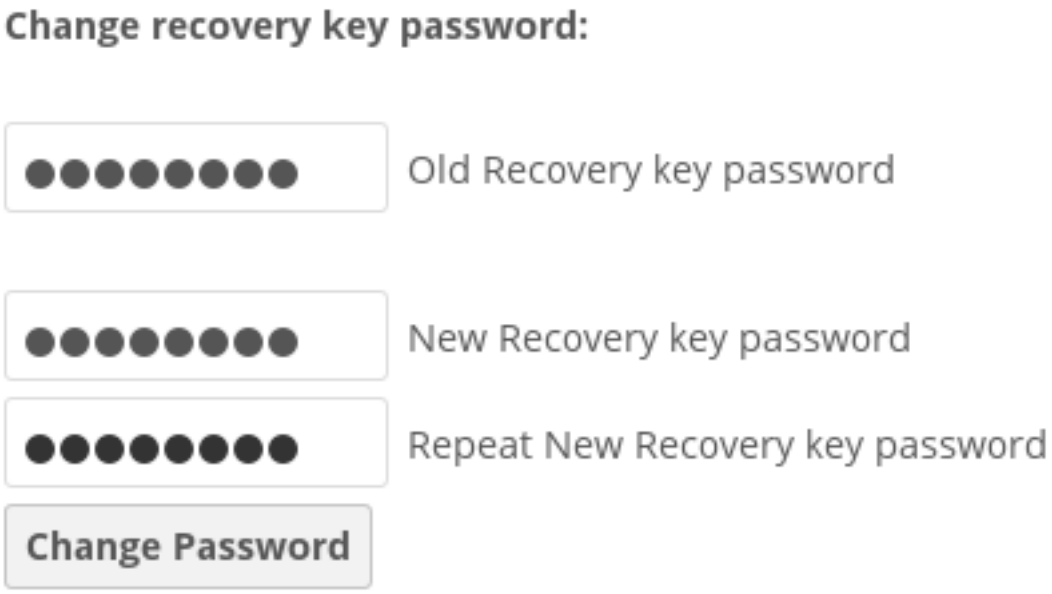

Then your users have the option of enabling password recovery on their Personal pages. If they do not do this, then the Recovery Key won't work for them.




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 an Nextcloud server only new and changed files are encrypted. This command gives you the option to encrypt all files. You must first put your Nextcloud server into single-user mode to prevent any user activity until encryption is completed:

```
occ maintenance:singleuser --on
Single user mode is currently enabled
```

Then 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 root 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. This example is for Ubuntu Linux. Note that the new folder is relative to your `occ` directory:

```
mkdir /etc/keys
chown -R root:www-data /etc/keys
chmod -R 0770 /etc/keys
occ encryption:change-key-storage-root ../../../etc/keys
Start to move keys:
   4 [============================]
Key storage root successfully changed to ../../../etc/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 single-user mode, and then disable your encryption module with this command:

```
occ maintenance:singleuser --on
occ encryption:disable
```

Take it out of single-user mode when you are finished:

```
occ maintenance:singleuser --off
```

### 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.

## 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 Configuring 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.

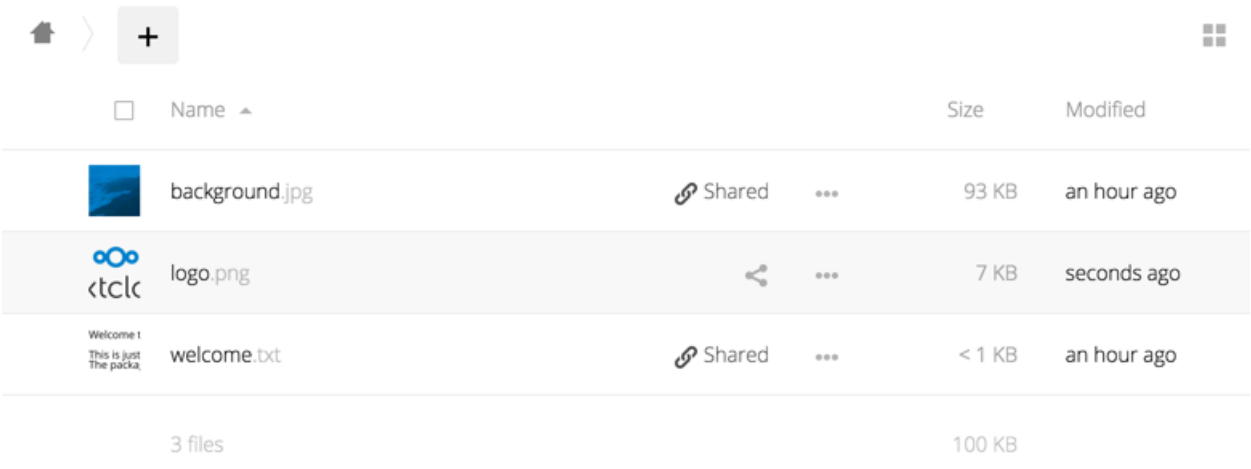
Learn more about Reds at Redis. Memcached, the popular distributed memory caching system, is not suitable for the new file locking because it is not designed to store locks, and data can disappear from the cache at any time. Redis is a key-value store, and it guarantees that cached objects are available for as long as they are needed.

Debian Jesse users, please see this Github discussion if you have problems with LDAP authentication.

## 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,
```

## 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 that 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 Versions; no files will be deleted.

## 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*.

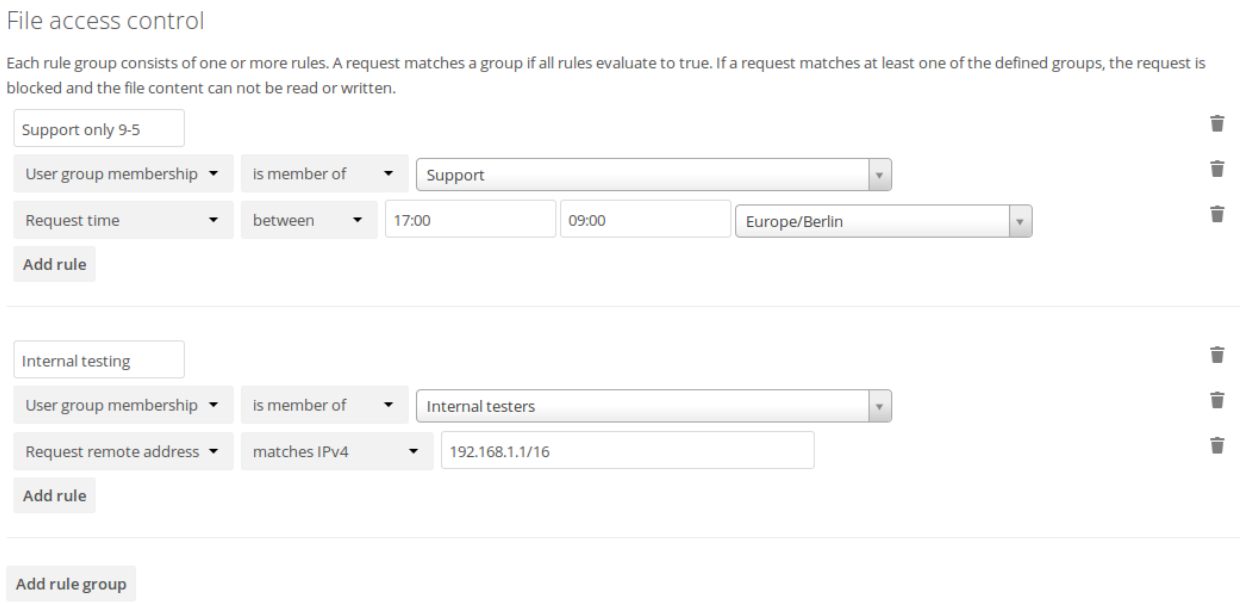
### Denied access

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
- Syncronise the file with clients, such as the Nextcloud desktop and mobile clients

## Examples




The first rule group `Support only 9-5` denies any access to files for users of the Support user group, between 5pm and 9am.

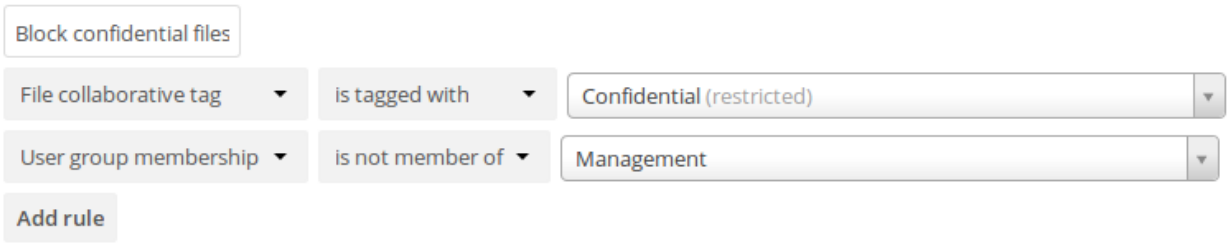
The second rule group `Internal testing` prevents users of the Internal testers group to access files from outside of the local network.

## 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.

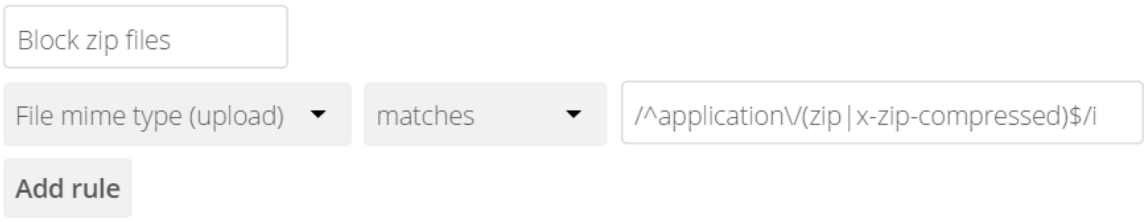
This example blocks access to any folder with the tag `Confidential`.




## 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.
- **File mimetype:** The mimetype of the file, e.g. `text/plain`
- **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.

CHAPTER

SIX

# DATABASE CONFIGURATION

## Converting Database Type

You can convert a SQLite database to a more 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 setup 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)

*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`.

### Unconvertible Tables

If you updated your Nextcloud installation there might exist old tables, which are not used anymore. The converter will tell you which ones.

```
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_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.

---

### MySQL / MariaDB with Binary Logging Enabled

Nextcloud is currently using a `TRANSACTION_READ_COMMITTED` transaction isolation to avoid data loss under high load scenarios (e.g. by using the sync client with many clients/users and many parallel operations). This requires a disabled or correctly configured binary logging when using MySQL or MariaDB. Your system is affected if you see the following in your log file during the installation or update of Nextcloud:

> An unhandled exception has been thrown: exception 'PDOException' with message 'SQLSTATE[HY000]: General error: 1665 Cannot execute statement: impossible to write to binary log since BINLOG_FORMAT = STATEMENT and at least one table uses a storage engine limited to row-based logging. InnoDB is limited to row-logging when transaction isolation level is READ COMMITTED or READ UNCOMMITTED.'

There are two solutions. One is to disable binary logging. Binary logging records all changes to your database, and how long each change took. The purpose of binary logging is to enable replication and to support backup operations.

The other is to change the BINLOG_FORMAT = STATEMENT in your database configuration file, or possibly in your database startup script, to BINLOG_FORMAT = MIXED. See Overview of the Binary Log and The Binary Log for detailed information.

### Database "READ COMMITED" 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:

- 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/php5/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.sock
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;
GRANT ALL PRIVILEGES ON nextcloud.* TO 'username'@'localhost' IDENTIFIED BY 'password';
```

You can quit the prompt by entering:

```
quit
```

An 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_",
```

### 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/php5/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
```

An 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
```

An 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 workaround General error: 2006 MySQL server has gone away

The database request takes too long and therefore the MySQL server times out. Its 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 Config.php 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.dom
```

```
PING db.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=3.64 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 can be accessed 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       | 5.1.67 |
+---------------+--------+
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 MySQL 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), 32-bit
(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
```

CHAPTER

SEVEN

# 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.

CHAPTER

EIGHT

# MAINTENANCE

## Backup

To backup an Nextcloud installation there are four main things you need to retain:

1. The config folder
2. The data folder
3. The database
4. The theme folder

### 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 -Aax 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-sqlbkp_`
```

#### 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 +"%Y%m%d"`.bak
```

## 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, see the `Setting Strong Directory Permissions` section of Installation Wizard.

### 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

#### Clean Database Before Restoring

**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 emoijs 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 utf8mb4 COLL
```

#### PostgreSQL

```
PGPASSWORD="password" psql -h [server] -U [username] -d nextcloud -c "DROP DATABASE \"nextcloud\";"
PGPASSWORD="password" psql -h [server] -U [username] -d nextcloud -c "CREATE DATABASE \"nextcloud\";"
```

### 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" pg_restore -c -d nextcloud -h [server] -U [username]
nextcloud-sqlbkp.bak
```

## How to Upgrade

There are three ways to upgrade your Nextcloud server:

- With the Updater App.
- 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 without skipping any of them, as skipping releases increases the risk of errors. Major releases are 9, 10, and 11. Point

releases are intermediate releases for each major release. For example, 9.0.52 and 10.0.2 are point releases. **Skipping major releases is not supported.**

**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.

> **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 availablilty of an update. Then you decide which update method to use.

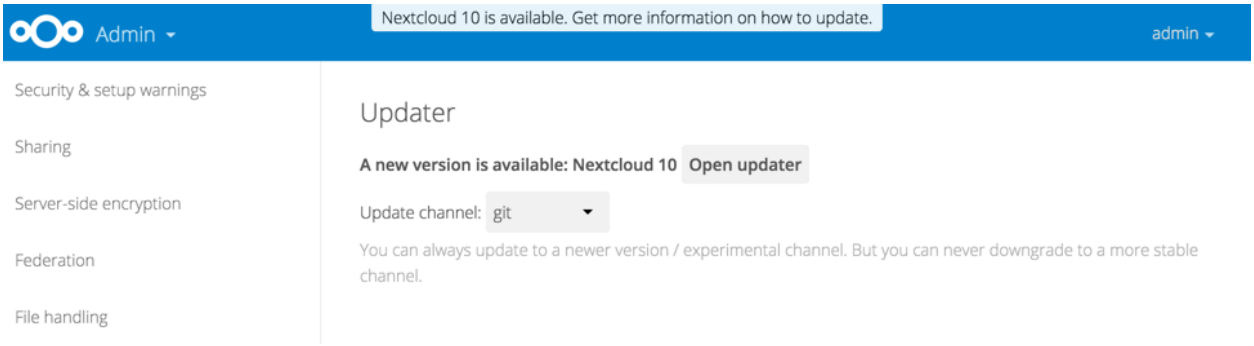



Fig. 8.1: *Figure 1: The top banner is the update notification that is shown on every page, and the Updates section can be found in the admin page*

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.

## Upgrade via build-in Updater

The build-in updater automates many of the steps of upgrading an 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 build-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 build-in updater performs these operations:

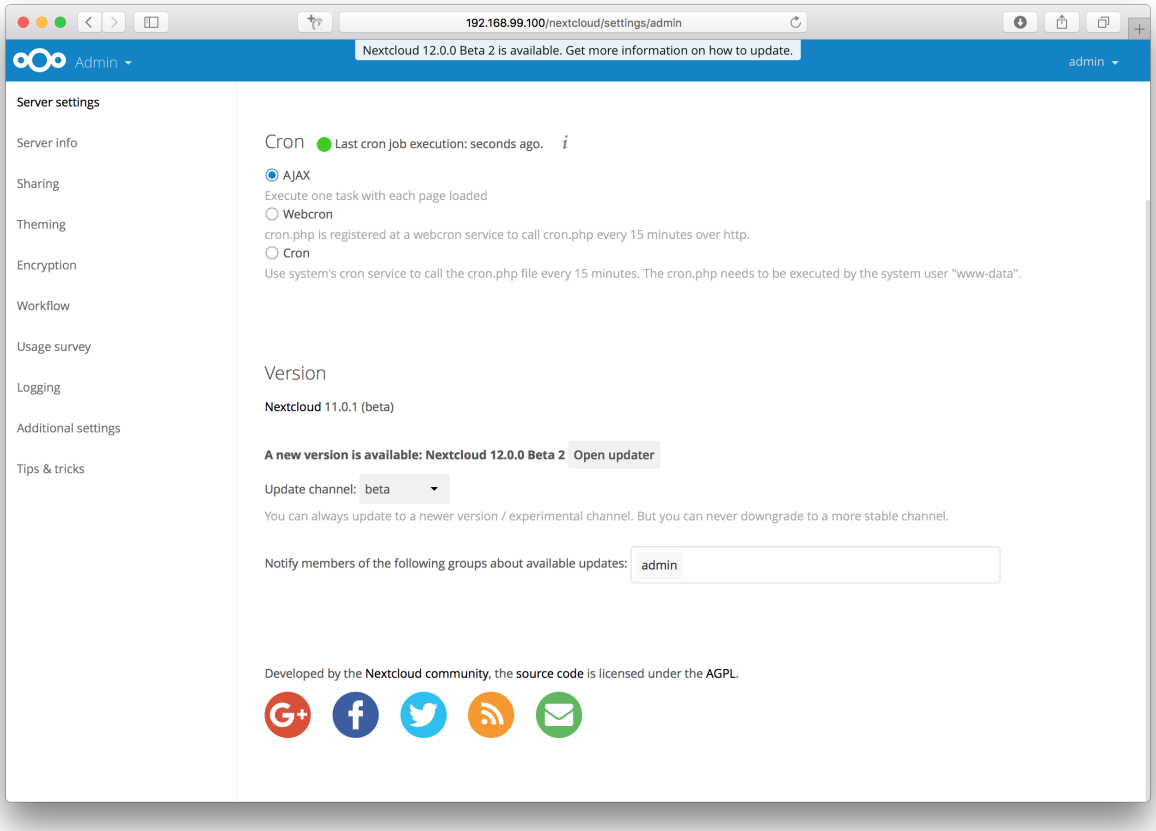
- **Check for expected files:** checks if only the expected files of a Nextcloud installation are present, because it turned out that some files that were left in the Nextcloud directory caused side effects that risked the update procedure.
- **Check for write permissions:** checks if all files that 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.

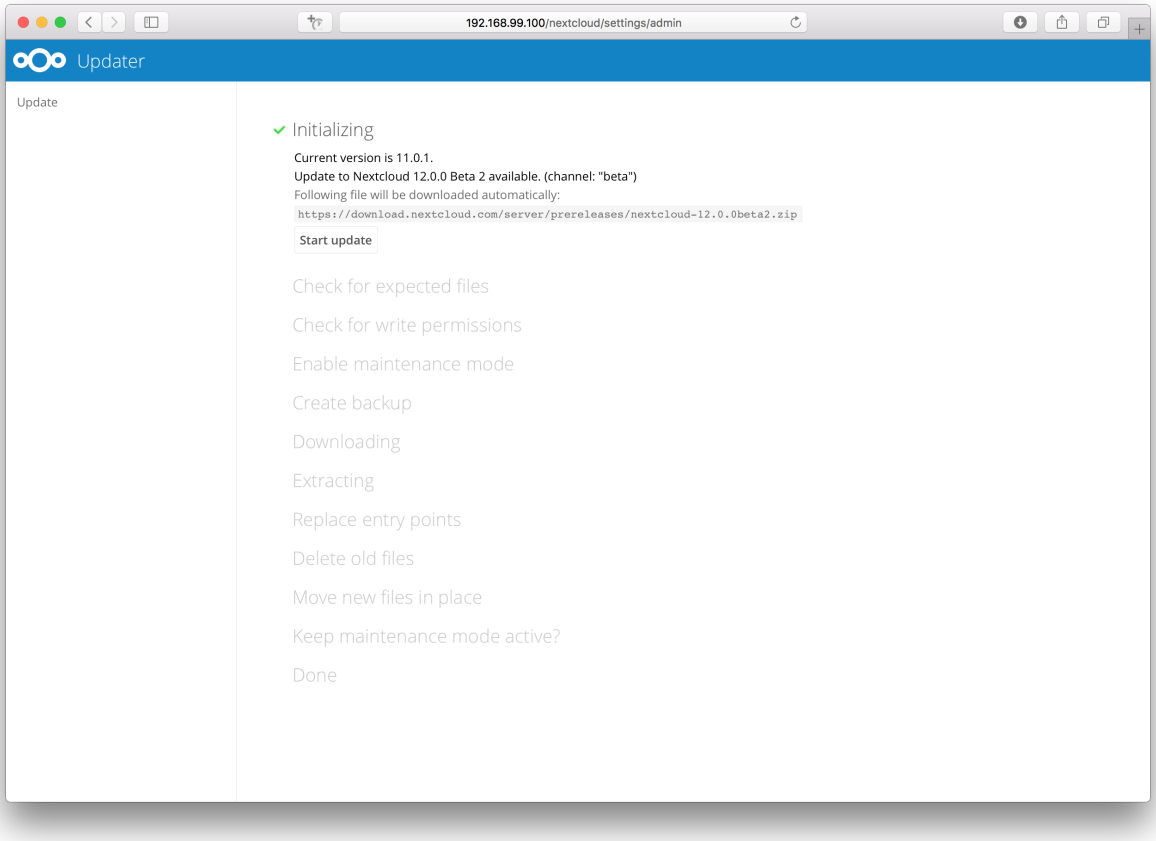
## Using the web based Updater

Using the build-in Updater to update your Nextcloud installation is just a few steps:

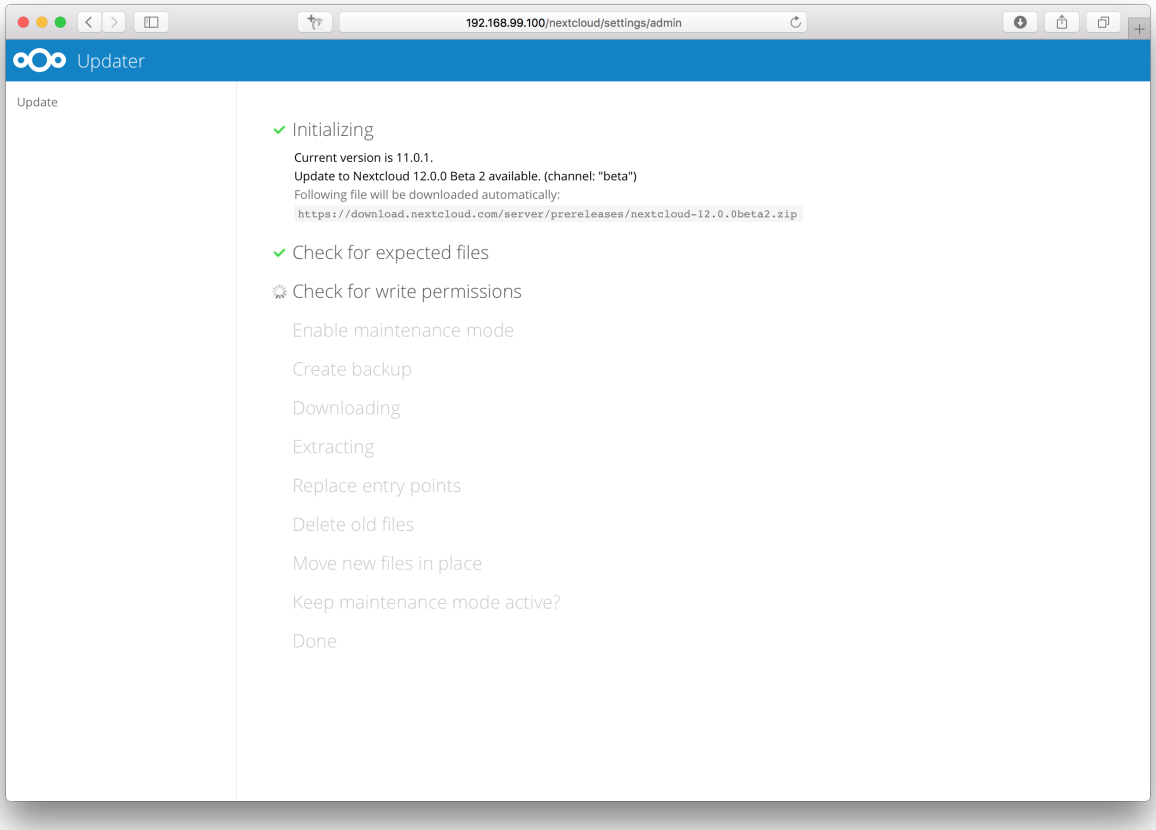
1. You should see a notification at the top of any Nextcloud page when there is a new update available. Go to the admin settings page and scroll to the section "Version". This section has a button to open the updater. This section as well as the update notification is only available if the update notication app is enabled in the apps management.



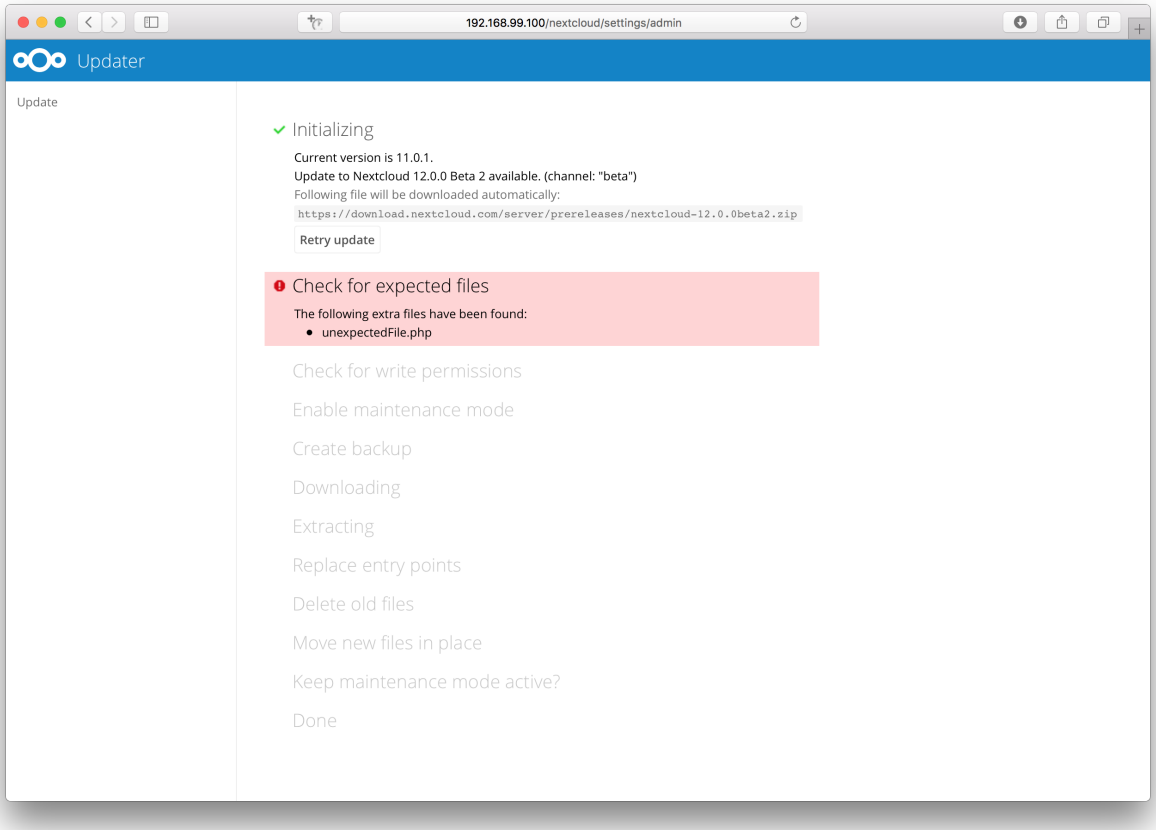
2. Click the button “Open updater”.



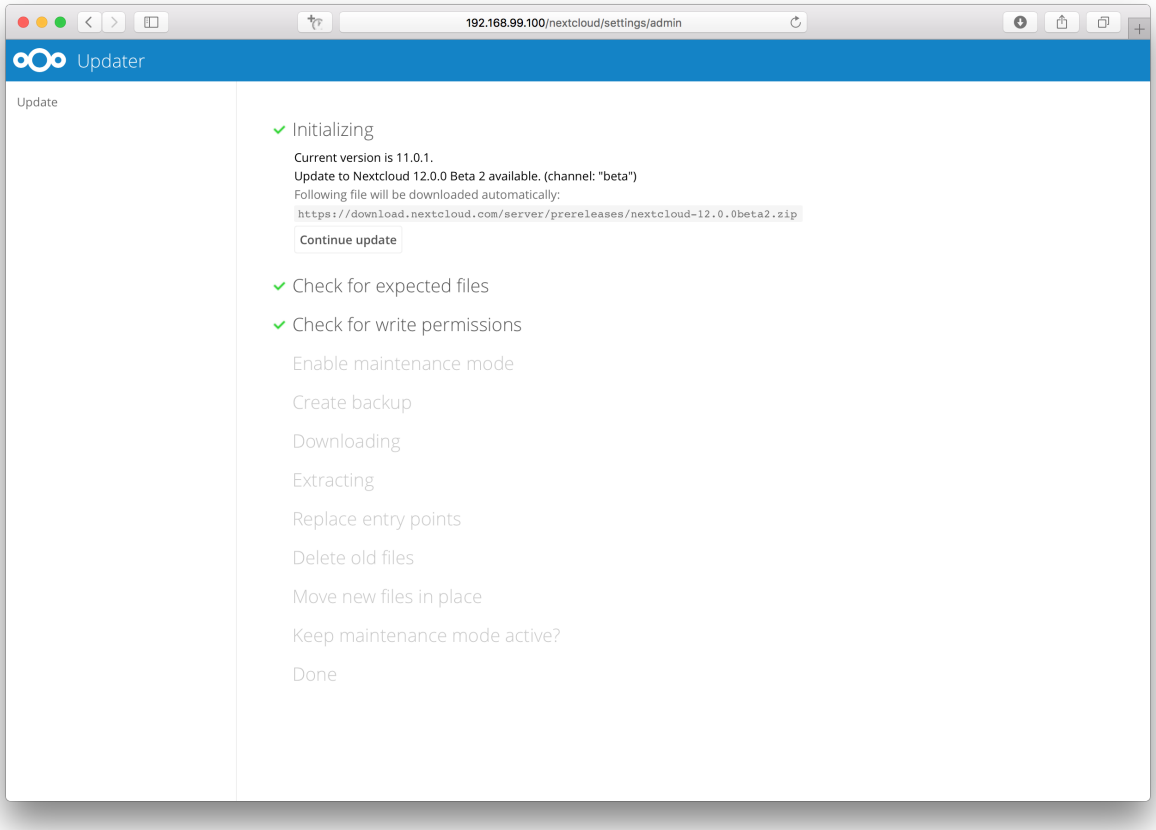

3. Verify the information that is shown and click the button “Start update” to start the update.



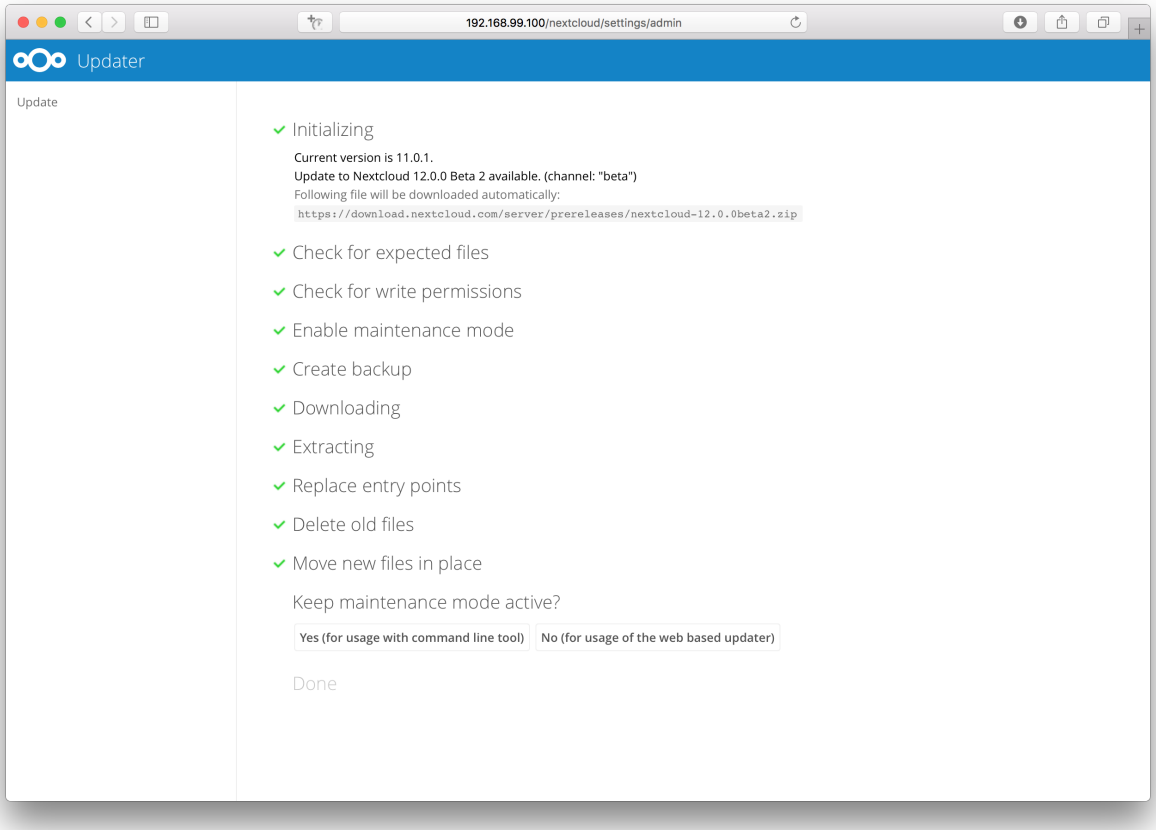

4. In case an error happens or the check failed the updater stops processing and gives feedback. You can now try to solve the problem and click the “Retry update” button. This will continue the update and re-run the failed step. It will not re-run the previous succeeded steps.




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.



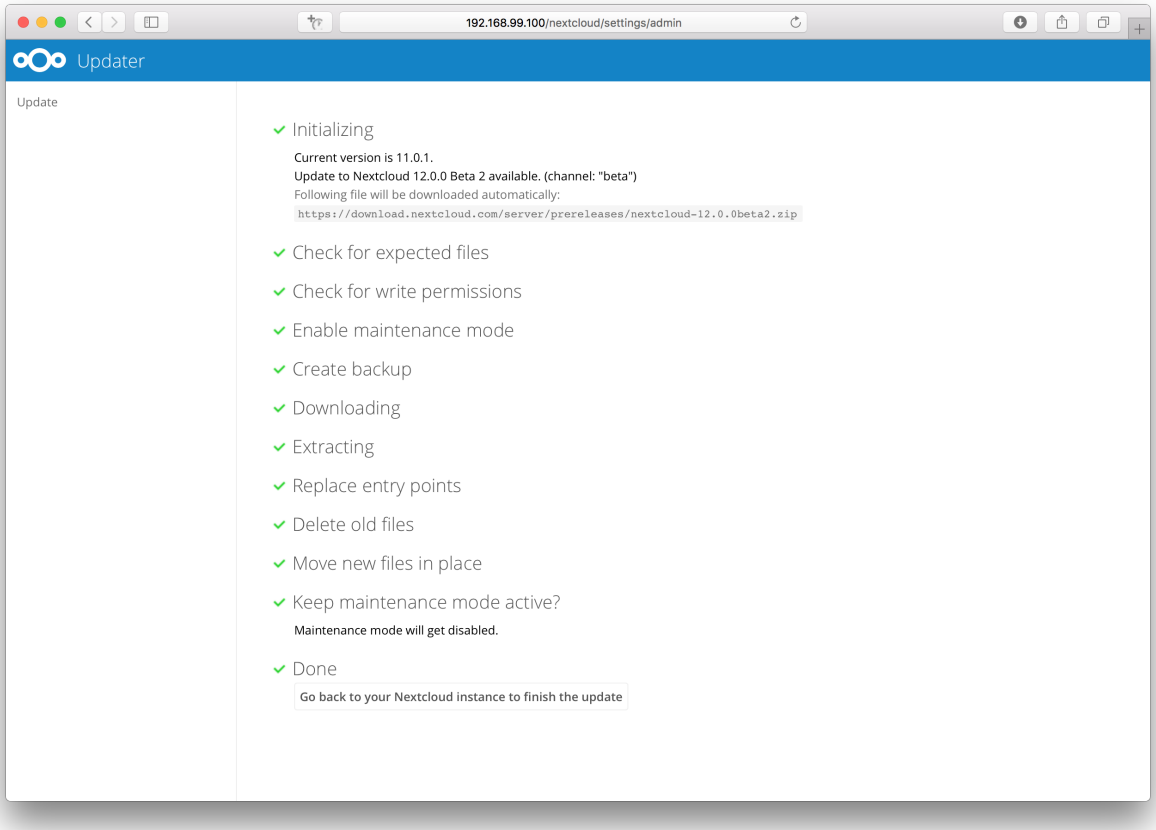

6. Once all steps are executed the updater will ask you a final question: "Keep maintenance mode active?". This allows you to use either the web based upgrade page or the command line based upgrade procedure (`occ upgrade`). Command line access is required if the maintenance mode is kept active.

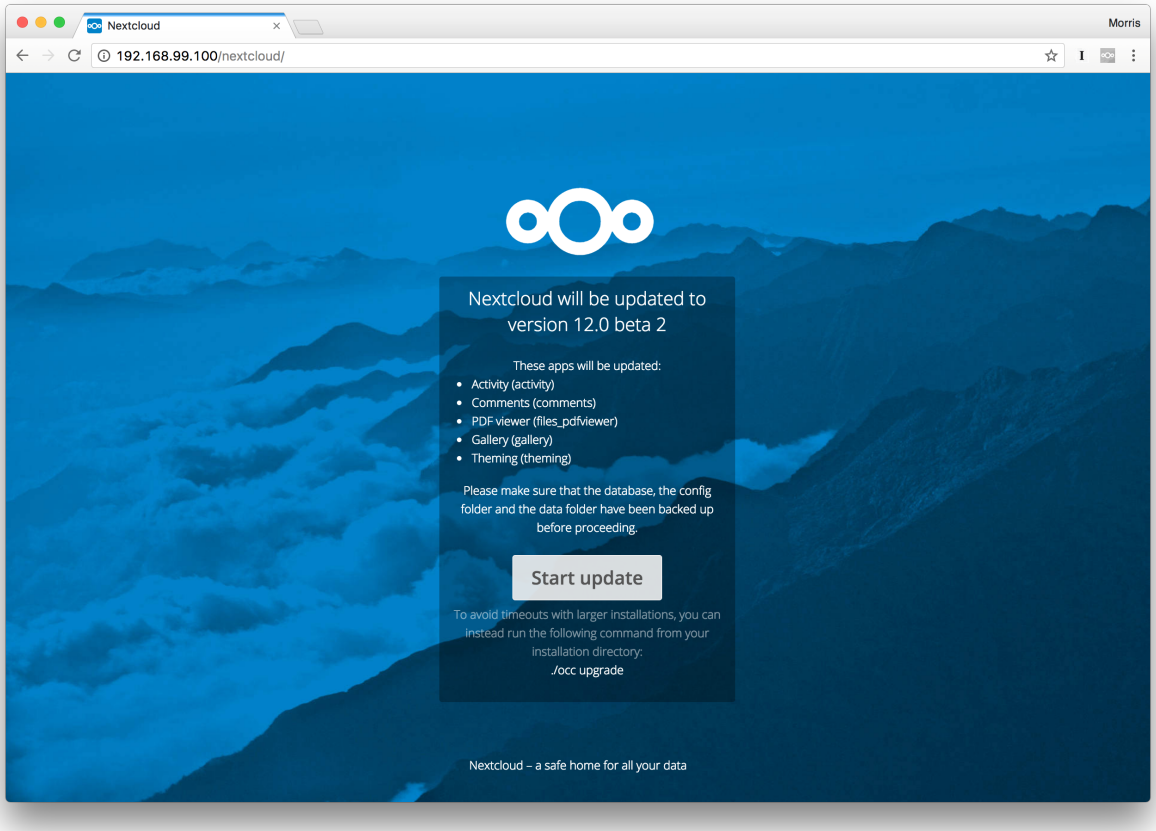



7. Done. You now can continue either to the web based upgrade page or run `occ upgrade`. The two examples "Web based upgrade" and "Command line based upgrade" shows how the screens then look like.

**Web based upgrade**

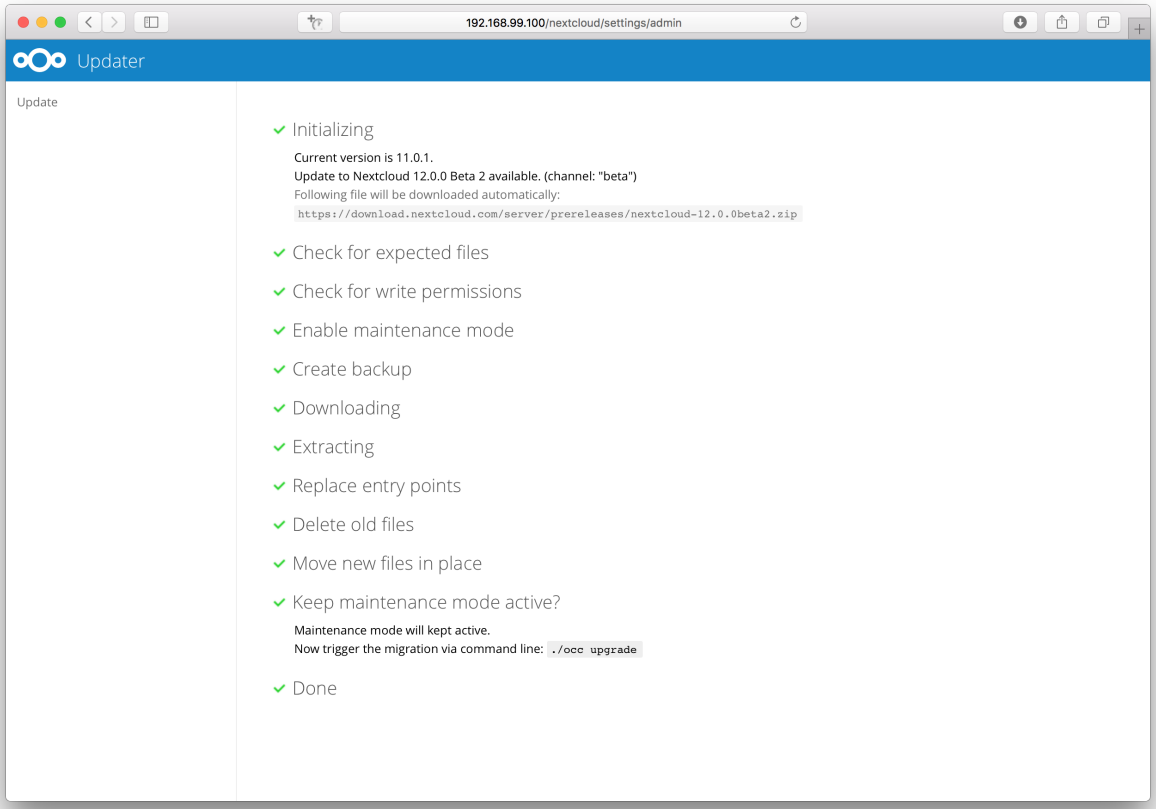
This is how the web based update would continue:


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:



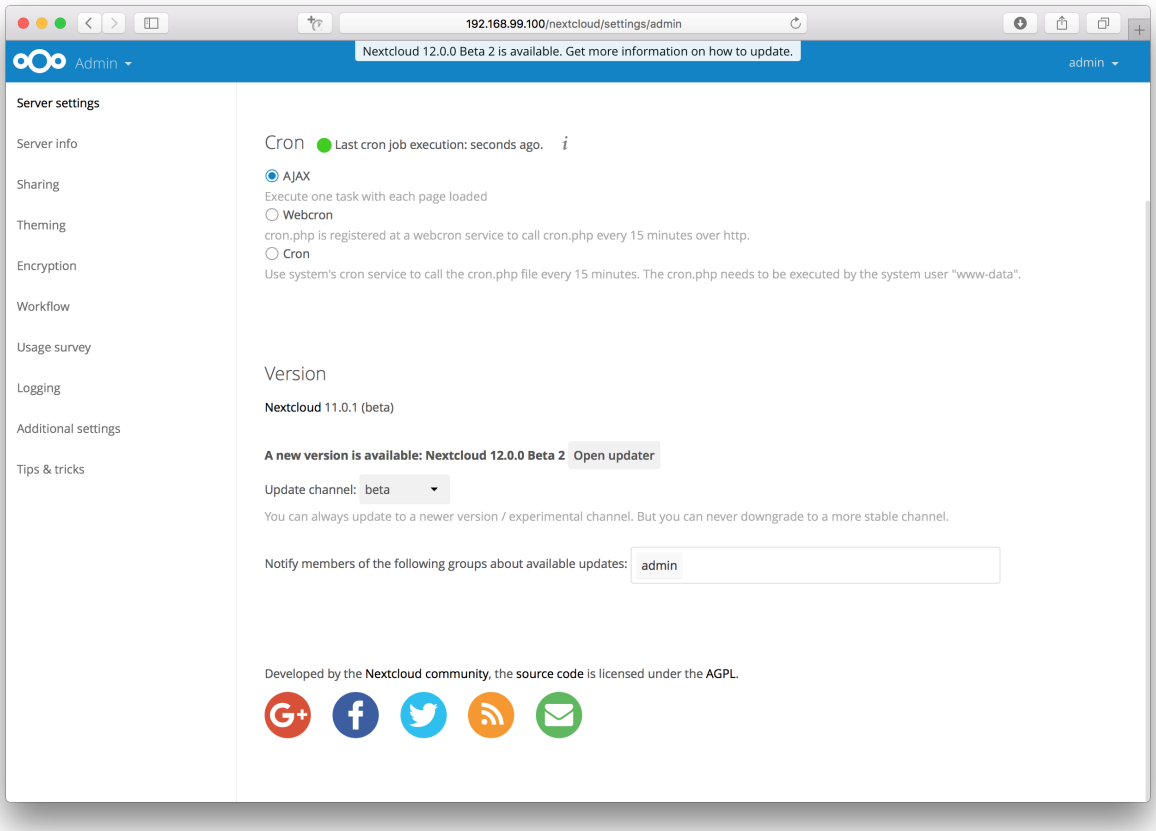

```
$ sudo -u www-data php ./occ upgrade
Nextcloud or one of the apps require upgrade - only a limited number of commands are available
You may use your browser or the occ upgrade command to do the upgrade
Set log level to debug
Updating database schema
Updated database
Updating <files_pdfviewer> ...
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.

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 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 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

3. Stop your Web server.
4. Rename your current Nextcloud directory, for example `nextcloud-old`.
5. Unpacking the new archive creates a new `nextcloud` directory populated with your new server files. Copy 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`.
6. Copy the `config.php` file from your old Nextcloud directory to your new Nextcloud directory.

7. 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.
8. If you are using 3rd party applications, look in your new `nextcloud/apps/` directory to see if they are there. If not, copy them from your old `apps/` directory to your new one. Make sure the directory permissions of your third party application directories are the same as for the other ones.
9. 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 {} \;
```

10. Restart your Web server.
11. Now launch the upgrade from the command line using `occ`, like this example on Ubuntu Linux:

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

12. 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.

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

When upgrading Nextcloud and you are running MySQL or MariaDB with binary logging enabled, your upgrade may fail with these errors in your MySQL/MariaDB log:

```
An unhandled exception has been thrown:
exception 'PDOException' with message 'SQLSTATE[HY000]: General error: 1665
Cannot execute statement: impossible to write to binary log since
BINLOG_FORMAT = STATEMENT and at least one table uses a storage engine limited
to row-based logging. InnoDB is limited to row-logging when transaction
isolation level is READ COMMITTED or READ UNCOMMITTED.'
```

Please refer to *MySQL / MariaDB with Binary Logging Enabled* on how to correctly configure your environment.

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 http://snapcraft.io/71 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.

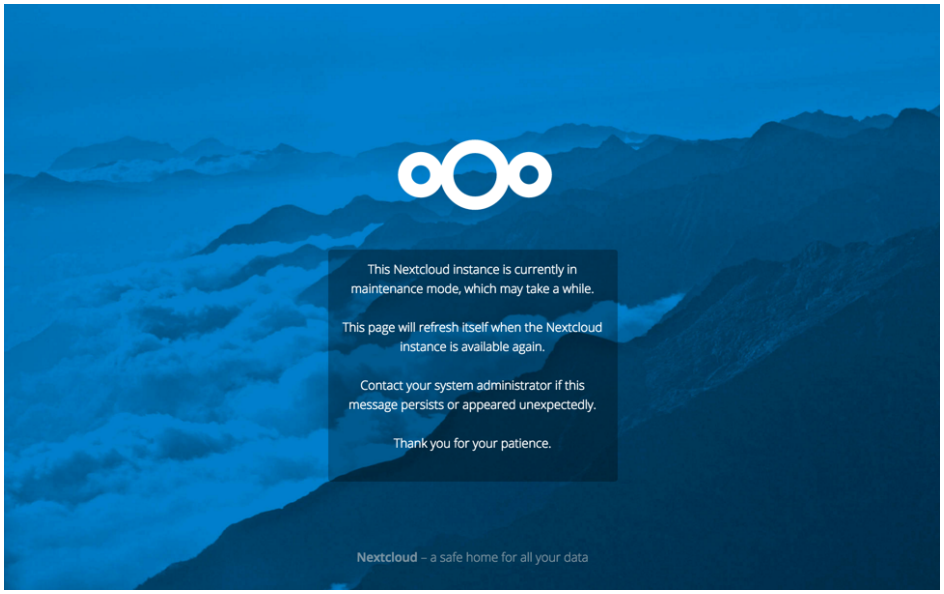
- Make a fresh backup.
- Upgrade your Nextcloud snap: sudo snap refresh nextcloud
- Run *occ upgrade*.
- Take your Nextcloud server out of *maintenance mode*.
- Re-enable third-party apps.

## 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 an 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 adopt the paths in the Nextcloud config.php file. Note: This step might take several hours, depending on your installation.
5. 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.
6. Change the `CNAME` entry in the DNS to point your users to the new location.

## Enabling MySQL 4-byte support

**Note:** This feature is currently **experimental**.

In order to use Emojis (textbased smilies) on your Nextcloud server with a MySQL database, the installation needs to be tweaked a bit.

1. Update your Nextcloud server to Nextcloud 11 or later.
2. Make sure the following InnoDB settings are set on your MySQL server:

```
[mysqld]
innodb_large_prefix=1
innodb_file_format=barracuda
innodb_file_per_table=1
```

3. Restart the MySQL server in case you changed the configuration in step 2.
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 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 occ maintenance:repair
```

Now you should be able to use Emojis in your file names, calendar events, comments and many more.

## MariaDB support

**Note:** This is even more **experimental**.

1. Follow MySQL steps 1, 2 and 3
2. Figure out whether the file formate was changed to Barracuda:

```
MariaDB> SELECT NAME, SPACE, FILE_FORMAT FROM INFORMATION_SCHEMA.INNODB_SYS_TABLES WHERE NAME li
```

If the file format is "Barracuda" for every single table, nothing is left to do. Continue with the MySQL instructions. While testing, all tables' file format was "Antelope".

3. The tables needs to be migrated to "Barracuda" manually, one by one. SQL commands can be created easily, however:

```
MariaDB> USE INFORMATION_SCHEMA;
MariaDB> SELECT CONCAT("ALTER TABLE `", TABLE_SCHEMA,"`.`", TABLE_NAME, "` ROW_FORMAT=DYNAMIC;")
```

This will return an SQL command for each table in the nextcloud database. The rows can be quickly copied into a text editor, the "l"s replaced and the SQL commands copied back to the MariaDB shell. If no error appeared (in doubt check step 2) all is done and nothing is left to do here. It can be proceded with the MySQL steps.

4. It is possible, however, that some tables cannot be altered. The operations fails with: "ERROR 1478 (HY000): Table storage engine 'InnoDB' does not support the create option 'ROW_FORMAT"'. In that case the failing tables have a SPACE value of 0 in step 2. It basically means that the table does not have an index file of its own, which is required for the Barracuda format. This can be solved with a slightly different SQL command:

```
MariaDB> ALTER TABLE `nextcloud`.`oc_tablename` ROW_FORMAT=DYNAMIC, ALGORITHM=COPY;
```

Replace oc_tablename with the failing table. If there are too many (did not happen here), SQL commands can be generated in a batch (task for the reader).

Now everything should be fine and the MySQL instructions should be proceeded.

CHAPTER

NINE

# OPERATIONS

Advanced operation including monitoring, scaling across multiple machines, and creating a custom theme for your Nextcloud server.

## Considerations on Monitoring

Large scale Nextcloud deployments are typically installed as load balanced n-tier web applications. Successfully managing such an installation requires active monitoring of the application and supporting infrastructure components. The purpose of this section is to outline the components of Nextcloud that need to be monitored, and provide guidance on what to look for in Nextcloud in an enterprise installation.

### Nextcloud Deployment Architecture

Before discussing how to monitor Nextcloud, it is important to understand the architecture of a typical Nextcloud deployment. These monitoring best practices are developed based on the use of load balanced Web servers, a clustered database running a distributed database storage engine, such as MySQL NDB, and a clustered filesystem, such as Red Hat Storage.

It is assumed that specific enterprise tools (monitoring, log management, etc) to monitor operations are available, and that Nextcloud is simply a new target for these tools.

### The Important Components of Nextcloud

Nextcloud is a PHP application that depends on a filesystem for file storage, and a database for storing user and file meta data, as well as some application specific information. While the loss of an app server or a node in the database or storage clusters should not bring the system down, knowing that this happened and resolving it is essential to keeping the service running efficiently. Therefore it is important to monitor the Nextcloud servers, the Load Balancer, the Storage Cluster and the Database. This documentation starts with the Nextcloud application and works out from there through the layers of infrastructure.

#### Status.php

Nextcloud provides a very simple mechanism for determining if an application server is up and functioning – call the status.php file on each Nextcloud server. This file can be found in the root Nextcloud directory on the server, which by default is /status.php. If the server is functioning normally, the response looks something like this:

```
{"installed":"true","version":"6.0.0.16","versionstring":"6.0.1","edition":""}
```

We recommend monitoring this file on each Nextcloud application server to provide a basic check that the server is operating properly.

#### Nextcloud.log

Nextcloud also provides a built in logging function. If the Nextcloud logging application is enabled, this file will track user logins and shared file activity. If these logging applications are not enabled, this log file still tracks basic Nextcloud health. Given the potential for this file to get quite large, the log file should be rotated on a daily basis, and given the importance of the error information in the log file, this should be integrated with an enterprise log manager.

Logfile entries that start with the keyword "Error" should be logged and reported to Nextcloud support.

#### Apache

The apache error and access log should also be monitored. Significant spontaneous changes of the number of requests per second should also be monitored and looked into.

#### Database server

The load and general health of the database server or cluster has to be monitored also. All mysql vendors provide tools to monitor this.

### Clustered Filesystem

The available space of the filesystem should be monitored to prevent a full Nextcloud. This functionality is provided by the operating-system and/or the cluster filesystem vendor.

### Load Balancer

The load balancer is monitoring the health of the application servers and is distributing the traffic in the optimal way. The application-servers should also be monitored to detect long lasting OS or hardware problems. Monitoring solutions like Nagios provide built in functionality to do this.

## Scaling Across Multiple Machines

This document will cover the reference architecture for the Nextcloud Scale Out model for a single datacenter implementation. The document will focus on the three main elements of an Nextcloud deployment:

- Application layer
- Database Layer
- Storage Layer

At each layer the goal is to provide the ability to scale, and providing a high availability while maintaining the needed level of performance.

## Application Layer

For the application layer of this reference architecture we used Oracle Enterprise Linux as the front end servers to host the Nextcloud code. In this instance we made `httpd` a permissive domain, allowing it to operate within the SELinux environment. In this example we also used the standard directory structure placing the Nextcloud code in the Apache root directory. The following components were installed on each application server:

- Apache
- PHP 5.6.x
- PHP-GD
- PHP-XML
- PHP-MYSQL
- PHP-CURL

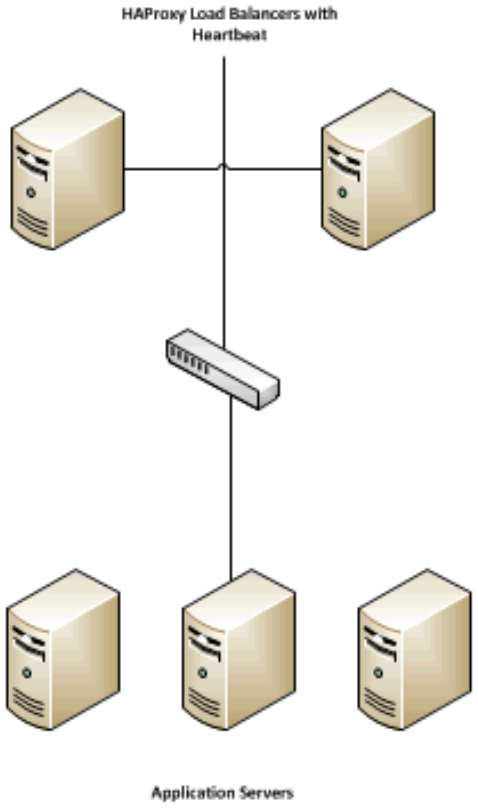
Also required is the PHP smbclient module or smbclient (see SMB/CIFS).

It is also worth mentioning that the appropriate exceptions where made in the firewall to allow the Nextcloud traffic (for the purpose of testing we enable both encrypted SSL via port 443 and unencrypted via port 80).

The next step was to generate and import the needed SSL certificates following the standard process in the OEL documentation.

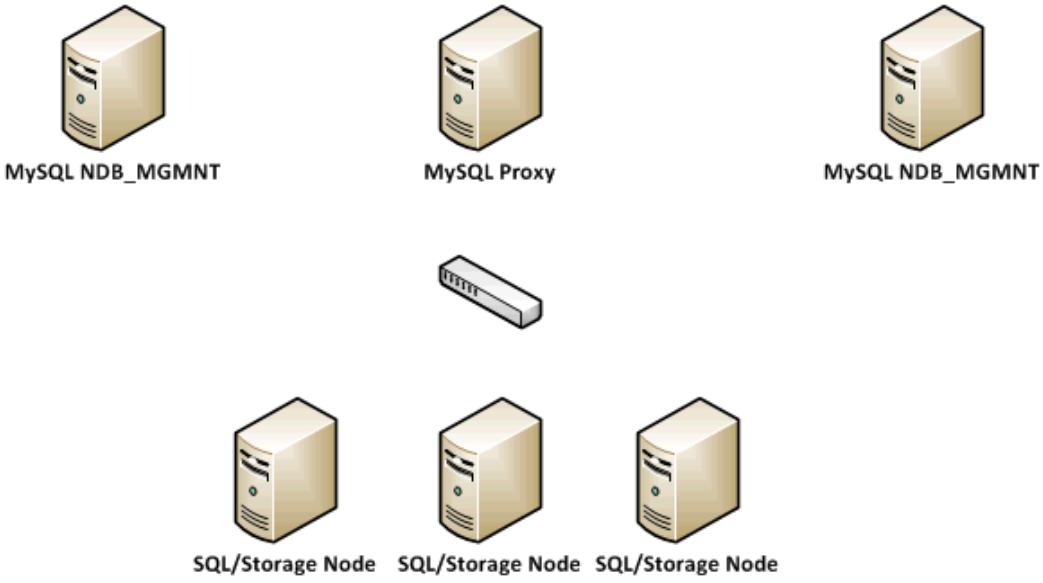
The next step is to create a scalable environment at the application layer, which introduces the load balancer. Because the application servers here are stateless, simply taking the configuration above and replicating (once configured with storage and database connections) and placing behind a load balancer will provide a scalable and highly available environment. For this purpose we chose HAProxy and configured it for HTTPS traffic following the documentation found here http://haproxy.1wt.eu/#doc1.5

It is worth noting that this particular load balancer is not required, and the use of any commercial load balancer (i.e. F5) will work here. It is also worth noting that the HAProxy servers were setup with a heartbeat and IP Shift to failover the IP address should one fail.




## Database Layer

For the purposes of this example, we have chosen a MySQL cluster using the NDB Storage engine. The cluster was configured based on the documentation found here http://dev.mysql.com/doc/refman/5.1/en/mysql-cluster.html with a sample looking like this:



Taking a closer look at the database architecture, we have created redundant MySQL NDB Management nodes for redundancy and we have configured 3 NDB SQL/Storage nodes across which we are able to spread the database traffic. All of the clients (Nextcloud Application Servers) will connect to the database via the My SQL Proxy. It is worth noting that MySQL proxy is still in beta and that using another load balancing method like HAProxy or F5 is supported, in that you will be distributing traffic among the various SQL/Storage nodes. Here, we simply swap out the MySQL Proxy for a properly configured HAProxy giving us the following:

Note: UP with load-balancing disabled is reported as "NOLB".

mysql-cluster

| | Queue | | | Session rate | | | Sessions | | | | | Bytes | | Denied | | Errors | | | Warnings | | Server | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Cur | Max | Limit | Cur | Max | Limit | Cur | Max | Limit | Total | LbTot | In | Out | Req | Resp | Req | Conn | Resp | Retr | Redis | Status | LastChk | Wght | Act | Bck | Chk | Dwn | Dwntme | Thrtle |
| Frontend | | | | 0 | 12 | - | 0 | 3 | 2 000 | 5 512 | | 32 079 803 | 65 469 296 | 0 | 0 | 0 | | | | | OPEN | | | | | | | | |
| mysql-1 | 0 | 0 | - | 0 | 4 | | 0 | 2 | - | 1 810 | 1 810 | 10 615 863 | 21 656 848 | | 0 | | 0 | 0 | 0 | 0 | 4d17h UP | L7OK/0 in 1ms | 1 | Y | - | 1 | 1 | 18d22h | - |
| mysql-2 | 0 | 0 | - | 0 | 4 | | 0 | 2 | - | 1 853 | 1 850 | 10 780 136 | 22 007 349 | | 0 | | 0 | 0 | 3 | 0 | 13d46m UP | L7OK/0 in 1ms | 1 | Y | - | 66482 | 8737 | 1d18h | - |
| mysql-3 | 0 | 0 | - | 0 | 4 | | 0 | 2 | - | 1 851 | 1 850 | 10 683 804 | 21 805 099 | | 0 | | 0 | 0 | 1 | 0 | 13d47m UP | L7OK/0 in 1ms | 1 | Y | - | 24878 | 3001 | 10h40m | - |
| Backend | 0 | 0 | | 0 | 12 | | 0 | 3 | 2 000 | 5 512 | 5 510 | 32 079 803 | 65 469 296 | 0 | 0 | | 1 | 0 | 4 | 0 | 13d50m UP | | 3 | 3 | 0 | | 361 | 1h22m | |

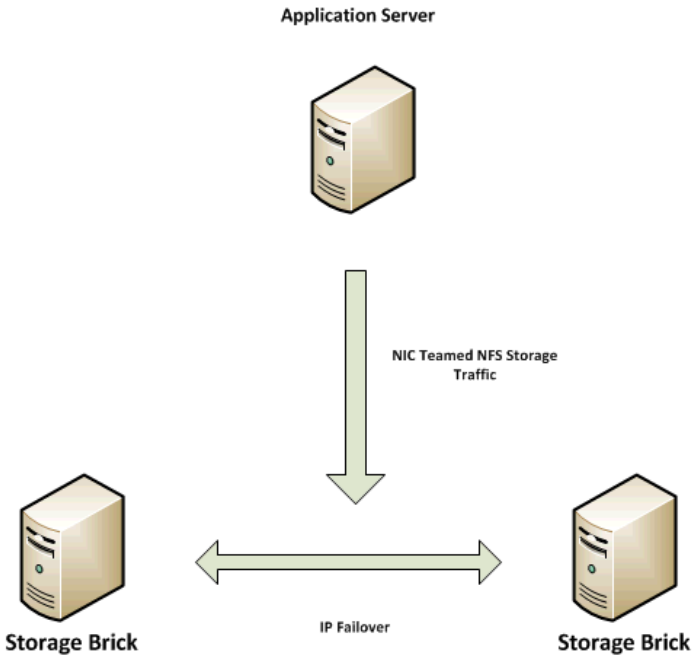
In this example we have also added a second HAProxy server with Heartbeat to prevent any single point of failure. We have also implemented NIC bonding to load balance the traffic across multiple physical NICs.

## Storage Layer

Storage was deployed using the Red Hat Storage server with the GlusterFS (pre-configured as part of the Red Hat Storage Server offering).

The Red Hat Storage Servers where configured based on documentation found here https://access.redhat.com/site/documentation/en-US/Red_Hat_Storage/2.0/html/Administration_Guide/Admin_Guide_Part_1.html

For the purposes of scale and high availability we configured a distributed replicated volume with IP Failover. The storage was configured on a separate subnet with bonded NICs at the application server level. We have chosen to address the storage using NFS, and for high availability we have chosen to implement IP Failover of the storage as documented here https://access.redhat.com/site/documentation/en-US/Red_Hat_Storage/2.0/html/Administration_Guide/ch09s04.html

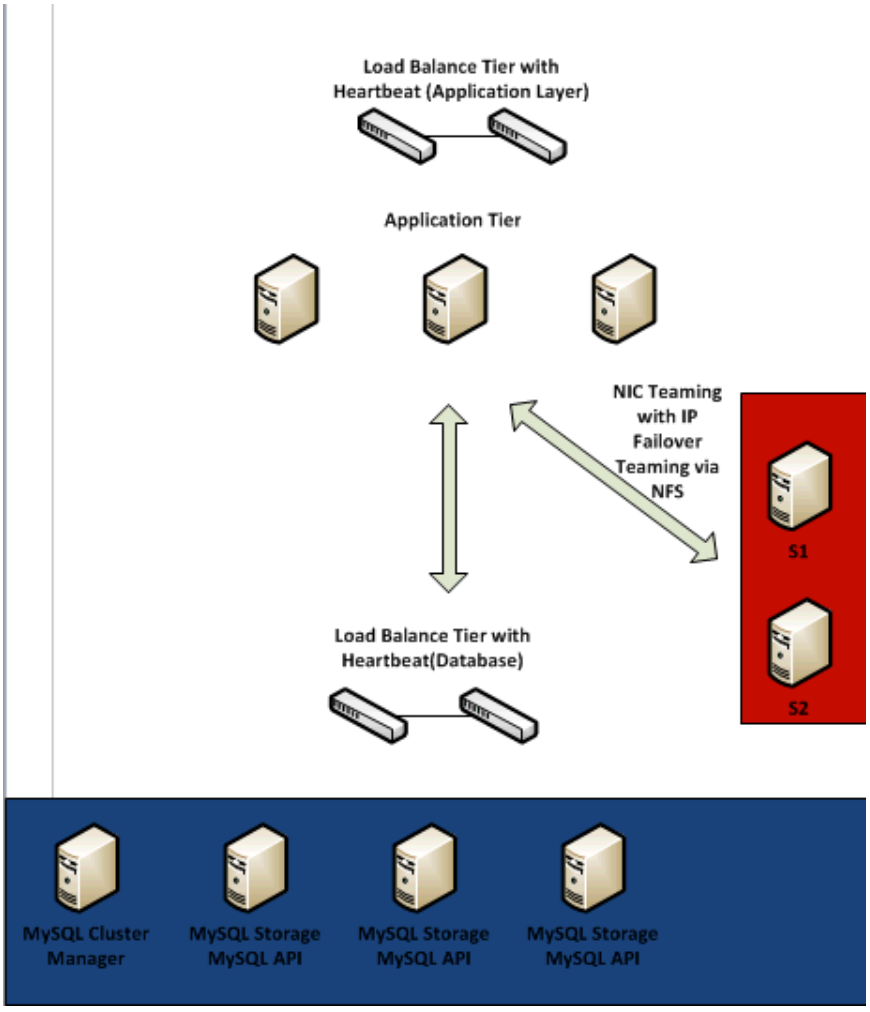



We chose to deploy the storage in this fashion to address both HA and extensibility of the storage, along with managing performance by simply adding additional bricks to the storage volume, back-ended by additional physical disk.

It is worth noting that several options are available for storage configuration (such as striped replicated volumes). A discussion around the type of IO performance required and the needed HA configuration needs to take place to understand the best possible option here.

If we add up the parts, we have the following:

- An application layer that supports dynamic expansion through the addition of additional servers and provides HA behind a load balancer
- A database layer that can also be scaled through the addition of additional SQL/Storage nodes and will provide HA behind a load balancer
- A storage layer that can dynamically expand to meet storage needs, will scale based on backend hardware, and provides HA via IP Failover




## Theming Nextcloud

Theming can be done very easily using the shipped theming app, which is enabled by default.

For more individual theming options please head over to the developers documentation.

CHAPTER

TEN

# 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
- The Nextcloud forums
- The Nextcloud IRC chat channel `irc://#nextcloud@freenode.net` on freenode.net, also accessible via webchat

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 Configuration 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. For Firefox we recommend to installing the Firebug extension.

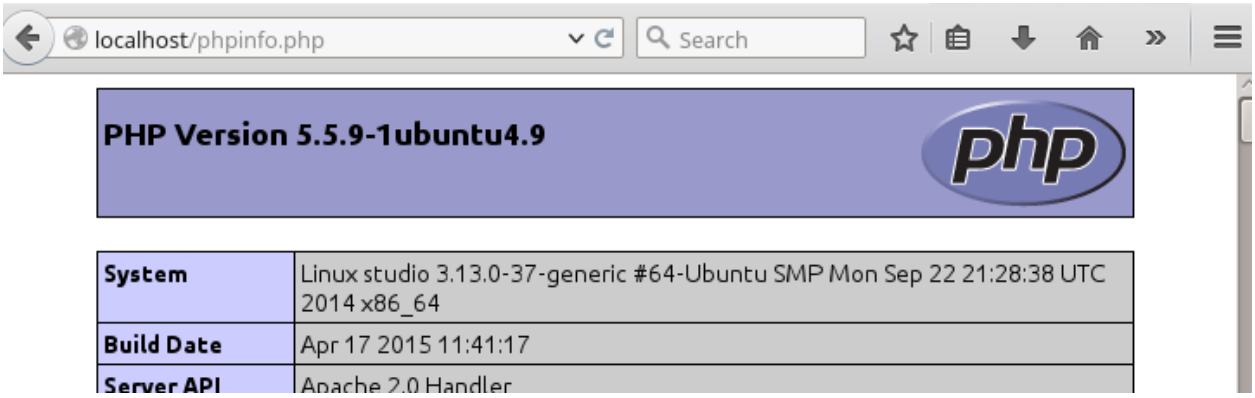
**Note:** The logfile of Nextcloud is located in the data directory `nextcloud/data/nextcloud.log`.

### 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 behaviours 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. Information on how to correctly configure your environment can be found at the forums.

## 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/php5/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-php5filter (use libapache2-mod-php5 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.

## Troubleshooting Contacts & Calendar

### Service discovery

Some clients - especially on iOS/Mac OS X - have problems finding the proper sync URL, even when explicitly configured to use it.

If you want to use CalDAV or CardDAV clients 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 DAV endpoints. If running Nextcloud at the document root of your Web server the correct URL is:

`https://example.com/remote.php/dav`

and if running in a subfolder like `nextcloud`:

`https://example.com/nextcloud/remote.php/dav`

For the first case the `.htaccess` file shipped with Nextcloud should do this work for your when running Apache. You only need to make sure that your Web server is using this file. When running NGINX please refer to Nginx Example Configurations.

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:

```
Redirect 301 /.well-known/carddav /nextcloud/remote.php/dav
Redirect 301 /.well-known/caldav /nextcloud/remote.php/dav
```

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.

### 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.

## 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.

# 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.

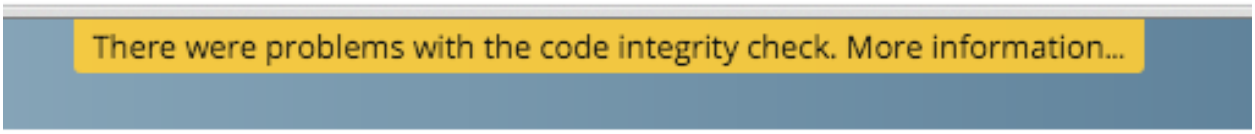
For custom distributions of Nextcloud it is recommended to change the release version branch in version.php to something else than "stable".

### Is Code Signing Mandatory For Apps?

Code signing is optional for all third-party applications. Applications with a tag of "Official" on apps.owncloud.com require code signing.

## Fixing Invalid Code Integrity Messages

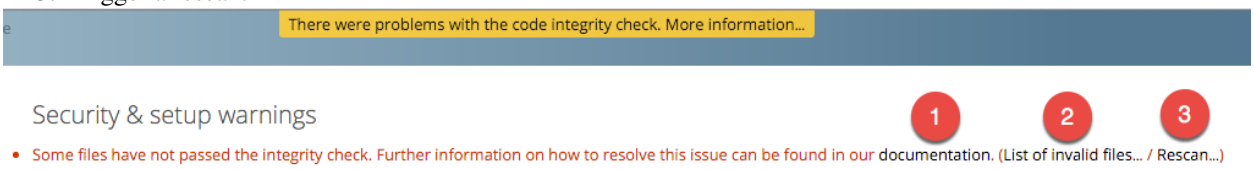
A code integrity error message ("There were problems with the code integrity check. More information…") appears in a yellow banner at the top of your Nextcloud Web interface:




**Note:** The yellow banner is only shown for admin users.

Clicking on this link will take you to your Nextcloud admin page, 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`.