Nextcloud Server Administration Manual

Release 36

The Nextcloud developers

Sep 17, 2026

CONTENTS

1 Introduction **1**
- 1.1 Target Audience . . . 1
- 1.2 Core Components . . . 1
- 1.3 Editions . . . 2
- 1.4 Further Resources . . . 2
- 1.5 Getting Started . . . 2

2 Maintenance and release schedule **3**
- 2.1 Overview . . . 3
- 2.2 Release types . . . 3
- 2.3 Release schedule . . . 5
- 2.4 Installation version . . . 5
- 2.5 Release channels . . . 6
- 2.6 Major version upgrades . . . 6
- 2.7 Beta releases and Release candidates . . . 7
- 2.8 Downgrading . . . 7
- 2.9 Bug reporting . . . 7

3 GDPR compliance **9**
- 3.1 Personal data stored . . . 9
- 3.2 Responding to data subject requests . . . 11
- 3.3 Data retention . . . 13
- 3.4 Helpful apps . . . 16
- 3.5 Cookies . . . 18

4 Declarations **21**
- 4.1 Energy consumption . . . 21

5 Release notes **23**
- 5.1 Critical changes . . . 23
- 5.2 Changelog . . . 30

6 Installation and server configuration **31**
- 6.1 System requirements . . . 31
- 6.2 Deployment recommendations . . . 34
- 6.3 Preparing PHP . . . 34
- 6.4 Installation on Linux . . . 39
- 6.5 Installation wizard . . . 49
- 6.6 Install from the command line . . . 54
- 6.7 Automated installation with `autoconfig.php` . . . 56
- 6.8 SELinux configuration . . . 60

6.9 NGINX configuration . . . 63
6.10 Hardening and security guidance . . . 76
6.11 Server tuning . . . 85
6.12 Example installation on Ubuntu 24.04 LTS . . . 91
6.13 Example installation on CentOS 8 . . . 93
6.14 Example installation on OpenBSD . . . 97
6.15 Uninstallation . . . 101

7 Database setup and configuration **103**
7.1 Set up a database server . . . 103
7.2 Enabling MySQL/MariaDB 4-byte support . . . 110
7.3 Convert to a different database type . . . 111
7.4 Prevent identifier exhaustion in large databases . . . 114
7.5 Using database read replicas . . . 114
7.6 Splitting tables or apps across databases . . . 115

8 Nextcloud configuration **119**
8.1 Warnings on admin page . . . 119
8.2 Configuration Parameters . . . 121
8.3 Activity app . . . 185
8.4 Administration privileges (Delegation) . . . 189
8.5 Android Deep Link Handling . . . 190
8.6 Antivirus scanner . . . 191
8.7 Background jobs . . . 198
8.8 Brute force protection . . . 201
8.9 Memory caching . . . 205
8.10 Dashboard app . . . 214
8.11 Domain Change . . . 215
8.12 Email . . . 215
8.13 Linking external sites . . . 223
8.14 Language & Locale . . . 226
8.15 Logging . . . 227
8.16 OAuth2 . . . 234
8.17 Reverse proxy . . . 235
8.18 Text app . . . 239
8.19 Theming . . . 240

9 Using the occ command **245**
9.1 Running occ . . . 245
9.2 Enabling autocompletion . . . 248
9.3 Limitations in maintenance mode . . . 248
9.4 Debugging . . . 249
9.5 Command reference . . . 249

10 Reference management **323**
10.1 Link previews . . . 323
10.2 The Smart Picker . . . 324

11 Webhook Listeners **327**
11.1 Introduction . . . 327
11.2 Overview . . . 327
11.3 Installation . . . 327
11.4 Listening to events . . . 327
11.5 Nextcloud Webhook Events . . . 330

12 Windmill Workflows **337**
12.1 Installation . . . 337
12.2 Setting up the workspace connection . . . 337
12.3 Building a workflow . . . 338
12.4 Nextcloud Scripts . . . 339
12.5 FAQ . . . 342

13 File sharing and management **343**
13.1 File Sharing . . . 343
13.2 Configuring Federation Sharing . . . 348
13.3 Uploading big files > 512MB . . . 353
13.4 Providing default files . . . 357
13.5 Configuring Object Storage as Primary Storage . . . 358
13.6 External Storage . . . 365
13.7 External Storage authentication mechanisms . . . 379
13.8 Server-side Encryption . . . 380
13.9 Server-side encryption details . . . 390
13.10 Server-side encryption migration . . . 398
13.11 Transactional file locking . . . 398
13.12 Previews configuration . . . 399
13.13 Controlling file versions and aging . . . 402
13.14 Deleted Files . . . 403
13.15 File conversion . . . 407
13.16 Windows compatible filenames . . . 408

14 Flow **411**
14.1 Flow configuration . . . 411
14.2 Files access control . . . 411
14.3 Automated tagging of files . . . 414
14.4 Retention of files . . . 415

15 Mimetypes management **417**
15.1 Mimetype aliases . . . 417
15.2 Mimetype mapping . . . 418
15.3 Icon retrieval . . . 418

16 Apps management **419**
16.1 Apps . . . 419
16.2 Managing apps . . . 420

17 Apps management API **425**
17.1 Get list of apps . . . 425
17.2 Get app info . . . 426
17.3 Enable an app . . . 427
17.4 Disable an app . . . 427

18 ExApps management **429**
18.1 AppAPI and External Apps . . . 429
18.2 Deployment configurations . . . 433
18.3 Managing Deploy Daemons . . . 447
18.4 Test Deploy Daemon . . . 450
18.5 Managing ExApps . . . 454
18.6 Advanced Deploy Options . . . 456

19 Artificial Intelligence **459**

19.1 Overview . . . 459
19.2 Nextcloud Assistant . . . 468
19.3 App: Local Machine translation 2 (translate2) . . . 473
19.4 App: Local large language model (llm2) . . . 474
19.5 App: Local Whisper Speech-To-Text (stt_whisper2) . . . 478
19.6 App: Local Image Generation (text2image_stablediffusion2) . . . 480
19.7 App: Local Image Generation: Flux (text2image_flux) . . . 481
19.8 App: Recognize . . . 483
19.9 App: Context Chat . . . 486
19.10 App: Context Agent (context_agent) . . . 491
19.11 App: Summary Bot (Talk chat summarize bot) . . . 500
19.12 App: Local Text-To-Speech (text2speech_kokoro) . . . 503
19.13 App: Live Transcription and Translation in Nextcloud Talk (live_transcription) . . . 504
19.14 App: Local Optical Character Recognition (ocr_paddle) . . . 507
19.15 AI as a Service . . . 508
19.16 Insight and debugging . . . 509
19.17 Legal: Compliance with EU AI Act . . . 511

20 User management **513**
20.1 User management . . . 513
20.2 Resetting a lost admin password . . . 521
20.3 Resetting a user password . . . 521
20.4 User password policy . . . 521
20.5 Authentication . . . 522
20.6 Two-factor authentication . . . 522
20.7 User authentication with LDAP . . . 526
20.8 LDAP user cleanup . . . 547
20.9 The LDAP configuration API . . . 548
20.10 User provisioning API . . . 553
20.11 Profiles . . . 568
20.12 User authentication with OpenID Connect . . . 576

21 Desktop Clients **577**
21.1 Desktop Client Deployment and Setup . . . 577
21.2 Block desktop client access at the HTTP layer . . . 584
21.3 Troubleshooting . . . 588

22 Groupware **595**
22.1 Calendar / CalDAV . . . 595
22.2 Contacts / CardDAV . . . 600
22.3 Contacts Interaction . . . 603
22.4 Mail . . . 604
22.5 Out-of-office feature . . . 615
22.6 Troubleshooting . . . 616

23 Office **623**
23.1 Nextcloud Office . . . 623
23.2 Collabora Online . . . 638

24 Collectives **647**
24.1 Runtime Dependencies . . . 647
24.2 Collectives and *group_everyone* . . . 647
24.3 Collectives and guest users . . . 647
24.4 Public shares . . . 647
24.5 Configuration . . . 647

24.6 Allow for groups in your collectives . . . 648
24.7 Importing existing data . . . 648

25 Monitoring **649**
25.1 OpenMetrics . . . 649

26 Maintenance **651**
26.1 Backup . . . 651
26.2 Restoring backup . . . 652
26.3 How to upgrade . . . 655
26.4 Upgrade via built-in updater . . . 657
26.5 Upgrade manually . . . 673
26.6 Upgrade via snap packages . . . 675
26.7 Migrating to a different server . . . 676
26.8 Migrating from ownCloud . . . 679

27 Issues and troubleshooting **681**
27.1 General troubleshooting . . . 681
27.2 Patching Nextcloud . . . 689
27.3 Code signing . . . 690

CHAPTER

ONE

INTRODUCTION

Welcome to the Nextcloud Server Administration Guide.

This guide explains how to perform administrative tasks in Nextcloud, a highly versatile and scalable open-source platform for file synchronization and content collaboration.

With over 400,000 deployments, Nextcloud can run on a simple two-user Raspberry Pi or scale to support global, distributed installations serving tens of millions of users. It can be deployed on-premises, in private or public clouds, or in hybrid environments. Supported by a large and growing community, Nextcloud is available in more than 60 languages.

The latest editions of the Nextcloud manuals are always available online at docs.nextcloud.com.

1.1 Target Audience

This guide is for users who want to:

- Install Nextcloud Server
- Administer and manage their instance
- Optimize server performance

For documentation on Nextcloud web, desktop, or mobile clients, see:

- Nextcloud User Manual
- Nextcloud Desktop Client

For documentation on development topics, see:

- The individual repositories on GitHub within the @nextcloud organization
- Nextcloud Development Manual

1.2 Core Components

Nextcloud includes:

- Nextcloud Server (backend running on Linux)
- A responsive, integrated web client
- Cross-platform desktop clients (Windows, macOS, and Linux) for file synchronization and local access
- Dedicated mobile clients (Android and iOS)
- An extensive app ecosystem for expanded functionality

1.3 Editions

Nextcloud Server is available in two editions:

- **Community:** Community-supported (peer-to-peer help), 100% free.
- **Enterprise:** Supported by core developers or authorized partners, with official packaging, extensive enterprise-specific documentation, and support options.

Both editions include all functionality and source code.

Enterprise editions can include deployment guidance, phone and email access to Nextcloud developers, official support for integrations and add-ons, custom branding, and extended support cycles, among other benefits.

This guide primarily focuses on the Community edition, but the information applies to both editions.

1.4 Further Resources

- Video tutorials, overviews, and conference presentations: Nextcloud YouTube channel.
- Latest news and updates: Nextcloud blog.
- Community support: Nextcloud Help Forum.
- Commercial support: Nextcloud GmbH.
- Documentation: Nextcloud Documentation.

1.5 Getting Started

To get started, see the Installation section.

Who develops Nextcloud?

The development of the Nextcloud platform is a cooperative endeavor overseen by the core maintainers – primarily employed by Nextcloud GmbH – along with thousands of partners, providers, collaborators, project members, and community participants.

The Nextcloud community includes everyone – from individual users and independent developers to large organizations.

Community members share and discuss their experiences with the platform every day. They suggest improvements, collaborate on design, write code, create documentation, test for bugs, triage bug reports and enhancement ideas, support others, improve translations, and help fund and organize the logistics of a project this size. Most importantly, they use Nextcloud in their daily lives and work.

CHAPTER

TWO

MAINTENANCE AND RELEASE SCHEDULE

2.1 Overview

Nextcloud releases multiple major versions *throughout* the year, but maintains support for *each* major version for one full year each through "lighter" maintenance updates (and regularly backporting applicable security and bug fixes). This permits a high velocity development cadence, while still giving administrators flexibility when planning deployments, upgrades, and maintenance activities.

A detailed schedule for upcoming major and maintenance releases (as well as end-of-life projections) is regularly updated to facilitate planning deployment, testing, and upgrade planning.

Whether you want the latest features and optimizations, want to help with testing, or just want to wait until everything is perfectly ready to go, you've got options with regards to which version of Nextcloud Server to initially deploy as well as how frequently to do major upgrades.

Danger

We always recommend installing the latest **maintenance** releases as soon as possible, regardless of which major version of Nextcloud Server you use. And we also always highly recommend upgrading from **end-of-life** releases as soon as possible.

Tip

Extended maintenance and additional support is available through subscriptions options for enterprise support offered by Nextcloud developers through Nextcloud GmbH.

2.2 Release types

Nextcloud has two types of releases in the default release channel:

1. Major releases
2. Maintenance releases

Major releases of Nextcloud Server (e.g. `28.X.X`) introduce new features and functionality.

Every major release is, in turn, supported for *one year* via periodic **maintenance** releases (e.g. `X.X.4`), which correct critical bugs and security vulnerabilities.

2.2.1 Major releases

Major releases usually introduce new features and often also include changes "under the hood". These changes may be extensive.

A specific major release is indicated by the first part of the version string. For example, Nextcloud Server `28.0.4` is major release `28`. And `27.1.7` is major release `27`.

Tip

The highest numbered major release offers the latest features. While the lowest numbered major release offers the most time in the field.

Note

You may need to meet new system requirements before the Updater will offer you a new major version. Even if offered, there may be other changes required that the Updater cannot check for fully. We try to highlight these, in each new edition of the Admin Manual, in the Critical changes section of the *Release notes* chapter.

Warning

Apps generally define their compatibility based on the major version(s) of Nextcloud Server they support. Consider the compatibility of your favorite and most critical apps, with a prospective major version of Nextcloud Server, before choosing which major version to deploy or deciding when to upgrade to a newly available major version. Also, since many apps are community provided and maintained by volunteers, you may want to offer to test the app against a new major version of Nextcloud (or to adapt it, if you're in a position to do so) in order to encourage a faster (or higher quality) release.

2.2.2 Maintenance releases

Maintenance releases deliberately **do not** introduce new features or breaking changes. This is meant to reduce the risks and impact associated with deploying updates so that critical bugs or security vulnerabilities can be rapidly and routinely addressed.

Maintenance releases are published (generally simultaneously) for all stable major releases that have not reached end-of-life status.

These releases should not have app compatibility concerns or introduce changes requiring retraining end users.

A specific maintenance release is indicated by the last part of the version number. For example, `28.0.4` is the *fourth* maintenance release for major version `28` of Nextcloud Server. It offers fixes for any critical bugs and security vulnerabilities addressed since the last maintenance release (`28.0.3` in this example).

Note

All critical bug fixes, including security related ones, are backported to **all** maintained major releases.

2.3 Release schedule

New **major** releases of Nextcloud Server are published approximately every sixteen weeks.

New **maintenance** releases are published approximately every four weeks.

2.3.1 Length of support ("maintenance")

Our release schedule means that several major releases (e.g. 26.X.X, 27.X.X, 28.X.X) are supported simultaneously. Whenever a critical bug or vulnerability is addressed, if it impacts more than one major release, it is **backported** to all applicable major releases and published in the next maintenance release (e.g. `28.0.3` -> `28.0.4`). Any major release that has not reached end-of-life status receives these maintenance updates.

This overlapping schedule and predictable cadence permits rapid development while giving administrators visibility, access to critical bug fixes, and flexibility as to how aggressively to upgrade to new majors.

Note

Since every major release is supported for one year from initial release, the minimum you need to do to stay up-to-date is to install maintenance releases as they're published and upgrade to the next higher up major release when the one you're currently on reaches end-of-life status. Since maintenance releases only patch your Server with the latest bug and security vulnerability fixes - and do **not** introduce other significant changes - the risk of upgrading to a new maintenance release is far less than upgrading to a new major release.

2.3.2 End-of-life

End-of-life status means that support/maintenance ends. Maintenance releases cease for a major version on the one year anniversary of initial release. The major version then moves into end-of-life status and will not receive any further bug fixes or corrections for security vulnerabilities.

Note

Support for major releases may be extended through subscription services for enterprises offered by Nextcloud developers via Nextcloud GmbH.

The end-of-life dates for all major releases are published ahead of time to ease planning.

Note

As long as a major release is still listed on the maintenance schedule as being *Currently Maintained*, you can expect to receive all relevant fixes for critical bugs or security vulnerabilities (even those made available for newer major releases, if they are relevant to a still supported earlier major).

2.4 Installation version

Since multiple major releases are published throughout the year and each is supported for a year with any relevant bug and security fixes, you have discretion as to which which major to deploy initially as well as when to upgrade to a new major.

Note

If you're planning to deploy Nextcloud in an enterprise setting and your usage will be mission-critical, the developers can help you choose, via an Enterprise services arrangement, the major version most suitable for your particular use case as well as help make sure it's deployed optimally while addressing any critical problems that arise with you one-on-one.

2.5 Release channels

By default all Nextcloud installations utilize the `stable` release channel. This channel delivers the latest features that are ready for most users at minimal risk.

Note

Nextcloud does staged roll-outs of new releases to further reduce the risk of widespread updates. New releases, particularly major releases, are usually only made available to a small percentage of systems initially. After a week (or more) has passed with no reported widespread critical bugs, more systems will be offered the update. Sometimes major versions are limited to <100% of systems until after the first maintenance (bug fix) release has been published.

Warning

When using the `stable` channel it is possible you'll be *offered* a newer major version to upgrade to *even if* your existing major version has **not** reached end-of-life. It is up to you to decide whether to upgrade then or wait until a better time for deploying a major new release. On the other hand, new **maintenance** releases (within the major version you're already running) should be deployed as soon as possible to keep up-to-date with security and other critical bug fixes.

Danger

Making sure you're running an actively maintained **major** release is critical. Once a major release reaches End of Life status it will not receive any further maintenance releases to correct critical bugs or vulnerabilities.

You can find the detailed schedule for all stable channel major releases and maintenance releases, including end-of-life dates, in our regularly updated Maintenance and Release Schedule.

2.6 Major version upgrades

Before upgrading from one one major release to another, we strongly recommend reviewing the *Critical changes* section of the **Release Notes** chapter to minimize the chance of introducing unexpected breaking changes in your environment.

Warning

Having good data backups (and a tested data restore approach!) is recommendeded in general, but definitely before performing an update - whether major or merely maintenance.

2.7 Beta releases and Release candidates

Before a new final major release is published, typically at least four beta releases are published followed by two release candidates, with an interval of one week between each.

Before a new final maintenance release is published, one release candidate is published approximately one week beforehand.

Anticipated dates for each release can be found on detailed schedule.

Tip

To update sooner to a new major version or beta version, you may at your discretion adjust your instance to use the `beta` channel. Around big releases the `beta` channel also delivers the newest major version earlier regardless of staging parameters.

Everyone in the community benefits considerably from the generous testing and feedback of those that choose to evaluate beta releases or release candidates in either their test environments or, for the bold, under real-world conditions.

If you are in a position to evaluate a pre-final release, the developers and the entire community thank you!

Tip

We suggest focusing your testing efforts on verifying the functionality and features you rely on every day (to make sure these operate as expected). Then, if you are so inclined, to consider evaluating any new functionality that interests you. Please discuss problems that arise at the Help Forum and report suspected bugs to the GitHub repository.

2.8 Downgrading

Downgrading is not supported officially between any major, maintenance, or pre-release version.

2.9 Bug reporting

Before reporting bugs, please make sure you're running a still supported major release *and* the latest maintenance release for it.

Tip

Nextcloud GmbH - which employs many of the core developers - offers Nextcloud Enterprise services providing direct access to Nextcloud engineering expertise where usage is mission-critical. Among other things, they can help you choose the major version most appropriate to your use case (and make sure it's deployed optimally).

CHAPTER

THREE

GDPR COMPLIANCE

Nextcloud is designed to be operated fully on-premises, keeping personal data under your organisation's control. This chapter provides guidance for administrators on meeting their obligations under the General Data Protection Regulation (GDPR).

Note

This documentation describes Nextcloud's built-in capabilities. It does not constitute legal advice. Consult your Data Protection Officer or legal counsel to confirm that your specific deployment meets your regulatory obligations.

3.1 Personal data stored

This page gives an overview of the categories of personal data that Nextcloud stores, to help you prepare your Records of Processing Activities (RoPA) and privacy notices as required under GDPR Article 30.

3.1.1 Account data

Every user account holds the following data:

- **Username** — the unique identifier used to log in.
- **Display name** — the name shown to other users.
- **Email address** — used for notifications, password resets, and sharing invitations.
- **Password hash** — stored using a one-way hash; the plaintext password is never stored.
- **Last login timestamp** — the date and time of the user's most recent login.
- **Account creation timestamp** — when the account was created.
- **Authentication tokens** — long-lived tokens created for clients and app passwords.

3.1.2 Profile data

Nextcloud stores additional profile fields that users may populate:

- Phone number
- Address
- Website URL
- Fediverse handle, Bluesky handle
- Biography

- Profile picture (avatar)

Each field has a configurable **scope** that controls who can see it: private, local (visible to other users on the same instance), federated (shared with trusted federation partners), or published (shared with the public lookup server). Administrators can restrict or enforce these scopes server-wide. See *Profiles* for details.

3.1.3 Files and metadata

- **File contents** — all files stored by users, including files in external storage mounts.
- **File metadata** — filename, path, size, MIME type, creation time, modification time, and owner.
- **File share records** — who has shared what with whom, including public share tokens and passwords.
- **Trash bin** — deleted files and their original paths, retained according to the configured retention policy.
- **File versions** — previous versions of modified files, retained according to the configured retention policy.

3.1.4 Activity and audit logs

- **Activity log** — a per-user record of file and sharing events (uploads, downloads, shares, edits). Retained for the number of days configured via `activity_expire_days` (default: 365 days).
- **System audit log** — if the `admin_audit` app is enabled, a server-wide log of administrative actions and login events is written to the Nextcloud log file.

3.1.5 Groupware data

When the groupware apps are enabled:

- **Contacts** — vCards including names, email addresses, phone numbers, physical addresses, photos, and any other fields users add.
- **Calendar events** — event titles, times, locations, descriptions, and attendee lists.
- **Tasks** — task titles, descriptions, due dates, and completion status.

3.1.6 Talk (chat and calls)

When the Talk app is enabled:

- **Chat messages** — text messages and reactions in one-to-one and group conversations.
- **Call metadata** — call participants and timestamps (call recordings, if enabled, are stored as files in the initiating user's storage).
- **Conversation membership** — which accounts belong to which conversations.

3.1.7 Server and web server logs

Personal data is also present in server logs, independent of what users intentionally store:

- **Web server access logs** — your web server (Apache, nginx) records the IP address, timestamp, and URL of every request. IP addresses are personal data under GDPR.
- **Nextcloud application log** — records errors, warnings, and (at higher log levels) user actions including IP addresses.
- **Brute-force protection list** — Nextcloud temporarily stores the IP address of failed login attempts. These are deleted automatically after 24 hours or upon a successful login.

Storing logs indefinitely is not considered legitimate usage under GDPR. Rotate logs at a reasonable interval and consider encrypting them, as they contain personal data you are responsible for securing. See *Data retention* for guidance on log rotation.

3.1.8 Session data

Nextcloud stores the following data client-side as cookies (see *Cookies* for the full list) and server-side in the session store:

- PHP session data — a random session identifier and an encrypted session passphrase.
- Remember-me tokens — stored if the user selected "Remember me" at login. Lifetime is controlled by `remember_login_cookie_lifetime` (default: 15 days).

3.1.9 Data held by third-party services

Some optional features transmit personal data outside the server:

- **Public lookup server** — profile fields with "Published" scope are sent to `lookup.nextcloud.com` (or a custom server configured via `lookup_server`) to enable cross-instance user discovery. This can be disabled by setting `lookup_server` to an empty string.
- **Push notifications** — Talk and other apps may route notification payloads through Nextcloud's push notification proxy. The payload contains the account name and a notification message.
- **Federated sharing** — when sharing files with users on other Nextcloud instances, the sharer's display name and email are transmitted to the remote server.

3.2 Responding to data subject requests

GDPR grants individuals several rights over their personal data. This page describes how to fulfil the most common requests using Nextcloud's built-in tools.

You must respond to subject access requests within **one month**. If a request is complex or you receive a high volume, you can extend this by a further two months, but you must inform the requester within the first month that an extension is needed and explain why. You cannot charge a fee for complying unless a request is manifestly unfounded or excessive.

3.2.1 Right of access (Article 15)

A user can view most of their own personal data directly in Nextcloud:

- **Profile and account info** — Personal Settings → Personal Info.
- **Files** — the Files app shows all stored files.
- **Activity log** — the Activity app shows file and sharing events.
- **Sharing history** — the Sharing section in Personal Settings.
- **Connected clients and app passwords** — Personal Settings → Security.

As an administrator you can retrieve a summary of an account's metadata using:

```
sudo -E -u www-data php occ user:info <uid>
```

This outputs display name, email, last login, quota, groups, and backend.

3.2.2 Right to erasure (Article 17)

To permanently delete a user account and all associated data:

```
sudo -E -u www-data php occ user:delete <uid>
```

This removes the account, all files owned by the user, their profile data, authentication tokens, and groupware data (contacts, calendars).

Warning

Comments must be deleted manually before running `user:delete`. File comments and Talk messages are stored in `oc_comments` and `oc_comments_read_markers`. These are not removed by `user:delete` and will be left attributed to "unknown/anonymous user" if not cleaned up first. Run these SQL queries before deleting the account:

```
DELETE FROM oc_comments
  WHERE actor_type = 'users' AND actor_id = '$USERID';

DELETE FROM oc_comments_read_markers
  WHERE user_id = '$USERID';
```

Replace `$USERID` with the account's username. To find all comment types on your instance (file comments, Talk, etc.) run:

```
SELECT DISTINCT object_type FROM oc_comments;
```

Note

What else is not automatically removed:

- Files the user has shared with others remain accessible to recipients until the recipient or an administrator deletes them.
- Federated shares the user accepted from other instances are held on those remote servers and must be removed there separately.
- Entries in the system audit log (`admin_audit`) are retained as they form part of the administrative record. Consult your legal team on the appropriate retention period for audit logs.
- Chat messages in Talk conversations are retained in the conversation history. Use the Talk admin panel to delete individual conversations if needed.
- Data in backups — see *Data retention* for guidance.

Before deleting the account you may want to transfer file ownership to another user so that shared files remain available:

```
sudo -E -u www-data php occ files:transfer-ownership <uid> <destination-uid>
```

3.2.3 Right to data portability (Article 20)

Users can export their own data directly from Nextcloud:

- **Files** — downloadable as a ZIP from the Files app, or accessible in full via WebDAV at `https://<your-server>/remote.php/dav/files/<uid>/`.

- **Contacts** — exportable as a `.vcf` file from the Contacts app (select all → Export).
- **Calendar** — exportable as an `.ics` file from the Calendar app (Calendar settings → Export).
- **Personal data export** — users can request a full data export in Personal Settings → Personal Info → scroll to the bottom → **Download your data**. This produces a ZIP archive containing files, contacts, and calendar data.

As an administrator you can export a user's calendar from the command line:

```
sudo -E -u www-data php occ dav:export-calendar <uid> <calendar-name> <output-file>
```

3.2.4 Right to rectification (Article 16)

Users can correct their own profile data in Personal Settings → Personal Info. Administrators can update display name, email, and other profile fields using:

```
sudo -E -u www-data php occ user:setting <uid> settings email <new-email>
sudo -E -u www-data php occ user:setting <uid> settings displayname <new-name>
```

For fields stored in a connected LDAP or SAML directory, corrections must be made in the upstream directory rather than in Nextcloud.

3.2.5 Right to restriction of processing (Article 18)

To suspend processing without deleting an account, disable it:

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

A disabled account cannot log in and no background jobs run for it. The account and all its data remain intact and can be re-enabled at any time:

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

3.3 Data retention

Nextcloud retains several categories of data beyond the point when a user considers them "deleted." This page describes the configurable retention periods and how to align them with your data minimisation obligations under GDPR Article 5(1)(e).

All settings below are configured in `config/config.php`. See *Configuration Parameters* for the full parameter reference.

3.3.1 Trash bin

When a user deletes a file it moves to the trash bin. The retention policy is controlled by `trashbin_retention_obligation`:

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

Available values:

- `auto` (default) — keep deleted files for at least 30 days; remove sooner if the user is running low on quota.
- `D1, auto` — keep for at least D1 days; remove sooner if quota is low.
- `auto, D2` — remove sooner if quota is low; guarantee deletion after D2 days regardless.

- `D1, D2` — keep for at least D1 days; guarantee deletion after D2 days.
- `disabled` — never automatically empty the trash bin.

For GDPR purposes, `disabled` means deleted files are retained indefinitely — avoid it unless you have a specific reason. Setting a firm maximum (e.g. `'30, 60'`) gives users a predictable deletion guarantee.

Users with the appropriate permissions can empty their own trash bin at any time from the Files app, or an administrator can run:

```
sudo -E -u www-data php occ trashbin:cleanup <uid>
```

To expire files across all users according to the current policy:

```
sudo -E -u www-data php occ trashbin:expire
```

3.3.2 File versions

The Versions app stores previous copies of modified files. Retention is controlled by `versions_retention_obligation`:

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

Available values:

- `auto` (default) — versions are pruned according to a built-in schedule (more versions kept for recent changes, fewer for older ones). See *Controlling file versions and aging*.
- `D, auto` — keep versions for at least D days, then apply the automatic schedule.
- `auto, D` — apply the automatic schedule; guarantee deletion after D days.
- `D1, D2` — keep for at least D1 days; guarantee deletion after D2 days.
- `disabled` — never automatically remove versions.

To remove versions immediately:

```
sudo -E -u www-data php occ versions:cleanup <uid>
sudo -E -u www-data php occ versions:expire <uid>
```

3.3.3 Activity log

The activity log records file and sharing events per user. Entries older than the configured number of days are deleted by the daily cron job:

```
'activity_expire_days' => 365,
```

Set this to a lower value to reduce the personal data footprint. Setting it to `0` disables automatic expiry (entries are kept indefinitely).

Note

The activity log is distinct from the system audit log produced by the `admin_audit` app. Audit log retention is controlled by your log rotation configuration, not by this setting.

3.3.4 Remember-me tokens

When users select "Remember me" at login, Nextcloud stores a long-lived authentication cookie. Its lifetime is controlled by:

```
'remember_login_cookie_lifetime' => 60 * 60 * 24 * 15,
```

The default is 15 days. Reducing this value means users must re-authenticate more frequently but limits the exposure window for stolen tokens.

3.3.5 Session lifetime

Active sessions expire when the browser is closed (session cookies). There is no server-side session expiry setting in core; sessions are invalidated when the user logs out or an administrator revokes them via:

```
sudo -E -u www-data php occ user:auth-tokens:delete <uid>
```

3.3.6 Server and web server logs

Web server access logs and the Nextcloud application log contain IP addresses and other personal data. Storing them indefinitely is not considered legitimate usage under GDPR. Rotate logs regularly and encrypt archived logs to protect the personal data they contain.

A minimal `logrotate` configuration that rotates daily and keeps logs for a limited period:

```
/var/log/nextcloud/*.log {
    daily
    rotate 90
    compress
    shred
    missingok
    notifempty
}
```

Adjust the `rotate` value to match your legal obligations and security requirements. If you are legally required to retain logs for a specific period (e.g. for compliance with national cybersecurity laws), that overrides the minimisation principle — but you must disclose the retention period in your privacy policy.

> **Note**
>
> Nextcloud's brute-force protection stores IP addresses of failed logins, but these are automatically deleted after 24 hours or upon a successful login and do not require manual management.

3.3.7 Backups

When you fulfil a right-to-erasure request by deleting an account, the data also exists in any backups you hold. GDPR's right to erasure extends to backup copies unless retaining them is required by law or necessary for legal defence.

Practical approaches:

- **Time-limited backups** — set a backup retention policy (e.g. 90 days) so that personal data is eventually purged from backups automatically, even if you cannot remove it on demand.
- **Isolated backups** — ensure backups can never be restored to a live production instance without a deliberate recovery procedure, so that deleted data cannot accidentally reappear.

- **Encrypted backups** — encrypt backup media so that the data cannot be read if the media is lost or transferred.

> **Note**
>
> Removing a specific user's data from an existing backup is technically complex (often impractical for tape or snapshot-based backups) and may require rethinking your backup strategy if you are subject to frequent erasure requests at scale.

3.3.8 Summary table

Data category	Config key	Default retention	Minimum recommended
Trash bin	`trashbin_retention_obligation`	30 days (auto)	Set a firm maximum (e.g. `auto, 60`)
File versions	`versions_retention_obligation`	auto schedule	Set a firm maximum (e.g. `auto, 180`)
Activity log	`activity_expire_days`	365 days	90–180 days
Remember-me tokens	`remember_login_cookie_lifetime`	15 days	7–15 days

3.4 Helpful apps

Several apps are available from the Nextcloud App Store to help you meet specific GDPR obligations.

3.4.1 Imprint (Theming app)

The **right to be informed** (GDPR Article 13/14) requires you to make your privacy policy accessible to users. The built-in Theming app includes an **Imprint** feature that lets you add links to your imprint and privacy policy on the Nextcloud login screen and in the footer.

To configure it:

1. Go to **Administration Settings → Theming**.
2. Fill in the **Imprint URL** and **Privacy policy URL** fields.
3. Save. The links appear on the login page and in the web interface footer.

This ensures all users — including those accessing public share links — can find your privacy policy without needing to be logged in.

3.4.2 Drop Account app

The Drop Account app allows users to delete their own Nextcloud account directly from Personal Settings, without needing to contact an administrator.

Install it from the Apps page or via occ:

```
sudo -E -u www-data php occ app:install drop_account
```

Once installed, users see a **Delete account** button in **Personal Settings → Personal Info**. They must confirm their password before the deletion proceeds.

Warning

Account deletion via this app also removes all of the user's files. Make sure users are aware of this before they proceed. Consider adding a note in your privacy policy explaining what is and is not removed (see *Responding to data subject requests* for the full list of what `user:delete` does and does not clean up, including the comments tables which require manual cleanup).

3.4.3 Data Request app

The Data Request app gives users a self-service way to invoke their GDPR rights directly from their Personal Settings, without needing to contact an administrator by email or through an external channel.

Once installed, two buttons appear in every user's **Personal Settings → Personal Info** page:

- **Request data export** — sends an email to all administrators notifying them that the user is requesting a copy of their data.
- **Request account deletion** — sends an email to all administrators notifying them that the user wants their account deleted.

Both actions require the user to confirm their password and are rate-limited to one request per action per hour.

The app is a notification bridge only. It does not perform the export or deletion automatically — administrators must act on each request manually using the steps described in *Responding to data subject requests*.

Note

Administrators receive the notification at the email address configured for their account. Make sure all administrator accounts have a valid email address set, otherwise notifications will be silently dropped.

Install it from the Apps page or via occ:

```
sudo -E -u www-data php occ app:install data_request
```

No further configuration is required after installation.

Handling incoming requests

When a request email arrives:

For a data export request:

1. Export the user's files via WebDAV, or ask the user to use the **Download your data** button in their Personal Settings.
2. Export contacts and calendars from the Contacts and Calendar apps, or use `occ dav:export-calendar`.
3. Deliver the exported data to the user securely.

For an account deletion request:

1. Optionally transfer file ownership before deleting:

   ```
   sudo -E -u www-data php occ files:transfer-ownership <uid> <new-owner>
   ```

2. Delete the account:

   ```
   sudo -E -u www-data php occ user:delete <uid>
   ```

See *Responding to data subject requests* for full details on what each operation removes and what data may remain after deletion.

3.5 Cookies

Nextcloud only stores cookies that are necessary for it to function. All cookies are set by your Nextcloud server directly — no third-party cookies are involved.

Under GDPR, only cookies that store or transmit personal data require a legal basis or consent. Of the cookies listed below, only the remember-me cookies contain personal data (the username). All other cookies contain randomly generated tokens with no inherent personal information.

Note

The `__Host-` prefix is applied to the same-site cookies only when Nextcloud is accessed over HTTPS. On plain HTTP the prefix is omitted and the cookies are named `nc_sameSiteCookiestrict` and `nc_sameSiteCookielax`.

3.5.1 Cookies stored by Nextcloud

Type	Name	Purpose	Personal data	Lifetime
Session cookie	`<instance_id>`	Carries a random PHP session ID used to identify the user's session on the server.	No	Until browser is closed.
Session cookie	`oc_sessionPassphrase`	Carries a random token used to decrypt the session data stored on the server.	No	Until browser is closed.
Same-site cookie	`__Host-nc_sameSiteCoo`	Used to detect whether a request originates from the same site (`SameSite=Strict`). Helps prevent CSRF attacks. Contains no user information.	No	Expires 2100-12-31 (effectively permanent).
Same-site cookie	`__Host-nc_sameSiteCoo`	Used to detect cross-site navigation requests (`SameSite=Lax`). Helps prevent CSRF attacks. Contains no user information.	No	Expires 2100-12-31 (effectively permanent).
Remember-me cookie	`nc_username`	Stores the user's login name to enable persistent login across browser sessions.	**Yes** — contains the username.	Defaults to 15 days. Configurable via `remember_login_cookie_life`
Remember-me cookie	`nc_token`	A random token paired with `nc_username` to authenticate the persistent login without storing the password.	No	Same as `nc_username`.
Remember-me cookie	`nc_session_id`	The original session ID, retained to allow session continuity when the remember-me token is used.	No	Same as `nc_username`.
Download helper	`ocDownloadStarted`	A short-lived random token set when a file download begins, used to signal the browser that the download has started (e.g. to hide a loading indicator).	No	20 seconds.

3.5.2 Remember-me cookies

The remember-me cookies (`nc_username`, `nc_token`, `nc_session_id`) are only set when the user explicitly selects **Remember me** at login. They are cleared immediately when the user logs out.

Because `nc_username` contains the user's login name, it is personal data under GDPR. The legal basis for storing it is

typically **legitimate interest** or **contract performance** (enabling the service the user has requested), provided the user has been informed of this in your privacy policy.

The lifetime defaults to 15 days and can be shortened in `config/config.php`:

```
'remember_login_cookie_lifetime' => 60 * 60 * 24 * 15,
```

CHAPTER

FOUR

DECLARATIONS

4.1 Energy consumption

The Green Metrics Tool is certified under the Blue Angel for Software DE-UZ 215.

This page contains the typical energy consumption according to the Standard Usage Scenarios used in the certification.

4.1.1 Initial Certification 2024/2025

Download Report 2024/2025

4.1.2 Certification Extension 2025/2026

Download Report 2025/2026

CHAPTER

FIVE

RELEASE NOTES

5.1 Critical changes

Once you've installed and configured your server, you will want to keep it up to date with the latest Nextcloud features.

These sub pages will cover the most important changes in Nextcloud, as well as some guides on how to upgrade existing installations.

5.1.1 Upgrade to Nextcloud 35

System requirements

- PHP 8.2 is no longer supported.
- The list of officially supported operating system has been updated:
  - The minimum supported version of *SUSE Linux Enterprise Server 15* has been bumped to SP7.
  - The minimum supported version of *Debian Linux* has been bumped to *13 (Trixie)*.
  - The minimum supported version of *Ubuntu Linux* has been bumped to *24.04*.
- The list of officially supported databases has been updated:
  - MariaDB 10.6 is out of support and thus Nextcloud dropped support for it. The minimum supported version of MariaDB is now 10.11 LTS.
  - MariaDB 12.3 is released as a new LTS version and is now supported by Nextcloud.
  - MySQL 8.0 is out of support and thus Nextcloud dropped support for it. The minimum supported version of MySQL is now 8.4 LTS.
  - MySQL 9.7 is released as a new LTS version and is now supported by Nextcloud.

> **Note**
>
> MySQL 9+ dropped support for MD5, some parts of the Nextcloud ecosystem might stilly rely on it. You need to make sure that after upgrading to MySQL 9+ you load the MySQL component for MD5 support, to do so run this on your MySQL 9+ server:
>
> ```
> INSTALL COMPONENT 'file://component_classic_hashing';
> ```

5.1.2 Upgrade to Nextcloud 33

System requirements

- PHP 8.5 is now supported.
- PHP 8.2 is now deprecated but still supported.
- PHP 8.1 is no longer supported.
- Oracle 11g is no longer supported.
- PostgreSQL 13 is no longer supported.

If you configured restrictions on which domains can be contacted on the internet, you need to add connectivity.nextcloud.com to the allowlist, as it's now used by default to test internet connectivity instead of www.nextcloud.com. You can also configure any other URL to use in the configuration instead. See *Connections to remote servers*.

Previews

The preview provider for MP3 files, which reads cover images embedded in the files, is disabled by default for performance and stability reasons. See *Previews configuration* for details on how to enable or disable the preview provider.

Snowflake IDs

This version of Nextcloud ships with Snowflake IDs. Those IDs include the creation time of object, a sequence ID and a server ID. The server ID should now be configured in your config.php file or using environment variables. See *Configuration Parameters* for more information.

OpenMetrics endpoint

Nextcloud 33 introduces a `/metrics` endpoint that can be integrated into every OpenMetrics (Prometheus) system. For security, it only answers on localhost by default.

See *Monitoring* for more information about it.

Default user agent for outgoing requests changed

Starting with this release, the default user agent for requests done by the instance was changed from `Nextcloud Server Crawler` to `Nextcloud-Server-Crawler/X.Y.Z`, where `X.Y.Z` is the current server version.

TaskProcessing worker command

Previously we instructed admins to run *occ background-job:worker <JobClass>* to speed up AI task processing. This recommendation has changed to running *occ taskprocessing:worker*, which handles parallel execution better. Make sure to update your setup.

See *Overview* for more information about this.

5.1.3 Upgrade to Nextcloud 32

System requirements

- PHP 8.1 is now deprecated but still supported.
- PHP 8.4 is now supported, but 8.3 is recommended.

Web server configuration

- Setup checks do not check for the `X-XSS-Protection` response header anymore. It has been removed from Nextcloud's `.htaccess` and you may want to adjust your webserver config to not serve it anymore. XSS filtering was supported only until Chromium 78 and similarly old browsers, but had been found to cause more issues, including attack vectors, than it solved. Nowadays, aside of not serving the header at all, the only generally recommended value is `0`. More context can be found in the OWASP Cheat Sheet Series.

Monitoring: Counting of active users

The monitoring app was adjusted to count the active users in the same way as occ user:report and the support app.

System address book

During the upgrade to Nextcloud 32 the system address book might become disabled if the amount of system users exceeds the default limit of 5000 users. This is to prevent performance issues. You can re-enable the system address book using the command line or administration interface.

For more information about the system address book, see the documentation. *System Address Book*

Previews

Starting with Nextcloud 32.0.1, the preview provider for MP3 files, which reads cover images embedded in the files, is disabled by default for performance and stability reasons. See *Previews configuration* for details on how to enable or disable the preview provider.

AppAPI (app_api) setup checks expanded

Starting with Nextcloud 30.0.1, the AppAPI app is included and enabled by default. See *ExApps management* for details. Additionally, as of version 32.0.0, the AppAPI has expanded its setup checks.

You can disable this app in the standard manner via the *Apps* menu if you do not expect to use AppAPI integrations in the near future.

If AppAPI is disabled, other apps that depend on it will not be visible in the app store. AppAPI-related setup checks will also be deactivated.

S3 integrity protections enabled, configuration update may be needed

The AWS SDK for PHP was updated and now supports the data integrity protections for S3.

>= Nextcloud 32.0.2: If your S3 backend does not support the data integrity protection, you can disable it by adding `'request_checksum_calculation' => 'when_required',` and `'response_checksum_validation' => 'when_required',` to the object store configuration.

>= Nextcloud 32.0.3: S3 data integrity protections are disabled by default and are now opt-in.

If your S3 backend does not support this, you may see an error such as `Checksum Type mismatch occurred, expected checksum Type: null, actual checksum Type: crc32` in your logs when uploading files.

More details about data integrity protections for S3 can be found at https://docs.aws.amazon.com/sdkref/latest/guide/feature-dataintegrity.html and https://github.com/aws/aws-sdk-php/discussions/3100.

5.1.4 Upgrade to Nextcloud 31

System requirements

- PHP 8.1 is now deprecated but still supported.
- PHP 8.4 is now supported, but 8.3 is recommended.

Database configuration

Other row formats than `DYNAMIC` for MySQL and MariaDB databases will issue a warning since Nextcloud 24, as they often cause performance issues. With Nextcloud 31 a more prominent new setup warning for this was added.

The row format can be changed via `ALTER TABLE` DDL commands during a maintenance window. Changing the row format from `COMPRESSED` to `DYNAMIC` requires about 2x the disk space and may take a long time depending on the size of the database. See the MySQL documentation for more information. If you're not sure how to do this, you can find some tips and tricks from the community.

PHP configuration

We have a new setup warning to check if the memory reserved for APCu is high enough. If you see this warning, you should increase the memory reserved for APCu. You can do this by increasing the value of the `apc.shm_size` directive in your `php.ini` file. It is generally advised to review this value and increase it if necessary depending on your instance size.

Nextcloud configuration

Maximum chunk size

We have adjusted the default maximum chunk size for big file uploading. Previously it was set to 10MiB, it is now increased to 100MiB.

Also the configuration was moved from an app configuration to the system configuration (`config.php`). If you set up a custom value previously the value will be automatically migrated to the system configuration during the update. But if you need to set a new custom value you need now to use the system configuration, see also *Adjust chunk size on Nextcloud side*.

Monitoring: Counting of active users

Starting with Nextcloud 31.0.6 the monitoring app was adjusted to count the active users in the same way as occ user:report and the support app.

Previews

Starting with Nextcloud 31.0.10, the preview provider for MP3 files, which reads cover images embedded in the files, is disabled by default for performance and stability reasons. See *Previews configuration* for details on how to enable or disable the preview provider.

AppAPI (app_api) is now a default app

Starting with Nextcloud 30.0.1, the AppAPI app is included and enabled by default. See *ExApps management* for details.

You can disable this app in the standard manner via the *Apps* menu if you do not expect to use AppAPI integrations in the near future.

If AppAPI is disabled, other apps that depend on it will not be visible in the app store. AppAPI-related setup checks will also be deactivated.

5.1.5 Upgrade to Nextcloud 30

System requirements

- PHP 8.1 is now deprecated but still supported.
- PHP 8.0 is no longer supported.
- PostgreSQL 9.4 is no longer supported.

- MariaDB 10.3 and 10.5 are no longer supported.

Web server configuration

Make sure that your web server is serving files with the `webp` extension (WebP images) correctly as static assets. This is included in the shipped `.htaccess` file but if you use another web server or custom configuration you need to check this manually.

Nextcloud configuration

Changes to the available options in `config.php`.

- The option `blacklisted_files` is now deprecated and replaced with `forbidden_filenames`
- The option `forbidden_chars` is now deprecated and replaced with `forbidden_filename_characters`
- The option `forbidden_filename_basenames` was added to allow bocking files with specific basenames (the filename without extension (before the first dot))
- The option `forbidden_filename_extensions` was added to allow blocking extensions from being used on filenames

Previews for PDF files with Imaginary

The preview provider `OC\Preview\Imaginary` is no longer generating previews for PDF files. Add the new preview provider `OC\Preview\ImaginaryPDF` to `enabledPreviewProviders` to enable preview generation with Imaginary for PDF files.

Automated clean-up of app password

Nextcloud 30 will *clean-up unused app passwords*.

Monitoring: Counting of active users

Starting with Nextcloud 30.0.12 the monitoring app was adjusted to count the active users in the same way as occ user:report and the support app.

AppAPI (app_api) is now a default app

Starting with Nextcloud 30.0.1, the AppAPI app is included and enabled by default. See *ExApps management* for details.

You can disable this app in the standard manner via the *Apps* menu if you do not expect to use AppAPI integrations in the near future.

If AppAPI is disabled, other apps that depend on it will not be visible in the app store. AppAPI-related setup checks will also be deactivated.

5.1.6 Upgrade to Nextcloud 28

System requirements

- PHP 8.3 is now supported, but 8.2 is recommended.

Web server configuration

- The recommended *nginx configuration* changed as Nextcloud Talk now serves audio files with `.ogg` / `.flac` extension, make sure to add these extensions to the list of static files.

- As some core app now make use of JavaScript modules, make sure your web server is not rewriting requests to `.mjs` files, but serves them with `text/javascript` MIME type and proper `Cache-Control` header, like `.js` and other static file extensions.
  - When using Apache with `.htaccess` configuration, this will be done automatically.
  - For Nginx, please refer to our recommended *Nginx configuration*.
  - For other setups, make sure to add `.mjs` to the list of static file extensions in web server configs and in case define its MIME type in `/etc/mime.types`.

Setup Checks

The setup checks (the ones visible under *Administration settings->Overview*) that previously ran from the web browser now run server-side rather than from the browser.

This means that some false positives may be triggered in existing installations after upgrading. This does not mean the checks are invalid or broken. It does mean that local configuration matters that may not have had obvious side effects previously may now prevent the tests from getting accurate results.

In nearly all cases the resolution is one or more of the following:

- verifying all entries in `trusted_domains` and the value of `overwrite.cli.url` are valid, resolvable in DNS, and reachable *from the Nextcloud Server itself*
- verifying that the Server can reach its own URL(s)
- verifying all `overwrite*` config values are reasonable

In diagnosing the above, many admins have found it useful to review not only their *config.php* (for cleanup) but also:

- their local DNS resolvers and `/etc/hosts` files for reasonableness
- their firewall configurations
- their container network configuration if using Docker/etc (especially for outbound connectivity)

> **Tip**
>
> Testing of connectivity and reachability of specific URLs can usually be tested from servers or containers via `curl` or `wget`.

Monitoring

Beginning with Nextcloud 28, the monitoring endpoint no longer provides information about available app updates, as gathering the data always involves at least one external request to apps.nextcloud.com.

You can still ask the monitoring endpoint to show new app updates by using the URL parameter skipApps=false. However, please do not check this endpoint too often.

https://github.com/nextcloud/serverinfo#api

Previews for Office files using LibreOffice

Nextcloud can generate previews for Office files using LibreOffice.

Since Nextcloud 28, you can also create previews for EMF files. To enable it, add `'OC\Preview\EMF'` to `enabledPreviewProviders`.

Until Nextcloud 28, the same LibreOffice user profile was used to generate the previews. LibreOffice can only be invoked once per user profile, so the generation of a preview for an office file would fail if another one were created right now.

Beginning with Nextcloud 28, a different LibreOffice user profile is used for each file. Downside: If you upload 100 emf files, you may end up with 100 LibreOffice invocations. Though, you can use `preview_concurrency_new` and `preview_concurrency_all` to limit the number of previews that can be generated concurrently when php-sysvsem is available.

The configuration option `preview_office_cl_parameters` was removed with Nextcloud 28. We expect LibreOffice to be started with the given parameters, so it's unfavorable to have a configuration option to change the parameters. Please reach out to us via https://github.com/nextcloud/server/pull/41395 if that's causing any trouble for you.

Tip

Previews for EMF files can be enabled without a local LibreOffice installation if you are already using Nextcloud Office / Collabora. Make sure you have Nextcloud Office 8.3.0 installed and add `'OCA\Richdocuments\Preview\EMF'` to `enabledPreviewProviders`.

5.1.7 Upgrade to Nextcloud 27

System requirements

- PHP 8.2 is recommended over PHP 8.1.
- PHP 8.0 is deprecated and might be removed in Nextcloud 28.

Exposed system address book

Nextcloud 27 exposes the *system address book*. Restrict the enumeration settings if your users should not see other users.

Web server configuration

- The recommended *nginx configuration* changed as Nextcloud now supports module javascript with the `.mjs` and audio files with `.ogg` / `.flac` extension, make sure to add these extensions to the list of static files.

5.1.8 Upgrade to Nextcloud 26

System requirements

- PHP 8.2 is now supported, but 8.1 is recommended.
- PHP 7.4 is no longer supported.

System email

The software component to send system emails (notifications, invites, password reset, etc) had to be replaced. The new library should work without any changes out of the box for most setups.

A brief overview of changes:

- STARTTLS cannot be enforced. It will be used automatically if the mail server supports it. The encryption type should be set to 'None/STARTTLS' in this case.
- Self signed certificates now need to be explicitly enabled, see *this guide* for an example on how to configure this.
- NTLM authentication for Microsoft Exchange is not supported by the new mailer library. Try using basic authentication instead.

See for more information: *Mail Providers*.

DAV sync tokens retention

A mechanism to clean up old CalDAV and CardDAV sync tokens has been added. See *CalDAV retention* and *CardDAV retention* and make sure it fits your installation size.

Web server configuration

- The recommended *nginx configuration* changed.

5.2 Changelog

See the official changelog for a complete list of changes.

CHAPTER

SIX

INSTALLATION AND SERVER CONFIGURATION

6.1 System requirements

6.1.1 Server

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

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

Platform	Options
Operating System (64-bit)	• **Ubuntu 26.04 LTS** (recommended) • Ubuntu 24.04 LTS • **Red Hat Enterprise Linux 10** (recommended) • Red Hat Enterprise Linux 9 • Debian 13 (Trixie) • SUSE Linux Enterprise Server 16 • SUSE Linux Enterprise Server 15 SP7 (or later) • openSUSE Leap 16 • CentOS Stream • Alpine Linux
Database	• MySQL 8.4 / 9.7 • MariaDB 10.11 / 11.4 / **11.8** / 12.3 (recommended) • Oracle Database 19c, 21c, 23ai (*only as part of an enterprise subscription*) • PostgreSQL 14 / 15 / 16 / 17 / **18** (recommended) • SQLite 3.24+ (*only recommended for testing and minimal-instances*)
Webserver	• **Apache 2.4 with** `mod_php` **or** `php-fpm` (recommended) • nginx with `php-fpm`
PHP Runtime	• 8.3 • 8.4 • **8.5** (*recommended*)

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

To ensure the full functionality of your Nextcloud, please make sure that the server can reach the *required remote systems*.

CPU Architecture and OS

A 64-bit CPU, OS and PHP is strongly recommended for Nextcloud.

32-bit systems are supported, with the following known limitations:

- Dates before Unix Epoch (1970-01-01) are not supported
- Dates after 2038 are not supported
- Some external apps may not work with 32-bit systems

Memory

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

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

In low memory environments, some features or apps may require adjustments to their default settings in order to function (or, in some cases, may need to be disabled outright).

Warning

To use the built-in Updater, at least 256MB is required.

Database requirements for MySQL / MariaDB

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

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

Why we drop old PHP versions

Every year, a new PHP version is added and old PHP versions are deprecated. This also affects our documented recommended PHP version.

We try to support old PHP versions for as long as reasonably possible. However the list of security, performance, and bug fixes will only increase, some of those fixes might be considered critical and thus at some point the deprecation will be inevitable.

Thus it is recommended to keep your PHP version up to date.

Advantages of upgrading PHP

- **Security**

  PHP deprecates security fixes of old versions. Nextcloud cannot implement security fixes that come with new PHP versions as long as we support deprecated PHP versions, since the syntax that we are allowed to use must be the lowest one of the supported versions, thus the upstream packages of third parties break because they dropped this support.

- **Performance**

  The language continuously improves over time which makes it possible to do more requests in significantly less time.

Long term support

If you are running Nextcloud for an organisation-critical use case, you could consider upgrading your subscription to a premium subscription which comes with 5 years of long term support. This means you continue to receive maintenance releases for high and critical security issues, data loss fixes, and regressions within version over this extended period of time.

6.1.2 Desktop client

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

- **Windows** 10+

- **macOS** Monterey (12.0)+ (64-bits only) * Please note that your server may need to be Apple App Transport Security compliant in order for the desktop client to connect successfully. This may involve using a digital certificate that is adequately signed to the standards established by Apple. More information is provided by Apple in their developer documentation: https://developer.apple.com/documentation/security/preventing-insecure-network-connections
- **Linux** (64-bits only) Should run on any distribution newer than Ubuntu 18.04 with our official AppImage package

6.1.3 Mobile apps

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

Files App

- **iOS** 17.0+
- **Android** 9.0+

Talk App

- **iOS** 16.0+
- **Android** 8.0+
- **Nextcloud Server** 22.0+
- **Nextcloud Talk** 12.0+

6.1.4 Web browser

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

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

> **Note**
>
> If you want to use Nextcloud Talk you should use the latest version of Mozilla **Firefox** or Google **Chrome**/Chromium to have the full experience with video calls and screensharing.

6.2 Deployment recommendations

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

6.3 Preparing PHP

Before installing Nextcloud Server, ensure your PHP environment is properly configured. This includes installing the correct PHP version, enabling required PHP modules, and adjusting important *php.ini* settings. This guide explains which

PHP modules are necessary, which are recommended for optimal performance and compatibility, and how to configure your PHP environment for both web server and command-line usage.

Note

You can safely ignore this chapter if you plan to use a turnkey Nextcloud Server installation method (such as AIO, Snap, NCP, or Community Docker). Those installation methods provide PHP environments that are already pre-configured for use with Nextcloud Server. For guidance regarding customizing PHP in those environments, refer to the documentation provided specifically for or by those install methods.

- *PHP Installation*
- *Required PHP Modules*
- *Required PHP Database Connectors*
- *Recommended General PHP Modules*
- *Recommended PHP Caching Modules*
- *Recommended PHP CLI Modules*
- *PHP Modules for Media Management*
- *PHP Modules for Specific Applications*
- *PHP* ini *Settings*
- *Notes on PHP* ini *Configuration*
- *PHP Module Quick Reference Table*
- *Further Resources*

6.3.1 PHP Installation

Refer to your OS distribution's documentation for instructions for establishing a base PHP installation. It may be possible to choose among several versions of PHP. Refer to *System requirements* to see which versions of PHP are supported by this release of Nextcloud Server. After completing a base PHP installation, follow the below guidance to configure your new PHP installation for your new Nextcloud Server deployment.

6.3.2 Required PHP Modules

The following PHP modules **must** be installed and enabled for Nextcloud Server to function:

- *ctype* (included with PHP)
- *curl*
- *DOM*
- *fileinfo* (included with PHP)
- *filter* (only on Mageia and FreeBSD)
- *GD*
- *xml* (provides SimpleXML, XMLReader and XMLWriter; requires Linux package *libxml2* version >= 2.7.0)
- *mbstring*

- *OpenSSL* (included with PHP)
- *posix*
- *session* (included with PHP)
- *zip*
- *zlib*

Note

The PHP "xml" extension is commonly packaged as *php-xml* or shown as *libxml* by OS package managers. This extension provides the underlying libxml2 bindings and exposes SimpleXML, XMLReader and XMLWriter. Ensure the corresponding *php-xml* (or distribution-specific) package is installed so that SimpleXML, XMLReader and XMLWriter are available to PHP.

The *ctype*, *fileinfo*, and *OpenSSL* modules are generally included and enabled in PHP by default. Often some of the other required modules are automatically installed by OS distribution package managers.

Note

PHP Version-Specific Information:

- **PHP 8.3 and 8.4:** Several modules are now bundled with PHP by default, such as *curl*, *zlib*, and others. When upgrading to these versions, some modules that were previously optional may already be enabled. If you encounter "module not found" errors when running standard package installation commands, the module may already be bundled and enabled — verify this by using the module check command below.
- **Debian/Ubuntu packages:** Module availability varies by distribution version. If a listed module cannot be found in your package manager, check the PHP documentation or your distribution's PHP module list to confirm if the feature is bundled with your PHP version.

How to check if a module is enabled:

- Run `php -m | grep -i <module_name>`. If you see output, the module is active.

Note

The *filter* module is required only on Mageia and FreeBSD.

6.3.3 Required PHP Database Connectors

Install the PHP connector module for the database you plan to use (choose one):

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

6.3.4 Recommended General PHP Modules

These modules are not required, but are highly recommended to improve functionality or security:

- *intl*: Fixes sorting of non-ASCII characters and improves language translation performance.
- *sodium*: Provides Argon2 password hashing (needed if using PHP < 8.4 and PHP was built without *libargon2*). Starting with PHP 8.0, *sodium* is usually enabled by default. In PHP 8.4, *sodium* is typically bundled.

  bcrypt will be used if Argon2 is unavailable, but if passwords were previously hashed with Argon2 (such as when migrating an existing Nextcloud Server installation to a new server environment) and this module is missing, accounts will not be able to log-in).
- *sysvsem*: Enables System V semaphores used by Nextcloud to coordinate preview generation across PHP processes. Recommended; if missing, previews still work but may be less reliable under heavy load.

6.3.5 Recommended PHP Caching Modules

Memory caching is not required so these modules are not required, but are highly recommended for optimal performance and reliability. Choose and install your preferred combination of memory caching modules:

- *APCu* (>= 4.0.6)
- *redis* / *phpredis* (>= 2.2.6, required for Transactional File Locking)
- *memcached* (an older alternative to *redis* that is not recommended for new installations)

> **Note**
>
> Memory caching is highly recommended for optimal performance. In most cases, a combination of *APCu* and *redis* are the best choice for new installations. You can also use the `KeyValueCache` cache option which provides Redis-like functionality without requiring installing a PHP module, see *KeyValueCache configuration in Nextcloud (config.php)*.

See *Memory caching* for configuration details.

6.3.6 Recommended PHP CLI Modules

For command-line processing (optional):

- *pcntl*: Allows command interruption (e.g., via `ctrl-c`).

  Ensure `pcntl_signal` and `pcntl_signal_dispatch` are *not* disabled in your *php.ini* by the `disable_functions` option.

For command-line updater (optional):

- *phar*: Required to run the updater with:

  ```
  sudo -E -u www-data php /var/www/nextcloud/updater/updater.phar
  ```

6.3.7 PHP Modules for Media Management

Image meta data and orientation (optional):

- *exif*: Image meta data loading and rotation

Preview Generation (optional):

- *imagick* (for image previews)
- *avconv* or *ffmpeg* (for video previews)

- OpenOffice or LibreOffice (for document previews)

> **Note**
>
> If previewing PDF files fails with a "not authorized" error, you may need to adjust the *imagick* policy file. See https://cromwell-intl.com/open-source/pdf-not-authorized.html

See *Previews configuration* for additional preview generation context.

6.3.8 PHP Modules for Specific Applications

Some optional Nextcloud apps/functionality require additional modules. Install as needed:

- *ldap*: LDAP integration
- *smbclient*: SMB/CIFS integration (see *SMB/CIFS*)
- *ftp*: FTP storage or external user authentication
- *imap*: External user authentication

Recommended/Optional:

- *gmp*: SFTP storage

6.3.9 PHP *ini* Settings

Adjust the following settings in your *php.ini* as needed for Nextcloud:

- `disable_functions`: Avoid disabling functions unless necessary.
- `max_execution_time`: See *Uploading big files > 512MB*
- `memory_limit`: Should be at least 512MB. See also *Uploading big files > 512MB*
- `opcache.enable` and related settings: See *Memory caching* and *Server tuning*
- `open_basedir`: See *Hardening and security guidance*
- `upload_tmp_dir`: See *Uploading big files > 512MB*

6.3.10 Notes on PHP *ini* Configuration

- **Multiple php.ini files:**
  - You may need to configure settings in more than one *php.ini* file (e.g., for web server and CLI).
    - **Web server:**
      /etc/php/<version>/apache2/php.ini or */etc/php/<version>/fpm/php.ini*
    - **CLI (used by Nextcloud CRON jobs):**
      /etc/php/<version>/cli/php.ini
- **Find which php.ini is active for each SAPI:**
  - Use `php --ini` for CLI, or check `phpinfo()` in a web page.
- **Search for a parameter:**
  - Run `grep -r <parameter_name> /etc/php` (e.g., `grep -r date.timezone /etc/php`)
- **Replace `<version>` with your actual PHP version (e.g., 8.1, 8.2, etc.).**

6.3.11 PHP Module Quick Reference Table

Module	Required	Recommended	For Specific App	Description
ctype	✓			Core functionality
curl	✓			HTTP requests
DOM	✓			Document Object Model (XML/HTML handling)
fileinfo	✓			File type detection
filter*	✓*			Data filtering and validation (Mageia/FreeBSD)
GD	✓			Image processing
xml	✓			XML parsing (libxml2 >= 2.7.0) — provides the php 'xml' ex
mbstring	✓			Multibyte character handling
OpenSSL	✓			Secure communications
posix	✓			POSIX functions
session	✓			Session support
zip	✓			Zip file handling
zlib	✓			Compression and decompression
intl		✓		Improves translations and sorting
sodium		✓		Argon2 password hashing
sysvsem		✓		System V semaphore support used to coordinate preview gene
ldap			✓	LDAP integration
smbclient			✓	SMB/CIFS integration
ftp			✓	FTP storage/authentication
imap			✓	External user authentication
gmp			✓ (optional)	SFTP storage
exif			✓ (optional)	Image rotation in Pictures app
apcu		✓		Performance caching
memcached		✓		Performance caching
redis		✓		Transactional File Locking
imagick			✓ (optional)	Image previews
avconv/ffmpeg			✓ (optional)	Video previews
Open/LibreOffice			✓ (optional)	Document previews
pcntl			✓ (optional)	Command interruption in CLI
phar			✓ (optional)	Needed for command-line updater

*The filter module is required only on Mageia and FreeBSD.

6.3.12 Further Resources

- For more details on each module, consult the official PHP documentation.
- Refer to your OS distribution's documentation for the specifics of installing PHP modules in your environment.
- The words *extension* and *module* are interchangeable within PHP. We use the word *modules* in our documentation.
- Always restart your web server and PHP-FPM after making changes to an *php.ini* file or installed modules.

6.4 Installation on Linux

There are multiple ways of installing Nextcloud depending on your preferences, requirements and goals.

If you prefer an automated installation, you have the option to:

- use the official Nextcloud installation method. Nextcloud AIO provides easy deployment and maintenance with most features included in this one Nextcloud instance. It includes Office, a turnkey Backup solution, Imaginary

(for previews of heic, heif, illustrator, pdf, svg, tiff and webp) and more.

- use the community Snap Package. This includes a full production-ready stack, will maintain your HTTPS certificates for you, and will automatically update as needed to stay secure.
- use the community Nextcloud VM Appliance (aka Nextcloud Virtual Machine or NcVM). This helps you create a personal or corporate Nextcloud Server faster and easier. It can be used install directly on a clean Ubuntu Server or downloaded as a fully functioning VM.
- use the community NextcloudPi scripts (based on Debian). It will setup everything for you and include scripts for automated installation of apps like: Collabora, OnlyOffice, Talk and so on.
- use the community Nextcloud Docker image. This image is designed to be used in a micro-service environment. There are two versions of the image you can choose from: the Apache one contains a full Nextcloud installation including an Apache web server. The second option is an FPM installation and runs a FastCGI process that serves your Nextcloud installation (you will need to supply your preferred web, database and other desired supplementary services).

Note

Please note that the community options are not officially supported by Nextcloud GmbH.

Tip

For an enterprise-ready and scalable installation based on Helm Charts (also available for Podman), please contact Nextcloud GmbH.

In case you prefer installing from the source tarball, you can setup Nextcloud from scratch using a classic LAMP stack (Linux, Apache, MySQL/MariaDB, PHP). This document provides a complete walk-through for installing Nextcloud on Ubuntu 24.04 LTS Server with Apache and MariaDB, using the Nextcloud .tar archive. This method is recommended to install Nextcloud.

This installation guide is giving a general overview of required dependencies and their configuration. For a distribution specific setup guide have a look at the *Example installation on Ubuntu 24.04 LTS* and *Example installation on CentOS 8*.

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.

6.4.1 Prerequisites for manual installation

The Nextcloud .tar archive contains all of the required PHP modules. Your Linux distribution should have packages for all required modules. See *Preparing PHP* for a list of required and suggested modules.

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.)

6.4.2 Apache Web server configuration

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

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

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

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

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

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

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

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

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

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

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

```
a2ensite nextcloud.conf
```

Additional Apache configurations

Required modules:

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

  ```
  a2enmod rewrite
  ```

- If you're using `mod_fcgi` or `php-fpm` (PHP FastCGI Process Manager), you must enable the proxy modules:

```
a2enmod proxy
a2enmod proxy_fcgi
```

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

and apply the following modifications to the configuration::

  ProxyFCGIBackendType FPM

  <FilesMatch remote.php>
    SetEnvIf Authorization "(.*)" HTTP_AUTHORIZATION=$1
  </FilesMatch>
```

Verifying modules are enabled:

To verify that required modules are enabled, use the `apache2ctl` command:

```
apache2ctl -M | grep -E "rewrite|proxy|proxy_fcgi"
```

If you see matching output for the modules you've enabled, they are active. If any required module is missing, troubleshoot your OS package manager to ensure the Apache modules are installed (package names may vary by distribution; for example, on Debian/Ubuntu look for `libapache2-mod-fcgid` or similar).

- 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.

6.4.3 Pretty URLs

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

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

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

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

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

if it isn't installed in a subfolder.

Note

htaccess.RewriteBase must match the path relative to Apache's DocumentRoot where Nextcloud is served on the backend, not the public URL prefix. In a direct Apache setup these are identical. Behind a reverse proxy that strips the URL prefix — for example https://domain.com/nextcloud/ forwarded to http://localhost:8080/ — the correct value is / even though the public URL contains /nextcloud.

Finally run this occ-command to update your .htaccess file:

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

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

Note

In case the automatically added .htaccess configuration *SetEnv front_controller_active true* does not work for your environment: Edit config/config.php and add 'htaccess.IgnoreFrontController' => true. See *Configuration Parameters* for a detailed description.

6.4.4 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. Consider getting a certificate signed by a signing authority. Check with your domain name registrar or hosting service for good deals on commercial certificates. Or use a free Let's Encrypt ones.

6.4.5 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

`www-data` is the default web server user on Debian/Ubuntu systems. On RHEL/CentOS/Fedora use `apache`. The key requirement is that the web server process user has **read and write access** to all Nextcloud directories — ownership is not strictly necessary. In environments where changing ownership is not possible (shared hosting, Docker bind-mounts, etc.), it is sufficient to add the web server user to the group that owns the directories and ensure the directories are group-writable, for example:

```
usermod -a -G vboxsf www-data
```

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 *Install from the command line*.

To use the graphical Installation Wizard see *Installation wizard*.

6.4.6 Setting up background jobs

Nextcloud requires that some tasks are run regularly. These may include maintenance tasks to ensure optimal performance or time sensitive tasks like sending notifications.

See *Background jobs* for a detailed description and the benefits.

6.4.7 SELinux configuration tips

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

6.4.8 PHP-FPM configuration

Overview

PHP-FPM is a FastCGI based implementation of PHP containing features useful for busy web sites and large web applications. Using it with Nextcloud is an advanced topic and requires getting familiar with how PHP-FPM functions. In most cases the defaults are not ideal for use with Nextcloud. Here we'll highlight a few of the most important areas that should be adjusted.

Process manager

The default value for `pm.max_children` in many PHP-FPM installations is lower than appropriate. Having a low value may cause client connectivity problems, unexplained errors, and performance problems. It is a common cause of *Gateway Timeouts*. Having too high of a value in relation to available resources (such as memory), however, will also lead to problems. The default value is often `5`. This greatly limits simultaneously connections to your Nextcloud instance and, unless you are under severe resource constraints, will underutilize your hardware. Check the *Server tuning* chapter for some guidance and resources for coming up with appropriate values, as well as other related parameters.

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 appropriate `php-fpm` ini/config file.

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

Debian/Ubuntu/Mint	CentOS/Red Hat/Fedora
`/etc/php/8.3/fpm/`	`/etc/php-fpm.d/`

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

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

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

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

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

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

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

```
/etc/php/8.3/fpm/pool.d/www.conf
```

and uncommenting the line:

```
clear_env = no
```

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

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

Maximum upload size

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

.htaccess

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.

6.4.9 Other Web servers

- *NGINX configuration*

6.4.10 Installing on Windows (virtual machine)

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

- **Enterprise/SME appliance**

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

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

Download the the Appliance here:

- Univention Corporate Server (UCS)

- **Home User/SME appliance**

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

The VM comes in different sizes and versions.

You can find all the available versions here.

For complete instructions and downloads see:

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

Note

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

6.4.11 Installing via Snap packages

Nextcloud snap is a community driven installation method and is designed to be easy to install and simple to maintain. The ideal Nextcloud snap is an "install and forget" Nextcloud instance that works on most architectures and updates itself without needing administrative skills. Combining Nextcloud with snapd makes it a perfect fit for IoT or scalable environments. Snapd is a secure and robust technology which the Nextcloud snap team has embraced.

Most importantly snaps are designed to be secure, sandboxed, containerized applications isolated from the underlying system and from other applications.

However, the snap is opinionated and there are requirements to be met.

- Nextcloud snap uses recommended Apache.
- Nextcloud snap uses recommended MySQL.
- Nextcloud snap uses recommended PHP.

Installation

On Ubuntu

- https://snapcraft.io/nextcloud
- Install Nextcloud `sudo snap install nextcloud`

All other distros be warned

By default the latest stable Nextcloud snap release will be installed and it will automatically update to subsequent stable releases, but there are other releases available as well and you have full control of automatic updates.

After installation, Nextcloud will start automatically. Assuming you and the device on which it was installed are on the same network, you will reach the Nextcloud installation by visiting `<hostname>.local` or the IP address of the instance in your browser. If your hostname is `localhost` or `localhost.localdomain`, like on an Ubuntu Core device, `nextcloud.local` will be used instead.

1st login

Upon visiting the Nextcloud installation for the first time, you will be prompted to enter an admin username and password before Nextcloud is initialised. This may take a while depending on resources and the device. After you provide that information you will be logged in and able to install apps, create users, and upload files.

HTTPS encryption

Nextcloud snap includes a service for automated HTTPS encryption and automated renewal using Lets Encrypt, or self-signed certificates. Run `nextcloud.enable-https -h` for more information. Managing encryption.

Configuration

While the default Nextcloud configurations are mostly fine, it may be necessary to fine tune Nextcloud snap by editing configuration files manually or using the management console. Configuring Nextcloud snap.

External media

Snap confinement is a security feature and determines the amount of access an application has to system resources, such as files, the network, peripherals and services. Thus your Nextcloud snap is securely confined from the host system. Unless you specifically allow the Nextcloud snap to access the `/media` or `/mnt` directories on the host system, you will not be able to access any other directory outside of the confinement.

Removable media or external storage must be mounted to either `/media` or `/mnt` as root with root permissions and connected to Snap! Managing external media and storage

The interface providing the ability to access removable media is not automatically connected upon install, to use external storage (or otherwise use a device in `/media` or `/mnt` for data), you need to give the snap permission to access removable media by connecting that interface:

```
sudo snap connect nextcloud:removable-media
```

Further documentation, an extensive Wiki and FAQ's can be found on the developers GitHub.

Note

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

6.4.12 Installation via web installer on a VPS or web space

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

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

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

Note

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

6.4.13 Installation on TrueNAS

See the TrueNAS installation documentation.

6.4.14 Installation via install script

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

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

```
sudo bash nextcloud_install_production.sh
```

or

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

```
sudo bash install.sh
```

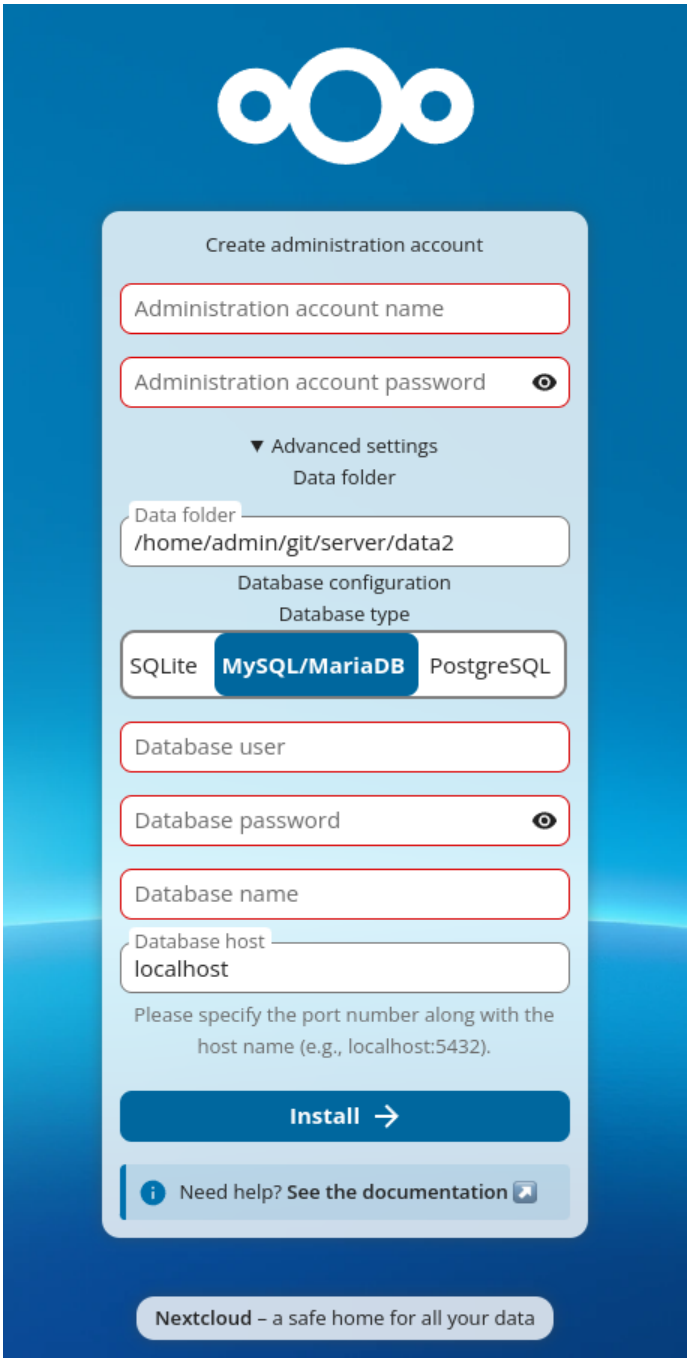
A guided setup will follow and the only thing you have to do it to follow the on screen instructions, when given to you.

6.5 Installation wizard

6.5.1 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 administration account name and password.
3. Click **Install**.



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

Note

The wizard includes a real-time password strength indicator that rates your chosen password from "too weak" to "extremely strong". For security, choose a password rated at least "strong".

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*

6.5.2 Data directory location

Expand the **Storage & database** section 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 it is located in `/opt/nextcloud/`. This directory must already exist, and must be owned by your HTTP user.

Note

If the wizard detects that your `.htaccess` file is not working (for example, because you are using Nginx or another non-Apache web server), it will display a **Security warning** indicating that your data directory and files may be accessible from the internet. Refer to the *Hardening and security guidance* documentation for guidance on securing your data directory.

6.5.3 Database choice

SQLite is the default database for Nextcloud Server. When SQLite is selected, the wizard displays a **Performance warning**:

> *SQLite should only be used for minimal and development instances. For production we recommend a different database backend. If you use clients for file syncing, the use of SQLite is highly discouraged.*

Supported databases are MySQL, MariaDB, Oracle, and PostgreSQL, and we recommend *MySQL/MariaDB*. Your database and PHP connectors must be installed before you run the Installation Wizard. When you install Nextcloud from packages all the necessary dependencies will be satisfied (see *Installation on Linux* for a detailed listing of required and optional PHP modules). If only one database driver is available, the wizard will show a notice and a link to the documentation on how to install additional PHP modules.

When you select a database other than SQLite, the wizard exposes additional fields:

- **Database user**: The username to connect to the database server. If this user has sufficient privileges (e.g. the ability to query `mysql.user` for MySQL, or the `CREATEROLE` privilege for PostgreSQL), the wizard will attempt to create a dedicated Nextcloud database user with limited privileges (see below). If the user lacks those privileges, the wizard gracefully falls back to using the provided credentials directly.
- **Database password**: The password for the database user above.
- **Database name**: The name you want for your Nextcloud database. The wizard will create it if it does not already exist and the user has `CREATE DATABASE` privileges.
- **Database host**: The hostname (and optionally port) of your database server, e.g. `localhost` or `db.example.com:3306`. The default is `localhost`. You can also specify a Unix socket path here. The wizard shows a helper hint: *"Please specify the port number along with the host name (e.g., localhost:5432)."*
- **Database tablespace** *(Oracle only)*: Shown only when Oracle is selected.

Automatic database user creation

When the provided database user has administrative privileges, the installer attempts to create a dedicated database user with privileges limited to the Nextcloud database. This avoids storing your administrative database credentials in `config.php`.

If privileges are sufficient, the install creates a user named `oc_admin`. If that user already exists, a numeric suffix is appended (`oc_admin1`, `oc_admin2`, etc.) until an available username is found.

A random password is generated for the new user. The resulting credentials are written into `config.php`:

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

If the provided user lacks the privileges to create new database users, the installer falls back to using the provided credentials directly.

Tip

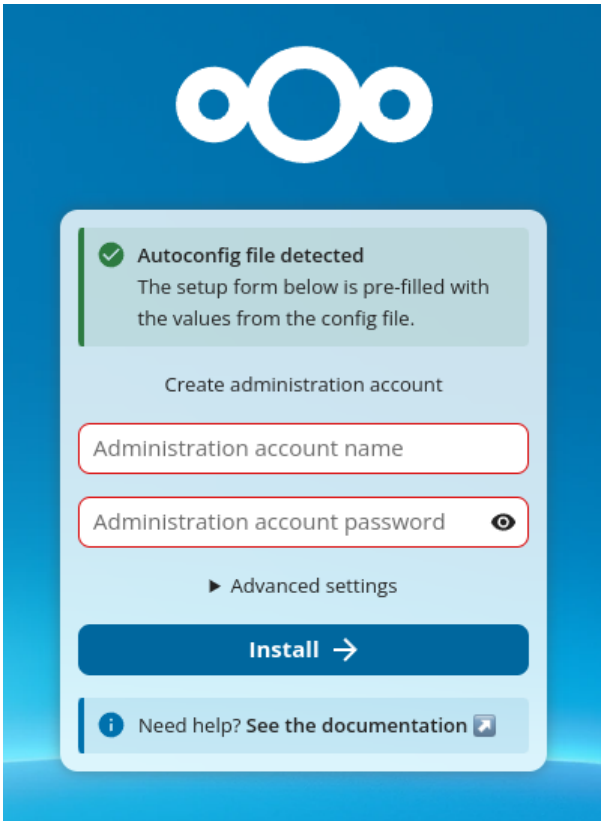
You can also explicitly prevent automatic user creation by setting the following in your `config.php` before running the wizard (or via an autoconfig file):

```
'setup_create_db_user' => false,
```

This is useful when your database administrator has already created a dedicated user for Nextcloud. In that case the wizard will use the database credentials you provide directly, without attempting to create a new user or query administrative privileges.

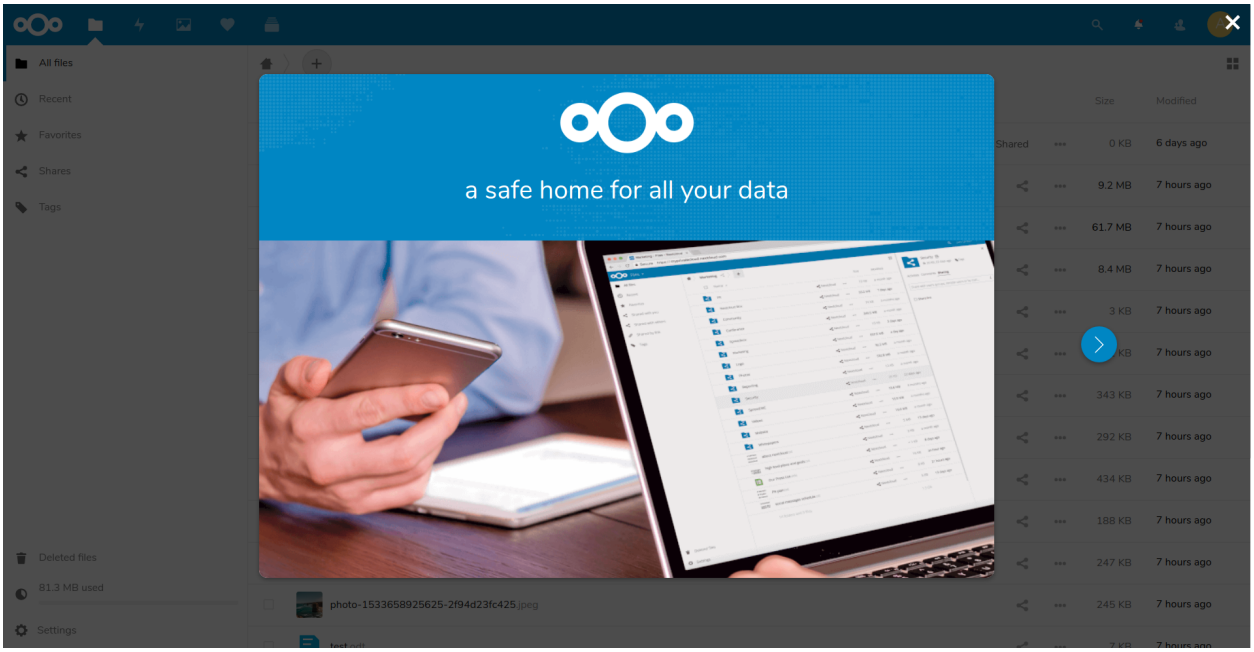
Autoconfig

If an autoconfig file is detected, the wizard displays a success notice: *"Autoconfig file detected — The setup form below is pre-filled with the values from the config file."* The **Storage & database** section is automatically collapsed when the autoconfig provides valid values. For details on autoconfig files, see *Automated installation with autoconfig.php*.



Completing Installation

Click **Install**, and start using your new Nextcloud server.



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

6.5.4 Trusted domains

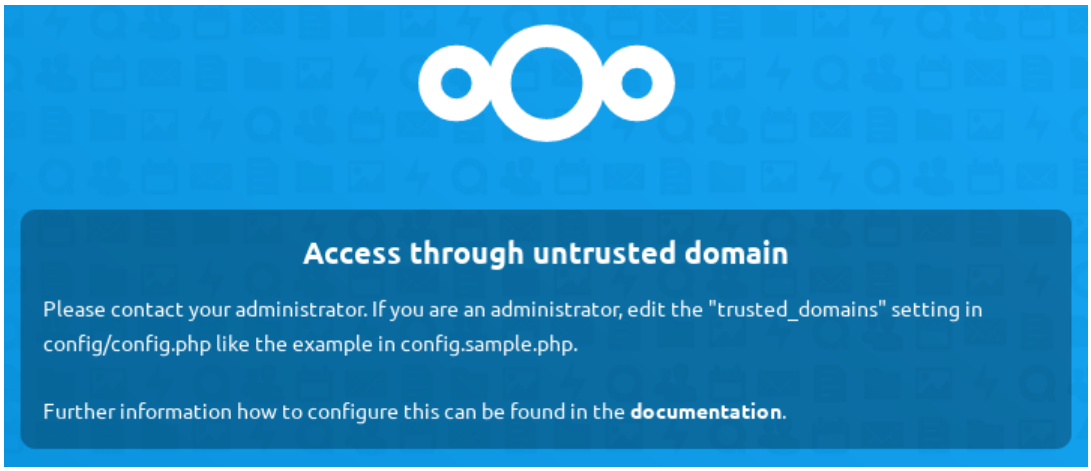
All URLs used to access your Nextcloud server must be whitelisted in your `config.php` file, under the `trusted_domains` setting. Users are allowed to log into Nextcloud only when they point their browsers to a URL that is listed in the `trusted_domains` setting. This is not a list of allowed client-side domains or IP addresses. You may use IP addresses and domain names. Wildcard patterns using `*` are also supported (e.g. `*.example.com`). A typical configuration looks like this:

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

Note

The loopback addresses `localhost`, `127.0.0.1`, and `[::1]` are always treated as trusted regardless of the `trusted_domains` configuration. This means that as long as you have access to the physical server you can always log in. In the event that a load balancer or reverse proxy 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:



6.6 Install from the command line

You can install Nextcloud without using the web-based Installation Wizard. This is useful for automated deployments, headless servers, and administrators who prefer the command line.

Before continuing, download and unpack the Nextcloud archive, configure the web server and PHP, and create the database and database user when using MySQL, MariaDB, PostgreSQL, or Oracle.

For the complete manual-installation prerequisites, see *Installation on Linux*.

1. Ensure that the HTTP user has read and write access to all Nextcloud directories. Changing ownership is the usual approach; for example, on Debian/Ubuntu:

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

Ownership is not strictly required. For example, where changing ownership is not possible, add the HTTP user to the group that owns the directories and ensure that they are group-writable. See *Running occ*.

2. Use the `occ` command to complete the installation. This takes the place of the graphical Installation Wizard. Run `occ` as the HTTP user so that ownership and permissions remain consistent with the web server:

```
$ sudo -E -u www-data php /var/www/nextcloud/occ maintenance:install \
  --database mysql --database-name nextcloud \
  --database-user nextcloud \
  --admin-user admin
```

This command prompts for the database and administrator passwords. For unattended installations, specify `--database-pass` and `--admin-pass` using your deployment system's protected secret-handling mechanism. Avoid exposing production passwords in shell history or process listings.

You can invoke `occ` from any directory when using its absolute path, as in the example above. Alternatively, change to the Nextcloud root directory and run `sudo -E -u www-data php occ maintenance:install`.

Use `--database-host` and, if necessary, `--database-port` when the database runs on another host. Use `--data-dir` to select a data directory other than the default. To view all supported options, run:

```
$ sudo -E -u www-data php /var/www/nextcloud/occ maintenance:install --help
```

When the database runs on another host, encrypt the connection using the `--database-ssl-*` options described in *Encrypted database connection* below.

Supported databases are:

- `sqlite` (SQLite)
- `mysql` (MySQL/MariaDB)
- `pgsql` (PostgreSQL)
- `oci` (Oracle; contact Nextcloud GmbH for enterprise support)

The selected database requires its corresponding PHP extension or driver to be installed and enabled.

6.6.1 Encrypted database connection

Added in version 35.

When the database does not run on the same host as Nextcloud, the connection should be encrypted so that the credentials and all queries are not sent in plaintext. The following options configure this during the installation, so that the installation itself already uses an encrypted connection:

Table 2: SSL/TLS options of `maintenance:install`

Option	Supported by	Description
`--database-ssl-mode`	`pgsql`	Encryption mode of the connection, for example `require` or `verify-full`. See the PostgreSQL documentation for the available modes.
`--database-ssl-ca`	`mysql`, `pgsql`	Path to the CA certificate the database server is verified against.
`--database-ssl-cert`	`mysql`, `pgsql`	Path to the client certificate used to authenticate against the database server.
`--database-ssl-key`	`mysql`, `pgsql`	Path to the private key belonging to the client certificate.
`--database-ssl-crl`	`pgsql`	Path to the certificate revocation list.
`--database-ssl-no-verify`	`mysql`	Do not verify that the certificate of the database server matches the hostname used to connect. MySQL and MariaDB verify this by default, PostgreSQL only in the `verify-full` mode.

`--database-ssl-cert` and `--database-ssl-key` have to be provided together. Passing an option that the selected database does not support aborts the installation with an error — `sqlite` and `oci` support none of them. The certificates and keys have to be readable by the PHP process.

This example installs Nextcloud with a remote MySQL database, verifying the server against a CA certificate:

```
$ sudo -E -u www-data php /var/www/nextcloud/occ maintenance:install \
  --database mysql --database-name nextcloud \
  --database-host db.example.com \
  --database-user nextcloud \
  --database-ssl-ca /etc/ssl/nextcloud/ca-cert.pem \
  --admin-user admin
```

Nextcloud writes the resulting configuration to `dbdriveroptions` (MySQL/MariaDB) or `pgsql_ssl` (PostgreSQL) in `config.php`, see *Configuration Parameters*. The same connection can be configured for the Installation Wizard, see *Encrypted database connection*.

See *Command line installation* for more information.

6.7 Automated installation with `autoconfig.php`

Use `config/autoconfig.php` to prefill, or fully automate, the web-based initial setup of Nextcloud. This is useful when installing multiple identical or similar instances.

Create `config/autoconfig.php` and include the installation parameters that you want to prefill. Any values not provided in the file can be completed in the setup form when you first launch Nextcloud.

To complete the installation without interaction, provide a valid database configuration, data directory, and both `adminlogin` and `adminpass`. Otherwise, Nextcloud displays the setup form so that missing or invalid values can be corrected.

Nextcloud removes `config/autoconfig.php` after a successful installation. The file remains in place if installation fails.

Note

The supplied database account must be able to connect to the database server and have sufficient privileges for the selected configuration. Depending on the database backend and the account's privileges, Nextcloud may create the database and a dedicated database user. Alternatively, create the database and database user in advance, and grant the user the required privileges, as described in *Set up a database server*.

Warning

`autoconfig.php` commonly contains plaintext database and administrator passwords. Restrict access to the file before writing secrets to it. If installation fails, remove the file or rotate any exposed secrets.

6.7.1 Parameters

The following parameters are commonly used:

- `directory` (written to `config.php` as `datadirectory`)
- `dbtype`, `dbname`, `dbuser`, `dbpass`, `dbhost`, and `dbtableprefix`
- `adminlogin` and `adminpass`
- optionally, `trusted_domains` and `adminemail`
- optionally, the parameters of an *encrypted database connection*

Two parameters have different names from their corresponding `config.php` settings:

autoconfig.php	config.php
directory	datadirectory
dbpass	dbpassword

6.7.2 Encrypted database connection

Added in version 35.

An SSL/TLS encrypted connection to the database server can be configured during the installation. Use this when the database does not run on the same host as Nextcloud, so that the credentials and all queries are not sent in plaintext.

The following parameters are independent of the database backend. Nextcloud translates them into the corresponding `config.php` settings before the first connection is opened, so the installation itself already uses an encrypted connection.

Table 3: Connection encryption parameters

Parameter	Supported by	Description
`dbsslmode`	PostgreSQL	Encryption mode of the connection, for example `require` or `verify-full`. See the PostgreSQL documentation for the available modes.
`dbsslca`	MySQL/MariaDB, PostgreSQL	Path to the CA certificate the database server is verified against.
`dbsslcert`	MySQL/MariaDB, PostgreSQL	Path to the client certificate used to authenticate against the database server.
`dbsslkey`	MySQL/MariaDB, PostgreSQL	Path to the private key belonging to the client certificate.
`dbsslcrl`	PostgreSQL	Path to the certificate revocation list.
`dbsslnoverify`	MySQL/MariaDB	Set to `true` to not verify that the certificate of the database server matches the hostname used to connect. MySQL and MariaDB verify this by default, PostgreSQL only in the `verify-full` mode.

Note the following restrictions:

- `dbsslcert` and `dbsslkey` have to be provided together.
- A parameter that the selected database does not support is rejected with an error instead of being ignored, and the installation does not proceed. SQLite and Oracle support none of them — an Oracle connection is encrypted through the connect string and `sqlnet.ora` instead.
- The certificates and keys have to be readable by the PHP process.

Alternatively, the backend-specific `dbdriveroptions` (MySQL/MariaDB) and `pgsql_ssl` (PostgreSQL) settings, as documented in *Configuration Parameters*, can be written to `autoconfig.php` verbatim. They are passed to `config.php` as provided and are not validated. Values set through the parameters above take precedence over individual entries of these arrays.

6.7.3 Examples

The following examples show partial and complete automatic configurations.

Data Directory

This configuration prefills the data directory. Complete the database and administrator-account settings in the setup form.

```
<?php
$AUTOCONFIG = [
  "directory"     => "/www/htdocs/nextcloud/data",
];
```

SQLite database

This configuration prefills the SQLite database settings. Complete the data directory and administrator-account settings in the setup form.

```
<?php
$AUTOCONFIG = [
  "dbtype"        => "sqlite",
  "dbname"        => "nextcloud",
```

(continues on next page)

(continued from previous page)

```
  "dbtableprefix" => "",
];
```

MySQL / MariaDB database

This configuration prefills the MySQL or MariaDB settings. Complete the data directory and administrator-account settings in the setup form.

```
<?php
$AUTOCONFIG = [
  "dbtype"        => "mysql",
  "dbname"        => "nextcloud",
  "dbuser"        => "username",
  "dbpass"        => "password",
  "dbhost"        => "localhost",
  "dbtableprefix" => "",
];
```

PostgreSQL database

This configuration prefills the PostgreSQL settings. Complete the data directory and administrator-account settings in the setup form.

```
<?php
$AUTOCONFIG = [
  "dbtype"        => "pgsql",
  "dbname"        => "nextcloud",
  "dbuser"        => "username",
  "dbpass"        => "password",
  "dbhost"        => "localhost",
  "dbtableprefix" => "",
];
```

Encrypted MySQL / MariaDB connection

This configuration connects to a remote MySQL or MariaDB server over an encrypted connection, verifying the server against a CA certificate and authenticating with a client certificate.

```
<?php
$AUTOCONFIG = [
  "dbtype"        => "mysql",
  "dbname"        => "nextcloud",
  "dbuser"        => "username",
  "dbpass"        => "password",
  "dbhost"        => "db.example.com",
  "dbtableprefix" => "",
  "dbsslca"       => "/etc/ssl/nextcloud/ca-cert.pem",
  "dbsslcert"     => "/etc/ssl/nextcloud/client-cert.pem",
  "dbsslkey"      => "/etc/ssl/nextcloud/client-key.pem",
];
```

Nextcloud stores these paths as `dbdriveroptions` in `config.php`.

Encrypted PostgreSQL connection

This configuration requires an encrypted connection to a remote PostgreSQL server and verifies both the certificate of the server and the hostname it was issued for.

```
<?php
$AUTOCONFIG = [
  "dbtype"        => "pgsql",
  "dbname"        => "nextcloud",
  "dbuser"        => "username",
  "dbpass"        => "password",
  "dbhost"        => "db.example.com",
  "dbtableprefix" => "",
  "dbsslmode"     => "verify-full",
  "dbsslca"       => "/etc/ssl/nextcloud/ca-cert.pem",
];
```

Nextcloud stores these values as `pgsql_ssl` in `config.php`.

Complete non-interactive setup

When all required values are present and valid, the installation proceeds without requiring user interaction.

The following configuration bypasses the setup form:

```
<?php
$AUTOCONFIG = [
  "dbtype"        => "mysql",
  "dbname"        => "nextcloud",
  "dbuser"        => "username",
  "dbpass"        => "password",
  "dbhost"        => "localhost",
  "dbtableprefix" => "",
  "adminlogin"    => "root",
  "adminpass"     => "root-password",
  "directory"     => "/www/htdocs/nextcloud/data",
];
```

6.8 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.

> **Tip**
>
> Permission problems may be caused by SELinux *even if the denial is not indicated in the audit logs.* This is because SELinux does not log all system calls used for verifying access. See Possible Causes of Silent Denials to solve.

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(/.*)?'
```

(continues on next page)

(continued from previous page)

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

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

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

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

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

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

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

6.8.1 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
```

6.8.2 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
```

6.8.3 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
```

6.8.4 Allow access to LDAP server

Use this setting to allow LDAP connections:

```
setsebool -P httpd_can_connect_ldap on
```

6.8.5 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
```

6.8.6 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
```

6.8.7 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
```

6.8.8 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
```

6.8.9 Allow access to NFS

If you have placed your datadir on an NFS share use the following setting:

```
setsebool -P httpd_use_nfs on
```

6.8.10 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
```

6.8.11 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
```

6.8.12 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.

6.9 NGINX configuration

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

- You need to insert the following code into **your Nginx configuration file**. Choose the appropriate example based on whether you are deploying *Nextcloud in the webroot of NGINX* (i.e. `https://cloud.example.com/`) or *Nextcloud in a subdir of the NGINX webroot* (i.e. `https://cloud.example.com/nextcloud`).
- Adjust the server directive under `upstream php-handler` to match your PHP installation's configured FPM listener (a misconfiguration here will result in a `502 Bad Gateway` - see *PHP-Handler Configuration / Avoiding "502 Bad Gateway"* for details)
- Adjust the existing `server_name` directives found under *both* `server` sections to your real hostname
- Adjust `root` to the webroot of your Nextcloud installation
- Adjust the `ssl_certificate` and `ssl_certificate_key` directives to the real paths for your signed certificate and private key. Make sure your SSL certificates are readable by the nginx server process (see nginx HTTPS SSL Module documentation).
- If using Let's Encrypt as TLS certificate and nginx as webserver, set *ssl_stapling* and *ssl_stapling_verify* to *off* in main nginx config (see [Let's Encrypt blog post](https://letsencrypt.org/2024/12/05/ending-ocsp)).
- Be careful about line breaks if you copy the examples, as long lines may be broken for page display and result in an invalid configuration files.
- Some environments might need a `cgi.fix_pathinfo` set to `1` in their `php.ini`.

6.9.1 Nextcloud in the webroot of NGINX

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

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

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

(continues on next page)

(continued from previous page)

```
}

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

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

    # Prevent nginx HTTP Server Detection
    server_tokens off;

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

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

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

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

    # Prevent nginx HTTP Server Detection
    server_tokens off;

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

    # set max upload size and increase upload timeout:
    client_max_body_size 512M;
```

(continues on next page)

(continued from previous page)

```
    client_body_timeout 300s;
    fastcgi_buffers 64 4K;

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

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

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

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

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

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

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

(continues on next page)

(continued from previous page)

```
    }

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

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

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

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

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

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

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

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

(continues on next page)

(continued from previous page)

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

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

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

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

        include fastcgi_params;
        fastcgi_pass php-handler;

        fastcgi_param SCRIPT_FILENAME            $document_root$fastcgi_script_name;
        fastcgi_param PATH_INFO                  $path_info;
        fastcgi_param HTTPS                      on;       # Assumes TLS terminates␣
↪here
        fastcgi_param modHeadersAvailable        true;     # Avoid duplicate security␣
```

(continues on next page)

(continued from previous page)

```
↪headers
        fastcgi_param front_controller_active    true;     # Enable pretty URLs

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

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

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

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

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

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

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

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

(continues on next page)

(continued from previous page)

```
    }

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

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

6.9.2 Nextcloud in a subdir of the NGINX webroot

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

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

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

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

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

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

(continues on next page)

(continued from previous page)

```
    # Prevent nginx HTTP Server Detection
    server_tokens off;

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

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

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

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

    # Prevent nginx HTTP Server Detection
    server_tokens off;

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

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

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

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

(continues on next page)

(continued from previous page)

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

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

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

    location ^~ /nextcloud {
        # set max upload size and increase upload timeout:
        client_max_body_size 512M;
        client_body_timeout 300s;
        fastcgi_buffers 64 4K;

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

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

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

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

        # HSTS settings
        # WARNING: Only add the preload option once you read about
        # the consequences in https://hstspreload.org/. This option
        # will add the domain to a hardcoded list that is shipped
```

(continues on next page)

(continued from previous page)

```
        # in all major browsers and getting removed from this list
        # could take several months.
        #add_header Strict-Transport-Security "max-age=31536000; includeSubDomains;␣
↪preload" always;

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

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

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

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

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

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

        # Pass PHP requests to PHP-FPM.
        #
        # Important: this block must appear above the static asset locations
        # below. Those locations fall back to `/nextcloud/index.php$request_uri`;
        # if they appear first, nginx can repeatedly rewrite to
```

(continues on next page)

(continued from previous page)

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

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

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

            include fastcgi_params;
            fastcgi_pass php-handler;

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

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

            # Required for uploads: PHP-FPM does not support chunked
```

(continues on next page)

(continued from previous page)

```
            # transfer encoding and needs a Content-Length header.
            fastcgi_request_buffering on;

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

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

        # Serve static files
        location ~ \.(?
↪:css|js|mjs|svg|gif|ico|jpg|png|webp|wasm|tflite|map|ogg|flac|mp4|webm)$ {
            try_files $uri /nextcloud/index.php$request_uri;
            # HTTP response headers borrowed from Nextcloud `.htaccess`
            add_header Cache-Control                    "public, max-age=15778463
↪$asset_immutable";
            add_header Referrer-Policy                  "no-referrer"       always;
            add_header X-Content-Type-Options           "nosniff"           always;
            add_header X-Frame-Options                  "SAMEORIGIN"        always;
            add_header X-Permitted-Cross-Domain-Policies "none"             always;
            add_header X-Robots-Tag                     "noindex, nofollow" always;
            access_log off;     # Optional: Don't log access to assets
        }

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

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

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

6.9.3 Tips and tricks

PHP-Handler Configuration / Avoiding “502 Bad Gateway”

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

for NC.

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

Look for the line that is set to something like:

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

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

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

Suppressing log messages

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

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

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

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

This could be caused by the:

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

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

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

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

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

Upload of files greater than 10 MiB fails

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

You may require to change:

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

to the following to re-allow file uploads:

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

See issue #8802 on nextcloud/server for more information.

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

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

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

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

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

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

Trusted proxy not detected when using a Unix domain socket

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

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

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

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

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

"Access through untrusted domain" error with HTTP/3

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

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

```
fastcgi_param  HTTP_HOST  $host;
```

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

6.10 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 where 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.

6.10.1 Passwords

Storage of account passwords

Nextcloud's built-in database user backend stores a salted, one-way hash of each account password. It prefers Argon2id (when supported by the PHP installation), with Argon2i and bcrypt used as fallbacks. The algorithm, salt, and cost parameters are included in the stored hash. Existing hashes are automatically upgraded following successful password verification when they no longer match the preferred algorithm or parameters.

The hash is used to verify password-based login attempts and is not designed to be decrypted. When an external user backend (such as LDAP) is used, storage and verification of the account password are controlled by that backend.

This account-password hash is separate from any recoverable copy of the login password that Nextcloud stores in connection with authentication tokens, as described below.

Storage of authentication tokens

After successful authentication, Nextcloud issues an authentication token that the client presents with subsequent authenticated requests. A valid token can authenticate as the associated user, subject to the token's scope, expiration, type, and server-side validity checks. Depending on the token type and client, the token may be transmitted in a session cookie, used as an app password, or sent as a bearer token.

Nextcloud does not store the plaintext authentication token in the database. Instead, it stores a SHA-512 hash derived from the token and the instance-specific `secret`. The corresponding server-side token record contains the associated user identity, authentication metadata, and cryptographic key material. Authentication tokens must therefore be protected like passwords. They should not be logged, placed in URLs, or intentionally persisted outside the client that uses them.

Token-associated storage of login passwords

By default, `auth.storeCryptedPassword` is enabled. When this setting is enabled and the login password is available during token creation, Nextcloud stores a reversibly encrypted copy of it in the server-side authentication-token record.

For an account using the built-in database user backend, this encrypted copy is separate from the one-way account-password hash. The database contains the one-way account-password hash and also contains an encrypted password copy for each authentication-token record created with password storage enabled and a login password supplied.

The recoverable copy is used by features that need the original login credentials, such as connecting to external storage, autoconfiguring accounts in the Mail app, and periodically checking whether the login credentials remain valid. When token creation receives no login password, the resulting token record contains no recoverable password. Such a token record remains without a stored password until Nextcloud receives the password during a later login or password update.

The authentication token itself does not contain the login password. Each token record has a separate RSA key pair. Nextcloud encrypts the login password with the record's public key and encrypts the corresponding private key using the authentication token together with the instance-specific `secret` from `config.php`. Possession of the authentication token, instance secret, and corresponding database record is therefore sufficient to decrypt a password stored in that record.

Administrators can disable this behavior with `auth.storeCryptedPassword`. Disabling it does not affect the one-way account-password hash used by the built-in database user backend. Features that rely on recovering the login password from an authenticate-token record cannot retrieve it from records created without a stored password.

When an authentication token contains a stored password, Nextcloud periodically checks that password against the user backend. If the password is no longer valid, Nextcloud marks the token as having an invalid password and rejects authentication with that token. When the token contains no stored password, Nextcloud skips this password check. Consequently, changing a password directly in an external user backend does not cause a token without a stored password to be rejected through the periodic credential check. The password change alone does not invalidate the token; the token remains valid until it expires, is otherwise invalidated, or the user is disabled.

Security consequences

Leakage of authentication data has the following security consequences:

- An actor with a valid authentication token can authenticate as the associated user, subject to the token's scope, expiration, type, and server-side validity checks.
- An actor with the authentication token, the instance-specific `secret` from `config.php`, and the corresponding database record can decrypt the login password stored in that record.
- An account-password hash does not reveal the original password directly, but an actor who obtains it can perform offline password-guessing attacks.

Password Length Limits

Nextcloud accepts account passwords of up to 469 bytes through its standard account-creation, password-change, and password-reset interfaces. This is the maximum account-password length enforced by these interfaces. Because the limit is measured in bytes, a password containing multibyte characters (such as emojis or characters from non-Latin scripts) can reach the limit with fewer than 469 characters.

Administrators can use the *Password Policy app* to configure requirements such as a minimum password length and other complexity rules. External user backends can impose additional or different requirements.

The following implementation details do not change the 469-byte account-password maximum, but are relevant when selecting a password policy:

Token Encryption Performance
: When `auth.storeCryptedPassword` is enabled and an account password is longer than 214 bytes, Nextcloud uses a larger RSA key when creating authentication-token records. This increases token-generation overhead, but does not prevent passwords between 215 and 469 bytes from being accepted. The 214-byte threshold is therefore a performance consideration, not a password-length limit.

  Administrators who expect very long or one-time passwords to be used may consider disabling `auth.storeCryptedPassword` to avoid this overhead, subject to the functional consequences described above.

Algorithmic Truncation (bcrypt fallback)
: Nextcloud prefers Argon2id for one-way password hashing (when supported by the PHP installation), with Argon2i and bcrypt as fallbacks. Bcrypt considers only the first 72 bytes of its input. Therefore, if bcrypt is selected, input after the first 72 bytes does not contribute to password verification. This is a bcrypt-specific behavior, not a general 72-byte limit imposed by Nextcloud.

Passwords protecting public link and mail shares use the same one-way password hasher and are subject to the applicable share-password policy. They are not stored in authentication-token records, so the encryption-related performance considerations above do not apply.

6.10.2 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.

6.10.3 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.

Place config directory outside of the web root

You can move the `config/` directory outside the web root using the `NEXTCLOUD_CONFIG_DIR` environment variable. This ensures `config.php` — which contains database credentials, secret keys, and other sensitive values — is not accessible via HTTP even in the event of a web server misconfiguration.

Set the variable in your web server virtual host configuration:

```
# Apache
SetEnv NEXTCLOUD_CONFIG_DIR /etc/nextcloud
```

```
# nginx — set via fastcgi_param or the PHP-FPM pool's env[] setting
fastcgi_param NEXTCLOUD_CONFIG_DIR /etc/nextcloud;
```

Also set it for CLI work (`occ`, cron):

```
export NEXTCLOUD_CONFIG_DIR=/etc/nextcloud
```

> **Note**
>
> The variable must be set for **both** the web server process and CLI invocations. Verify with `occ config:list system` after changing it.

> **See also**
>
> *Configuration Parameters* for full details on `NEXTCLOUD_CONFIG_DIR` and other configuration loading behaviour.

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. 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.

Disable Debug Mode

Verify that `debug` is `false` in your `config.php`. The default is `false` in new installations (or when not specified). It should not be enabled in production environments or outside of targeted troubleshooting situations. When enabled, things like server-wide WebDAV collection listings are permitted. It is intended for local development and usage in controlled environments only.

6.10.4 Use HTTPS

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

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

Redirect all unencrypted traffic to HTTPS

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

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

Enable HTTP Strict Transport Security

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

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

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

Warning

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

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

This requires the `mod_headers` extension in Apache.

Proper SSL configuration

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

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

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

6.10.5 Restrict admin actions to a specific range of IP addresses

Configure `allowed_admin_ranges` in `config.php` to restrict the admin actions to trusted IP ranges.

This can be achieved with this kind of setting, usually using private IP ranges:

```
'allowed_admin_ranges' => [
  '127.0.0.1/8',
  '192.168.0.0/16',
  'fd00::/8',
],
```

All requests originating from IP addresses outside of these ranges will not be able to execute admin actions.

Administrators connected from untrusted IP addresses will be able to use Nextcloud, but all admin specific actions will be hidden.

6.10.6 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.

6.10.7 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.

6.10.8 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-Robots-Tag: noindex, nofollow`**
  - Instructs search machines to not index these pages and not follow any links there.
- **`X-Frame-Options: SAMEORIGIN`**
  - Prevents embedding of the Nextcloud instance within an iframe from other domains to prevent Clickjacking and other similar attacks.
- **`Referrer-Policy: no-referrer`**
  - The default *no-referrer* policy instructs the browser not to send referrer information along with requests to any origin.

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

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

- mod_headers
- mod_env

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

6.10.9 Connections to remote servers

Some functionalities require the Nextcloud server to be able to connect remote systems via https/443. This paragraph also includes the data which is being transmitted to the Nextcloud GmbH. Depending on your server setup, these are the possible connections:

- **connectivity.nextcloud.com, www.eff.org, edri.org**
  - optional (config)
  - for checking the internet connection
- **cloud.nextcloud.com**
  - used for enterprise license monitoring
  - submitted data: subscription key, user count
- **updates.nextcloud.com**
  - to check for available Nextcloud server updates
  - submitted data: server version, subscription key, install time, instance id, instance size
- **apps.nextcloud.com, ltd[1-3].nextcloud.com, garm[1-5].nextcloud.com**
  - to check for available apps and their updates
  - source is apps.nextcloud.com the ltd and garm servers are just mirroring the apps.json file
  - submitted data: subscription key
- **github.com, objects.githubusercontent.com, release-assets.githubusercontent.com**
  - to download Nextcloud standard apps
  - to download Nextcloud server releases
- **push-notifications.nextcloud.com**
  - sending push notifications to mobile clients

  - submitted data: unique device identifier, public key, push token

- **pushfeed.nextcloud.com**
  - optional
  - checking for updates to be shown in the Nextcloud Announcements app
- **lookup.nextcloud.com**
  - optional
  - for updating and lookups to the federated sharing addressbook
  - submitted data: *pending*
- **surveyserver.nextcloud.com**
  - optional
  - if the admin has agreed to share anonymized server data
  - submitted data: statistical data. see here for the detailed field list
- **nominatim.openstreetmap.org**
  - optional
  - if the weather status app is enabled and used
  - submitted data: address manually entered by the user to resolve to longitude and latitude
- **api.opentopodata.org**
  - optional
  - if the weather status app is enabled and used
  - submitted data: address manually entered by the user to resolve the altitude of the location
- **api.met.no**
  - optional
  - if the weather status app is enabled and used
  - submitted data: longitude and latitude configured in the weather status app by the individual user
- Any remote Nextcloud server that is connected with federated sharing
- When downloading apps from the App store other domains might be accessed, based on the choice of the app developers where they host the releases. For all official Nextcloud apps this is not the case though, because they are hosted on Github.

6.10.10 Setup fail2ban

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

This guide will enable blocking of the originating IP addresses at an operating system level, so the webserver, PHP and the database do not need to handle this unnecessary traffic at all.

Nextcloud prerequisites

Nextcloud logs failed login attempts in `nextcloud.log` with log level `2`, so you need to define a `loglevel` of `2` or less in `config.php`.

Make sure your `nextcloud.log` is writeable by your webserver user, possibly by defining a correct `logfilemode` in `config.php`.

Perform a bad login attempt and check whether it does get logged to `nextcloud.log`.

Note that `audit.log` (if enabled) currently only logs successful logins and cannot be used.

Fail2ban introduction

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

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

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

Setup a filter and a jail for Nextcloud

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

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

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

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

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

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

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

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

```
fail2ban-client status nextcloud
```

If you need to unban certain IP addresses (`1.2.3.4` in this example), you may do so by issuing:

```
fail2ban-client unban 1.2.3.4
```

There may be scenarios where you want to more permanently ban certain IP addresses that repeatedly generate bad login attempts (or other attacks) by using fail2ban's `recidive` feature.

6.11 Server tuning

This page collects configuration changes that can improve the performance of your Nextcloud server. Most items only require editing a configuration file or installing a package, while a few involve additional services. Start with the ones that match your setup and revisit the rest as your instance grows.

6.11.1 Using cron to perform background jobs

See *Background jobs* for a description and the benefits.

6.11.2 Reducing system load

High system load will slow down Nextcloud and may also lead to other unwanted side effects. To reduce load, you should first identify the source of the problem. Tools such as htop, iotop, netdata, or glances can help you identify the process or drive that slows down your system. First, make sure that you have installed and assigned enough RAM. Minimize swap usage as much as possible, as excessive swapping can severely degrade performance. If you run your database inside a VM, use a dedicated block device for database storage rather than storing it inside the VM's disk image file, to reduce latency caused by multiple abstraction layers.

6.11.3 Log Levels

Verify the `loglevel` in your `config.php` file. The default log level is set to `2` (WARN) in new installations. Sometimes this parameter is inadvertently left at the DEBUG level (`0`) after troubleshooting. In some older installations, this parameter may also be something other than the default. Use `0` (DEBUG) when you have a problem to diagnose, and then reset your log level to a less-verbose level. DEBUG outputs a lot of information, and can affect your server performance.

6.11.4 Debug Mode

Verify that `debug` is set to `false` in your `config.php` file. The default is `false` in new installations (or when not specified). While similar to the DEBUG logging level, this option also disables various optimizations (to facilitate easier debugging) and generates additional debug output both at the browser level and server-side. It should not be enabled in production environments except during isolated troubleshooting.

6.11.5 Caching

Caching improves performance by storing data, code, and other objects in memory. Memory caching is not enabled by default because it requires optional extensions (such as APCu) and/or system components (e.g., Redis). Although these add-ons are generally not challenging to install and activate—at least in single-server deployments—you must install them before enabling their use in Nextcloud. See *Memory caching* for guidance.

6.11.6 Compression

Enabling compression in your web server for JavaScript, CSS, and SVG files improves performance because less data is transferred to clients.

6.11.7 Replacing SQLite

SQLite is a suitable database for some use cases, but using MariaDB, MySQL, or PostgreSQL can be more beneficial with Nextcloud.

If you do not select a database at installation time, SQLite is used by default because it does not require any external components.

However, MySQL/MariaDB or PostgreSQL are generally recommended for Nextcloud because of the performance limitations of SQLite with highly concurrent applications, like Nextcloud.

If your installation is already running on SQLite, you can convert to MySQL or MariaDB using the steps provided in *Convert to a different database type*.

See the section *Set up a database server* for instructions on configuring Nextcloud for MySQL or MariaDB.

6.11.8 Tuning your database

Databases are not plug-and-play. They benefit not only from basic configuration for compatibility with Nextcloud, but also from tuning within the environment in which they are deployed. This tuning should be based on your hardware, storage, usage patterns, underlying operating system, priorities, and other factors.

For more details and help tuning your database:

- MariaDB – Optimization and Tuning
- PostgreSQL – Resource Consumption
- PostgreSQL – Tuning Your PostgreSQL Server

6.11.9 Using Redis-based transactional file locking

Transactional File Locking uses the database as the default backend. This places additional load on your database. See the section *Transactional file locking* for instructions on configuring Nextcloud to use Redis-based Transactional File Locking.

6.11.10 TLS / encryption app

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

Here are some examples of 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 `aes`, the extension is present.
- For Intel processors, you can search the Intel ARK database to check if your CPU supports AES-NI. Use the Intel Processor Feature Filter, filtering by "AES New Instructions".
- 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 supported hardware. You can check the OpenSSL version via `openssl version -a`.
- If your processor supports AES-NI but it does not show up via `grep` or `coreinfo`, it may be just be disabled in the BIOS. Check your BIOS settings.
- If your environment runs virtualized, check the virtualization vendor for support.

6.11.11 Enable HTTP/2 for faster loading

HTTP/2 has huge speed improvements over HTTP with multiple requests. Most *browsers already support HTTP/2 over TLS (HTTPS)*.

6.11.12 Tune PHP-FPM

PHP-FPM is required for Nginx setups and is widely used with Apache as well. Its default configuration is extremely conservative: the default pool has `pm.max_children = 5`, which limits Nextcloud to five simultaneous PHP requests and is a common cause of gateway timeouts, slow page loads, and sync client errors under any real load.

Process manager modes

The `pm` directive controls how PHP-FPM manages its worker processes:

`dynamic`
: Keeps between `pm.min_spare_servers` and `pm.max_spare_servers` idle workers alive, up to a ceiling of `pm.max_children`. Good default for most Nextcloud installations: balances RAM efficiency with burst capacity. Set `pm.min_spare_servers` high enough that sync-client poll bursts do not stall waiting for new processes to spawn.

`static`
: Always keeps exactly `pm.max_children` processes running. Highest memory use, lowest latency. Use on dedicated servers with predictable load. Always set `pm.max_requests` to recycle workers and prevent memory leaks.

`ondemand`
: Spawns a worker only when a request arrives; kills idle workers after `pm.process_idle_timeout` (default `10s`). Lowest memory use but adds cold-start latency on every burst. Not recommended for Nextcloud: desktop and mobile clients poll every 30 seconds, repeatedly triggering cold starts.

Key parameters

`pm.max_children`
: Maximum (or fixed, under `static`) number of simultaneous worker processes. This is the most important value to tune. If all workers are busy, new requests queue up; a full queue produces 502/504 errors.

  Estimate it from available RAM:

```
pm.max_children = floor(available_RAM_for_PHP / average_worker_RSS)
```

  Measure the average RSS of a running pool:

```
ps --no-headers -o rss -C php-fpm | awk '{sum+=$1; count++} END {if (count>0)␣
↪print sum/count/1024 " MB"; else print "No php-fpm processes found"}'
```

  A typical Nextcloud worker uses **50–100 MB** (more if Imagick or LDAP is loaded). Leave headroom for the OS, web server, database, and cache. Setting `pm.max_children` too high causes swapping, which is worse than queuing.

`pm.start_servers` ***(dynamic only)***
: Workers started at FPM boot. Defaults to `(pm.min_spare_servers + pm.max_spare_servers) / 2` if not set.

`pm.min_spare_servers` / `pm.max_spare_servers` ***(dynamic only)***
: Range of idle workers kept warm. For Nextcloud, keep `pm.min_spare_servers` high enough to absorb a sync-client burst without spawning new processes:

```
pm.min_spare_servers = 4     # adjust upward for many connected clients
pm.max_spare_servers = 16
```

pm.max_requests
: Recycle a worker after this many requests. `0` means never recycle. Setting a value of `500-1000` guards against slow memory growth from leaky extensions (Imagick, LDAP, SAML XML parsers). Essential under `static` mode.

pm.process_idle_timeout ***(ondemand only)***
: How long an idle worker lives before being killed. Default: `10s`.

Example configuration

A starting point for `dynamic` mode on a server with 2 GB of RAM dedicated to PHP (adjust `pm.max_children` based on your measured worker RSS):

```
pm = dynamic
pm.max_children = 30
pm.start_servers = 8
pm.min_spare_servers = 4
pm.max_spare_servers = 16
pm.max_requests = 500
```

Use the PHP-FPM process calculator as a cross-check for your values.

Slow log

Enable the slow log to identify PHP scripts that are taking too long:

```
slowlog = /var/log/php-fpm-slow.log
request_slowlog_timeout = 5s
```

Each entry records the full PHP backtrace of the slow request. This is the fastest way to find the root cause of gateway timeouts and sluggish pages.

Troubleshooting

502 Bad Gateway
: All `pm.max_children` workers are busy. Increase `pm.max_children` if RAM allows. Enable the slow log to check whether a slow query is tying up workers. Also check that a `request_terminate_timeout` is not killing workers mid-request.

504 Gateway Timeout
: A worker is running but not responding within the web server's upstream timeout (nginx `fastcgi_read_timeout`, Apache `ProxyTimeout`). Common Nextcloud causes: large file operations, slow database queries during sync, or PROPFIND over large directory trees. Use the slow log to identify the bottleneck.

Memory grows over time
: Memory leaks can occur in worker processes. Common culprits include libraries that manage external resources (like Imagick for image processing). Use the slow log and RSS monitoring to identify which requests cause growth. Set `pm.max_requests = 500` to recycle them before they grow too large.

Slow first request after idle
: `pm = ondemand` or `pm.min_spare_servers` too low. Switch to `pm = dynamic` and raise `pm.min_spare_servers`.

After any configuration change, reload PHP-FPM — changes do not take effect until you do:

```
sudo systemctl reload php8.3-fpm    # Debian/Ubuntu – adjust version as needed
sudo systemctl reload php-fpm       # RHEL/Fedora
```

For pool configuration details (environment variables, upload sizes, Unix socket vs TCP), see *PHP-FPM configuration* in the installation guide.

6.11.13 Enable PHP OPcache

The OPcache improves the performance of PHP applications by caching precompiled bytecode.

Revalidation

OPcache revalidation in PHP handles changes made to PHP application code stored on disk. Code changes occur whenever:

- Nextcloud or a Nextcloud app is upgraded
- a configuration change is made (e.g. when `config.php` is modified)

Nextcloud, as much as possible, handles cache revalidation internally when required. However, this is not foolproof. In a default PHP environment, revalidation is enabled, and cached scripts are checked for changes on disk every `2` seconds. In many environments, these default values are reasonable and may never need to be changed.

However, the revalidation frequency can be adjusted and may *potentially* enhance performance. We make no recommendations here about appropriate values for revalidation (other than the PHP defaults).

Danger

Increasing the time between revalidations (or disabling it completely) means that changes to scripts, including `config.php`, will take longer to become active (or may never do so if revalidation is disabled completely). Increasing the interval also raises the risk of transient server and application upgrade problems and prevents the proper toggling of maintenance mode.

Warning

If you adjust these parameters, you are more likely to need to restart/reload your web server (`mod_php`) or PHP-FPM after making configuration changes or performing upgrades. If you forget to do so, you may experience unusual behavior due to a mismatch between what is on disk and what is in memory. These may appear to be bugs, but will go away as soon as you restart/reload `mod_php` / fpm.

To change the default from `2` and check for changes on disk at most every `60` seconds, add the following setting to your `php.ini` file:

```
opcache.revalidate_freq = 60
```

Any Server/app upgrades or changes to `config.php` will then require restarting PHP (or otherwise manually clearing the cache or invalidating this particular script).

Warning

Please do not report bugs or odd behavior after upgrading Nextcloud or Nextcloud apps until after you've restarted mod_php/fpm (to confirm the issue is not caused by local revalidation configuration).

Sizing

If any OPcache size limit exceeds 90% of its allocated size, the admin panel will show a related warning and suggest changes.

For more details, check the official PHP documentation. To monitor OPcache usage and clear individual or all cache entries, you can use opcache-gui.

Comments

Nextcloud strictly requires code comments to be preserved in opcode, which is the default. If your PHP settings have changed, ensure the following is set in your `php.ini`:

```
opcache.save_comments = 1
```

JIT

PHP ships with a JIT compiler that can be enabled on x86 platforms to benefit any CPU-intensive apps you might be running. To enable a tracing JIT with all optimizations, add to your `php.ini`:

```
opcache.jit = 1255
opcache.jit_buffer_size = 8M
```

Note

Most Nextcloud instances use less than 2 MiB of the configured JIT buffer size, so 8 MiB is generally sufficient. The overall OPcache usage, however, increases by a larger margin. The PHP parameter `opcache.memory_consumption` might need to be raised in some cases. JIT buffer usage can be monitored with opcache-gui as well.

6.11.14 Previews

It is possible to speed up preview generation using an external microservice: Imaginary.

Warning

Imaginary is currently incompatible with server-side encryption. See https://github.com/nextcloud/server/issues/34262

We strongly recommend running our custom Docker image, which is more up to date than the official image. You can find the image at https://ghcr.io/nextcloud-releases/aio-imaginary. When running it, map a port by adding *-p <port>:9000* to the *docker run* command (or Compose equivalent), e.g.

```
docker run -d -p 9000:9000 --name nextcloud_imaginary --restart always ghcr.io/
↪nextcloud-releases/aio-imaginary:latest
```

Ensure the service is only accessible from your internal servers. Then, configure Nextcloud to use Imaginary by editing your `config.php` file:

```
'enabledPreviewProviders' => [
    'OC\Preview\TXT',
    'OC\Preview\MarkDown',
    'OC\Preview\OpenDocument',
```

(continues on next page)

(continued from previous page)

```
    'OC\Preview\Krita',
    'OC\Preview\Imaginary',
],
'preview_imaginary_url' => 'http://<url of imaginary>:<port>',
```

Warning

Make sure to start Imaginary with the `-return-size` command line parameter. Otherwise, there will be a minor performance impact. The flag requires a recent version of Imaginary (newer than v1.2.4). Also, if running Imaginary in Docker, ensure to add the Docker container capability `SYS_NICE` via `--cap-add=sys_nice` (Docker CLI) or `cap_add: - SYS_NICE` (Docker Compose), as it is required by Imaginary to generate HEIC previews. This is not applicable when running Imaginary outside of Docker.

Note

For large instances, follow Imaginary's scalability recommendation.

Settings

To set the preview format for Imaginary (default is jpeg), add to your `config.php`:

```
'preview_format' => 'webp',
```

To set an API key for Imaginary:

```
'preview_imaginary_key' => 'secret',
```

The default WebP quality setting for preview images is '80'. Change this with:

```
occ config:app:set preview webp_quality --value="30"
```

6.12 Example installation on Ubuntu 24.04 LTS

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

```
sudo apt update && sudo apt upgrade
sudo apt install apache2 mariadb-server libapache2-mod-php php-gd php-mysql \
php-curl php-mbstring php-intl php-gmp php-xml php-imagick php-zip
```

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

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

To start the MySQL command line mode use the following command:

```
sudo mysql
```

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

```
CREATE USER 'username'@'localhost' IDENTIFIED BY 'password';
CREATE DATABASE IF NOT EXISTS nextcloud CHARACTER SET utf8mb4 COLLATE utf8mb4_general_
↪ci;
GRANT ALL PRIVILEGES ON nextcloud.* TO 'username'@'localhost';
FLUSH PRIVILEGES;
```

You can quit the prompt by entering:

```
quit;
```

Now download the archive of the latest Nextcloud version:

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

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

- You may also verify the PGP signature:

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

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

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

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

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

- Finally, change the ownership of your Nextcloud directories to your HTTP user:

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

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

6.12.1 Next steps

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

6.13 Example installation on CentOS 8

Warning

CentOS 8 reached end-of-life on December 31, 2021 and no longer receives security updates. This guide is preserved for reference only and may be outdated. For production deployments, use a currently supported Linux distribution.

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

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

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

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

Now make sure your system is up to date:

```
dnf update -y
```

6.13.1 Apache

```
dnf install -y httpd
```

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

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

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

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

  </Directory>
</VirtualHost>
```

See *Apache Web server configuration* for further details.

Make sure the apache web service is enabled and started:

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

6.13.2 PHP

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

Setting up remirepo with PHP 8.2

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

Command to install the Remi repository configuration package:

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

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

```
dnf install yum-utils
```

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

You have to enable the module stream for 8.2:

```
dnf module reset php
dnf module install php:remi-8.2
dnf update
```

Installing PHP and the required modules

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

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

Installing optional modules redis/imagick

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

6.13.3 Database

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

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

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

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

Improve MariaDB security.:

```
mysql_secure_installation
```

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

6.13.4 Redis

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

6.13.5 Installing Nextcloud

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

Now download the archive of the latest Nextcloud version:

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

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

- You may also verify the PGP signature:

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

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

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

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

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

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

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

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

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

Restart apache:

```
systemctl restart httpd.service
```

Create a firewall rule for access to apache:

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

6.13.6 SELinux

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

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

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

setsebool -P httpd_can_network_connect on
```

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

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

Note

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

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

Example config:

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

(continues on next page)

(continued from previous page)

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

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

6.14 Example installation on OpenBSD

Warning

Nextcloud does not have official OpenBSD or other BSDs support

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

From a base installed OpenBSD system you can just do:

```
# pkg_add nextcloud
```

The extra packages:

```
# pkg_add postgresql-server redis pecl82-redis php-pdo_pgsql
```

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

6.14.1 HTTPD(8)

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

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

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

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

  root "/nextcloud"
  directory index "index.php"
```

(continues on next page)

(continued from previous page)

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

    location "/data/*" {
        block drop
    }

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

    location "/apps/*" {
        pass
    }

    location "/core/*" {
        pass
    }

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

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

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

    location match "/ocs-provider/*" {
        pass
    }
}
```

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

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

6.14.2 PHP

Assuming that you are on OpenBSD -current (or >= 6.8-stable) you could use PHP 8.2 so I will keep this version, but the concept is the same for other version.

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

It is recommended to add opcache to it:

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

And increase some limits:

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

We can enable the PHP modules with:

```
# cd /etc/php-8.2.sample
# for i in *; do ln -sf ../php-8.2.sample/$i ../php-8.2/; done
```

And then we just enable and start PHP:

```
# rcctl enable php82_fpm
# rcctl start php82_fpm
```

6.14.3 Database

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

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

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

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

We need to check, enable and start postgres:

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

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

6.14.4 Redis

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

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

6.14.5 Cron job

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

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

6.14.6 Chroot

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

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

6.14.7 Nextcloud final steps

The remaining installation step are completed in the web-based installation wizard.

To activate this wizard, create a file named CAN_INSTALL inside the installation's config folder:

```
# touch /var/www/nextcloud/config/CAN_INSTALL
```

Use your browser to navigate to the installation's URL:

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

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

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

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

And on -stable:

```
# pkg_add -u
```

Then you just follow the steps from your browser.

6.14.8 NOTE

Remember always to read all the READMES from the OpenBSD packages on:

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

All this information and more is available for you there.

6.15 Uninstallation

The application is stored in a server directory and works with a database to store the metadata for files and their shares (EFSS functionality).

There are no general uninstallation instructions, as Nextcloud offers a high degree of flexibility with regard to the operating model or operating platform; examples include abstract containers, virtual machines or "bare metal", i.e. installation directly on one or more servers.

It is therefore important for the uninstallation to understand where the Nextcloud application is installed and where the corresponding data is located.

- Application directory (created before installation)
- File storage of the users (configured within the application directory or outside)
- Metadata storage in the database (within the application directory when using SQLite or outside on the same or another server)
- Caching via Redis server or similar (if used)

For uninstallation, a decision must be made as to whether the file storage should be backed up or whether the data should also be deleted. In addition, either the corresponding servers must be completely deprovisioned or the application directory deleted, as well as the database schemas and Redis entries, depending on the deployment scenario. If dedicated containers or virtual machines are used, these must be deprovisioned and the Nextcloud application must also be deprovisioned.

To uninstall, you can read values from your configuration in `config` directory. Check:

- Source code (manually installed, usually in `/var/www` or `/opt/nextcloud`): remove the directory on all servers
- Database (related configuration keys: `dbtype`, `dbhost`): remove the corresponding database on all your database servers (you may want to make a backup first)
- Cache (related configuration keys: `memcache.*`): if persistent, remove the corresponding database or key from all cache servers
- Data (related configuration keys: `datadirectory`): delete the directory on all servers (you may need to create a backup beforehand). Nextcloud has the option to store data in different locations. Also check external storage and objectstore
- Logs (related configuration keys: `logfile`, `logfile_audit`): normally in the data directory, but can also be in another location such as `/var/log/`

CHAPTER

SEVEN

DATABASE SETUP AND CONFIGURATION

7.1 Set up a database server

Nextcloud requires a database in which administrative data is stored. The following databases are currently supported:

- MySQL / MariaDB
- PostgreSQL
- Oracle (*only for Nextcloud Enterprise*)

The PostgreSQL or MariaDB databases are the recommended database engines.

> **Tip**
>
> Not all versions of every supported database are recommended. Please review the Nextcloud *System Requirements* before settling on a particular version.

7.1.1 Requirements

- Decide whether you wish to use MySQL / MariaDB, PostgreSQL, or Oracle as your database
- Pick a recommended version of your database by checking the Nextcloud *System Requirements*
- Install and set up the chosen database server software (and preferred version) before deploying Nextcloud Server

> **Note**
>
> The steps for configuring a third party database are beyond the scope of this document. Please refer to the documentation for your specific database choice for instructions.

Database "READ COMMITTED" transaction isolation level

As discussed above Nextcloud is using the `TRANSACTION_READ_COMMITTED` transaction isolation level. Some database configurations are enforcing other transaction isolation levels. To avoid data loss under high load scenarios (e.g. by using the sync client with many clients/users and many parallel operations) you need to configure the transaction isolation level accordingly. Please refer to the MySQL manual for detailed information.

7.1.2 Parameters

For setting up Nextcloud to use any database, use the instructions in *Installation wizard*. You should not have to edit the respective values in the `config/config.php`. However, in special cases (for example, if you want to connect your Nextcloud instance to a database created by a previous installation of Nextcloud), some modification might be required.

Configuring a MySQL or MariaDB database

If you decide to use a MySQL or MariaDB database, ensure the following:

- The transaction isolation level is set to "READ-COMMITTED" in your MariaDB server configuration `/etc/mysql/my.cnf` to persist even after a restart of your database server.

  Verify the **transaction_isolation** and **binlog_format**:

```
[mysqld]
...
transaction_isolation = READ-COMMITTED
binlog_format = ROW
...
```

Your `/etc/mysql/my.cnf` could look like this:

```
[server]
skip_name_resolve = 1
innodb_buffer_pool_size = 128M
innodb_buffer_pool_instances = 1
innodb_flush_log_at_trx_commit = 2
innodb_log_buffer_size = 32M
innodb_max_dirty_pages_pct = 90
query_cache_type = 1
query_cache_limit = 2M
query_cache_min_res_unit = 2k
query_cache_size = 64M
tmp_table_size= 64M
max_heap_table_size= 64M
slow_query_log = 1
slow_query_log_file = /var/log/mysql/slow.log
long_query_time = 1

[client-server]
!includedir /etc/mysql/conf.d/
!includedir /etc/mysql/mariadb.conf.d/

[client]
default-character-set = utf8mb4

[mysqld]
character_set_server = utf8mb4
collation_server = utf8mb4_bin
transaction_isolation = READ-COMMITTED
binlog_format = ROW
innodb_large_prefix=on
innodb_file_format=barracuda
innodb_file_per_table=1
```

Please refer to the page in the MySQL manual.

- That you have installed and enabled the pdo_mysql extension in PHP
- That the **mysql.default_socket** points to the correct socket (if the database runs on the same server as Nextcloud).

> **Note**
>
> MariaDB is backwards compatible with MySQL. All instructions work for both. You will not need to replace mysql with anything.

The PHP configuration in `/etc/php7/conf.d/mysql.ini` could look like this:

```
# configuration for PHP MySQL module
extension=pdo_mysql.so

[mysql]
mysql.allow_local_infile=On
mysql.allow_persistent=On
mysql.cache_size=2000
mysql.max_persistent=-1
mysql.max_links=-1
mysql.default_port=
mysql.default_socket=/var/lib/mysql/mysql.sock  # Debian squeeze: /var/run/mysqld/
↪mysqld.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
```

When using MariaDB use:

```
mariadb -uroot -p
```

Then a **mysql>** or **MariaDB [root]>** prompt will appear. Now enter the following lines and confirm them with the enter key:

```
CREATE USER 'username'@'localhost' IDENTIFIED BY 'password';
CREATE DATABASE IF NOT EXISTS nextcloud CHARACTER SET utf8mb4 COLLATE utf8mb4_bin;
GRANT ALL PRIVILEGES on nextcloud.* to 'username'@'localhost';
```

You can quit the prompt by entering:

```
quit;
```

A Nextcloud instance configured with MySQL would contain the hostname on which the database is running, a valid username and password to access it, and the name of the database. The `config/config.php` as created by the *Installation wizard* would therefore contain entries like this:

```
<?php

  "dbtype"        => "mysql",
  "dbname"        => "nextcloud",
  "dbuser"        => "username",
  "dbpassword"    => "password",
  "dbhost"        => "localhost",
  "dbtableprefix" => "oc_",
```

In case of UTF8MB4 you will also find:

```
"mysql.utf8mb4" => true,
```

SSL for MySQL Database

Enabling SSL is only necessary if your database does not reside on the same server as your Nextcloud instance. If you do not connect over localhost and need to allow remote connections then you should enable SSL. This just covers the SSL database configuration on the Nextcloud server. First you need to configure your database server accordingly.

```
'dbdriveroptions' => [
  \PDO::MYSQL_ATTR_SSL_KEY => '/../ssl-key.pem',
  \PDO::MYSQL_ATTR_SSL_CERT => '/../ssl-cert.pem',
  \PDO::MYSQL_ATTR_SSL_CA => '/../ca-cert.pem',
  \PDO::MYSQL_ATTR_SSL_VERIFY_SERVER_CERT => true,
],
```

Adjust the paths to the pem files for your environment.

Tip

The encrypted connection can also be set up during the installation, so that no unencrypted connection is used at any point. See *Encrypted database connection* when installing with an autoconfig file, or *Encrypted database connection* when installing with `occ maintenance:install`.

PostgreSQL database

In order to run Nextcloud securely on PostgreSQL, it is assumed that only Nextcloud uses this database and thus only one user accesses the database. For further services and users, we recommend to create a separate database or PostgreSQL instance.

If you decide to use a PostgreSQL database make sure that you have installed and enabled the PostgreSQL extension in PHP. The PHP configuration in `/etc/php7/conf.d/pgsql.ini` could look like this:

```
# configuration for PHP PostgreSQL module
extension=pdo_pgsql.so
extension=pgsql.so

[PostgreSQL]
pgsql.allow_persistent = On
pgsql.auto_reset_persistent = Off
pgsql.max_persistent = -1
pgsql.max_links = -1
```

(continues on next page)

(continued from previous page)

```
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 TEMPLATE template0 ENCODING 'UTF8';
GRANT CREATE ON SCHEMA public TO username;
```

You can quit the prompt by entering:

```
\q
```

A Nextcloud instance configured with PostgreSQL would contain the path to the socket on which the database is running as the hostname, the system username the PHP process is using, and an empty password to access it, and the name of the database. The `config/config.php` as created by the *Installation wizard* would therefore contain entries like this:

```
<?php

  "dbtype"        => "pgsql",
  "dbname"        => "nextcloud",
  "dbuser"        => "username",
  "dbpassword"    => "",
  "dbhost"        => "/var/run/postgresql",
  "dbtableprefix" => "oc_",
```

Note

The host actually points to the socket that is used to connect to the database. Using localhost here will not work if postgreSQL is configured to use peer authentication. Also note that no password is specified, because this authentication method doesn't use a password.

If you use another authentication method (not peer), you'll need to use the following steps to get the database setup: Now you need to create a database user and the database itself by using the PostgreSQL command line interface. The database tables will be created by Nextcloud when you login for the first time.

To start the postgres command line mode use:

```
psql -hlocalhost -Upostgres
```

Then a **postgres=#** prompt will appear. Now enter the following lines and confirm them with the enter key:

```
CREATE USER username WITH PASSWORD 'password' CREATEDB;
CREATE DATABASE nextcloud TEMPLATE template0 ENCODING 'UTF8';
ALTER DATABASE nextcloud OWNER TO username;
GRANT ALL PRIVILEGES ON DATABASE nextcloud TO username;
GRANT ALL PRIVILEGES ON SCHEMA public TO username;
```

You can quit the prompt by entering:

```
\q
```

A Nextcloud instance configured with PostgreSQL would contain the hostname on which the database is running, a valid username and password to access it, and the name of the database. The `config/config.php` as created by the *Installation wizard* would therefore contain entries like this:

```
<?php

  "dbtype"        => "pgsql",
  "dbname"        => "nextcloud",
  "dbuser"        => "username",
  "dbpassword"    => "password",
  "dbhost"        => "localhost",
  "dbtableprefix" => "oc_",
```

SSL for PostgreSQL database

As with MySQL, enabling SSL is only necessary if your database does not reside on the same server as your Nextcloud instance. First configure your database server accordingly, then add the `pgsql_ssl` configuration to your `config/config.php`.

```
'pgsql_ssl' => [
  'mode' => 'verify-full',
  'rootcert' => '/../ca-cert.pem',
  'cert' => '/../ssl-cert.pem',
  'key' => '/../ssl-key.pem',
],
```

The `mode` corresponds to the `sslmode` connection parameter of PostgreSQL, see the PostgreSQL documentation for the available modes. Only `verify-full` verifies that the certificate of the database server was issued for the hostname used to connect. Adjust the paths to the pem files for your environment.

Tip

The encrypted connection can also be set up during the installation, so that no unencrypted connection is used at any point. See *Encrypted database connection* when installing with an autoconfig file, or *Encrypted database connection* when installing with `occ maintenance:install`.

7.1.3 Troubleshooting

How to work around "general error: 2006 MySQL server has gone away"

The database request takes too long and therefore the MySQL server times out. It's also possible that the server is dropping a packet that is too large. Please refer to the manual of your database for how to raise the configuration options `wait_timeout` and/or `max_allowed_packet`.

Some shared hosters are not allowing the access to these config options. For such systems Nextcloud is providing a `dbdriveroptions` configuration option within your `config/config.php` where you can pass such options to the database driver. Please refer to *Configuration Parameters* for an example.

How can I find out if my MySQL/PostgreSQL server is reachable?

To check the server's network availability, use the ping command on the server's host name (db.server.com in this example):

```
ping db.server.com
```

```
PING db.server.com (ip-address) 56(84) bytes of data.
64 bytes from your-server.local.lan (192.168.1.10): icmp_req=1 ttl=64 time=3.64 ms
64 bytes from your-server.local.lan (192.168.1.10): icmp_req=2 ttl=64 time=0.055 ms
64 bytes from your-server.local.lan (192.168.1.10): icmp_req=3 ttl=64 time=0.062 ms
```

For a more detailed check whether the access to the database server software itself works correctly, see the next question.

How can I find out if a created user can access a database?

The easiest way to test if a database is accessible is by starting the command line interface:

MySQL:

Assuming the database server is installed on the same system you're running the command from, use:

```
mysql -uUSERNAME -p
```

To access a MySQL installation on a different machine, add the -h option with the respective host name:

```
mysql -uUSERNAME -p -h HOSTNAME
```

```
mysql> SHOW VARIABLES LIKE "version";
+---------------+--------+
| Variable_name | Value  |
+---------------+--------+
| version       | 8.0.36 |
+---------------+--------+
1 row in set (0.00 sec)
mysql> quit
```

PostgreSQL:

Assuming the database server is installed on the same system you're running the command from, use:

```
psql -Uusername -dnextcloud
```

To access a PostgreSQL installation on a different machine, add the -h option with the respective host name:

```
psql -Uusername -dnextcloud -h HOSTNAME
```

```
postgres=# SELECT version();
PostgreSQL 16.2 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
```

7.2 Enabling MySQL/MariaDB 4-byte support

Note

Be sure to backup your database before performing this database upgrade.

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

Warning

This guide applies only to MySQL 8 or newer and MariaDB 10.6 or newer. For a list of supported MySQL and MariaDB versions, see our *system requirements documentation*.

1. Make sure the following InnoDB settings are set on your MySQL server:

   ```
   [mysqld]
   innodb_file_per_table=1
   ```

2. Restart the MySQL server in case you changed the configuration in step 1.

You can then verify that the change worked:

```
SHOW VARIABLES LIKE 'innodb_file_per_table';
```

The result should look like this:

```
mysql> SHOW VARIABLES LIKE 'innodb_file_per_table';
+-----------------------+-------+
| Variable_name         | Value |
```

(continues on next page)

(continued from previous page)

```
+-----------------------+-------+
| innodb_file_per_table | ON    |
+-----------------------+-------+
1 row in set (0.00 sec)
```

3. Open a shell, change dir (adjust `/var/www/nextcloud` to your nextcloud location if needed), and put your nextcloud instance in maintenance mode, if it isn't already:

```
$ cd /var/www/nextcloud
$ sudo -E -u www-data php occ maintenance:mode --on
```

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 -E -u www-data php occ config:system:set mysql.utf8mb4 --type boolean --
↪value="true"
```

6. Convert all existing tables to the new collation by running the repair step:

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

Note

This will also change the *ROW_FORMAT* to *DYNAMIC* for your tables.

7. Disable maintenance mode:

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

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

Note

Also make sure your backup strategy still work. If you use `mysqldump` make sure to add the `--default-character-set=utf8mb4` option. Otherwise your backups are broken and restoring them will result in `?` instead of the emojis, making files inaccessible.

7.3 Convert to a different database type

You can convert a SQLite database to a better performing MySQL, MariaDB or PostgreSQL database with the Nextcloud command line tool. SQLite is good for testing and simple single-user Nextcloud servers, but it does not scale for multiple-user production servers.

7.3.1 Run the conversion

Conversion consists of two steps:

1. Establishing the target database (including its credentials)
2. Triggering the conversion tool which migrates the contents of the existing database to the target database

Establishing the target database

First create up the target (new) database (along with its associated username and password) by following the manual database configuration instructions for your chosen target database type:

- *Configuring a MySQL or MariaDB database*
- *PostgreSQL database*

Since the above db instructions uses the database name `nextcloud` for the newly created database we will do so here for consistency, but you are free to use whatever database name you prefer. Use the database name, database username, and database password you specified when creating the new database.

Triggering the conversion

The `occ db:convert-type` command handles all the tasks of the conversion. The following are the parameters available:

```
sudo -E -u www-data php occ db:convert-type [options] type username hostname database
```

`type` should be the target database type. The same values are available here as for the `config.php` `dbtype` parameter. It should be one of: `mysql` for MariaDB/MySQL, `pgsql` for PostgreSQL, or `oci` for Oracle.

The options:

- `--port="3306"` the database port (optional) [defaults to "3306"]
- `--password="mysql_user_password"` password for the new database. If omitted the tool will ask you (optional)
- `--clear-schema` clear schema (optional)
- `--all-apps` by default, tables for enabled apps are converted, use to convert also tables of deactivated apps (optional)
- `-n, --no-interaction` do not ask any interactive question

Note

The conversion tool searches for apps in your configured app folders and uses the schema (table) definitions in the apps to create the new tables. Any tables that still exist for removed apps will not be converted (even with option `--all-apps`).

Let's convert our existing (functioning) sqlite3 installation to be MariaDB/MySQL based:

```
sudo -E -u www-data php occ db:convert-type --password="<password>" --port="3306" --
↪all-apps mysql <username> <hostname> nextcloud
```

> **Note**
>
> It was unnecessary to specify the port in this example because `3306` is already the default. We did so merely for demonstration purposes and completeness in case the reader is using a non-standard port on their target database server.

On success the converter will automatically configure the new database in your Nextcloud config `config.php`.

If you are converting to a MySQL/MariaDB database, you will also want to set `mysql.utf8mb4` parameter to true in your `config.php`:

```
sudo -E -u www-data php occ config:system:set mysql.utf8mb4 --type boolean --value=
↪"true"
```

If you like, you can view the changes that were made by looking for the `db*` parameters in your `config.php` (you could also use this command before doing the conversion to compare your configuration before/after):

```
grep db config/config.php
```

7.3.2 Inconvertible tables

If you updated your Nextcloud instance, there might be remnants of old tables which are not used any more. The updater will tell you which ones these are.

```
The following tables will not be converted:
oc_permissions
...
```

You can ignore these tables. Here is a list of known old tables:

- oc_calendar_calendars
- oc_calendar_objects
- oc_calendar_share_calendar
- oc_calendar_share_event
- oc_fscache
- oc_log
- oc_media_albums
- oc_media_artists
- oc_media_sessions
- oc_media_songs
- oc_media_users
- oc_permissions
- oc_privatedata - this table was later added again by the app *privatedata* (https://apps.nextcloud.com/apps/privatedata) and is safe to be removed if that app is not enabled
- oc_queuedtasks
- oc_sharing

7.4 Prevent identifier exhaustion in large databases

Database identifiers stored as 32-bit integers have a limited range. Larger Nextcloud installations can exhaust this range in heavily used tables, preventing new records from being created. Converting these identifiers and their auto-increment keys to 64-bit integers provides more capacity for future growth.

Changing columns in large tables can take several hours or even days, depending on the number of files in the Nextcloud instance. For this reason, conversion of the `filecache` and activity tables must be triggered manually with a console command.

The command can safely be executed. It will show a success message when there is nothing to do:

```
sudo -E -u www-data php occ db:convert-filecache-bigint
All tables already up to date!
```

or otherwise ask for confirmation, before performing the heavy actions:

```
sudo -E -u www-data php occ db:convert-filecache-bigint
This can take up to hours, depending on the number of files in your instance!
Continue with the conversion (y/n)? [n]
```

to suppress the confirmation message append `--no-interaction` to the argument list:

```
sudo -E -u www-data php occ db:convert-filecache-bigint --no-interaction
```

Note

Similar to a normal update, you should shutdown your Apache or nginx server or enable maintenance mode before running the command to avoid issues with your sync clients.

7.5 Using database read replicas

Added in version 29.

Nextcloud can natively split read and write operations at the level of individual database queries. Read-only replicas serve reads, while the default (primary) database connection handles writes as well as reads that must be consistent with a preceding write.

Note

Nextcloud only routes queries to the endpoints you provide. Setting up the replication itself (provisioning the replicas, keeping them in sync, and monitoring replication health) is the responsibility of your database administrator.

7.5.1 Configuration

Add the read-only replicas to `config.php` with the `dbreplica` parameter. Each entry describes one replica and accepts the same connection options as the primary database connection:

```
'dbreplica' => [
        ['user' => 'nextcloud', 'password' => 'password1', 'host' => '10.0.3.1',
↪'dbname' => 'nextcloud'],
        ['user' => 'nextcloud', 'password' => 'password2', 'host' => '10.0.3.2',
```

(continues on next page)

(continued from previous page)

```
↪'dbname' => 'nextcloud'],
    ],
```

When more than one replica is configured, Nextcloud picks one of them at random for the reads of a given connection. There is no load weighting or health checking, so if you need balanced or fault-tolerant read distribution, place the replicas behind your own load balancer and point `dbreplica` at it.

7.5.2 How queries are routed

Nextcloud does not inspect the SQL to decide where a query goes. The routing is based on which method sends the query: read operations are sent to a replica, and write operations are sent to the primary.

To avoid reading stale data right after a change, Nextcloud keeps a request consistent with itself: once a request has written to the database, its subsequent reads are served from the primary as well. This guarantees *read-your-writes* consistency: data written earlier in the same request is always read back in its up-to-date state.

7.5.3 Connection scope

This consistency guarantee is scoped to a single PHP process, that is one web request, one cron execution, or one `occ` invocation. It is therefore short-lived and does not carry over to other processes.

A single user operation may span more than one request, and there the guarantee no longer applies. Consider two requests belonging to the same operation:

1. Request 1 creates a row.
2. Request 2 reads that row.

Request 2 does not know about the write in request 1, so it is free to read from a replica. If the two requests arrive within a shorter timeframe than the replication lag, the replica may not have received the new row yet, the read returns nothing, and the application logic fails.

> **Warning**
>
> Because the consistency guarantee does not span multiple requests, the replicas must not lag significantly behind the primary. Noticeable replication delay leads to symptoms such as newly created or changed content that seems to disappear after a page refresh.
>
> For this reason the feature works best with a synchronous, always-consistent cluster such as Galera. Setups with asynchronous replication are only safe when the replication delay stays negligible.

7.6 Splitting tables or apps across databases

> **Warning**
>
> This is still proof-of-concept level. Use with care.

In order to scale at some point it might make sense to split out some tables or apps which allow it as they might be better off with a different replication methods, etc.

A first attempt we do right now with the activity table. In order to make use of this, the app/table needs to match the following criteria:

- No other apps are allowed to have queries directly to the table
- No JOINs are performed between this table and any other tables not on this new separate connection
- The app needs to support a connection parameter prefix

In case of the activity app the prefix is `activity_`. If a database config is not specified it falls back to the normal database configuration option for this value:

- `activity_dbuser` falling back to `dbuser`
- `activity_dbpassword` falling back to `dbpassword`
- `activity_dbname` falling back to `dbname`
- `activity_dbhost` falling back to `dbhost`
- `activity_dbport` falling back to `dbport`
- `activity_dbdriveroptions` falling back to `dbdriveroptions`

Note

It is not possible to use a different database type (SQLite, MySQL, PostrgreSQL, Oracle) for a split database. Also in case of MySQL and MariaDB the utf8mb4 option needs to be the same on both databases.

7.6.1 Initial splitting

For the initial split the affected tables have to be copied over to the new database, in case of the activity app these are:

- `oc_activity`
- `oc_activity_mq`

1. Enable maintenance mode
2. Make sure optional database changes are applied:
    1. `occ db:convert-mysql-charset`
    2. `occ db:convert-filecache-bigint`
    3. `occ db:add-missing-columns`
    4. `occ db:add-missing-indices`
    5. `occ db:add-missing-primary-keys`
3. Specify the desired configuration values
4. Copy the 2 tables to the new database
5. Disable maintenance mode

7.6.2 Migrations on updates

We will try to avoid migrations on those tables in the future, but it might be necessary at some point. We hope to have a dedicated plan by the time this happens. For now a potential way would be:

1. Enable maintenance mode
2. Update as usual
3. Execute manual queries for schema changes provided by the app authors

4. Execute manual queries for data changes provided by the app authors
5. Disable maintenance mode

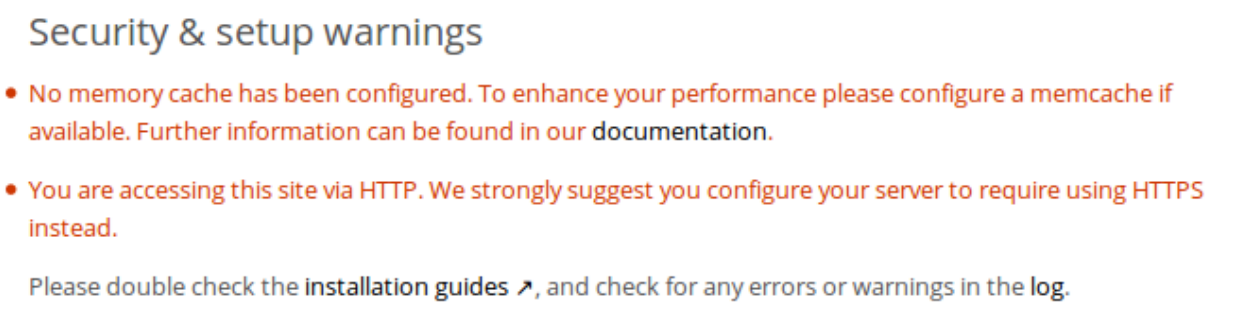
CHAPTER

EIGHT

NEXTCLOUD CONFIGURATION

8.1 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.

8.1.1 Cache warnings

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

- APCu (minimum required PHP extension version 4.0.6)
- Memcached
- Redis (minimum required PHP extension version: 2.2.6)

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

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

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

8.1.2 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.

8.1.3 You are accessing this site via HTTP

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

Enabling SSL (on Apache)

Use HTTPS

NGINX configuration

8.1.4 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* provides the information about how to configure your environment.

8.1.5 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

You can see if the header is appearing in requests by using your browser inspector or using a tool such as cURL: `curl --head https://cloud.domain.tld`.

8.1.6 /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.

8.1.7 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.

8.1.8 Outdated NSS / OpenSSL version

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

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

There are known bugs in older OpenSSL and NSS versions leading to misbehavior in combination with remote hosts using SNI. A technology used by most of the HTTPS websites. To ensure that Nextcloud will work properly you need to

update OpenSSL to at least 1.0.2b or 1.0.1d. For NSS the patch version depends on your distribution and an heuristic is running the test which actually reproduces the bug.

8.1.9 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.

8.1.10 Some files have not passed the integrity check

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

8.1.11 Your database does not run with "READ COMMITTED" transaction isolation level

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

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

8.1.12 The "__Host-" prefix is not used for the cookie name

"The `__Host-` prefix is not used for the cookie name. It is recommended to enable this in your configuration."

Nextcloud applies the `__Host-` prefix to its same-site CSRF cookies (`__Host-nc_sameSiteCookiestrict` and `__Host-nc_sameSiteCookielax`) when it detects that the connection is served over HTTPS. The prefix instructs browsers to only accept those cookies over a secure connection and from the exact host that set them, which strengthens CSRF protection.

This warning appears when Nextcloud cannot confirm it is running over HTTPS. The most common cause is a **reverse proxy** that terminates TLS and forwards requests to Nextcloud over plain HTTP. In that case Nextcloud sees HTTP internally and omits the prefix.

To fix this, tell Nextcloud to treat the connection as HTTPS by adding `overwriteprotocol` to `config/config.php`:

```
'overwriteprotocol' => 'https',
```

If you are not behind a reverse proxy, ensure your web server is configured to serve Nextcloud exclusively over HTTPS. See the *Use HTTPS* documentation.

For background on the cookies themselves, see *Cookies*.

8.2 Configuration Parameters

8.2.1 Introduction

Nextcloud uses `config/config.php` as its main configuration file. This file controls various fundamental aspects of server operations. It is typically modified as part of initial deployment, when troubleshooting, and when making adjustments to surrounding infrastructure.

This is a required file for all Nextcloud deployments and thus it is critical for Nextcloud administrators to be familiar with managing it.

This section of the *Administration Manual* documents how to adjust this essential file, certain special characteristics of the `config/` directory, and all of the supported parameters that can be specified in a `config/config.php` file.

Note

While `config/config.php` is a required file, many Nextcloud or Nextcloud app settings are managed elsewhere and thus not included in it. These settings are typically managed via individual apps.

8.2.2 Loading

Configuration files located in `config/` are parsed automatically when Nextcloud starts up. They are also checked for changes periodically (approximately every two seconds in a standard PHP environment running with default *OPcache* settings; approximately every sixty seconds in many pre-packaged Nextcloud installation methods).

The `config/config.php` file may be supplemented by additional `*.config.php` files placed in the `config/` directory (if appropriately named and formatted).

Danger

Be cautious when naming or creating backup copies of your active `config/config.php`. If a backup is located within `config/` and is named `(ANYTHING).config.php`, it will be loaded as part of your live configuration and override your `config/config.php` values!

Tip

If your configuration changes don't seem to be taking effect, check: (a) your PHP opcache configuration; (b) for additional `*.config.php` files located in `config/`; (c) the documentation for your Nextcloud installation method/package; (d) the output of `occ config:list system`.

8.2.3 Format

The short answer is that `config/` files are plain text files with some special formatting requirements for different types of parameters and values. This makes it extensible and easy for Nextcloud to interact with. It also makes it easy for administrators to view with any text viewer and from the command-line.

Technically these configuration files are PHP files containing a special (to Nextcloud) PHP array called `$CONFIG`. This array consists of various Nextcloud specific "key-value" pairs (in some cases arrays themselves). Each pair has the form `key => value` and is comma-separated.

Types of Values

Strings:

- `"thisIsAnImportantValue"`
- Note: These must be either single or double quoted - i.e. `"string"` or `'string'`.
- Note: IP addresses are considered strings.
- **Examples:**

```
- 'logo_url' => 'https://example.org',
- 'versions_retention_obligation' => 'auto, D',
- 'logtimezone' => 'Europe/Berlin',
```

Boolean:

- `true` or `false`
- Note: These should **not** be surrounded by quote marks within the configuration file itself.
- **Examples:**

```
- 'session_keepalive' => true,
- 'hide_login_form' => false,
```

Numerical:

- `12`
- This includes both integers and floating point numbers.
- Note: These should **not** be surrounded by quote marks within the configuration file itself.
- **Examples:**

```
- 'loglevel' => 2,
- 'session_lifetime' => 60 * 60 * 24,
```

Arrays of any of the above types:

- `[ 'value1', 'value2' ]`
- All value types (including other arrays) can be included in arrays.
- Note: Only some parameters support array style values.
- **Examples:**

```
- 'connectivity_check_domains' => [ 'www.nextcloud.com', 'www.eff.org', ],
- 'enabledPreviewProviders' => [ 'OC\Preview\BMP', 'OC\Preview\GIF', 'OC\
  Preview\JPEG', ],
```

Tip

Nextcloud attempts to remedy some value type/formatting mistakes, but this is not foolproof. Use the correct formatting (for the type of value in question) to avoid unexpected results arising from values being cast in unexpected ways.

8.2.4 Modifying

Parameters may be modified in a standard text editor (i.e. via the command-line or externally then re-uploaded). They may also, in most cases, be modified using the commands in the `occ config:system:*` namespace.

Tip

Incorrectly formatted `key => value` entries (or incorrectly specified values) may not generate immediate errors or problems (such as parsing / syntax errors), but may still lead to unexpected and undesirable results. Review your fully parsed (by PHP) configuration by using the command `occ config:list system` and/or `occ config:list system --private` to identify anything unexpected.

8.2.5 Defaults

Nextcloud creates a base `config/config.php` file at installation time containing the most essential parameters for operations. These values are a mixture of auto-generated and drawn from information provided by the administrator at installation time.

The file `config/config.sample.php` lists all the parameters within Nextcloud that can be specified in `config/` files, along with example and default values for each. The content of that sample configuration file is included *below* for ease of reference and alongside additional context.

Tip

Only add parameters to `config/config.php` that you wish to modify.

Danger

Do not copy everything from `config/config.sample.php` into your own `config/config.php`! Besides being unnecessary, it will break things and possibly even require re-installation.

8.2.6 Multiple/Merged Configuration Files

Nextcloud supports loading configuration parameters from multiple files. You can add arbitrary files ending with `.config.php*` (i.e. `*.config.php`) in the `config/` directory. The values in these files take precedence over `config/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.

For example, you could place your email server configuration in `config/email.config.php` and whatever parameters you specify in it will be merged with your `config/config.php`.

Note

The values in these additional configuration files **always** take precedence over `config/config.php`.

Tip

To view your fully merged configuration (i.e. incorporating all config files), use `occ config:list system` and/or `occ config:list system --private`.

Danger

Be cautious when naming or creating backup copies of your active `config/config.php`. If a backup config file is located within `config/` and happens to be named `(ANYTHING).config.php`, it will be loaded as part of your live configuration and override your `config/config.php` values!

8.2.7 Environment Variables

The `NEXTCLOUD_CONFIG_DIR` environment variable overrides the default config directory path. When set, Nextcloud loads `config.php` (and any `*.config.php` files) from that path instead of the `config/` directory inside the webroot.

```
NEXTCLOUD_CONFIG_DIR=/etc/nextcloud php /var/www/nextcloud/cron.php
```

This is useful for:

- Moving `config.php` outside the webroot as a hardening measure — credentials are not accessible via HTTP even if directory listing is enabled or misconfigured.
- Running multiple Nextcloud instances that share a single codebase but require separate config directories.

Note

`NEXTCLOUD_CONFIG_DIR` must be set for **both** the web server process and any CLI invocations (`occ`, cron jobs). Set it in your web server virtual host configuration and in the shell environment used for CLI work.

See also

Place config directory outside of the web root in the hardening guide for a deployment recommendation.

8.2.8 Examples

These are some examples of the content of typical `config/config.php` files immediately after a basic installation of Nextcloud.

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:

```
<?php
$CONFIG = array (
  'instanceid' => 'occ6f7365735',
  'passwordsalt' => '2c57784763467863063',
  'trusted_domains' =>
  array (
    0 => 'localhost',
    1 => 'studio',
  ),
  'datadirectory' => '/var/www/nextcloud/data',
  'dbtype' => 'sqlite3',
  'version' => '7.0.2.1',
  'installed' => true,
);
```

Note

SQLite is a simple, lightweight embedded database that is fine for testing and simple installations, but production environments you should use MySQL/MariaDB, Oracle, or PosgreSQL.

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,
);
```

8.2.9 Default Parameters

These parameters are configured by the Nextcloud installer, and are required for your Nextcloud server to operate.

instanceid

```
'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.

serverid

```
'serverid' => -1,
```

This is a unique identifier for your server.

It is useful when your Nextcloud instance is using different PHP servers. Once it's set it shouldn't be changed.

Value must be an integer, comprised between 0 and 511.

When config.php is shared between different servers, this value should be overridden with "NC_serverid=<int>" on each server. Note that it must be overridden for CLI and for your webserver.

Example for CLI: NC_serverid=42 occ config:list system

passwordsalt

```
'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.

Deprecated since version 9.0.0: This salt is deprecated and only used for legacy-compatibility, developers should *NOT* use this value for anything nowadays.

secret

```
'secret' => '',
```

Secret used by Nextcloud for various purposes, e.g., to encrypt data. If you lose this string, there will be data corruption.

trusted_domains

```
'trusted_domains' => [
        'demo.example.org',
        'otherdomain.example.org',
        '10.111.112.113',
        '[2001:db8::1]'
    ],
```

Your list of trusted domains that users can log into. Specifying trusted domains prevents host header poisoning. Do not remove this, as it performs necessary security checks.

You can specify:

- The exact hostname of your host or virtual host, e.g. `demo.example.org`.
- The exact hostname with permitted port, e.g. `demo.example.org:443`. This disallows all other ports on this host
- Use `*` as a wildcard, e.g., `ubos-raspberry-pi*.local` will allow `ubos-raspberry-pi.local` and `ubos-raspberry-pi-2.local`
- The IP address with or without permitted port, e.g. `[2001:db8::1]:8080` Using TLS certificates where `commonName=<IP address>` is deprecated

cookie_domain

```
'cookie_domain' => '',
```

The validity domain for cookies, for example `''` (cookies will be sent only the domain that defined it, e.g. `'demo.example.org'`), `'demo.example.org'` (cookies will be valid for the domain and all subdomains), …

Defaults to `''` (safe option)

datadirectory

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

Where user files are stored. The SQLite database is also stored here, when you use SQLite.

Default to `data/` in the Nextcloud directory.

version

```
'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

```
'dbtype' => 'sqlite3',
```

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

Available:

- sqlite3 (SQLite3)
- mysql (MySQL/MariaDB)
- pgsql (PostgreSQL)

Defaults to `sqlite3`

dbhost

```
'dbhost' => '',
```

Your host server name, for example `localhost`, `hostname`, `hostname.example.com`, or the IP address.

To specify a port, use `hostname:####`; for IPv6 addresses, use the URI notation `[ip]:port`. To specify a Unix socket, use `localhost:/path/to/directory/containing/socket` or `:/path/to/directory/containing/socket`, e.g., `localhost:/run/postgresql/`.

dbname

```
'dbname' => 'nextcloud',
```

The name of the Nextcloud database, which is set during installation. You should not need to change this.

dbuser

```
'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

```
'dbpassword' => '',
```

The password for the database user. This is set up during installation, so you shouldn't need to change it.

dbtableprefix

```
'dbtableprefix' => 'oc_',
```

Prefix for the Nextcloud tables in the database.

Default to `oc_`

dbpersistent

```
'dbpersistent' => '',
```

Enable persistent connections to the database.

This setting uses the `persistent` option from Doctrine DBAL, which in turn uses the PDO::ATTR_PERSISTENT option from the PDO driver.

dbreplica

```
'dbreplica' => [
        ['user' => 'nextcloud', 'password' => 'password1', 'host' => 'replica1',
↪'dbname' => ''],
        ['user' => 'nextcloud', 'password' => 'password2', 'host' => 'replica2',
↪'dbname' => ''],
    ],
```

Specify read-only replicas to be used by Nextcloud when querying the database

db.log_request_id

```
'db.log_request_id' => false,
```

Add request ID to the database query in a comment.

This can be enabled to assist in mapping database logs to Nextcloud logs.

installed

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

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

Defaults to `false`

8.2.10 User Experience

These optional parameters control some aspects of the user interface. Default values, where present, are shown.

default_language

```
'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. The default_language parameter is only used when the browser does not send any language, and the user hasn't configured their own language preferences.

Nextcloud has two distinguished language codes for German, `de` and `de_DE`. `de` is used for informal German and `de_DE` for formal German. By setting this value to `de_DE`, you can enforce the formal version of German unless the user has chosen something different explicitly.

Defaults to `en`

force_language

```
'force_language' => 'en',
```

With this setting, a language can be forced for all users. If a language is forced, the users are also unable to change their language in the personal settings. If users shall be unable to change their language, but users have different languages, this value can be set to `true` instead of a language code.

Defaults to `false`

default_locale

```
'default_locale' => 'en_US',
```

This sets the default locale on your Nextcloud server, using ISO_639 language codes such as `en` for English, `de` for German, and `fr` for French, and ISO-3166 country codes such as `GB`, `US`, `CA`, as defined in RFC 5646. It overrides automatic locale detection on public pages like login or shared items. User's locale preferences configured under "personal -> locale" override this setting after they have logged in.

Defaults to `en`

reduce_to_languages

```
'reduce_to_languages' => [],
```

With this setting, it is possible to reduce the languages available in the language chooser. The languages have to be set as array values using ISO_639-1 language codes such as `en` for English, `de` for German, etc.

For example: Set to `['de', 'fr']` to only allow German and French languages.

default_phone_region

```
'default_phone_region' => 'GB',
```

This sets the default region for phone numbers on your Nextcloud server, using ISO 3166-1 country codes such as `DE` for Germany, `FR` for France, … It is required to allow inserting phone numbers in the user profiles starting without the country code (e.g., +49 for Germany).

No default value!

force_locale

```
'force_locale' => 'en_US',
```

With this setting, a locale can be forced for all users. If a locale is forced, the users are also unable to change their locale in the personal settings. If users shall be unable to change their locale, but users have different languages, this value can be set to `true` instead of a locale code.

Defaults to `false`

default_timezone

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

This sets the default timezone on your Nextcloud server, using IANA identifiers like `Europe/Berlin` or `Pacific/Auckland`. The default timezone parameter is only used when the timezone of the user cannot be determined.

Defaults to `UTC`

knowledgebaseenabled

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

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

`false` removes the Help item.

knowledgebase.embedded

```
'knowledgebase.embedded' => false,
```

`true` embeds the documentation in an iframe inside Nextcloud.

`false` only shows buttons to the online documentation.

allow_user_to_change_display_name

```
'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.

skeletondirectory

```
'skeletondirectory' => '/path/to/nextcloud/core/skeleton',
```

The directory containing initial content copied to a user's personal files during their first login. This content can include ordinary files and folders, such as welcome documents, example files, or organization-wide reusable documents and folder structures.

The path may include the optional `{lang}` placeholder to select language-specific content. The placeholder is replaced with the user's language code. For example, when this is set to `/path/to/skeleton/{lang}` and a user's language is `de_DE`, Nextcloud first looks for `/path/to/skeleton/de_DE`. If that directory does not exist, Nextcloud falls back to the base language code (`de` in this example), looking for `/path/to/skeleton/de`. If that directory also does not exist, Nextcloud looks for `/path/to/skeleton/default`.

If the path does not include `{lang}`, Nextcloud uses the configured directory directly. Only one matching directory is used; content from fallback directories is not merged. If no matching directory exists, no content is copied.

Set to an empty string (`''`) to disable copying skeleton content.

> **Note**
>
> Copying skeleton content also triggers automatic initialization of the user's template directory (see `templatedirectory`). Setting this parameter to an empty string therefore prevents template initialization on first login. Users can still initialize a template directory later through the Files UI, which uses the OCS API, but doing so does not copy skeleton content. The template source directory itself is configured separately and is not derived from this directory.

Defaults to `core/skeleton` relative to the Nextcloud installation root.

templatedirectory

```
'templatedirectory' => '/path/to/nextcloud/templates',
```

The directory containing template content that can be copied to a user's template directory. Template files are ordinary files that users can select as reusable templates when creating supported file types in the Files app.

Like `skeletondirectory`, this path may contain the optional `{lang}` placeholder. Nextcloud resolves it using the same language, base-language, and `default` fallback order described for `skeletondirectory`. The placeholder is resolved independently for this setting.

If the path does not include `{lang}`, Nextcloud uses the configured directory directly. Only one matching directory is used; content from fallback directories is not merged. If no matching directory exists, no template content is copied.

This setting is independent of `skeletondirectory`. Its default value is `core/skeleton/Templates` relative to the Nextcloud installation root. Changing `skeletondirectory` does not change that value. If a custom skeleton directory contains a `Templates` subdirectory, Nextcloud copies it as ordinary skeleton content; it does not automatically use it as the user's template directory.

Template content is copied only when the target template folder is empty.

Set to an empty string (`''`) to disable copying template content from this source. During automatic first-login initialization, Nextcloud may still create or reuse the default template folder, but it will not copy content from this source or configure that folder as the user's template directory.

Defaults to `core/skeleton/Templates` relative to the Nextcloud installation root, regardless of the `skeletondirectory` value.

8.2.11 User session

remember_login_cookie_lifetime

```
'remember_login_cookie_lifetime' => 60 * 60 * 24 * 15,
```

Lifetime of logins where the user selected "Remember me", in seconds.

A value > `0` means "Remember me" is available. To make "Remember me" unavailable to users, set to `0`.

To avoid unexpected expiry, set this higher than `session_lifetime`.

Despite the key name, this also affects server-side expiration of remembered login tokens. Clearing browser cookies removes remembered login on that browser, but does not itself revoke server-side remember tokens.

Defaults to `60*60*24*15` seconds (15 days).

session_lifetime

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

Lifetime of authenticated sessions after inactivity, in seconds.

When "Remember me" is enabled, users may be transparently re-authenticated after session expiry/logout while the remember-login cookie remains valid.

To avoid earlier-than-expected remembered-login expiry, set `remember_login_cookie_lifetime` higher than this value.

Effective behavior also depends on related settings: - `session_keepalive` can extend active Web UI sessions via heartbeat requests. - `session_relaxed_expiry` may allow sessions to persist longer than this value. - `auto_logout` can enforce logout behavior in the Web UI.

The effective maximum retention also depends on PHP settings and external session-backend cleanup policies, including (but not limited to) PHP's `session.gc_maxlifetime` and environment-specific cleanup behavior (e.g., distro cron/tmpfiles policies and handler-specific GC behavior). These may override this value.

Defaults to `60*60*24` seconds (24 hours).

davstorage.request_timeout

```
'davstorage.request_timeout' => 30,
```

The timeout in seconds for requests to servers made by the DAV component (e.g., needed for federated shares).

carddav_sync_request_timeout

```
'carddav_sync_request_timeout' => 30,
```

The timeout in seconds for synchronizing address books, e.g., federated system address books (as run by `occ federation:sync-addressbooks`).

Defaults to `30` seconds

carddav_sync_request_truncation

```
'carddav_sync_request_truncation' => 2500,
```

The limit applied to the synchronization report request, e.g. federated system address books (as run by `occ federation:sync-addressbooks`).

session_relaxed_expiry

```
'session_relaxed_expiry' => false,
```

`true` enables a relaxed session timeout, where the session timeout would no longer be handled by Nextcloud but by either the PHP garbage collection or the expiration of potential other session backends like Redis.

This may lead to sessions being available for longer than what `session_lifetime` uses but comes with performance benefits as sessions are no longer a locking operation for concurrent requests.

session_keepalive

```
'session_keepalive' => true,
```

Enable or disable session keep-alive when a user is logged in to the Web UI.

Enabling this sends a "heartbeat" to the server to keep it from timing out.

Defaults to `true`

auto_logout

```
'auto_logout' => false,
```

Enable or disable the automatic logout after session_lifetime, even if session keepalive is enabled. This will make sure that an inactive browser will log itself out even if requests to the server might extend the session lifetime.

> **Note**
>
> The logout is handled on the client side. This is not a way to limit the duration of potentially compromised sessions.

Defaults to `false`

token_auth_enforced

```
'token_auth_enforced' => false,
```

Enforce token authentication for clients, which blocks requests using the user password for enhanced security. Users need to generate tokens in personal settings which can be used as passwords on their clients.

Defaults to `false`

token_auth_activity_update

```
'token_auth_activity_update' => 60,
```

The interval at which token activity should be updated.

Increasing this value means that the last activity on the security page gets more outdated.

Tokens are still checked every 5 minutes for validity max value: 300

Defaults to `60`

auth.bruteforce.protection.enabled

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

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

> **Warning**
>
> Disabling this is discouraged for security reasons.

Defaults to `true`

auth.bruteforce.protection.force.database

```
'auth.bruteforce.protection.force.database' => false,
```

Whether the brute force protection should write into the database even when a memory cache is available

Using the database is most likely worse for performance, but makes investigating issues a lot easier as it's possible to look directly at the table to see all logged remote addresses and actions.

Defaults to `false`

auth.bruteforce.protection.testing

```
'auth.bruteforce.protection.testing' => false,
```

Whether the brute force protection shipped with Nextcloud should be set to testing mode.

In testing mode, brute force attempts are still recorded, but the requests do not sleep/wait for the specified time. They will still abort with "429 Too Many Requests" when the maximum delay is reached. Enabling this is discouraged for security reasons and should only be done for debugging and on CI when running tests.

Defaults to `false`

auth.bruteforce.max-attempts

```
'auth.bruteforce.max-attempts' => 10,
```

Brute force protection: maximum number of attempts before blocking

When more than max-attempts login requests are sent to Nextcloud, requests will abort with "429 Too Many Requests". For security reasons, change it only if you know what you are doing.

Defaults to `10`

ratelimit.protection.enabled

```
'ratelimit.protection.enabled' => true,
```

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

> **Warning**
>
> Disabling this is discouraged for security reasons.

Defaults to `true`

ratelimit_overwrite

```
'ratelimit_overwrite' => [
        'profile.profilepage.index' => [
            'user' => ['limit' => 300, 'period' => 3600],
            'anon' => ['limit' => 1, 'period' => 300],
        ]
    ],
```

Overwrite the individual rate limit for a specific route

From time to time it can be necessary to extend the rate limit of a specific route, depending on your usage pattern or when you script some actions. Instead of completely disabling the rate limit or excluding an IP address from the rate limit, the following config allows to overwrite the rate limit duration and period.

The first level key is the name of the route. You can find the route name from a URL using the `occ router:list` command of your server.

You can also specify different limits for logged-in users with the `user` key and not-logged-in users with the `anon` key. However, if there is no specific `user` limit, the `anon` limit is also applied for logged-in users.

Defaults to empty array `[]`

security.ipv6_normalized_subnet_size

```
'security.ipv6_normalized_subnet_size' => 56,
```

Size of subnet used to normalize IPv6

For Brute Force Protection and Rate Limiting, IPv6 addresses are truncated using subnet size. It defaults to /56, but you can set it between /32 and /64

Defaults to `56`

auth.webauthn.enabled

```
'auth.webauthn.enabled' => true,
```

By default, WebAuthn is available, but it can be explicitly disabled by admins

auth.storeCryptedPassword

```
'auth.storeCryptedPassword' => true,
```

Whether encrypted passwords should be stored in the database

The passwords are only decrypted using the login token stored uniquely in the clients and allow connecting to external storages, autoconfiguring mail accounts in the mail app, and periodically checking if the password is still valid.

This might be desirable to disable this functionality when using one-time passwords or when having a password policy enforcing long passwords (> 300 characters).

By default, the passwords are stored encrypted in the database.

Warning

If disabled, password changes on the user backend (e.g., on LDAP) no longer log connected clients out automatically. Users can still disconnect the clients by deleting the app token from the security settings.

hide_login_form

```
'hide_login_form' => false,
```

By default, the login form is always available. There are cases (SSO) where an admin wants to avoid users entering their credentials to the system if the SSO app is unavailable.

This will show an error. But the direct login still works with adding `?direct=1`

lost_password_link

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

If your user backend does not allow password resets (e.g., when it's a read-only user backend like LDAP), you can specify a custom link, where the user is redirected to, when clicking the "Reset password" link after a failed login attempt.

In case you do not want to provide any link, replace the URL with `'disabled'`

logo_url

```
'logo_url' => 'https://example.org',
```

URL to use as target for the logo link in the header (top-left logo)

Defaults to the base URL of your Nextcloud instance

8.2.12 Mail Parameters

These configure the email settings for Nextcloud notifications and password resets.

mail_domain

```
'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

```
'mail_from_address' => 'nextcloud',
```

FROM address that overrides the built-in `sharing-noreply` and `lostpassword-noreply` FROM addresses.

Defaults to different FROM addresses depending on the feature.

mail_smtpdebug

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

Enable SMTP class debugging.

> **Note**
>
> **`loglevel` will likely need to be adjusted too. See docs:**
> https://docs.nextcloud.com/server/36/admin_manual/configuration_server/email_configuration.html#enabling-debug-mode

Defaults to `false`

mail_smtpmode

```
'mail_smtpmode' => 'smtp',
```

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

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

For the `sendmail` option, you need an installed and working email system on the server, with `/usr/sbin/sendmail` installed on your Unix system.

For `qmail`, the binary is /var/qmail/bin/sendmail, and it must be installed on your Unix system.

Use the string `null` to send no mails (disable mail delivery). This can be useful if mails should be sent via APIs and rendering messages is not necessary.

Defaults to `smtp`

mail_smtphost

```
'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 semicolon. If you need to specify the port number, append it to the IP address separated by a colon, like this: `127.0.0.1:24`.

Defaults to `127.0.0.1`

mail_smtpport

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

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

Defaults to `25`

mail_smtptimeout

```
'mail_smtptimeout' => 10,
```

This depends on `mail_smtpmode`. This sets the SMTP server timeout, in seconds. You may need to increase this if you are running an anti-malware or spam scanner.

Defaults to `10` seconds

mail_smtpsecure

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

This depends on `mail_smtpmode`. Specify `ssl` when you are using SSL/TLS. Any other value will be ignored.

If the server advertises STARTTLS capabilities, they might be used, but they cannot be enforced by this config option.

Defaults to `''` (empty string)

mail_smtpauth

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

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

Defaults to `false`

mail_smtpname

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

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

Defaults to `''` (empty string)

mail_smtppassword

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

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

Default to `''` (empty string)

mail_template_class

```
'mail_template_class' => '\OC\Mail\EMailTemplate',
```

Replaces the default mail template layout. This can be utilized if the options to modify the mail texts with the theming app are not enough.

The class must extend `\OC\Mail\EMailTemplate`

mail_send_plaintext_only

```
'mail_send_plaintext_only' => false,
```

Email will be sent by default with an HTML and a plain text body. This option allows sending only plain text emails.

mail_smtpstreamoptions

```
'mail_smtpstreamoptions' => [],
```

This depends on `mail_smtpmode`. Array of additional streams options that will be passed to underlying Swift mailer implementation.

Defaults to an empty array.

mail_sendmailmode

```
'mail_sendmailmode' => 'smtp',
```

Which mode is used for sendmail/qmail: `smtp` or `pipe`.

For `smtp`, the sendmail binary is started with the parameter `-bs`:

- Use the SMTP protocol on standard input and output.

For `pipe`, the binary is started with the parameters `-t`:

- Read message from STDIN and extract recipients.

Defaults to `smtp`

8.2.13 Proxy Configurations

overwritehost

```
'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

```
'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

```
'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

```
'overwritecondaddr' => '',
```

This option allows you to define a manual override condition as a regular expression for the remote IP address. For example, defining a range of IP addresses starting with `10.0.0.` and ending with 1 to 3: `^10\.0\.0\.[1-3]$`

Defaults to `''` (empty string)

overwrite.cli.url

```
'overwrite.cli.url' => '',
```

Set the canonical base URL Nextcloud should use when generating URLs outside a normal web request, such as in background jobs or at the command-line.

The value should be the full base URL including the web root, for example: `https://www.example.com/nextcloud`

In most setups, this should match the URL your users normally use to access Nextcloud. If it is incorrect, URL generation may be wrong in cron jobs, `occ`, notifications, and similar contexts.

> **Note**
>
> During installation, Nextcloud seeds this value from the installer's current access context. If it was not preseeded (for example, via `autoconfig.php`), the inferred value may not match the public URL that users normally use to access Nextcloud and may need adjustment.

Defaults to `''` (empty string)

htaccess.RewriteBase

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

To have clean URLs without `/index.php`, this parameter needs to be configured.

This parameter will be written as "RewriteBase" on update and installation of Nextcloud to your `.htaccess` file. While this value is often simply the URL path of the Nextcloud installation, it cannot be set automatically properly in every scenario and needs thus some manual configuration.

In a standard Apache setup, this usually equals the folder that Nextcloud is accessible at. So if Nextcloud is accessible via `https://mycloud.org/nextcloud`, the correct value would most likely be `/nextcloud`. If Nextcloud is running under `https://mycloud.org/`, then it would be `/`.

Note that the above rule is not valid in every case, as there are some rare setup cases where this may not apply. However, to avoid any update problems, this configuration value is explicitly opt-in.

After setting this value, run `occ maintenance:update:htaccess`. Now, when the following conditions are met, Nextcloud URLs won't contain `index.php`:

- `mod_rewrite` is installed
- `mod_env` is installed

Defaults to `''` (empty string)

htaccess.IgnoreFrontController

```
'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 any more. For example, try accessing resources by leaving out `index.php` in the URL.

proxy

```
'proxy' => '',
```

The URL of your proxy server, for example, `proxy.example.com:8081`.

> **Note**
>
> Guzzle (the HTTP library used by Nextcloud) reads the environment variables `HTTP_PROXY` (only for CLI requests), `HTTPS_PROXY`, and `NO_PROXY` by default.

If you configure a proxy with Nextcloud, any default configuration by Guzzle is overwritten. Make sure to set `proxyexclude` accordingly if necessary.

Defaults to `''` (empty string)

proxyuserpwd

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

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

The format is: `username:password`.

Defaults to `''` (empty string)

proxyexclude

```
'proxyexclude' => [],
```

List of hostnames that should not be proxied to.

For example: `['.mit.edu', 'foo.com']`.

Hint: Use something like `explode(',', getenv('NO_PROXY'))` to sync this value with the global `NO_PROXY` option.

Defaults to empty array.

allow_local_remote_servers

```
'allow_local_remote_servers' => true,
```

Allow remote servers with local addresses, e.g., in federated shares, webcal services, and more

Defaults to `false`

http_client_add_user_agent_url

```
'http_client_add_user_agent_url' => false,
```

Add the URL of the Nextcloud server in User-Agent headers HTTP calls.

This helps service providers identifying calls from your server, which can be helpful for them, but can be a privacy issue on small Nextcloud servers.

Defaults to `false`

8.2.14 Deleted Items (trash bin)

These parameters control the Deleted files app.

trashbin_retention_obligation

```
'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.

If the user quota limit is exceeded due to deleted files in the trash bin, retention settings will be ignored and files will be cleaned up until the quota requirements are met.

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. A file may be deleted after the minimum number of days has expired if space is needed. The file will not be deleted if space is not needed.

Whether "space is needed" depends on whether a user quota is defined or not:

- If no user quota is defined, the available space on the Nextcloud data partition sets the limit for the trashbin (issues: see https://github.com/nextcloud/server/issues/28451).
- If a user quota is defined, 50% of the user's remaining quota space sets the limit for the trashbin.

Maximum time is the number of days at which it is *guaranteed to be* deleted. There is no further dependency on the available space.

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 (D1 and D2 are configurable numbers):

- **auto**

  Default setting. Keeps files and folders in the trash bin for at least **30** days.
  Then, **if space is needed**, deletes trashed files anytime after that.

- **D1, auto**

  Keeps files and folders in the trash bin for at least **D1** days.
  Then, **if space is needed**, deletes trashed files anytime after that.

- **auto, D2**

  If space is needed, deletes trashed files anytime.
  After **D2** days, delete all trashed files automatically

- **D1, D2**

  Keeps files and folders in the trash bin for at least **D1** days.
  Then, after **D2** days, delete all trashed files automatically.

- **disabled**

  Trash bin auto clean is disabled, files and folders will be kept forever.

Defaults to `auto`

8.2.15 File versions

These parameters control the Versions app.

versions_retention_obligation

```
'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 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 expire rules

- **D1, D2**
  keep versions for at least D1 days and delete when exceeds D2 days

- **disabled**
  versions auto clean disabled, versions will be kept forever

Defaults to `auto`

8.2.16 Nextcloud Verifications

Nextcloud performs several verification checks. There are two options, `true` and `false`.

appcodechecker

```
'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 installing or enabling apps that pass this check.

Defaults to `false`

updatechecker

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

Check if Nextcloud is up-to-date and shows a notification if a new version is available. It sends current version, PHP version, installation and last update time, and release channel to the updater server which responds with the latest available version based on those metrics.

Defaults to `true`

updater.server.url

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

URL that Nextcloud should use to look for updates

Defaults to `https://updates.nextcloud.com/updater_server/`

updater.release.channel

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

The channel that Nextcloud should use to look for updates

Supported values:

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

has_internet_connection

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

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

Defaults to `true`

connectivity_check_domains

```
'connectivity_check_domains' => [
        'https://www.nextcloud.com',
        'https://www.startpage.com',
        'https://www.eff.org',
        'https://www.edri.org'
    ],
```

Which domains to request to determine the availability of an Internet connection. If none of these hosts are reachable, the administration panel will show a warning. Set to an empty list to not do any such checks (warning will still be shown).

If no protocol is provided, both http and https will be tested. For example, `http://www.nextcloud.com` and `https://www.nextcloud.com` will be tested for `www.nextcloud.com` If a protocol is provided, only this one will be tested.

Defaults to the following domains:

- https://www.nextcloud.com
- https://www.startpage.com
- https://www.eff.org
- https://www.edri.org

check_for_working_wellknown_setup

```
'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://example.tld/.well-known/caldav/`

Defaults to `true`

check_for_working_htaccess

```
'check_for_working_htaccess' => true,
```

This is a crucial security check on Apache servers that should always be set to `true`. This verifies that the `.htaccess` file is writable and works.

If it is not, then any options controlled by `.htaccess`, such as large file uploads, will not work. It also runs checks on the `data/` directory, which verifies that it can't be accessed directly through the Web server.

Defaults to `true`

check_data_directory_permissions

```
'check_data_directory_permissions' => true,
```

In rare setups (e.g., on OpenShift or Docker on Windows), the permissions check might block the installation while the underlying system offers no means to "correct" the permissions. In this case, set the value to false.

In regular cases, if issues with permissions are encountered, they should be adjusted accordingly. Changing the flag is discouraged.

Defaults to `true`

config_is_read_only

```
'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`, writing to the config file will be forbidden. Therefore, 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 and to set this switch to `false` for the update process.

Defaults to `false`

8.2.17 Logging

log_type

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

This parameter determines where the Nextcloud logs are sent.

- `file`: the logs are written to file `nextcloud.log` in the default Nextcloud data directory. The log file can be changed with parameter `logfile`.
- `syslog`: the logs are sent to the system log. This requires a syslog daemon to be active.
- `errorlog`: the logs are sent to the PHP `error_log` function.
- `systemd`: the logs are sent to the Systemd journal. This requires a system that runs Systemd and the Systemd journal. The PHP extension `systemd` must be installed and active.

Defaults to `file`

log_type_audit

```
'log_type_audit' => 'file',
```

This parameter determines where the audit logs are sent. See `log_type` for more information.

Defaults to `file`

logfile

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

Name of the file to which the Nextcloud logs are written if parameter `log_type` is set to `file`.

Defaults to `[datadirectory]/nextcloud.log`

logfile_audit

```
'logfile_audit' => '/var/log/audit.log',
```

Name of the file to which the audit logs are written if parameter `log_type` is set to `file`.

Defaults to `[datadirectory]/audit.log`

logfilemode

```
'logfilemode' => 0640,
```

Log file mode for the Nextcloud logging type in octal notation.

Defaults to `0640` (writable by user, readable by group).

loglevel

```
'loglevel' => 2,
```

Loglevel to start logging at. Valid values are:

- `0` = Debug
- `1` = Info
- `2` = Warning
- `3` = Error
- `4` = Fatal.

Defaults to `2` (Warning)

loglevel_frontend

```
'loglevel_frontend' => 2,
```

Loglevel used by the frontend to start logging at. The same values as for `loglevel` can be used. If not set, it defaults to the value configured for `loglevel` or Warning if that is not set either.

Defaults to `2`

loglevel_dirty_database_queries

```
'loglevel_dirty_database_queries' => 0,
```

Loglevel used by the dirty database query detection. Useful to identify potential database bugs in production. Set this to loglevel or higher to see dirty queries in the logs.

Defaults to `0` (debug)

syslog_tag

```
'syslog_tag' => 'Nextcloud',
```

If you maintain different instances and aggregate the logs, you may want to distinguish between them. `syslog_tag` can be set per instance with a unique ID. Only available if `log_type` is set to `syslog` or `systemd`.

The default value is `Nextcloud`.

syslog_tag_audit

```
'syslog_tag_audit' => 'Nextcloud',
```

If you maintain different instances and aggregate the logs, you may want to distinguish between them. `syslog_tag_audit` can be set per instance with a unique ID. Only available if `log_type` is set to `syslog` or `systemd`.

The default value is the value of `syslog_tag`.

log.condition

```
'log.condition' => [
        'shared_secret' => '57b58edb6637fe3059b3595cf9c41b9',
        'users' => ['sample-user'],
        'apps' => ['files'],
        'matches' => [
            [
                'shared_secret' => '57b58edb6637fe3059b3595cf9c41b9',
                'users' => ['sample-user'],
                'apps' => ['files'],
                'loglevel' => 1,
                'message' => 'contains substring'
            ],
        ],
    ],
```

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 debugging 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
- **`matches`: if all the conditions inside a group match,**
  this condition is met. This allows logging only entries to an app by a few users.

Defaults to an empty array.

log.backtrace

```
'log.backtrace' => false,
```

Enables logging a backtrace with each log line. Normally, only Exceptions carry backtrace information, which are logged automatically. This switch turns them on for any log message. Enabling this option will lead to increased log data size.

Defaults to `false`.

logdateformat

```
'logdateformat' => 'F d, Y H:i:s',
```

This uses PHP.date formatting; see https://www.php.net/manual/en/function.date.php

Defaults to ISO 8601 `2005-08-15T15:52:01+00:00`, see `\DateTime::ATOM` https://www.php.net/manual/en/class.datetimeinterface.php#datetimeinterface.constants.atom

logtimezone

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

The timezone for logfiles. See https://www.php.net/manual/en/timezones.php

Defaults to `UTC`

log_query

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

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

log_rotate_size

```
'log_rotate_size' => 100 * 1024 * 1024,
```

Enables log rotation and limits the total size of logfiles. Set it to 0 for no rotation. Specify a size in bytes, for example, 104857600 (100 megabytes = 100 * 1024 * 1024 bytes). A new logfile is created with a new name when the old logfile reaches your limit. If a rotated log file is already present, it will be overwritten.

Defaults to 100 MB

profiler

```
'profiler' => false,
```

Enable built-in profiler. Helpful when trying to debug performance issues.

Note that this has a performance impact and shouldn't be enabled on production.

profiling.request

```
'profiling.request' => false,
```

Enable profiling for individual requests if profiling single requests is enabled or the secret is passed.

This requires the excimer extension to be installed. Be careful with this, as it can generate a lot of data.

The profile data will be stored as a JSON file in the profiling.path directory that can be analyzed with speedscope.

Defaults to `false`

profiling.request.rate

```
'profiling.request.rate' => 0.001,
```

The rate at which profiling data is collected for individual requests.

A lower value means more data points but higher overhead.

Defaults to `0.001`

profiling.secret

```
'profiling.secret' => '',
```

A secret token that can be passed via ?profile_secret=<secret> to enable profiling for a specific request.

This allows profiling specific requests in production without enabling it globally.

No default value.

profiling.sample

```
'profiling.sample' => false,
```

Enable sampling-based profiling. This collects profiling data periodically rather than per-request.

This requires the excimer extension to be installed. Be careful with this, as it can generate a lot of data.

The profile data will be stored as a plain text file in the profiling.path directory that can be analyzed with speedscope.

Defaults to `false`

profiling.sample.rate

```
'profiling.sample.rate' => 1,
```

The rate at which sampling profiling data is collected in seconds.

A lower value means more frequent samples but higher overhead.

Defaults to `1`

profiling.sample.rotation

```
'profiling.sample.rotation' => 60,
```

How often (in minutes) the sample log files are rotated.

Defaults to `60`

profiling.path

```
'profiling.path' => '/tmp',
```

The directory where profiling data is stored.

Note that this directory must be writable by the web server user and will not be cleaned up automatically.

8.2.18 Alternate Code Locations

Some Nextcloud code may be stored in alternate locations.

customclient_desktop

```
'customclient_desktop'
        => 'https://nextcloud.com/install/#install-clients',
    'customclient_android'
```

(continues on next page)

(continued from previous page)

```
    => 'https://play.google.com/store/apps/details?id=com.nextcloud.client',
  'customclient_ios'
    => 'https://itunes.apple.com/us/app/nextcloud/id1125420102?mt=8',
  'customclient_ios_appid'
    => '1125420102',
  'customclient_fdroid'
    => 'https://f-droid.org/packages/com.nextcloud.client/',
```

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

Defaults to:

- Desktop client: `https://nextcloud.com/install/#install-clients`
- Android client: `https://play.google.com/store/apps/details?id=com.nextcloud.client`
- iOS client: `https://itunes.apple.com/us/app/nextcloud/id1125420102?mt=8`
- iOS client app ID: `1125420102`
- F-Droid client: `https://f-droid.org/packages/com.nextcloud.client/`

8.2.19 Activity

Options for the activity app.

activity_expire_days

```
'activity_expire_days' => 365,
```

Retention of activities.

A daily cron job deletes all activities for all users which are older than the number of days specified here.

Defaults to `365`

activity_use_cached_mountpoints

```
'activity_use_cached_mountpoints' => false,
```

Activities in Team Folders and External Storages.

By default, activities in team folders or external storages are only generated for the current user. This is due to a limitations in current implementations. This config flag makes activities in group folders and external storages work like in normal shares (when set to `true`). Setting this flag does not allow past activities to be displayed (no retroactivity).

Warning

Enabling this comes with some CRITICAL trade-offs:

- If team folder "Advanced Permissions" (ACLs) are used, activities do not respect the permissions and therefore all users see all activities, even for files and directories they do not have access to.
- Users who had access to a team folder, share, or external storage can see activities in their stream and emails that happen after they are removed until they log in again.

- Users who are newly added to a team folder, share, or external storage cannot see activities in their stream or emails that happen after they are added until they log in again.

Defaults to `false`

8.2.20 Apps

Options for the Apps folder, Apps store, and App code checker.

defaultapp

```
'defaultapp' => 'dashboard,files',
```

Set the default app to open on login. The entry IDs can be retrieved from the Navigations OCS API endpoint: https://docs.nextcloud.com/server/36/developer_manual/_static/openapi.html#/operations/core-navigation-get-apps-navigation.

You can use a comma-separated list of app names, so if the first app is not enabled for a user, then Nextcloud will try the second one, and so on. If no enabled apps are found, it defaults to the dashboard app.

Defaults to `dashboard,files`

appstoreenabled

```
'appstoreenabled' => true,
```

When enabled, admins may install apps from the Nextcloud app store.

Defaults to `true`

appstoreurl

```
'appstoreurl' => 'https://apps.nextcloud.com/api/v1',
```

Enables the installation of apps from a self-hosted apps store.

Requires that at least one of the configured apps directories is writable.

Defaults to `https://apps.nextcloud.com/api/v1`

appsallowlist

```
'appsallowlist' => [],
```

Filters allowed installable apps from the appstore.

Empty array will prevent all apps from the store to be found.

apps_paths

```
'apps_paths' => [
        [
            'path' => '/var/www/nextcloud/apps',
            'url' => '/apps',
            'writable' => true,
```

(continues on next page)

(continued from previous page)

```
        ],
    ],
```

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.

8.2.21 Previews

Nextcloud supports generating previews for various file types, such as images, audio files, and text files. These options control enabling and disabling previews, and thumbnail size.

enable_previews

```
'enable_previews' => true,
```

By default, Nextcloud can generate previews for the following filetypes:

- Image files
- Text documents

Valid values are `true`, to enable previews, or `false`, to disable previews

Defaults to `true`

preview_concurrency_all

```
'preview_concurrency_all' => 8,
```

Number of all preview requests being processed concurrently, including previews that need to be newly generated, and those that have been generated.

This should be greater than `preview_concurrency_new`. If unspecified, defaults to twice the value of `preview_concurrency_new`.

preview_concurrency_new

```
'preview_concurrency_new' => 4,
```

Number of new previews that are being concurrently generated.

Depending on the max preview size set by `preview_max_x` and `preview_max_y`, the generation process can consume considerable CPU and memory resources. It's recommended to limit this to be no greater than the number of CPU cores. If unspecified, defaults to the number of CPU cores, or 4 if that cannot be determined.

preview_max_x

```
'preview_max_x' => 4096,
```

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

Defaults to `4096`

preview_max_y

```
'preview_max_y' => 4096,
```

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

Defaults to `4096`

preview_max_filesize_image

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

Max file size for generating image previews with imagegd (default behavior).

If the image is bigger, it'll try other preview generators, but will most likely either show the default mimetype icon or not display the image at all. Set to `-1` for no limit and try to generate image previews on all file sizes.

Defaults to `50` megabytes

preview_max_memory

```
'preview_max_memory' => 256,
```

Max memory for generating image previews with imagegd (default behavior) Reads the image dimensions from the header and assumes 32 bits per pixel.

If creating the image would allocate more memory, preview generation will be disabled and the default mimetype icon is shown. Set to `-1` for no limit.

Defaults to `256` megabytes

preview_libreoffice_path

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

Custom path for LibreOffice/OpenOffice binary

Defaults to `''` (empty string)

preview_ffmpeg_path

```
'preview_ffmpeg_path' => '/usr/bin/ffmpeg',
```

Custom path for ffmpeg binary

Defaults to `null` and falls back to searching `ffmpeg` in the configured `PATH` environment

preview_ffprobe_path

```
'preview_ffprobe_path' => '/usr/bin/ffprobe',
```

Custom path for ffprobe binary

Defaults to `null` and falls back to using the same path as ffmpeg. ffprobe is typically packaged with ffmpeg and is required for enhanced preview generation for HDR videos.

preview_imaginary_url

```
'preview_imaginary_url' => 'http://previews_hpb:8088/',
```

Set the URL of the Imaginary service to send image previews to.

Also requires the `OC\Preview\Imaginary` provider to be enabled in the `enabledPreviewProviders` array, to create previews for these mimetypes: bmp, x-bitmap, png, jpeg, gif, heic, heif, svg+xml, tiff, webp, and illustrator.

If you want Imaginary to also create preview images from PDF documents, you have to add the `OC\Preview\ImaginaryPDF` provider as well.

See https://github.com/h2non/imaginary

preview_imaginary_key

```
'preview_imaginary_key' => 'secret',
```

If you want to set an API key for Imaginary.

enabledPreviewProviders

```
'enabledPreviewProviders' => [
        'OC\Preview\PNG',
        'OC\Preview\JPEG',
        'OC\Preview\GIF',
        'OC\Preview\BMP',
        'OC\Preview\XBitmap',
        'OC\Preview\Krita',
        'OC\Preview\WebP',
        'OC\Preview\MarkDown',
        'OC\Preview\TXT',
        'OC\Preview\OpenDocument',
    ],
```

Only register providers that have been explicitly enabled

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

- `OC\Preview\EMF`
- `OC\Preview\Font`
- `OC\Preview\HEIC`
- `OC\Preview\Illustrator`
- `OC\Preview\Movie`
- `OC\Preview\MP3`
- `OC\Preview\MSOffice2003`
- `OC\Preview\MSOffice2007`
- `OC\Preview\MSOfficeDoc`
- `OC\Preview\PDF`
- `OC\Preview\Photoshop`
- `OC\Preview\Postscript`

- OC\Preview\SGI
- OC\Preview\StarOffice
- OC\Preview\SVG
- OC\Preview\TGA
- OC\Preview\TIFF

The following providers are disabled by default, because they provide an alternative to the built-in providers:

- OC\Preview\Imaginary
- OC\Preview\ImaginaryPDF

Defaults to the following providers:

- OC\Preview\PNG
- OC\Preview\JPEG
- OC\Preview\GIF
- OC\Preview\BMP
- OC\Preview\XBitmap
- OC\Preview\Krita
- OC\Preview\WebP
- OC\Preview\MarkDown
- OC\Preview\TXT
- OC\Preview\OpenDocument

metadata_max_filesize

```
'metadata_max_filesize' => 256,
```

Maximum file size for file metadata generation.

Files larger than this limit will be skipped.

This limit helps bound resource usage during metadata generation. Actual resource usage depends on the active metadata providers and how they process files. As a rough guide, memory usage may scale with file size.

Default: 256 MiB.

max_file_conversion_filesize

```
'max_file_conversion_filesize' => 100,
```

Maximum file size for file conversion.

Files larger than this limit will be skipped.

Raising this limit may increase conversion time, resource usage, and the risk of timeouts or conversion failures depending on the provider.

Default: 100 MiB.

8.2.22 LDAP

Global settings used by LDAP User and Group Backend

ldapUserCleanupInterval

```
'ldapUserCleanupInterval' => 51,
```

Defines the interval in minutes for the background job that checks user existence and marks them as ready to be cleaned up. The number is always minutes. Setting it to 0 disables the feature.

See command line (occ) methods `ldap:show-remnants` and `user:delete`

Defaults to `51` minutes

sort_groups_by_name

```
'sort_groups_by_name' => false,
```

Sort groups in the user settings by name instead of the user count

By enabling this, the user count beside the group name is disabled as well.

Deprecated since version 29.0.0: Use the frontend instead or set the app config value `group.sortBy` for `core` to `2`

8.2.23 Comments

Global settings for the Comments infrastructure

comments.managerFactory

```
'comments.managerFactory' => '\OC\Comments\ManagerFactory',
```

Replaces the default Comments Manager Factory. This can be utilized if an own or 3rd-party CommentsManager should be used that – for instance – uses the filesystem instead of the database to keep the comments.

Defaults to `\OC\Comments\ManagerFactory`

8.2.24 Maintenance

These options are for halting user activity when you are performing server maintenance.

maintenance

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

Enable maintenance mode to disable Nextcloud

If you want to prevent users from logging in to Nextcloud before you start doing some maintenance work, you need to set the value of the maintenance parameter to true. Please keep in mind that users who are already logged in are kicked out of Nextcloud instantly.

Defaults to `false`

maintenance_window_start

```
'maintenance_window_start' => 1,
```

UTC Hour for maintenance windows

Some background jobs only run once a day. When an hour is defined for this config, the background jobs which advertise themselves as not time sensitive will be delayed during the "working" hours and only run in the 4 hours after the given time. This is, e.g., used for activity expiration, suspicious login training, and update checks.

A value of 1, e.g., will only run these background jobs between 01:00am UTC and 05:00am UTC.

Defaults to `100` which disables the feature

ldap_log_file

```
'ldap_log_file' => '',
```

Log all LDAP requests into a file

Warning: This heavily decreases the performance of the server and is only meant to debug/profile the LDAP interaction manually. Also, it might log sensitive data into a plain text file.

8.2.25 SSL

openssl

```
'openssl' => [
        'config' => '/absolute/location/of/openssl.cnf',
    ],
```

Extra SSL options to be used for configuration.

Defaults to an empty array.

8.2.26 Memory caching backend configuration

Available cache backends:

- `\OC\Memcache\APCu` APC user backend
- `\OC\Memcache\ArrayCache` In-memory array-based backend (not recommended)
- `\OC\Memcache\Memcached` Memcached backend
- `\OC\Memcache\Redis` Redis backend

Advice on choosing between the various backends:

- APCu should be easiest to install. Almost all distributions have packages. Use this for single user environment for all caches.
- Use Redis or Memcached for distributed environments. For the local cache (you can configure two) take APCu.

memcache.local

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

Memory caching backend for locally stored data

- Used for host-specific data, e.g., file paths

Defaults to `none`

memcache.distributed

```
'memcache.distributed' => '\\OC\\Memcache\\Memcached',
```

Memory caching backend for distributed data

- Used for installation-specific data, e.g., database caching
- If unset, defaults to the value of memcache.local

Defaults to `none`

memcache_customprefix

```
'memcache_customprefix' => 'mycustomprefix',
```

Cache Key Prefix for Redis or Memcached

- Used for avoiding collisions in the cache system
- May be used for ACL restrictions in Redis

Defaults to `''` (empty string)

memcache.kvstore

```
'memcache.kvstore' => [
```

Connection details for the Key-Value store used for in-memory caching, for example when using Valkey or Redis.

This is the brand-independent version of the `redis` and `redis.cluster` options below without depending on `php-redis` meaning no additional PHP extension needs to be installed. Redis and Valkey 4.0 up to 8.0 are supported, for SSL support Redis v6 or higher is required, Valkey 9 is supported but without support for numbered databased when using cluster mode.

Warning: If the cache server is not hosted on the same machine as Nextcloud, this can have a network overhead compared to the `php-redis` based backend below.

Three topologies are supported: a single server, a Sentinel managed replication set, and a server cluster. Configure exactly one of `server`, `sentinel` or `seeds`.

For enhanced security, it is recommended to configure ACLs in the cache server and configure the `user` and `password` (or a TLS client certificate). Alternatively, you can also configure the cache server to just use a `password`. See https://valkey.io/topics/security/ for more information when using Valkey.

server

```
'server' => [
            'host' => 'localhost', // can also be a Unix domain socket: '/tmp/cache.
↪sock'
            'port' => 6379, // ignored for Unix domain sockets
            // Protocol used to connect. One of 'tcp', 'tls' or 'unix'.
            // When omitted it is derived from the host (a leading '/' means 'unix').
            'protocol' => 'tcp',
        ],
```

Single server setup.

Also used as the connection template for the `sentinel` managed primary / replica connections.

8.2.27 Sentinel managed replication setup.

Provide the name of the monitored service and one entry per Sentinel node. Each seed uses the same format as the `server` entry above. Uncomment to enable and remove the `server` / `seeds` entries.

user

```
//'seeds' => [
        //    ['host' => 'localhost', 'port' => 7000],
        //    ['host' => 'localhost', 'port' => 7001],
        //],

        // Optional: username, only sent when the cache server uses ACLs.
        'user' => '',
        // Optional: if not defined, no password will be used.
        'password' => '',
        // Optional: select a numbered database. Only supported for single
        // servers and Sentinel setups, clusters always use database 0.
        'dbindex' => 0,
        // Optional: connection timeout in seconds (float). 0 means no timeout.
        'timeout' => 0.0,
        // Optional: read/write timeout in seconds (float). 0 means no timeout.
        'read_timeout' => 0.0,
        // Optional: keep the connection open across requests. Defaults to false.
        'persistent' => false,
        // Optional: when the 'tls' protocol is used, provide the SSL context.
        // SSL context options, see https://www.php.net/manual/en/context.ssl.php
        //'ssl_context' => [
        //    'local_cert' => '/certs/cache.crt',
        //    'local_pk' => '/certs/cache.key',
        //    'cafile' => '/certs/ca.crt',
        //],
    ],
```

Cluster setup.

Provide some or all of the cluster nodes to bootstrap discovery. Each seed uses the same format as the `server` entry above. Uncomment to enable and remove the `server` / `sentinel` entries.

redis

```
'redis' => [
        'host' => 'localhost', // can also be a Unix domain socket: '/tmp/redis.sock'
        'port' => 6379,
        'timeout' => 0.0,
        'read_timeout' => 0.0,
        'user' => '', // Optional: if not defined, no password will be used.
        'password' => '', // Optional: if not defined, no password will be used.
        'dbindex' => 0, // Optional: if undefined, SELECT will not run and will use␣
↪Redis Server's default DB Index.
```

(continues on next page)

(continued from previous page)

```
        // If Redis in-transit encryption is enabled, provide certificates
        // SSL context https://www.php.net/manual/en/context.ssl.php
        'ssl_context' => [
            'local_cert' => '/certs/redis.crt',
            'local_pk' => '/certs/redis.key',
            'cafile' => '/certs/ca.crt'
        ]
    ],
```

Connection details for Redis to use for memory caching in a single server configuration.

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

We also support Redis SSL/TLS encryption as of version 6. See https://redis.io/topics/encryption for more information.

redis.cluster

```
'redis.cluster' => [
        'seeds' => [ // provide some or all of the cluster servers to bootstrap␣
↪discovery, port required
            'localhost:7000',
            'localhost:7001',
        ],
        'timeout' => 0.0,
        'read_timeout' => 0.0,
        'failover_mode' => \RedisCluster::FAILOVER_ERROR,
        'user' => '', // Optional: if not defined, no password will be used.
        'password' => '', // Optional: if not defined, no password will be used.
        // If Redis in-transit encryption is enabled, provide certificates
        // SSL context https://www.php.net/manual/en/context.ssl.php
        'ssl_context' => [
            'local_cert' => '/certs/redis.crt',
            'local_pk' => '/certs/redis.key',
            'cafile' => '/certs/ca.crt'
        ]
    ],
```

Connection details for a Redis Cluster.

Redis Cluster support requires the PHP module phpredis in version 3.0.0 or higher.

Available failover modes:

- `\RedisCluster::FAILOVER_NONE` - only send commands to master nodes (default)
- `\RedisCluster::FAILOVER_ERROR` - failover to slaves for read commands if master is unavailable (recommended)
- `\RedisCluster::FAILOVER_DISTRIBUTE` - randomly distribute read commands across master and slaves

Warning

`\RedisCluster::FAILOVER_DISTRIBUTE` is a not recommended setting, and we strongly suggest to not use it if you use Redis for file locking. Due to the way Redis is synchronized, it could happen that the read for an existing lock is scheduled to a slave that is not fully synchronized with the connected master which then causes a FileLocked exception.

See https://redis.io/topics/cluster-spec for details about the Redis cluster

Authentication works with phpredis version 4.2.1+. See https://github.com/phpredis/phpredis/commit/c5994f2a42b8a348af92d3acb4edff1328ad8ce1

memcached_servers

```
'memcached_servers' => [
        // hostname, port and optional weight
        // or path and port 0 for Unix socket. Also see:
        // https://www.php.net/manual/en/memcached.addservers.php
        // https://www.php.net/manual/en/memcached.addserver.php
        ['localhost', 11211],
        //array('other.host.local', 11211),
    ],
```

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

memcached_options

```
'memcached_options' => [
        // Set timeouts to 50ms
        \Memcached::OPT_CONNECT_TIMEOUT => 50,
        \Memcached::OPT_RETRY_TIMEOUT => 50,
        \Memcached::OPT_SEND_TIMEOUT => 50,
        \Memcached::OPT_RECV_TIMEOUT => 50,
        \Memcached::OPT_POLL_TIMEOUT => 50,

        // Enable compression
        \Memcached::OPT_COMPRESSION => true,

        // Turn on consistent hashing
        \Memcached::OPT_LIBKETAMA_COMPATIBLE => true,

        // Enable Binary Protocol
        \Memcached::OPT_BINARY_PROTOCOL => true,

        // Binary serializer will be enabled if the igbinary PECL module is available
        //\Memcached::OPT_SERIALIZER => \Memcached::SERIALIZER_IGBINARY,
    ],
```

Connection options for Memcached

cache_path

```
'cache_path' => '',
```

Location of the cache folder, defaults to `data/$user/cache` where `$user` is the current user. When specified, the format will change to `$cache_path/$user` where `$cache_path` is the configured cache directory and `$user` is the user.

Defaults to `''` (empty string)

cache_chunk_gc_ttl

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

TTL of chunks located in the cache folder before they're removed by garbage collection (in seconds). Increase this value if users have issues uploading very large files via the Nextcloud Client as upload isn't completed within one day.

Defaults to `60*60*24` (1 day)

cache_app_config

```
'cache_app_config' => true,
```

Enable caching of the app config values.

If enabled the app config will be cached locally for a short TTL, reducing database load significantly on larger setups.

Defaults to `true`

8.2.28 Using Object Store with Nextcloud

objectstore

```
'objectstore' => [
        'class' => 'OC\\Files\\ObjectStore\\Swift',
        'arguments' => [
            // trystack will use your Facebook ID as the username
            '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',
            // uploadPartSize: size of the uploaded chunks, defaults to 524288000
            'uploadPartSize' => 524288000,
            // required on dev-/trystack
            'tenantName' => 'facebook100000123456789',
```

(continues on next page)

(continued from previous page)

```
            // dev-/trystack uses swift by default, the lib defaults to 'cloudFiles'
            // if omitted
            'serviceName' => 'swift',
            // The Interface / URL Type, optional
            'urlType' => 'internal',
            // Maximum amount of data that can be uploaded
            'totalSizeLimit' => 1024 * 1024 * 1024,
        ],
    ],
```

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 I/O 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.

objectstore

```
'objectstore' => [
        'class' => 'OC\\Files\\ObjectStore\\Swift',
        'arguments' => [
            'autocreate' => true,
            'user' => [
                'name' => 'swift',
                'password' => 'swift',
                'domain' => [
                    'name' => 'default',
                ],
            ],
            'scope' => [
                'project' => [
                    'name' => 'service',
                    'domain' => [
                        'name' => 'default',
                    ],
                ],
            ],
            'tenantName' => 'service',
            'serviceName' => 'swift',
            'region' => 'regionOne',
            'url' => 'http://yourswifthost:5000/v3',
            'bucket' => 'nextcloud',
        ],
    ],
```

To use Swift V3

objectstore

```
'objectstore' => [
        'class' => 'OC\\Files\\ObjectStore\\S3',
        'arguments' => [
            'bucket' => 'nextcloud',
            'key' => 'your-access-key',
            'secret' => 'your-secret-key',
            'hostname' => 's3.example.com',
            'port' => 443,
            'use_ssl' => true,
            'region' => 'us-east-1',
            // optional: Maximum number of retry attempts for failed S3 requests
            // Default: 5
            'retriesMaxAttempts' => 5,
            // Data Integrity Protections for Amazon S3 (https://docs.aws.amazon.com/
↪sdkref/latest/guide/feature-dataintegrity.html)
            // Valid values are "when_required" (default) and "when_supported".
            // To ensure compatibility with 3rd party S3 implementations, Nextcloud␣
↪disables it by default. However, if you are
            // using Amazon S3 (or any other implementation that supports it) we␣
↪recommend enabling it by using "when_supported".
            'request_checksum_calculation' => 'when_required',
            'response_checksum_validation' => 'when_required',
        ],
    ],
```

To use S3 object storage

objectstore.multibucket.preview-distribution

```
'objectstore.multibucket.preview-distribution' => false,
```

If this is set to true and a multibucket object store is configured, then newly created previews are put into 256 dedicated buckets.

Those buckets are named like the multibucket version but with the postfix `-preview-NUMBER` where NUMBER is between 0 and 255.

Keep in mind that only previews of files are put in there that don't have some already. Otherwise, the old bucket will be used.

To migrate existing previews to this new multibucket distribution of previews, use the occ command `preview:repair`. For now, this will only migrate previews that were generated before Nextcloud 19 in the flat `appdata_INSTANCEID/previews/FILEID` folder structure.

8.2.29 Sharing

Global settings for Sharing

sharing.managerFactory

```
'sharing.managerFactory' => '\OC\Share20\ProviderFactory',
```

Replaces the default Share Provider Factory. This can be utilized if own or 3rd-party Share Providers are used that – for instance – use the filesystem instead of the database to keep the share information.

Defaults to `\OC\Share20\ProviderFactory`

sharing.enable_mail_link_password_expiration

```
'sharing.enable_mail_link_password_expiration' => false,
```

Enables expiration for link share passwords sent by email (sharebymail).

The passwords will expire after the configured interval; the users can still request a new one on the public link page.

sharing.mail_link_password_expiration_interval

```
'sharing.mail_link_password_expiration_interval' => 3600,
```

Expiration interval for passwords, in seconds.

sharing.maxAutocompleteResults

```
'sharing.maxAutocompleteResults' => 25,
```

Define max number of results returned by the search for auto-completion of users, groups, etc. The value must not be lower than 0 (for unlimited).

If more, different sources are requested (e.g., different user backends; or both users and groups), the value is applied per source and might not be truncated after collecting the results. I.e., more results can appear than configured here.

Default is 25.

sharing.minSearchStringLength

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

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

sharing.enable_share_accept

```
'sharing.enable_share_accept' => false,
```

Set to true to enable that internal shares need to be accepted by the users by default.

Users can change this for their account in their personal sharing settings

sharing.force_share_accept

```
'sharing.force_share_accept' => false,
```

Set to `true` to enforce that internal shares need to be accepted

sharing.allow_custom_share_folder

```
'sharing.allow_custom_share_folder' => true,
```

Set to `false` to prevent users from setting a custom share_folder

share_folder

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

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

Changes to this value will only have effect on new shares.

Defaults to `/`

sharing.enable_share_mail

```
'sharing.enable_share_mail' => true,
```

Set to `false` to stop sending a mail when users receive a share

sharing.allow_disabled_password_enforcement_groups

```
'sharing.allow_disabled_password_enforcement_groups' => false,
```

Set to true to enable the feature to add exceptions for share password enforcement

transferIncomingShares

```
'transferIncomingShares' => false,
```

Set to true to always transfer incoming shares by default when running `occ files:transfer-ownership`.

Defaults to `false`, so incoming shares are not transferred if not specifically requested by a command line argument.

8.2.30 Federated Cloud Sharing

sharing.federation.allowSelfSignedCertificates

```
'sharing.federation.allowSelfSignedCertificates' => false,
```

Allow self-signed certificates for federated shares

sharing.federation.ocm.apiVersion

```
'sharing.federation.ocm.apiVersion' => '',
```

Override the `apiVersion` advertised in the local OCM discovery document.

Setting this to a non-empty string also removes the non-standard `version` field. Leave empty for compatibility with older Nextcloud servers.

Defaults to `''` (empty string)

sharing.federation.ocm.removePublicKey

```
'sharing.federation.ocm.removePublicKey' => false,
```

Remove the non-standard `publicKey` field from the local OCM discovery document. Enable this only when all federated peers support RFC 9421 HTTP signatures, for example when all peers run Nextcloud 35 or higher.

Defaults to `false`

8.2.31 Hashing

hashing_default_password

```
'hashing_default_password' => false,
```

By default, Nextcloud will use the Argon2 password hashing if available.

However, if for whatever reason you want to stick with the PASSWORD_DEFAULT of your PHP version, then set the setting to true.

Nextcloud uses the Argon2 algorithm (with PHP >= 7.2) to create hashes by its own and exposes its configuration options as following. More information can be found at: https://www.php.net/manual/en/function.password-hash.php

hashingThreads

```
'hashingThreads' => PASSWORD_ARGON2_DEFAULT_THREADS,
```

The number of CPU threads to be used by the algorithm for computing a hash.

The value must be an integer, and the minimum value is `1`. Rationally, it does not help to provide a number higher than the available threads on the machine. Values that undershoot the minimum will be ignored in favor of the minimum.

hashingMemoryCost

```
'hashingMemoryCost' => PASSWORD_ARGON2_DEFAULT_MEMORY_COST,
```

The memory in KiB to be used by the algorithm for computing a hash. The value must be an integer, and the minimum value is 8 times the number of CPU threads.

Values that undershoot the minimum will be ignored in favor of the minimum.

hashingTimeCost

```
'hashingTimeCost' => PASSWORD_ARGON2_DEFAULT_TIME_COST,
```

The number of iterations that are used by the algorithm for computing a hash.

The value must be an integer, and the minimum value is `1`. Values that undershoot the minimum will be ignored in favor of the minimum.

hashingCost

```
'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

8.2.32 All other configuration options

dbdriveroptions

```
'dbdriveroptions' => [
        PDO::MYSQL_ATTR_SSL_CA => '/file/path/to/ca_cert.pem',
        PDO::MYSQL_ATTR_SSL_KEY => '/file/path/to/mysql-client-key.pem',
        PDO::MYSQL_ATTR_SSL_CERT => '/file/path/to/mysql-client-cert.pem',
```

(continues on next page)

(continued from previous page)

```
        PDO::MYSQL_ATTR_SSL_VERIFY_SERVER_CERT => false,
        PDO::MYSQL_ATTR_INIT_COMMAND => 'SET wait_timeout = 28800'
    ],
```

Additional driver options for the database connection, e.g., to enable SSL encryption in MySQL or specify a custom wait timeout on a cheap hoster.

When setting up TLS/SSL for encrypting the connections, you need to ensure that the passed keys and certificates are readable by the PHP process. In addition, `PDO::MYSQL_ATTR_SSL_VERIFY_SERVER_CERT` might need to be set to false, if the database server's certificate CN does not match with the hostname used to connect. The standard behavior here is different from the MySQL/MariaDB CLI client, which does not verify the server cert except `--ssl-verify-server-cert` is passed manually.

sqlite.journal_mode

```
'sqlite.journal_mode' => 'DELETE',
```

SQLite3 journal mode can be specified using this configuration parameter - can be `'WAL'` or `'DELETE'`. See https://www.sqlite.org/wal.html for more details.

mysql.utf8mb4

```
'mysql.utf8mb4' => false,
```

During setup, if requirements are met (see below), this setting is set to true to enable MySQL to handle 4-byte characters instead of 3-byte characters.

To convert an existing 3-byte setup to a 4-byte setup, configure the MySQL parameters as described below and run the migration command: `./occ db:convert-mysql-charset` This config setting will be automatically updated after a successful migration.

Refer to the documentation for more details.

MySQL requires specific settings for longer indexes (> 767 bytes), which are necessary for 4-byte character support:

```
[mysqld]
innodb_large_prefix=ON
innodb_file_format=Barracuda
innodb_file_per_table=ON
```

Tables will be created with:

- character set: `utf8mb4`
- collation: `utf8mb4_bin`
- row_format: `dynamic`

See:

- https://dev.mysql.com/doc/refman/5.7/en/charset-unicode-utf8mb4.html
- https://dev.mysql.com/doc/refman/5.7/en/innodb-parameters.html#sysvar_innodb_large_prefix
- https://mariadb.com/kb/en/mariadb/xtradbinnodb-server-system-variables/#innodb_large_prefix
- http://www.tocker.ca/2013/10/31/benchmarking-innodb-page-compression-performance.html
- http://mechanics.flite.com/blog/2014/07/29/using-innodb-large-prefix-to-avoid-error-1071/

mysql.collation

```
'mysql.collation' => null,
```

For search queries in the database, a default collation is chosen based on the character set. In some cases, a different collation is desired, such as for accent-sensitive searches.

MariaDB and MySQL share some collations, but also have incompatible ones, depending on the database server version.

This option allows overriding the automatic collation choice. Example:

```
'mysql.collation' => 'utf8mb4_0900_as_ci',
```

This setting does not affect table creation or setup, where utf8[mb4]_bin is always used. It applies only to SQL queries using LIKE comparison operators.

pgsql_ssl

```
'pgsql_ssl' => [
        'mode' => '',
        'cert' => '',
        'rootcert' => '',
        'key' => '',
        'crl' => '',
    ],
```

PostgreSQL SSL connection

supportedDatabases

```
'supportedDatabases' => [
        'sqlite',
        'mysql',
        'pgsql',
        'oci',
    ],
```

Database types supported for installation.

Available:

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

Defaults to:

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

tempdirectory

```
'tempdirectory' => '/tmp/nextcloudtemp',
```

Override the location where Nextcloud stores temporary files. Useful in setups where the system temporary directory is on a limited-space ramdisk, restricted, or when using external storage that does not support streaming.

The web server user/PHP must have write access to this directory. Ensure that PHP configuration recognizes this as a valid temporary directory by setting the TMP, TMPDIR, and TEMP environment variables accordingly. Additional permissions may be required for AppArmor or SELinux.

updatedirectory

```
'updatedirectory' => '',
```

Override the location where Nextcloud stores update files during updates.

Useful when the default `datadirectory` is on a network disk like NFS or is otherwise restricted. Defaults to the value of `datadirectory` if unset.

If set, the directory must be located outside the Nextcloud installation directory and writable by the web server user.

forbidden_filenames

```
'forbidden_filenames' => ['.htaccess'],
```

Block specific files or filenames, disallowing uploads or access (read and write).

`.htaccess` is blocked by default.

Warning

Use this only if you understand the implications.

Note

This list is case-insensitive.

Defaults to `['.htaccess']`

forbidden_filename_basenames

```
'forbidden_filename_basenames' => [],
```

Disallow uploads of files with specific basenames. Matching existing files cannot be updated, and no new files can be created in matching folders.

The basename is the filename without the extension, e.g., for “archive.tar.gz”, the basename is “archive”.

Note

This list is case-insensitive.

Defaults to `[]` (empty array)

forbidden_filename_characters

```
'forbidden_filename_characters' => [],
```

Block specific characters in filenames. Useful for filesystems or operating systems (e.g., Windows) that do not support certain characters. Matching existing files cannot be updated, and no new files can be created in matching folders.

The `/` and `\` characters, as well as ASCII characters [0-31], are always forbidden.

Example for Windows: `['?', '<', '>', ':', '*', '|', '"']` See: https://en.wikipedia.org/wiki/Comparison_of_file_systems#Limits

Defaults to `[]` (empty array)

forbidden_filename_extensions

```
'forbidden_filename_extensions' => ['.part', '.filepart'],
```

Deny specific file extensions. Matching existing files cannot be updated, and no new files can be created in matching folders.

The `'.part'` extension is always forbidden, as it is used internally by Nextcloud.

Defaults to `['.filepart', '.part']`

theme

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

Specify the name of a theme to apply to Nextcloud. Themes are located in `nextcloud/themes/` by default.

Defaults to the theming app, included since Nextcloud 9.

enforce_theme

```
'enforce_theme' => '',
```

Enforce a specific user theme, disabling user theming settings. Must be a valid ITheme ID, e.g., `dark`, `dark-highcontrast`, `default`, `light`, `light-highcontrast`, `opendyslexic`.

theming.standalone_window.enabled

```
'theming.standalone_window.enabled' => true,
```

Enable or disable Progressive Web App (PWA) functionality, which allows browsers to open web applications in dedicated windows.

Defaults to `true`

cipher

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

Specify the default cipher for encrypting files. Supported ciphers:

- AES-256-CTR

- AES-128-CTR
- AES-256-CFB
- AES-128-CFB

Defaults to `AES-256-CTR`

encryption.use_legacy_base64_encoding

```
'encryption.use_legacy_base64_encoding' => false,
```

Use the legacy base64 format for encrypted files instead of the more space-efficient binary format. This affects only newly written files; existing encrypted files remain readable regardless of the format.

Defaults to `false`

minimum.supported.desktop.version

```
'minimum.supported.desktop.version' => '3.3.50',
```

Specify the minimum Nextcloud desktop client version allowed to sync with this server. Connections from earlier clients will be denied. Defaults to the minimum officially supported version at the time of this server release.

Changing this may cause older, unsupported clients to malfunction, potentially leading to data loss or unexpected behavior.

Defaults to `3.3.50`

maximum.supported.desktop.version

```
'maximum.supported.desktop.version' => '99.99.99',
```

Specify the maximum Nextcloud desktop client version allowed to sync with this server. Connections from later clients will be denied.

Defaults to `99.99.99`

localstorage.allowsymlinks

```
'localstorage.allowsymlinks' => false,
```

Allow local storage to contain symlinks.

> **Warning**
>
> Not recommended, as this allows Nextcloud to access files outside the data directory, posing a potential security risk.

Defaults to `false`

localstorage.umask

```
'localstorage.umask' => 0022,
```

Nextcloud overrides umask to ensure suitable access permissions regardless of web server or PHP-FPM configuration. Modifying this value has security implications and may cause issues with the installation.

Most installations should not modify this value.

Defaults to `0022`

localstorage.unlink_on_truncate

```
'localstorage.unlink_on_truncate' => false,
```

Allow storage systems that do not support modifying existing files to overcome this limitation by removing files before overwriting.

Defaults to `false`

quota_include_external_storage

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

Include external storage mounts in quota calculations.

When enabled, user storage quotas will also include files stored on external storage mounts (such as SMB, SFTP, S3, etc.) that are configured for the user (either as personal or global/system mounts).

Only files visible to the user at these mount points are counted towards their quota. Files only visible to other users (on their own mounts) are not counted.

By default, system/global external storage mounts are shared: every user given access sees the same files and folders from the external storage. To have per-user isolation, configure the mount with user-specific path or credentials, or utilize a personal mount.

Enabling this option may impact performance if external storages are slow or unreliable.

Warning: This setting is considered EXPERIMENTAL and may not work with all external storage backends.

Defaults to `false`.

external_storage.auth_availability_delay

```
'external_storage.auth_availability_delay' => 1800,
```

When an external storage is unavailable (e.g., due to failed authentication), it is flagged as such for a specified duration. For authentication failures, this delay can be customized to reduce the likelihood of account lockouts in systems like Active Directory.

Defaults to `1800` seconds (30 minutes)

files_external_allow_create_new_local

```
'files_external_allow_create_new_local' => true,
```

Allow creation of external storages of type "Local" via the web interface and APIs. When disabled, local storages can still be created using the occ command:

```
occ files_external:create /mountpoint local null::null -c datadir=/path/to/data
```

Defaults to `true`

filesystem_check_changes

```
'filesystem_check_changes' => 0,
```

Specify how often the local filesystem (Nextcloud data/ directory and NFS mounts in data/) is checked for changes made outside Nextcloud. This does not apply to external storage.

- `0` -> Never check the filesystem for outside changes, improving performance when no external changes are expected.
- `1` -> Check each file or folder at most once per request, recommended for general use if outside changes are possible.

Defaults to `0`

part_file_in_storage

```
'part_file_in_storage' => true,
```

Control where temporary ".part" files are written during direct (non-chunked) uploads.

While an upload is in progress, Nextcloud writes data to a temporary ".part" file and renames it to the final filename when the upload completes.

- true: create the temporary ".part" file in the destination storage/path. This typically avoids cross-storage moves and can improve reliability and performance on backends where rename within the same storage is cheap/atomic.
- false: create the temporary ".part" file in the user's root folder first. This may help with some external storages that have limited rename/move behavior, but can add extra copy/move overhead.

Note: This setting applies to direct (non-chunked) uploads only. Chunked/ resumable uploads use a separate uploads staging mechanism and are not controlled by this option.

Defaults to `true`.

filesystem_cache_readonly

```
'filesystem_cache_readonly' => false,
```

Read-only mode for scan/detection reconciliation writes to filecache.

When true, Nextcloud does not store filecache metadata changes that are identified through scanner/change-detection reconciliation paths (global: all storages).

Scope note:

- Nextcloud-originated operations (UI/WebDAV/clients) are generally handled through normal application write paths and thus will still update filecache even when this is set to true.
- Reconciliation/refresh paths are prevented from writing back discovered metadata deltas while this is enabled.

Practical effect:

- Changes made directly on storage outside Nextcloud are generally not reflected while enabled.
- Some metadata-dependent behavior can appear stale until this parameter is disabled (permitting reconciliation writes again).

Warning: This is an expert/global setting for specialized environments and is intentionally not default-safe for general deployments.

Defaults to `false`.

trusted_proxies

```
'trusted_proxies' => ['203.0.113.45', '198.51.100.128', '192.168.2.0/24'],
```

List of trusted proxy servers. Supported formats:

- IPv4 addresses, e.g., `192.168.2.123`
- IPv4 ranges in CIDR notation, e.g., `192.168.2.0/24`
- IPv6 addresses, e.g., `fd9e:21a7:a92c:2323::1`
- IPv6 ranges in CIDR notation, e.g., `2001:db8:85a3:8d3:1319:8a20::/95`

If a request's `REMOTE_ADDR` matches an address here, it is treated as a proxy, and the client IP is read from the HTTP header specified in `forwarded_for_headers` instead of `REMOTE_ADDR`.

Ensure `forwarded_for_headers` is configured if `trusted_proxies` is set.

Defaults to `[]` (empty array)

forwarded_for_headers

```
'forwarded_for_headers' => ['HTTP_X_FORWARDED', 'HTTP_FORWARDED_FOR'],
```

Headers trusted as containing the client IP address when used with `trusted_proxies`. For example, use `HTTP_X_FORWARDED_FOR` for the `X-Forwarded-For` header.

Incorrect configuration allows clients to spoof their IP address, bypassing access controls and rendering logs unreliable.

Defaults to `['HTTP_X_FORWARDED_FOR']`

allowed_admin_ranges

```
'allowed_admin_ranges' => ['192.0.2.42/32', '233.252.0.0/24', '2001:db8::13:37/64'],
```

List of trusted IP ranges for admin actions. If non-empty, all admin actions must originate from IPs within these ranges.

Supported formats: - IPv4 addresses or ranges, e.g., `192.0.2.42/32`, `233.252.0.0/24` - IPv6 addresses or ranges, e.g., `2001:db8::13:37/64`

Defaults to `[]` (empty array)

allowed_no_password_confirmation_ranges

```
'allowed_no_password_confirmation_ranges' => ['192.0.2.42/32', '233.252.0.0/24',
↪'2001:db8::13:37/64'],
```

List of trusted IP ranges that can bypass password confirmation.

If non-empty, all endpoints marked with the PasswordConfirmationRequired attribute won't need a password confirmation when originating from IPs within these ranges.

Supported formats: - IPv4 addresses or ranges, e.g., `192.0.2.42/32`, `233.252.0.0/24` - IPv6 addresses or ranges, e.g., `2001:db8::13:37/64`

Defaults to `[]` (empty array)

max_filesize_animated_gifs_public_sharing

```
'max_filesize_animated_gifs_public_sharing' => 10,
```

Maximum file size (in megabytes) for animating GIFs on public sharing pages.

If a GIF exceeds this size, a static preview is shown.

Set to `-1` for no limit.

Defaults to `10` megabytes

filelocking.ttl

```
'filelocking.ttl' => 60 * 60,
```

Set the lock's time-to-live (TTL) in seconds. Locks older than this are automatically cleaned up.

Defaults to `3600` seconds (1 hour) or the PHP `max_execution_time`, whichever is higher.

memcache.locking

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

Memory caching backend for file locking. Redis is highly recommended to avoid data loss, as many memcache backends may evict values unexpectedly.

Defaults to `none`

filelocking.debug

```
'filelocking.debug' => false,
```

Enable debug logging for file locking. This can generate a large volume of log entries, potentially causing performance degradation and large log files on busy instances.

Use with `log.condition` to limit logging in production environments.

Defaults to `false`

upgrade.disable-web

```
'upgrade.disable-web' => false,
```

Disable the web-based updater.

Defaults to `false`

upgrade.cli-upgrade-link

```
'upgrade.cli-upgrade-link' => '',
```

Customize the CLI upgrade documentation link.

user_ini_additional_lines

```
'user_ini_additional_lines' => '',
```

Additional line(s) (string or array of strings) that will be added to .user.ini on each update by the updater.

Defaults to `''` (empty string)

documentation_url.server_logs

```
'documentation_url.server_logs' => '',
```

Customize the server logs documentation link for exception handling.

debug

```
'debug' => false,
```

Enable debugging mode for Nextcloud. Only use for local development, not in production, as it disables minification and outputs additional debug information.

Defaults to `false`

data-fingerprint

```
'data-fingerprint' => '',
```

Set the data fingerprint for the current data served. Used by clients to detect if a backup has been restored. Update this by running:

```
occ maintenance:data-fingerprint
```

Changing or deleting this value may cause connected clients to stall until conflicts are resolved.

Defaults to `''` (empty string)

configfilemode

```
'configfilemode' => 0640,
```

config.php file mode in octal notation.

Defaults to `0640` (writable by user, readable by group).

copied_sample_config

```
'copied_sample_config' => true,
```

This entry serves as a warning if the sample configuration was copied.

DO NOT ADD THIS TO YOUR CONFIGURATION!

Ensure all settings are modified only after consulting the documentation.

lookup_server

```
'lookup_server' => 'https://lookup.nextcloud.com',
```

Use a custom lookup server to publish user data.

Defaults to `https://lookup.nextcloud.com`

gs.enabled

```
'gs.enabled' => false,
```

Enable Nextcloud's Global Scale architecture.

Defaults to `false`

gs.federation

```
'gs.federation' => 'internal',
```

Configure federation for Global Scale setups. Set to `global` to allow federation outside the environment.

Defaults to `internal`

csrf.optout

```
'csrf.optout' => [
        '/^WebDAVFS/', // OS X Finder
        '/^Microsoft-WebDAV-MiniRedir/', // Windows WebDAV drive
    ],
```

List of user agents exempt from SameSite cookie protection due to non-standard HTTP behavior.

Warning

Use only if you understand the implications.

Defaults to:

- `/^WebDAVFS/` (OS X Finder)
- `/^Microsoft-WebDAV-MiniRedir/` (Windows WebDAV drive)

core.login_flow_v2.allowed_user_agents

```
'core.login_flow_v2.allowed_user_agents' => [],
```

Specify allowed user agents for Login Flow V2 using regular expressions.

User agents not matching this list are denied access to Login Flow V2.

Warning

Use only if you understand the implications.

Example: Allow only the Nextcloud Android app:

```
'core.login_flow_v2.allowed_user_agents' => ['/Nextcloud-android/i'],
```

Defaults to `[]` (empty array)

simpleSignUpLink.shown

```
'simpleSignUpLink.shown' => true,
```

Show or hide the "simple sign up" link on public pages.

See: https://nextcloud.com/signup/

Defaults to `true`

login_form_autocomplete

```
'login_form_autocomplete' => true,
```

Enable or disable autocompletion for the login form. Disabling this prevents browsers from remembering login credentials, which may be required for compliance with certain security policies.

Defaults to `true`

login_form_timeout

```
'login_form_timeout' => 300,
```

Set a timeout (in seconds) for the login form. After this period, the form is reset to prevent password leaks on public devices if the user forgets to clear it.

A value of 0 disables the timeout.

Defaults to `300` seconds (5 minutes)

no_unsupported_browser_warning

```
'no_unsupported_browser_warning' => false,
```

Suppress warnings for outdated or unsupported browsers. When enabled, users can bypass the warning after reading it.

Set to `true` to disable the warning.

Defaults to `false`

files_no_background_scan

```
'files_no_background_scan' => false,
```

Disable background scanning of files. When enabled, a background job runs every 10 minutes to sync the filesystem and database for up to 500 users with unscanned files (size < 0 in filecache).

Defaults to `false`

query_log_file

```
'query_log_file' => '',
```

Log all database queries to a file.

Warning

This significantly reduces server performance and is intended only for debugging or profiling query interactions. Sensitive data may be logged in plain text.

query_log_file_requestid

```
'query_log_file_requestid' => '',
```

Prefix all queries with the request ID when set to *yes*.

Requires `query_log_file` to be set.

query_log_file_parameters

```
'query_log_file_parameters' => '',
```

Include all query parameters in the query log when set to *yes*.

Requires `query_log_file` to be set.

Warning

This may log sensitive data in plain text.

query_log_file_backtrace

```
'query_log_file_backtrace' => '',
```

Include a backtrace in the query log when set to *yes*.

Requires `query_log_file` to be set.

redis_log_file

```
'redis_log_file' => '',
```

Log all Redis requests to a file.

Warning

This significantly reduces server performance and is intended only for debugging or profiling Redis interactions. Sensitive data may be logged in plain text.

diagnostics.logging

```
'diagnostics.logging' => true,
```

Enable diagnostics event logging. Logs timings of common execution steps at debug level. Use with `log.condition` to enable conditionally in production.

Defaults to `true`

diagnostics.logging.threshold

```
'diagnostics.logging.threshold' => 0,
```

Limit diagnostics event logging to events longer than the specified threshold (in milliseconds). A value of 0 disables diagnostics event logging.

profile.enabled

```
'profile.enabled' => true,
```

Toggle availability of user profiles.

User profile pages contain information that can be shared with other users, such as full name, phone number, organization, role, and similar fields.

Profiles are enabled by default, and profile data may be used by other features (for example, the system address book).

Profile visibility is layered: what is shared depends on a combination of system-wide and account-level privacy controls, and each field's visibility can be configured.

When set to false, profile functionality is disabled instance-wide.

Note: This affects user account profiles, not the developer performance profiler.

Defaults to *true*

account_manager.default_property_scope

```
'account_manager.default_property_scope' => [],
```

Override default scopes for account data. Valid properties and scopes are defined in `OCP\Accounts\IAccountManager`. Values are merged with defaults from `OC\Accounts\AccountManager`.

Example: Set phone property to private scope: `[\OCP\Accounts\IAccountManager::PROPERTY_PHONE => \OCP\Accounts\IAccountManager::SCOPE_PRIVATE]`

projects.enabled

```
'projects.enabled' => false,
```

Enable the deprecated Projects feature, superseded by Related Resources since Nextcloud 25.

Defaults to `false`

bulkupload.enabled

```
'bulkupload.enabled' => true,
```

Enable the bulk upload feature.

Defaults to `true`

reference_opengraph

```
'reference_opengraph' => true,
```

Enable fetching Open Graph metadata from remote URLs.

Defaults to `true`

unified_search.enabled

```
'unified_search.enabled' => false,
```

Enable the legacy unified search.

Defaults to `false`

enable_non-accessible_features

```
'enable_non-accessible_features' => true,
```

Enable features that do not yet comply with accessibility standards.

Defaults to `true`

binary_search_paths

```
'binary_search_paths' => [
        '/usr/local/sbin',
        '/usr/local/bin',
        '/usr/sbin',
        '/usr/bin',
        '/sbin',
        '/bin',
        '/opt/bin',
    ],
```

Directories where Nextcloud searches for external binaries (e.g., LibreOffice, sendmail, ffmpeg).

Defaults to: - /usr/local/sbin - /usr/local/bin - /usr/sbin - /usr/bin - /sbin - /bin - /opt/bin

files.chunked_upload.max_size

```
'files.chunked_upload.max_size' => 100 * 1024 * 1024,
```

Maximum chunk size for chunked uploads (in bytes). Larger chunks increase throughput but yield diminishing returns above 100 MiB. Services like Cloudflare may limit to 100 MiB.

Defaults to `100 * 1024 * 1024` (100 MiB)

files.chunked_upload.max_parallel_count

```
'files.chunked_upload.max_parallel_count' => 5,
```

Maximum number of chunks uploaded in parallel during chunked uploads. Higher counts increase throughput but consume more server resources, with diminishing returns. Value must be a positive integer.

Defaults to `5`

files.trash.delete

```
'files.trash.delete' => true,
```

Allow users to manually delete files from their trashbin. Automated deletions (e.g., due to low quota) are unaffected.

Defaults to `true`

enable_lazy_objects

```
'enable_lazy_objects' => true,
```

Enable PHP 8.4 lazy objects for Dependency Injection to improve performance by avoiding instantiation of unused objects.

Defaults to `true`

default_certificates_bundle_path

```
'default_certificates_bundle_path' => \OC::$SERVERROOT . '/resources/config/ca-bundle.
↪crt',
```

Change the default certificates bundle used for trusting certificates.

Nextcloud ships its own up-to-date certificates bundle, but in certain cases admins may wish to specify a different bundle, for example the one shipped by their distro.

Defaults to *\OC::$SERVERROOT . '/resources/config/ca-bundle.crt'*.

openmetrics_skipped_classes

```
'openmetrics_skipped_classes' => [
        'OC\OpenMetrics\Exporters\FilesByType',
        'OCA\Files_Sharing\OpenMetrics\SharesCount',
    ],
```

OpenMetrics skipped exporters Allows to skip some exporters in the OpenMetrics endpoint `/metrics`.

Default to `[]` (empty array)

openmetrics_allowed_clients

```
'openmetrics_allowed_clients' => [
        '192.168.0.0/16',
        'fe80::/10',
        '10.0.0.1',
    ],
```

OpenMetrics allowed client IP addresses Restricts the IP addresses able to make requests on the `/metrics` endpoint.

Keep this list as restrictive as possible as metrics can consume a lot of resources.

Default to `[127.0.0.0/16', '::1/128]` (allow loopback interface only)

preview_expiration_days

```
'preview_expiration_days' => 0,
```

Delete previews older than a certain number of days to reduce storage usage.

Less than one day is not allowed, so set it to 0 to disable the deletion.

Defaults to `0`.

background_jobs_expiration_days

```
'background_jobs_expiration_days' => 60,
```

Delete job runs older than a certain number of days.

Less than one day is not allowed.

Defaults to `60`.

8.3 Activity app

The Activity app records and summarizes user-visible events across your Nextcloud instance and can notify users via the activity stream, email, and push notifications. It is shipped and enabled by default.

> **Note**
>
> The Activity app is designed for user notifications, not for compliance or auditing. Users can enable or disable activity tracking for their own account, so the activity stream is not a reliable audit trail. If you need a complete log of all actions on your instance, use the **admin_audit** app instead — see *Logging*.

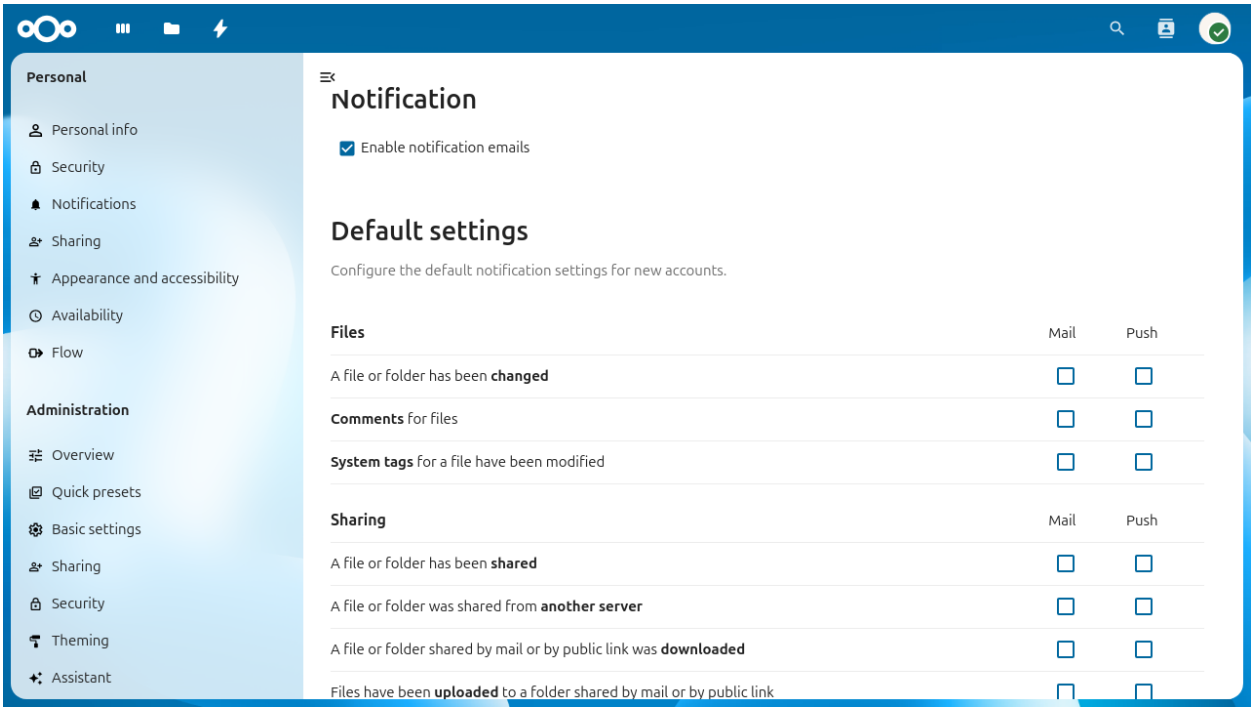


Fig. 1: *The Activity admin settings page showing default notification preferences for new accounts.*

8.3.1 Configuring your Nextcloud for the Activity app

Enabling email notifications

To send activity notification emails, a working *Email* is required.

It is also recommended to configure the background job run mode to a system run mode (`System Cron` or `systemd`) as described in *Background jobs*. The `Ajax` and `Webcron` modes may delay or skip email delivery.

Admin settings

As an administrator, you can:

- Enable or disable notification emails globally using the **Enable notification emails** checkbox in **Settings > Administration > Activity**.
- Configure the default notification preferences for new accounts. Existing users keep their personal settings. The defaults apply to accounts created after the change.

> **Note**
>
> The **Enable notification emails** toggle prevents **new** emails from being queued. It does not retroactively remove emails that are already waiting in the queue. Emails queued before the toggle was disabled will still be sent by the background job on its next run. To stop those emails immediately, truncate or clear the `oc_activity_mq` database table.

- Configure the maximum number of activities fully displayed in notification emails using the `mail_max_items` setting. Activities beyond this limit are summarised as *"and X more"* at the bottom of the email. The default is `200` and the maximum allowed value is `1000`. For example, to set it to `500`:

```
occ config:app:set activity mail_max_items --value=500
```

8.3.2 Configuration reference

The following `config.php` options control Activity app behavior.

Option	Default	Description
`activity_expire_days`	`365`	Number of days to retain activity records. A daily background job deletes all activities older than this value. Minimum is `1` day.
`activity_use_cached_mountpoint`	`false`	When `true`, activities in team folders and external storages are generated for all users with access, not just the acting user. See *Activities in Team Folders or External Storages* for details and caveats.
`activity_expire_exclude_users`	`[]`	An array of user IDs whose activity records are never deleted by the expiration job. See *Excluding users from activity expiration*.

8.3.3 Activities in Team Folders or External Storages

By default, activities in team folders or external storages are only generated for the current user. This is due to the underlying logic of these storage backends. The config flag `activity_use_cached_mountpoints` makes activities work like in normal shares when set to `true`.

```
'activity_use_cached_mountpoints' => true,
```

Danger

If "Advanced Permissions" (ACLs) are enabled in a team folder, activities do not respect those permissions. As a result, users may see activity entries for files and directories they cannot access. **This can leak sensitive information.** See this issue for more information.

Warning

Users who previously had access to a team folder, share, or external storage can continue to see new activity entries in their stream and emails until they log in again after access is removed.

Note

Users who are newly added to a team folder, share, or external storage will not see new activity entries in their stream or emails until they log in again after access is granted.

8.3.4 Excluding users from activity expiration

For certain users, it might make sense to never expire their activity data, for example administrators. Set the config value `activity_expire_exclude_users` in your `config.php`:

```
'activity_expire_exclude_users' => [
  'admin',
  'group_admin',
  'second_admin'
]
```

For these users, their activity records will never be deleted from the database.

8.3.5 Better scheduling of activity emails

In certain scenarios it makes sense to send the activity emails more regularly. For example, you may want to send the hourly emails always at the full hour, daily emails before people start to work in the morning, and weekly emails on Monday morning.

A console command is available to trigger sending those emails. This allows you to set up custom cron jobs with specific timing instead of relying on the Nextcloud background job:

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

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

8.3.6 Troubleshooting

Users are not receiving notification emails

1. Verify that email is configured correctly. Send a test email from **Settings > Administration > Basic settings**.
2. Check that **Enable notification emails** is turned on in **Settings > Administration > Activity**.
3. Verify that the user has email notifications enabled in their personal notification settings (**Settings > Personal > Notifications**).
4. Ensure the background job is running. If you use `Cron`, check that the system cron job is executing `php -f /var/www/nextcloud/cron.php` regularly.
5. Check the Nextcloud log (`data/nextcloud.log`) for mail-related errors.

Activities are missing for shared files or team folders

By default, activities in team folders and external storages are only generated for the acting user. See *Activities in Team Folders or External Storages* for how to enable activities for all users with access.

Database growing large due to activity data

The Activity app stores all events in the database. On instances with many users or high file turnover, this can lead to significant database growth. To manage this:

- Set `activity_expire_days` to a lower value (e.g. `90` or `180`) to automatically clean up older records.
- Ensure the Nextcloud background job is running so that the daily expiration job executes.
- To assess current database usage, check the size of the `oc_activity` and `oc_activity_mq` tables directly.

Emails are still being sent after disabling notification emails

The **Enable notification emails** toggle is checked when *queuing* new emails. The background job that actually delivers emails (`MailQueueHandler`) does not re-read the toggle before sending, so emails that were already in the queue when the toggle was turned off will still be delivered.

To stop those emails immediately, clear the `oc_activity_mq` table:

```
DELETE FROM oc_activity_mq;
```

After clearing the table, no further queued activity emails will be sent until new ones are queued (which requires the toggle to be re-enabled).

8.4 Administration privileges (Delegation)

8.4.1 Introduction

Nextcloud has built-in functionality which permits administrators to delegate authority to others without granting them full administration privileges (and without making them a member of the `admin` group).

This administration privilege delegation functionality is supported by many shipped and ecosystem apps that have their own settings areas under *Administration settings*.

Note

If you're an app developer and would like administrators to be able to utilize this functionality for your app, you need to enable support for delegation of your settings (see the Developer Manual for specifics).

Tip

Delegation of user management is possible, but you can also use *Group Administrators*.

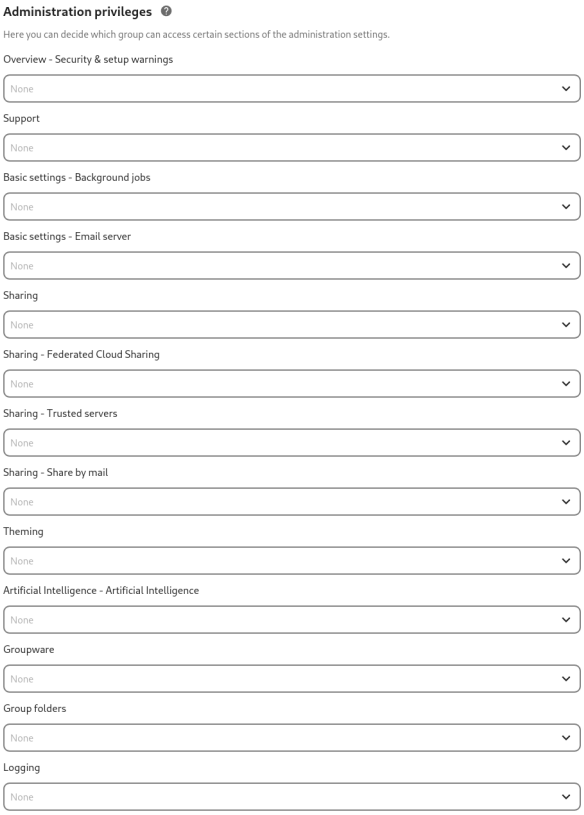
Warning

Delegation of user management allow the delegated users to add themselves to groups receiving delegation of other settings. This can be used to escalate privileges.

8.4.2 Usage

By default only members of the `admin` group can access *Administration settings*. You can create additional user groups (or use existing ones) and then grant these groups access to specific settings.

While logged in to an account that is a member of the `admin` group, go to *Administration settings* -> *Administration privilege*. You will be presented with the list of settings pages and sections, including for any installed apps, that support delegation.



By clicking on the combo box, you will be able to choose which groups are able to access the selected settings. You can revoke access at any time by removing the group from the selection (or, if you wish only to revoke access for an individual account, by removing that account from the configured group).

Tip

Not every settings page or section supports delegation. This is either because delegating access to that particular settings page would enable privilege escalation (i.e. bypassing of the limited administration authority) or delegation has not yet been implemented for that specific settings page or app.

8.5 Android Deep Link Handling

Deep linking in Android allows your application to be launched directly from a URL, making it easier for users to navigate to specific content within your app. Starting from Android 12, handling deep links requires additional configuration using an `assetlinks.json` file to ensure the app and the host domain are properly associated.

8.5.1 Android 11 and Below

For Android 11 and below, deep linking is straightforward and does not require additional configuration beyond the usual manifest settings.

8.5.2 Android 12 and Above

For Android 12 and above, an additional configuration step is required to verify the relationship between your app and the host domain using the `assetlinks.json` file.

Creating assetlinks.json

Create a file named `assetlinks.json` and host it in the .well-known directory of your website (e.g., https://www.cloud.example.com/.well-known/assetlinks.json).

Example `assetlinks.json`:

```
[
  {
    "relation": ["delegate_permission/common.handle_all_urls"],
    "target": {
      "namespace": "android_app",
      "package_name": "com.cloud.example.nextcloud",
      "sha256_cert_fingerprints": [

↪"FB:00:95:22:F6:5E:25:80:22:61:B6:7B:10:A4:5F:D7:0E:61:00:31:97:6F:40:B2:8A:64:9E:15:2D:ED:03:73
↪"
      ]
    }
  }
]
```

Nextcloud Configuration Limitation

Due to the additional requirement of hosting an `assetlinks.json` file for Android 12 and above, Nextcloud cannot configure the Android client for all different hosts. This is because each host needs its own `assetlinks.json` file to establish a verified relationship with the app, and Nextcloud cannot manage this file for every possible host domain.

8.6 Antivirus scanner

You can configure your Nextcloud server to automatically run a virus scan on newly-uploaded files with the Antivirus app for Files. The Antivirus app for Files integrates the open source anti-virus engine ClamAV with Nextcloud. ClamAV detects all forms of malware including Trojan horses, viruses, and worms, and it operates on all major file types including Windows, Linux, and Mac files, compressed files, executables, image files, Flash, PDF, and many others. ClamAV's Freshclam daemon automatically updates its malware signature database at scheduled intervals.

ClamAV runs on Linux and any Unix-type operating system, and Microsoft Windows. However, it has only been tested with Nextcloud on Linux, so these instructions are for Linux systems. You must first install ClamAV, and then install and configure the Antivirus app for Files on Nextcloud.

8.6.1 Installing ClamAV

As always, the various Linux distributions manage installing and configuring ClamAV in different ways.

Debian, Ubuntu, Linux Mint
On Debian and Ubuntu systems, and their many variants, install ClamAV with these commands:

```
apt-get install clamav clamav-daemon
```

The installer automatically creates default configuration files and launches the `clamd` and `freshclam` daemons. You don't have to do anything more, though it's a good idea to review the ClamAV documentation and your settings in `/etc/clamav/`. Enable verbose logging in both `clamd.conf` and `freshclam.conf` until you get any kinks worked out.

RedHat Enterprise Linux 7, CentOS 7

On RedHat Enterprise Linux 7 and related systems you must install the Extra Packages for Enterprise Linux (EPEL) repository, and then install ClamAV:

```
yum install epel-release
yum install clamav clamav-scanner clamav-scanner-systemd clamav-server
clamav-server-systemd clamav-update
```

This installs two configuration files: `/etc/freshclam.conf` and `/etc/clamd.d/scan.conf`. You must edit both of these before you can run ClamAV. Both files are well-commented, and `man clamd.conf` and `man freshclam.conf` explain all the options. Refer to `/etc/passwd` and `/etc/group` when you need to verify the ClamAV user and group.

First edit `/etc/freshclam.conf` and configure your options. `freshclam` updates your malware database, so you want it to run frequently to get updated malware signatures. Run it manually post-installation to download your first set of malware signatures:

```
freshclam
```

The EPEL packages do not include an init file for `freshclam`, so the quick and easy way to set it up for regular checks is with a cron job. This example runs it every hour at 47 minutes past the hour:

```
# m   h  dom mon dow  command
  47  *  *   *    *  /usr/bin/freshclam --quiet
```

Please avoid any multiples of 10, because those are when the ClamAV servers are hit the hardest for updates.

Next, edit `/etc/clamd.d/scan.conf`. When you're finished you must enable the `clamd` service file and start `clamd`:

```
systemctl enable clamd@scan.service
systemctl start clamd@scan.service
```

That should take care of everything. Enable verbose logging in `scan.conf` and `freshclam.conf` until it is running the way you want.

Docker, Docker-compose

To install ClamAV via docker or docker compose you can take official image of ClamAV, or build one by yourself. This example is based on docker image from https://github.com/Cisco-Talos/clamav.

You can mount ClamAV Socket from the Docker Container to the host System as volume. In this case you do not need to expose any port outside of container.

For a Docker run this command:

```
docker run --name clamav -d -v /var/run/clamav/:/var/run/clamav/ -v /var/docker/
↪clamav/virus_db/:/var/lib/clamav/ clamav/clamav:stable_base
```

For a Docker-compose use following settings:

```
version: "3.6"
services:
  clamav:
    image: "clamav/clamav:stable_base"
```

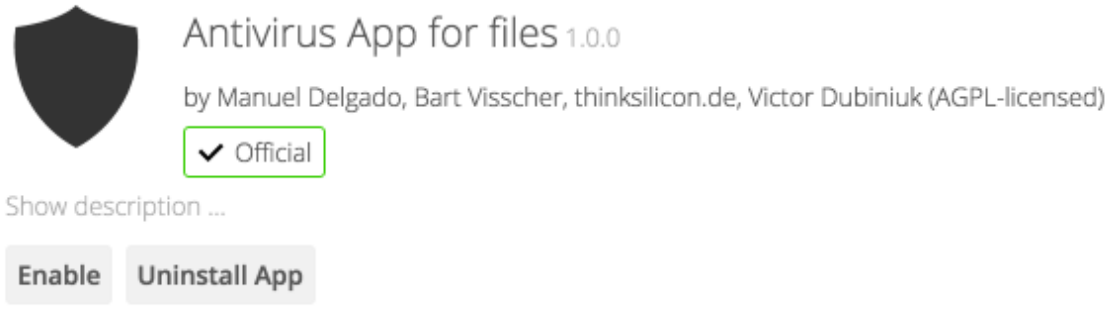
(continues on next page)

(continued from previous page)

```
    container_name: "clamav"
    volumes:
      # Socket
      - /var/run/clamav/:/var/run/clamav/
      # Virus DB
      - /var/docker/clamav/virus_db/:/var/lib/clamav/
    restart: unless-stopped
```

8.6.2 Enabling the antivirus app for files

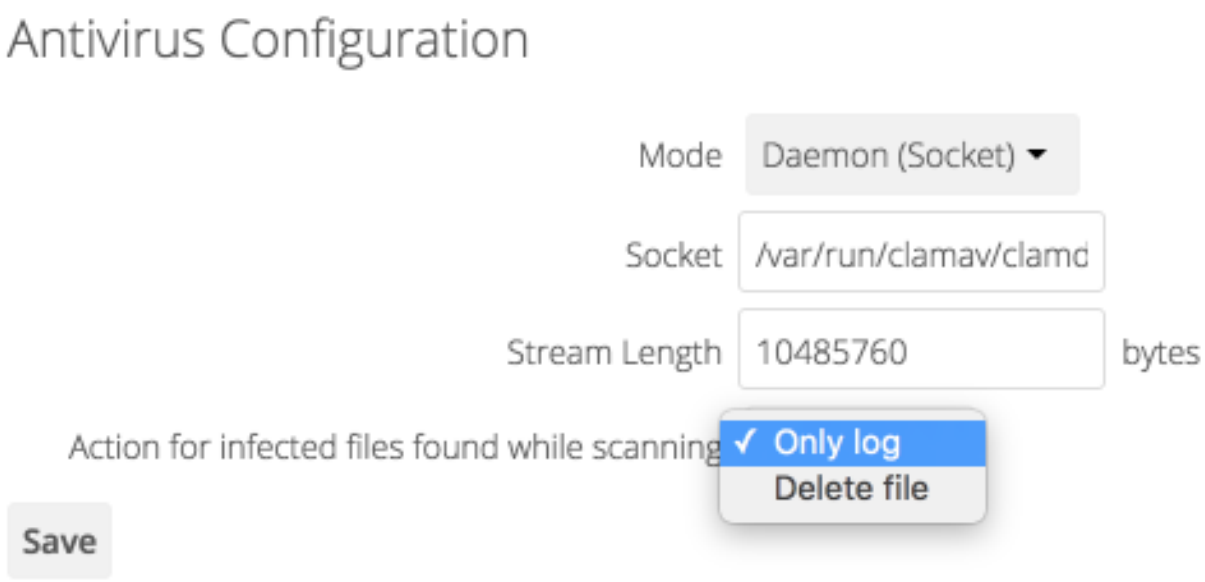
Place the `files_antivirus` app into the `apps` directory of your Nextcloud server. Then the app shows up on the Nextcloud Apps page where it simply can be enabled.



8.6.3 Configuring ClamAV on Nextcloud

Next, go to your Nextcloud Admin page and set your Nextcloud logging level to Everything.

Now find your Antivirus Configuration panel on your Admin page.



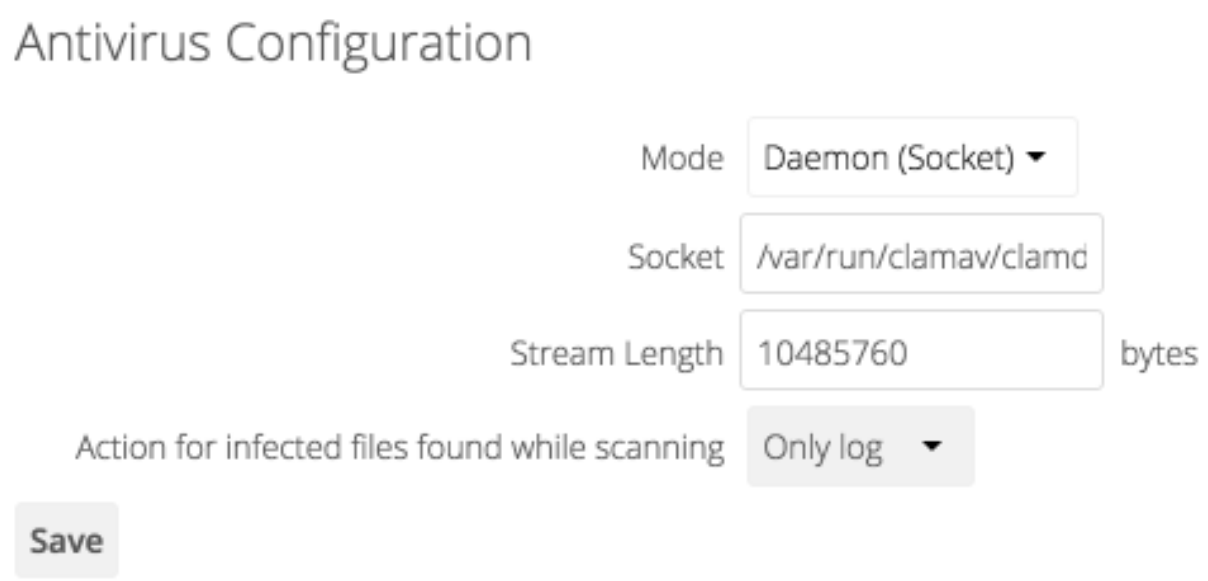
ClamAV runs in one of three modes:

- Daemon (Socket): ClamAV is running on the same server as Nextcloud. The ClamAV daemon, `clamd`, runs in the background. When there is no activity `clamd` places a minimal load on your system. If your users upload large volumes of files you will see high CPU usage.
- Daemon: ClamAV is running on a different server. This is a good option for Nextcloud servers with high volumes of file uploads.
- Executable: ClamAV is running on the same server as Nextcloud, and the `clamscan` command is started and then stopped with each file upload. `clamscan` is slow and not always reliable for on-demand usage; it is better to use one of the daemon modes.

Daemon (Socket)

Nextcloud should detect your `clamd` socket and fill in the `Socket` field. This is the `LocalSocket` option in `clamd.conf`. You can run `netstat` to verify:

```
netstat -a|grep clam
unix 2 [ ACC ] STREAM LISTENING 15857 /var/run/clamav/clamd.ctl
```

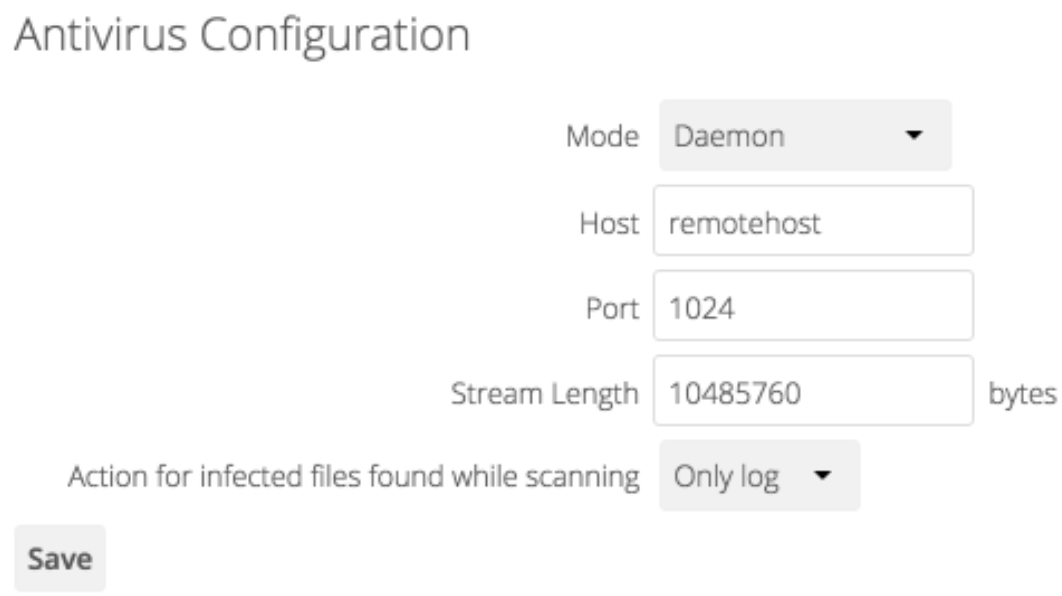


The `Stream Length` value sets the number of bytes read in one pass. 26214400 bytes, or 25 MiB, is the default. This value should be no larger than the PHP `memory_limit` settings, or physical memory if `memory_limit` is set to -1 (no limit).

`Action for infected files found while scanning` gives you the choice of logging any alerts without deleting the files, or immediately deleting infected files.

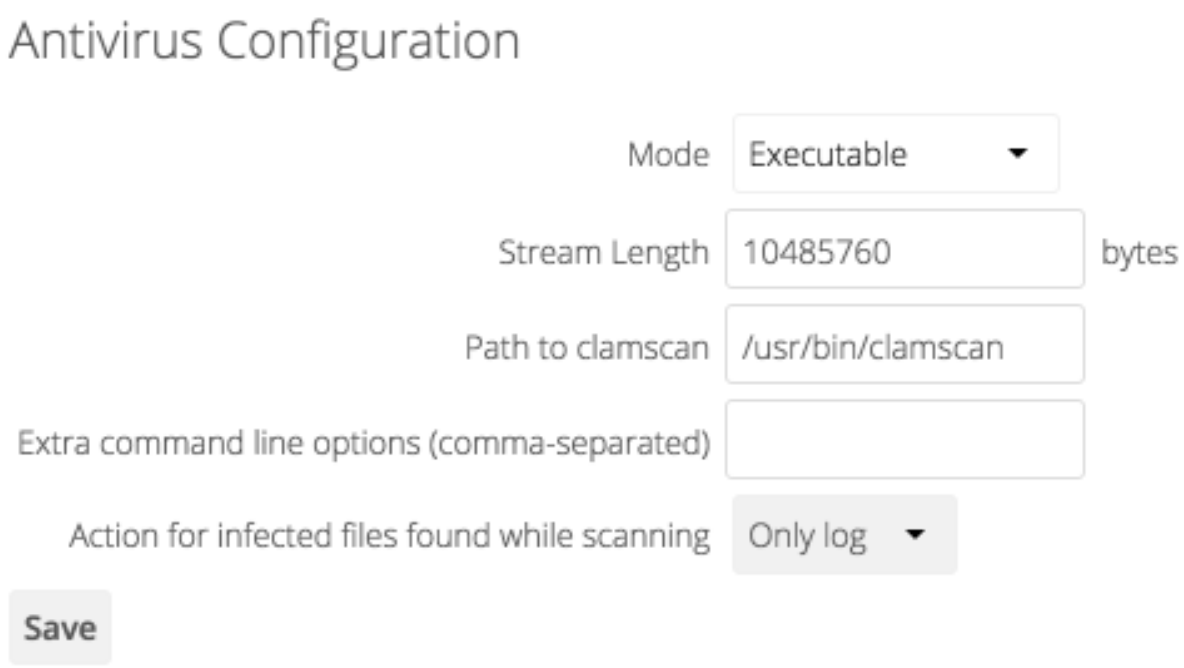
Daemon

For the Daemon option you need the hostname or IP address of the remote server running ClamAV, and the server's port number.



Executable

The Executable option requires the path to `clamscan`, which is the interactive ClamAV scanning command. Nextcloud should find it automatically.



When you are satisfied with how ClamAV is operating, you might want to go back and change all of your logging to less verbose levels.

8.6.4 Confirm everything is working

Every antivirus provider implements a test virus string, that way tests are quite easy. You find the files here: https://www.eicar.org/download-anti-malware-testfile/

Uploading the file will trigger an error:

> "Virus Win.Test.EICAR_HDB-1 is detected in the file. Upload cannot be completed."

8.6.5 Encrypted File Detection Limitations with ClamAV

By default, ClamAV may still return "OK" for password-protected archives and encrypted files. This known ClamAV behavior bypasses "Block unscannable files" option of Antivirus app. You may configure additional alert options in `clamd.conf`, that should catch it:

- `AlertEncryptedArchive` - Alert on encrypted archives with heuristic signature (encrypted .zip, .7zip, .rar).
- `AlertEncryptedDoc` - Alert on encrypted archives with heuristic signature (encrypted .pdf).
- `AlertEncrypted` - Alert on both encrypted archives and documents with heuristic signature.

For reliable detection and blocking of encrypted files, consult available antivirus backends documentation.

8.6.6 Manage the background scanner

The background scanner does not require any manual intervention. However at times you might want to inspect it or perform tasks on it.

How the background scanner works

Each time the background scanner runs, it processes files in three passes (in priority order), up to the configured batch size per run:

1. **Unscanned files** — files that have never been scanned, including files that were uploaded or existed before the Antivirus app was installed. All such files will eventually be scanned as the background job runs repeatedly.
2. **Modified files** — files whose modification time is newer than their last scan time, i.e. files that have been updated since they were last scanned.
3. **Outdated files** — files that were last scanned more than 28 days ago (configurable via `av_rescan_days`), to account for updated virus definitions.

The scanner runs at most once every 15 minutes by default. The effective frequency is also bounded by how often Nextcloud's background job mechanism (cron, webcron, or ajax) runs.

To change the minimum interval between runs (in seconds):

```
sudo -E -u www-data php occ config:app:set files_antivirus av_scan_interval --value=
↪"900"
```

To change the number of days before already-scanned files are rescanned:

```
sudo -E -u www-data php occ config:app:set files_antivirus av_rescan_days --value="28"
```

External storage: files on external storage are included in passes 1 and 2 (initial scan and rescan on modification). However, the periodic 28-day rescan (pass 3) applies only to files under `files/` (i.e. home storage), not external storage.

Get info about files in the scan queue

```
sudo -E -u www-data php occ files_antivirus:status [-v]
```

Manually trigger the background scan

```
sudo -E -u www-data php occ files_antivirus:background-scan [-v] [-m MAX]
```

Manually scan a single file

```
sudo -E -u www-data php occ files_antivirus:scan <path>
```

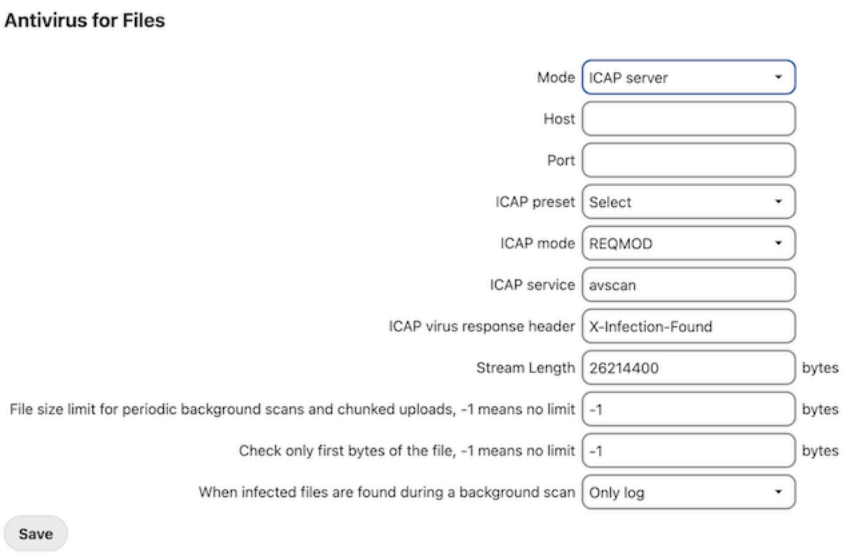
Mark a file as scanned or unscanned

```
sudo -E -u www-data php occ files_antivirus:mark <path> <scanned|unscanned>
```

Files marked as scanned will not be scanned for the next four weeks.

8.6.7 Configuring ICAP on Nextcloud

Nextcloud offers the integration of antivirus protection based on the ICAP protocol. The settings are outlined here. Additional documentation is work in progress.



8.6.8 Disabling background scan task

You can disable background scan with occ to only scan files during upload:

```
sudo -E -u www-data php occ config:app:set files_antivirus av_background_scan --value=
↪"off"
```

Note

Setting `av_rescan_days` to `0` does **not** disable the periodic rescan of already-scanned files. Any value below `1` is silently replaced with the default of 28 days, so the rescan continues on its normal schedule. Use `av_background_scan` as shown above to stop background scanning altogether.

8.7 Background jobs

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

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

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

8.7.1 Parameters

`maintenance_window_start`

Note

This setting is only taken into account in `cron` mode.

In the `config/config.php` file you can specify this config. Some background jobs only run once a day. When an hour is defined (timezone is UTC) for this config, the background jobs which advertise themselves as not time-sensitive will be delayed during the "working" hours and only run in the 4 hours after the given time. This is e.g. used for activity expiration, suspicious login training, and update checks.

A value of 1 e.g. will only run these background jobs between 01:00am UTC and 05:00am UTC:

```
'maintenance_window_start' => 1,
```

If you don't care when these jobs run, you can set the value to `100`, but beware that resource intensive jobs may then run unnecessarily during high usage periods. This may lead to slower performance and a lower quality user experience.

This setting may also be set directly via `occ` just like any other configuration parameter:

```
occ config:system:set maintenance_window_start --type=integer --value=1
```

8.7.2 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

Use case: Single user instance

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

Warning

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

Webcron

Use case: Very small instance (1–5 users depending on the usage)

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

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

Warning

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

Cron

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

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

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

And append this line:

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

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

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

Which returns:

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

Note

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

Note

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

Note

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
ExecCondition=php -f /var/www/nextcloud/occ status -e
```

(continues on next page)

(continued from previous page)

```
ExecStart=/usr/bin/php -f /var/www/nextcloud/cron.php
KillMode=process
```

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

The *ExecCondition* checks that the nextcloud instance is operating normally before running the background job, and skips it if otherwise.

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

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

nextcloudcron.timer should look like this:

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

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

[Install]
WantedBy=timers.target
```

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

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

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

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

Note

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

8.8 Brute force protection

8.8.1 Introduction

Nextcloud has built-in protection against brute force attempts.

The brute force protection feature is meant to protect Nextcloud servers from attempts to guess passwords and tokens in various ways. Besides the obvious “let’s try a big list of commonly used passwords” attack, it also makes it harder to use slightly more sophisticated attacks via the reset password page or trying to find app password tokens. It is used throughout the Nextcloud ecosystem, including by other apps, if they have sensitive entrypoints (and choose to enable support for it).

8.8.2 How it works

Overview

If triggered, brute force protection makes requests - coming from an IP address via a brute force protected entrypoint - slower for up to a 24 hour period. In extreme circumstances it may prevent access outright, for up to 30 minutes, from a problematic IP address.

This protects your system from attackers trying, for example, a lot of different passwords.

The primary filter is IP address-based. This means that any account - even one associated with a given brute force attempt - is not impacted when it is connecting from a different IP address than any brute force attempts. This helps minimize inadvertent denial of service attacks against legitimate connections, while maximizing attack resistance from problematic IP sources.

Nuisance triggers are minimized through reasonable built-in defaults appropriate to each type of action.

The attempts history is automatically managed by a daily cronjob. Individual entries expire after 48 hours (attempts, however, may be still *logged* indefinitely elsewhere through the usual mechanisms within Nextcloud Server and at the discretion of the admin).

Excluding (whitelisting) select IP addresses from brute force protection to prevent false positives is supported, but usually false positives are best handled by fixing the underlying causes (e.g. a misconfigured reverse proxy or misbehaving client).

Tip

If you do notice a problem with the authentication behavior of any the official Nextcloud clients, please report it to the appropriate repository so that it can be looked into.

Keeping brute force protection active and operating properly helps protects your Nextcloud Server from malicious actors while minimizing potential impact on legitimate usage.

Example: The login page

The brute force protection is easiest to see in action on the login page. If you try to log in the first time with an invalid username and/or password you will not notice anything. But if you do this a few times you start to notice that the verification of the login is taking longer each time. This is the brute force protection kicking in.

The maximum delay is 25 seconds, unless maximum number of attempts (currently 10) was reached within the last 30 minutes (in which case a `429 Too Many Requests` will be returned until the maximum attempts within the recent time has dropped below the threshold).

After a successful login (from the same source IP address), any prior invalid login attempts will be cleared and you will no longer be hit by the delay.

Note

Not all actions are necessarily viewed the same. It is possible for some activities to be more (or less) strict than others.

8.8.3 Usage

Activating

Brute force protection is enabled by default on Nextcloud. Its behavior can be adjusted through the `bruteforcesettings` app (shipped with Server and enabled by default), several `occ` commands, and several `config.php` parameters. Its effectiveness is highly dependent on having a properly configured environment, particularly when integrating a reverse proxy with Nextcloud (and associated parameters such as `trusted_proxies`).

The brute force settings app

This app, which shipped and enabled by default, makes it possible (via the Web UI) to view the status of a connection and modify certain parameters of the brute force protection built into Nextcloud Server.

The user interface added by this app is found under *Administration settings -> Security* under the *Brute-force IP whitelist* heading.

Currently an admin can view the status of the IP address they are connecting from as well as specify IPv4 or IPv6 addresses and ranges to exempt from brute force protection.

Additional enhancements may be made in the future, within this app and/or in combination with Nextcloud Server for additional monitoring or behavior adjustments related to brute force protection.

Warning

Disabling the `bruteforcesettings` app does **not** disable brute force protection. The protection is built into Nextcloud Server core and is always active. Disabling the app only removes the ability to manage brute force settings from the Web interface.

Danger

You would need to adjust the parameter `auth.bruteforce.protection.enabled` in your Nextcloud `config.php` to disable brute force protection, which is **heavily discouraged for production servers**, particularly if your server is reachable via a public IP address. It allows an attacker to iterate over all users and their passwords as well as two-factor verifications afterwards ultimately leading to admin access.

`occ` commands

There are several brute force related `occ` commands under `occ security`.

Brute force protection and load balancers/reverse proxies

If you are behind a reverse proxy or load balancer it is important you make sure it is setup properly. Especially the **trusted_proxies** and **forwarded_for_headers** *config.php* variables need to be set correctly. Otherwise it can happen that Nextcloud actually starts throttling all traffic coming from the reverse proxy or load balancer. For more information see *Reverse proxy*.

8.8.4 Troubleshooting

Overview

On most setups Nextcloud will work out of the box without any issues. If you run into a situation where logging in or connecting is often very slow for multiple users, the first step is to check your Nextcloud Server logs to see what IP addresses are being detected (you will need to adjust your `loglevel` to `1` temporarily to do so).

Look for entries that start with any of the following:

- *Bruteforce attempt from* [...]
- *IP address throttled* [...]
- *IP address blocked* [...]

If all clients appear to be coming from the same IP address and that IP address happens to be your proxy, you need to review your `trusted_proxies` configuration.

If the IP address is a common connection point, such as a multi-user office location, it can be an option to whitelist it, with the draw back that users have to be trust-worthy.

For testing purposes you want want to whitelist your own IP address to see if the problem disappears. If it does - and assuming your proxy configuration is correct - you may have a client/device in your network that is misbehaving and generating invalid login attempts from your IP address.

You can use the *occ security:bruteforce:attempts* command to check the realtime status for a given IP address.

Note

The *bruteforce_attempts* database table will be empty if you're using a distributed memory cache since the database backend is no longer used unless it is the only option available.

Excluding IP addresses from brute force protection

Note

Most nuisance triggering of brute force protection can be resolved through proper configuration of reverse proxies. In other cases, select IP addresses that need to be whitelisted can be configured within this app (while leaving brute force protection enabled). This can be useful for testing purposes or when there are a lot of people (or devices) connecting from a known, single IP address.

It's possible to exclude IP addresses from the brute force protection.

- Make sure the `bruteforcesettings` app is enabled (it is by default)
- Login as admin and go to **Administration settings -> Security**

Danger

Any excluded IP address can perform authentication attempts without any throttling. It's best to exclude as few IP addresses as you can, or even none at all.

8.8.5 Brute force protection vs fail2ban

Nextcloud's built-in brute force protection and fail2ban are complementary tools that operate at different layers of the stack. Using both together is recommended for production servers.

Nextcloud brute force protection is built into Nextcloud Server itself (the `bruteforcesettings` app provides the admin UI and exclusion settings, but the protection runs regardless). It works at the **application layer**: it detects suspicious login patterns and adds progressively longer delays to requests from the offending IP address. It has full context about Nextcloud-specific endpoints and credentials, and it activates automatically without any operating system configuration.

fail2ban works at the **OS/network layer**. It watches log files for failed login entries and instructs the system firewall (e.g. `iptables` or `nftables`) to block the offending IP outright. Blocked requests are dropped before they reach the web server, PHP, or the database, saving server resources entirely.

The two approaches are not mutually exclusive:

- Nextcloud brute force protection handles application-level throttling transparently, including for API clients and mobile apps, with no system configuration required.
- fail2ban reduces server load by blocking repeat offenders at the network level before their requests consume any application resources.

For setup instructions for fail2ban with Nextcloud, see *Setup fail2ban*.

8.9 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 cache for your web server, commonly called "memcache".

> **Note**
>
> 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. You may safely ignore the warning if you prefer.** If you enable only a distributed cache (`memcache.distributed`) in your `config.php` 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. Nextcloud supports multiple memory caching backends, so you can choose the type of memcache that best fits your needs. The supported caching backends are:

- APCu (APCu 4.0.6 and up required).

  A local cache for systems.

- Redis (4.0.0 and up required); Valkey and KeyDB are expected to work as Redis-compatible backends.

  > **Note**
  >
  > Automated/formal testing currently only occurs against Redis Open Source using the *redis* PHP extension and against Valkey using the Nextcloud *keyvalue* cache provider. KeyDB is expected to work as compatible backends as well, but are not formally tested.

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

- Memcached

  For distributed caching.

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

8.9.1 Recommendations based on type of deployment

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

> **Note**
>
> See specific cache configuration options under the appropriate section further down.

Small/Private home server

Only use APCu:

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

Organizations with single-server

Use Redis for everything except local memcache:

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

Organizations with clustered setups

Use APCu for local cache and either Redis cluster …:

```
'memcache.local' => '\OC\Memcache\APCu',
'memcache.distributed' => '\OC\Memcache\Redis',
'memcache.locking' => '\OC\Memcache\Redis',
'redis.cluster' => [
    'seeds' => [ // provide some/all of the cluster servers to bootstrap discovery,␣
↪port required
       'cache-cluster:7000',
       'cache-cluster:7001',
    ],
 ],
```

… or Memcached cluster …:

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

… for distributed and locking caches.

Note

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

Additional notes for Redis vs. APCu on memory caching

APCu is faster for 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.

8.9.2 APCu

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

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

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

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

Depending on your installation size and the number of users and interactions with the system you may want to adapt the `apc.shm_size` setting in your `php.ini`. The default value is 32M which is usually too low for Nextcloud. A good starting point is 128M. If you have a lot of users and/or a lot of apps installed you may want to increase this value further. Keep in mind that this memory needs to be available in your system's memory and kept in mind when sizing the amount of workers on your server.

A frequently resetting cache can lead to unexpected slow downs when the cache is being cleared and refilled.

There is an admin check trying to detect too low memory sizing, but make sure to monitor the APCu cache status to see if the cache is full and if you need to increase the size. APCu provides a script that can help with this, otherwise the serverinfo app in Nextcloud can also show the APCu cache status.

8.9.3 Redis / Valkey using the Nextcloud `KeyValueCache`

> **Note**
>
> The KeyValueCache was added with Nextcloud 34.0.2.

The `KeyValueCache` cache is a brand-independent cache backend for key-value stores like Valkey or Redis. These are excellent modern memcaches to use for distributed caching, and as a key-value store for *Transactional File Locking* because they guarantee that cached objects are available for as long as they are needed.

Unlike the `phpredis` backend described in the next section, the `KeyValueCache` does not require a PHP extension: it is built on the predis library which is bundled with Nextcloud. You only need to install and run a Valkey or Redis compatible server and configure it in `config.php`.

> **Note**
>
> When the cache server is not located on the same machine the performance may be impacted due to network overhead compared to the `php-redis` based backend. In those cases it may be preferable to use the `php-redis` backend instead.

On Debian/Ubuntu/Mint, install `valkey-server` or `redis-server`. On CentOS and Fedora, install `valkey` or `redis`. Depending on your distribution the server might not start automatically, so you may need to use your service manager to start the server, and to launch it at boot as a daemon.

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

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

Add the appropriate entries to your `config.php`, and refresh your Nextcloud admin page.

KeyValueCache configuration in Nextcloud (config.php)

For best performance, use the `KeyValueCache` for file locking by adding this:

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

Additionally, you should use it for the distributed server cache:

```
'memcache.distributed' => '\OC\Memcache\KeyValueCache',
```

Furthermore, you could use it for the local cache like so, but it's not recommended (see warning below):

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

Warning

Using a key-value store server for local cache on a multi-server setup can cause issues. Also, even on a single-server setup, APCu (see section above) should be faster.

When using the `keyvalue` cache for any of the above cache settings, you also need to specify the `memcache.kvstore` configuration in `config.php`. Three topologies are supported: a single server (`server`), a Sentinel managed replication set (`sentinel`), and a server cluster (`seeds`). Configure exactly one of them.

The following options are available when using a single server (all but `server` are optional):

```
'memcache.locking' => '\OC\Memcache\KeyValueCache',
'memcache.distributed' => '\OC\Memcache\KeyValueCache',
'memcache_customprefix' => 'mynextcloudprefix',
'memcache.kvstore' => [
   'server' => [
      'host' => 'cache-host.example.com',
      'port' => 6379,
      // Protocol used to connect. One of 'tcp', 'tls' or 'unix'.
      // When omitted it is derived from the host (a leading '/' means 'unix').
      'protocol' => 'tcp',
   ],
   'user'         => 'nextcloud', // only sent when the server uses ACLs
   'password'     => 'password',
   'dbindex'      => 0,
   'timeout'      => 1.5,
   'read_timeout' => 1.5,
   'persistent'   => false, // keep the connection open across requests
],
```

The same optional parameters are available for the Sentinel and cluster topologies below and are applied to every node.

For a Sentinel managed replication setup, provide the name of the monitored service and one seed entry per Sentinel node (each seed uses the same format as the `server` entry above):

```
'memcache.kvstore' => [
   'sentinel' => [
      'service' => 'mymaster',
      'seeds' => [
         ['host' => 'sentinel1.example.com', 'port' => 26379],
         ['host' => 'sentinel2.example.com', 'port' => 26379],
         ['host' => 'sentinel3.example.com', 'port' => 26379],
      ],
   ],
],
```

For a cluster, provide some or all of the cluster nodes to bootstrap discovery:

```
'memcache.kvstore' => [
   'seeds' => [
      ['host' => 'cache-cluster.example.com', 'port' => 7000],
      ['host' => 'cache-cluster.example.com', 'port' => 7001],
   ],
],
```

Note

Selecting a numbered database with `dbindex` is only supported for single servers and Sentinel setups; clusters always use database 0, the support for numbered databases added with Valkey v9 is pending upstream.

Connecting over TLS

To connect via TCP over TLS, set the `protocol` of the server (or of each seed) to `tls` and provide the SSL context (SSL context options):

```
'memcache.kvstore' => [
   'server' => [
      'host' => 'cache-host.example.com',
      'port' => 6379,
      'protocol' => 'tls',
   ],
   'ssl_context' => [
      'local_cert' => '/certs/cache.crt',
      'local_pk' => '/certs/cache.key',
      'cafile' => '/certs/ca.crt',
   ],
],
```

Connecting over UNIX socket

If you want to connect to a server configured to listen on a Unix socket (which is recommended if the server is running on the same system as Nextcloud), pass the socket path as the host. A leading slash in the host is automatically detected as a Unix socket, so no `protocol` or `port` is needed:

```
'memcache.kvstore' => [
   'server' => [
      'host' => '/run/valkey/valkey.sock',
```

(continues on next page)

(continued from previous page)

```
    ],
],
```

Update the server configuration (for example `/etc/valkey/valkey.conf`) accordingly: uncomment the Unix socket options and ensure the socket path matches your Nextcloud configuration.

Be sure to set the right permissions on the socket so that your web server can read and write to it. For this you typically have to add the web server user to the `valkey` (or `redis`) group:

```
usermod -a -G valkey www-data
```

And modify the `unixsocketperm` setting of the server configuration accordingly:

```
unixsocketperm 770
```

You might need to restart your web server and the cache server for the changes to take effect.

8.9.4 Redis using the phpredis PHP extension

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

Nextcloud uses the PhpRedis PHP extension. This extension provides an API for communicating with Redis-compatible key-value stores. Redis Open Source is the formally tested backend; Valkey and KeyDB are expected to work as compatible backends as well.

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 *Memcached*.

On Debian/Ubuntu/Mint, install `redis-server` (or equivalent) and `php-redis`. Package names vary by distribution and backend.

On CentOS and Fedora, install `redis` (or equivalent) and `php-pecl-redis`. It will not start automatically, so you must use your service manager to start the `redis` server, 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.

Redis configuration in Nextcloud (config.php)

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

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

Additionally, you should use Redis for the distributed server cache:

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

Furthermore, you could use Redis for the local cache like so, but it's not recommended (see warning below):

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

Warning

Using Redis for local cache on a multi-server setup can cause issues. Also, even on a single-server setup, APCu (see section above) should be faster.

When using Redis for any of the above cache settings, you also need to specify either the `redis` or `redis.cluster` configuration in `config.php`.

The following options are available to configure when using a single redis server (all but `host` and `port` are optional. For the latter two, see the next sections):

```
'memcache.locking' => '\OC\Memcache\Redis',
'memcache.distributed' => '\OC\Memcache\Redis',
'memcache.local' =>'\OC\Memcache\Redis' ,
'memcache_customprefix' => 'mynextcloudprefix',
'redis' => [
   // 'host'      => see connection parameters below
   // 'port'      => see connection parameters below
  'user'          => 'nextcloud',
  'password'      => 'password',
  'dbindex'       => 0,
  'timeout'       => 1.5,
  'read_timeout'  => 1.5,
],
```

The following options are available to configure when using a redis cluster (all but `seeds` are optional):

```
'memcache.locking' => '\OC\Memcache\Redis',
'memcache.distributed' => '\OC\Memcache\Redis',
'memcache.local' =>'\OC\Memcache\Redis' ,
'memcache_customprefix' => 'mynextcloudprefix',
'redis.cluster' => [
   'seeds' => [ // provide some/all of the cluster servers to bootstrap discovery,␣
↪port required
      'cache-cluster:7000',
      'cache-cluster:7001',
      'cache-cluster:7002',
      'cache-cluster:7003',
      'cache-cluster:7004',
      'cache-cluster:7005'
   ],
   'failover_mode'   => \RedisCluster::FAILOVER_ERROR,
   'timeout'         => 0.0,
   'read_timeout'    => 0.0,
   'user'            => 'nextcloud',
   'password'        => 'password',
   'dbindex'         => 0,
],
```

Note

The port is required as part of the server URL. However, it is not necessary to list all servers: for example, if all

servers are load balanced via the same DNS name, only that server name is required.

Connecting to single Redis server over TCP

To connect to a remote or local Redis server over TCP use:

```
'redis' => [
   'host' => 'redis-host.example.com',
   'port' => 6379,
],
```

Connecting to single Redis server over TLS

To connect via TCP over TLS, add the following configuration:

```
'redis' => [
   'host' => 'tls://127.0.0.1',
   'port' => 6379,
   'ssl_context' => [
      'local_cert' => '/certs/redis.crt',
      'local_pk' => '/certs/redis.key',
      'cafile' => '/certs/ca.crt',
      'verify_peer_name' => false,
   ],
],
```

Connecting to Redis cluster over TLS

To connect via TCP over TLS, add the following configuration:

```
'redis.cluster' => [
   'seeds' => [ // provide some/all of the cluster servers to bootstrap discovery,␣
↪port required
      'cache-cluster:7000',
      'cache-cluster:7001',
   ],
   'ssl_context' => [
      'local_cert' => '/certs/redis.crt',
      'local_pk' => '/certs/redis.key',
      'cafile' => '/certs/ca.crt',
      'verify_peer_name' => false,
   ],
],
```

Connecting to single Redis server over UNIX socket

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:

```
'redis' => [
   'host'     => '/run/redis/redis-server.sock',
   'port'     => 0,
],
```

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

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

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

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

And modify the `unixsocketperm` of the `redis.conf` accordingly:

```
unixsocketperm 770
```

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

```
systemctl restart apache2
systemctl restart redis-server
```

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

Using the Redis session handler

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

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

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

8.9.5 Memcached

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

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

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

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

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

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

Restart your Web server, add the appropriate entries to your `config.php`, and refresh your Nextcloud admin page.

Memcached configuration in Nextcloud (config.php)

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

8.9.6 Cache Directory location

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

8.9.7 Cache Key Prefix for Redis or Memcached

By default, Nextcloud generates a semi-unique prefix for cache keys using information like instance ID, version etc. to mitigate the problem of collisions when using the same cache for multiple Nextcloud instances. If you want to make sure to prevent collisions altogether, you can use the following setting to define your custom prefix:

```
'memcache_customprefix' => 'mynextcloudprefix',
```

This also allows you to create ACLs in Redis and limit the keys specific users can access (e.g. if you want to isolate specific Nextcloud instances when using the same cache). This may be security-relevant for you.

8.10 Dashboard app

The Nextcloud Dashboard is your starting point of the day, giving users an overview of your upcoming appointments, urgent emails, chat messages, incoming tickets, latest tweets and much more! Users can add the widgets they like and change the background to their liking.

8.10.1 Enabling the dashboard app

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

8.10.2 Configuring your Nextcloud for the dashboard app

The dashboard widgets are provided by apps and have a unique identifier. This can be used to customize the default layout of the dashboard as an administrator. The layout is stored as a comma-separated list of widget identifiers.

The layout of an existing user can be read with the following command:

```
occ user:setting admin dashboard layout
```

The layout of the dashboard for a specific user can be set with the following command:

```
occ user:setting admin dashboard layout "calendar,files,activity"
```

The default layout of the dashboard for all users can be set with the following command:

```
occ config:app:set dashboard layout --value="files,activity,calendar"
```

Changing the default layout will not affect existing users that already have a custom layout stored.

It is possible to replace the default app, which is the dashboard app, with a custom app with the following command:

occ config:app:set core defaultpage –value “/apps/files/extstoragemounts”

8.11 Domain Change

Changing the domain after the first setup is currently not supported by Nextcloud. That is mainly because Nextcloud's apps don't support changing the domain after they are set up.

Note

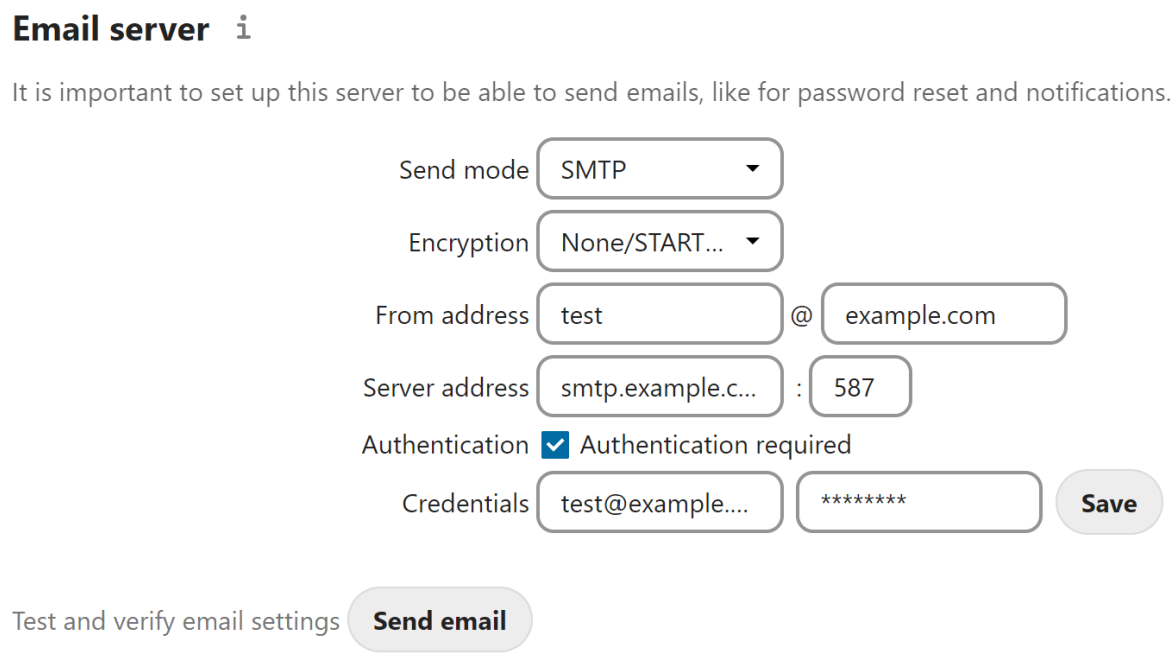
This documentation will get updated with steps what needs to be done if you want to change the domain as soon as this feature is available.

8.12 Email

Nextcloud is capable of sending password reset emails, notifying users of new file shares, changes in files, and activity notifications. Your users configure which notifications they want to receive on their Personal pages.

Nextcloud does not contain a full email server, but rather connects to your existing mail server. You must have a functioning mail server for Nextcloud to be able to send emails. You may have a mail server on the same machine as Nextcloud, or it may be a remote server.

To access the setup page below log in with an admin account. Click on your avatar and then click **Settings**. On the left side under Administration and click Basic settings.



With the wizard, connecting Nextcloud to your mail server is fast and easy. The wizard fills in the values in `config/config.php`, so you may use either or both as you prefer.

The Nextcloud Email wizard supports three types of mail server connections: SMTP, qmail, and Sendmail. Use the SMTP configurator for a remote server or Sendmail when your mail server is on the same machine as Nextcloud.

Note

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

8.12.1 Mail Providers

Added in version 30.

A mail provider is an app that provides outbound mail service to Nextcloud and allows the sending of system emails directly through a user's configured personal email account instead of the system email account. At present, this functionality is limited to calendar invitations. This feature automatically matches a users email address to a configured mail provider account, when a system message is sent. The only app that supports this functionality at present is Nextcloud Mail 4.1 or higher, a configured email account is required.

8.12.2 Configuring an SMTP server

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

Warning

There were changes to the 3rd party mailer library in Nextcloud 26:

- STARTTLS cannot be enforced; it is used automatically if the mail server supports it. Set the encryption type to **None/STARTTLS** to allow this automatic upgrade. See *here* for an example on how to configure self-signed

certificates.

- NTLM authentication for Microsoft Exchange is not supported by the new mailer library. Try using basic authentication instead.
- Outlook and Microsoft Exchange have discontinued support for Basic authentication. It is no longer possible to use their services as your default email handler.

- Encryption type: None/STARTTLS or SSL
- The From address you want your outgoing Nextcloud mails to use
- Whether authentication is required
- Authentication: when authentication is required, the underlying mailer will try the following authentication methods in the order they're listed:
  - CramMd5
  - Login
  - Plain
  - XOAuth2
- The server's IP address or fully-qualified domain name and the SMTP port
- Login credentials (if required)

Note

The `overwrite.cli.url` parameter from `config.php` will be used for the SMTP EHLO.

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

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

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

8.12.3 Configuring Sendmail/qmail

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

In most cases the `SMTP` option is best, since you will be able to control all of your mail server options in one place, in your mail server configuration then.

8.12.4 Using email templates

We designed a mechanism that generates emails which follow the theming settings and look the same in all the different email clients out there.

> **Note**
>
> If, for some reason, you need text-only emails, consider simply configuring this on the client side or let the receiving (or even sending) mail server drop the HTML part. Note that there is no security impact from **sending** HTML emails, just from displaying them and thus any security risk can only be mitigated by disabling showing HTML on the client (or removing the HTML part in the mail server).

Modifying the look of emails beyond the theming app capabilities

You can overwrite templates by writing a class that implements the template interface (or extends it to not need to copy over everything). Easiest way is to then put this class into an app and load it so you do not need to patch it on every update.

This is the interface of the class that needs to be implemented: https://github.com/nextcloud/server/blob/master/lib/public/Mail/IEMailTemplate.php

That is the implementation that could be extended and used to see how it works: https://github.com/nextcloud/server/blob/master/lib/private/Mail/EMailTemplate.php

An example from a Nextcloud portal article:

1. Look at the source code of extended class OC\Mail\EMailTemplate::class
2. Then override what you need in your own *OC\Mail\EMailTemplate::class* extension

Example:

Let's assume that we need to override the email header:

```
<?php

namespace \OCA\MyApp;

use OC\Mail\EMailTemplate;

class MyClass extends EMailTemplate
{
   protected string $header = <<<EOF
      <table align="center" class="wrapper">
            // your theme email header modification
      </table>
   EOF;
}
```

3. Then in `config/config.php` change `mail_template_class` to your class namespace:

```
'mail_template_class' => 'OCA\\MyApp\\MyClass',
```

You will find a detailed step by step guide in our support portal.

8.12.5 Setting mail server parameters in config.php

If you prefer, you may set your mail server parameters in `config/config.php`. The following examples are for SMTP, Sendmail, and Qmail.

SMTP

If you want to send email using a local or remote SMTP server it is necessary to enter the name or IP address of the server, optionally followed by a colon separated port number, e.g. **:425**. If this value is not given the default port 25/tcp will be used unless you change that by modifying the **mail_smtpport** parameter.

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

or

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

If a malware or SPAM scanner is running on the SMTP server it might be necessary that you increase the SMTP timeout to e.g. 30s:

```
"mail_smtptimeout"  => 30,
```

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

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

The connection will be upgraded automatically via STARTTLS if the SMTP server supports it.

If required by the SMTP server, a secure SSL/TLS connection can be enforced via the SMTPS protocol which uses the port 465/tcp:

```
"mail_smtphost"     => "smtp.server.dom:465",
"mail_smtpsecure"   => 'ssl',
```

And finally it is necessary to configure if the SMTP server requires authentication, if not, the default values can be taken as is.

```
"mail_smtpauth"     => false,
"mail_smtpname"     => "",
"mail_smtppassword" => "",
```

If SMTP authentication is required you have to set the required username and password.

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

Sendmail

If you want to use the well known Sendmail program to send email, it is necessary to have an installed and working email system on your *nix server. The sendmail binary (**/usr/sbin/sendmail**) is usually part of that system. Nextcloud should be able to send email out of the box.

```
"mail_smtpmode"     => "sendmail",
"mail_smtphost"     => "127.0.0.1",
"mail_smtpport"     => 25,
"mail_smtptimeout"  => 10,
```

(continues on next page)

(continued from previous page)

```
"mail_smtpsecure"   => "",
"mail_smtpauth"     => false,
"mail_smtpauthtype" => "LOGIN",
"mail_smtpname"     => "",
"mail_smtppassword" => "",
```

qmail

If you want to use the qmail program to send email, it is necessary to have an installed and working qmail email system on your server. The qmail binary installed on your server will then be used to send email. Nextcloud should be able to send email out of the box.

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

8.12.6 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.

8.12.7 Troubleshooting

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 and temporarily setting your NC loglevel to DEBUG:

```
"mail_smtpdebug" => true,
"loglevel" => 0,
```

Be cautious setting your `loglevel` to DEBUG (`0`) since it'll apply to everything occurring on your NC instance, not just email. And don't forget to set it back to a more reasonable level when you're done troubleshooting:

```
"mail_smtpdebug" => false,
"loglevel" => 2,
```

> **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.

Why is my web domain different from my mail domain?

The default domain name used for the sender address is the hostname where your Nextcloud installation is served. If you have a different mail domain name you can override this behavior by setting the following configuration parameter:

```
"mail_domain" => "example.com",
```

This setting results in every email sent by Nextcloud (for example, the password reset email) having the domain part of the sender address appear as follows:

```
no-reply@example.com
```

How can I find out if an SMTP server is reachable?

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
```

How can I find out if the SMTP server is listening on a specific TCP port?

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 submissions
- 587/tcp is opportunistically-encrypted submission
- 993/tcp/udp is encrypted imaps
- 995/tcp/udp is encrypted pop3s

How can I determine if the SMTP server supports the SMTPS protocol?

A good indication that the SMTP server supports the SMTPS protocol is that it is listening on the *submissions* port **465**.

How can I determine what authorization and encryption protocols the mail server supports?

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.
```

How can I send mail using self-signed certificates or use STARTTLS with self signed certificates?

To disable peer verification or to use self signed certificates, add the following to your `config/config.php`:

```
"mail_smtpstreamoptions" => array(
    'ssl' => array(
        'allow_self_signed' => true,
        'verify_peer' => false,
        'verify_peer_name' => false
    )
),
```

All emails keep getting rejected even though only one email address is invalid.

Partial sending, i. e. sending to all but the faulty email address is not possible.

> **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.

8.13 Linking external sites

You can embed external websites or documents inside your Nextcloud pages with the **External sites** app, as this screenshot shows.

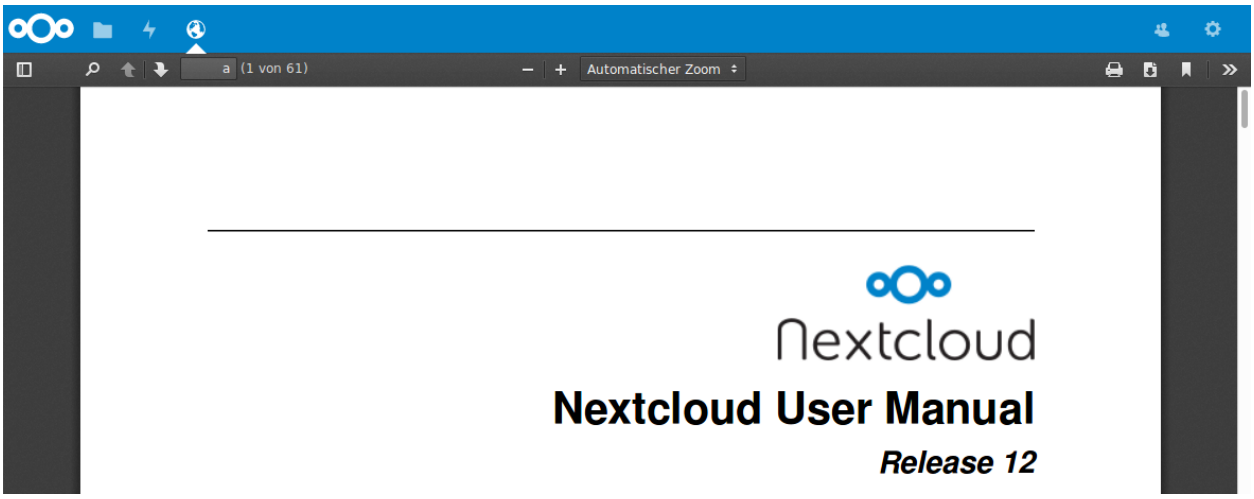


Fig. 2: *Click to enlarge*

This is useful for quick access to important pages such as the Nextcloud manuals and informational pages for your company, and for presenting external pages inside your custom Nextcloud branding, if you use your own custom themes.

The External sites app can be easily installed from the app store. Go to **Settings > Apps > Customization** to enable it. Then go to your Nextcloud **Settings > Administration > External sites** to create your links, which are saved automatically.

Each link can have a unique icon, which can be uploaded in the admin settings. If you select a language, the link will only be displayed for users with the selected language. This allows you to have different documentation links for users depending on their language.

It is also possible to add links for a special device (recognized by the user agent). Currently the following options are available: All devices, Android app, iOS app, Desktop client and all others (Browsers).

It is also possible to add links only for members of a given group.

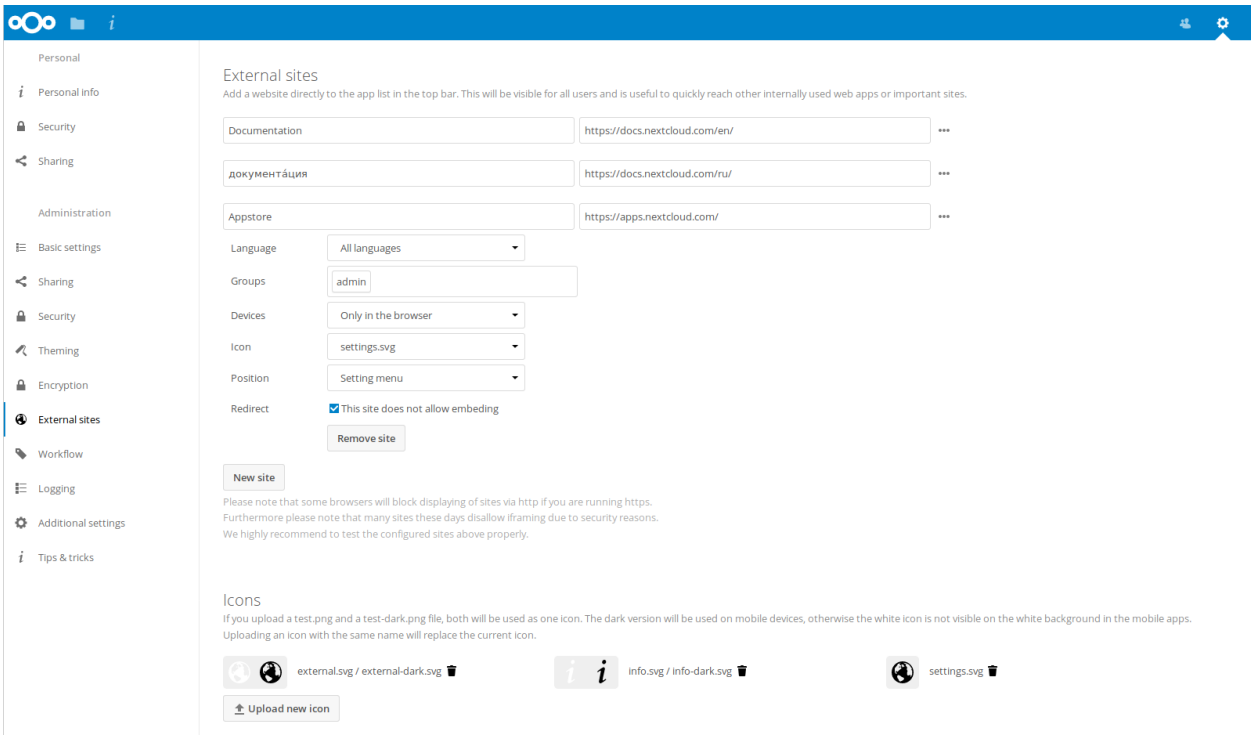
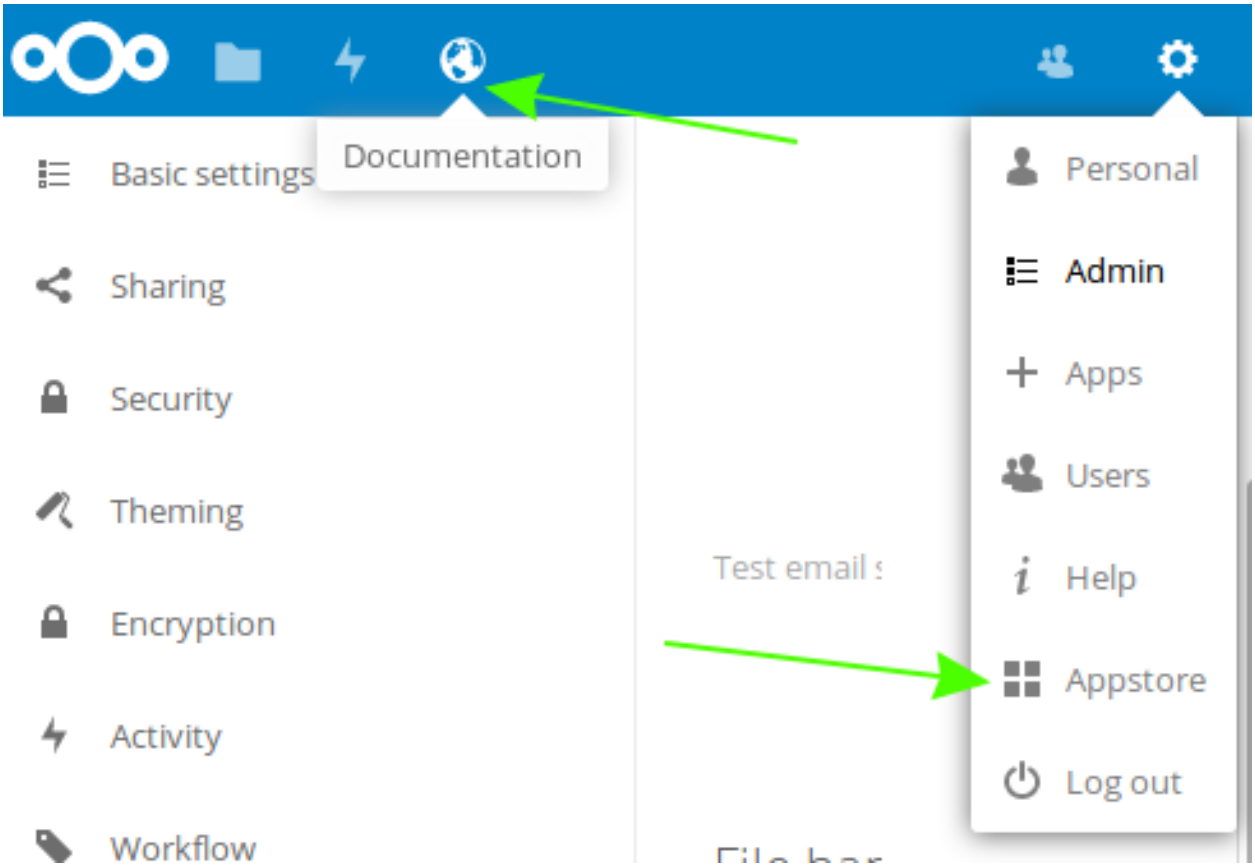


Fig. 3: *Click to enlarge*

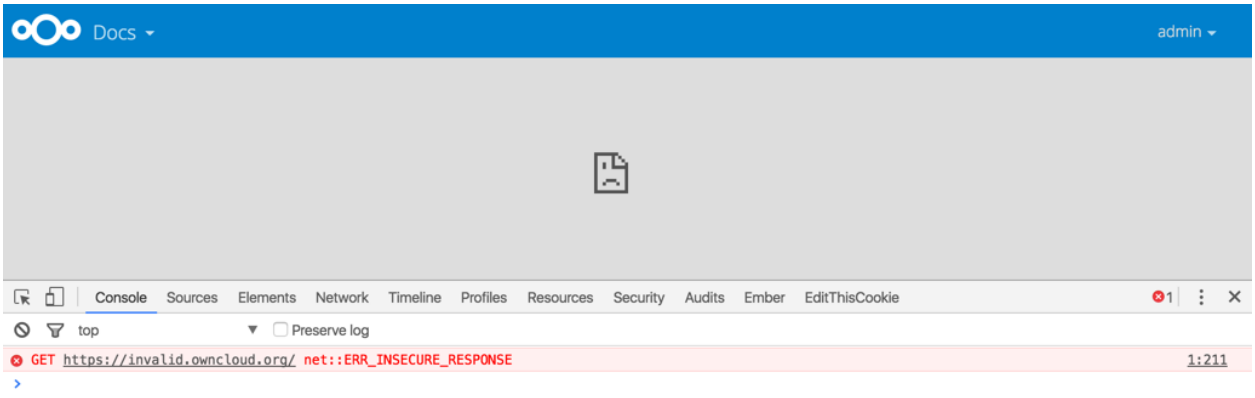
The links appear in the Nextcloud menu on the top or in the settings menu, after reloading the page.



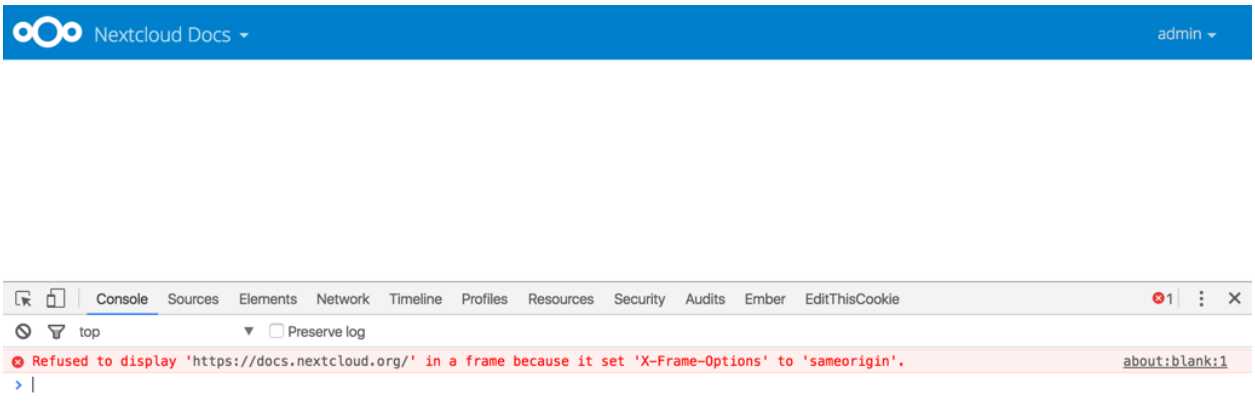
8.13.1 Configurations preventing embedding

Your links may or may not work correctly due to the various ways that Web browsers and Web sites handle HTTP and HTTPS URLs, and because the External Sites app embeds external links in IFrames. Modern Web browsers try very hard to protect Web surfers from dangerous links, and safety apps like Privacy Badger and ad-blockers may block embedded pages. It is strongly recommended to enforce HTTPS on your Nextcloud server; do not weaken this, or any of your security tools, just to make embedded Web pages work. After all, you can freely access them outside of Nextcloud.

Most Web sites that offer login functionalities use the `X-Frame-Options` or `Content-Security-Policy` HTTP header which instructs browsers to not allow their pages to be embedded for security reasons (e.g. "Clickjacking"). You can usually verify the reason why embedding the website is not possible by using your browser's console tool. For example, this page has an invalid SSL certificate.



On this page, X-Frame-Options prevents the embedding.



There is also a redirect option, which allows that those websites can still be added for quick access. Instead of embedding the website the user will be redirected to it.

8.14 Language & Locale

8.14.1 Default language

In normal cases Nextcloud will automatically detect the language of the Web-GUI. If this does not work properly or you want to make sure that Nextcloud always starts with a given language, you can set a **default_language** parameter in the `config/config.php`.

> **Note**
>
> The `default_language` parameter only applies when the browser sends no language preference and the user has not set their own. Accepts ISO 639-1 codes such as `en` (English), `fr` (French), `de` (informal German), or `de_DE` (formal German).

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

8.14.2 Force language

If you want to force a specific language, users will no longer be able to change their language in the personal settings. You can set a **force_language** parameter in the `config/config.php`.

```
"force_language" => "en",
```

If users shall be unable to change their language, but users have different languages, this value can be set to `true` instead of a language code.

> **Note**
>
> Please check Transifex language codes for the list of valid language codes.

8.14.3 Default locale

The locale is used to define how dates and other formats are displayed. Nextcloud should automatically pick an appropriate locale based on your current language. Users can modify their locale inside their settings panel. If that does not work properly or if you want to make sure that Nextcloud always starts with a given locale, you can set a **default_locale** parameter in the `config/config.php`.

> **Note**
>
> The default_locale parameter is only used when the user hasn't configured own locale preferences.

```
"default_locale" => "en_US",
```

8.14.4 Force locale

If you want to force a specific locale, users will no longer be able to change their locale in the personal settings. You can set a **force_locale** parameter in the `config/config.php`.

```
"force_locale" => "en_US",
```

> **Note**
>
> Please check the list of MomentJS supported locales for the list of valid locales.

8.15 Logging

Use your Nextcloud log to review system status, or to help debug problems. You may adjust logging levels, and choose how and where log data is stored. If additional event logging is required, you can optionally activate the **admin_audit** app.

When `file` based logging is utilized, both the Nextcloud log and, optionally, the **admin_audit** app log can be viewed within the Nextcloud interface under *Administration settings* -> *Logging* (this functionality is provided by the **logreader** app).

Further configuration and usage details for both the standard Nextcloud log and the optional **admin_audit** app log can be found below.

8.15.1 Log level

Logging levels range from **DEBUG**, which logs all activity, to **FATAL**, which logs only fatal errors.

- **0**: DEBUG: All activity; the most detailed logging.
- **1**: INFO: Activity such as user logins and file activities, plus warnings, errors, and fatal errors.
- **2**: WARN: Operations succeed, but with warnings of potential problems, plus errors and fatal errors.
- **3**: ERROR: An operation fails, but other services and operations continue, plus fatal errors.
- **4**: FATAL: The server stops.

By default the log level is set to **2** (WARN). Use **DEBUG** when you have a problem to diagnose, and then reset your log level to a less-verbose level as **DEBUG** outputs a lot of information, and can affect your server performance.

Logging level parameters are set in the `config/config.php` file.

8.15.2 Log type

errorlog

All log information will be sent to PHP `error_log()`.

```
"log_type" => "errorlog",
```

Log entries will be prefixed with `[nextcloud]`.

file

All log information will be written to a separate log file which can be viewed using the log viewer on your Admin page. By default, a log file named **nextcloud.log** will be created in the directory which has been configured by the **datadirectory** parameter in `config/config.php`.

The desired date format can optionally be defined using the **logdateformat** parameter in `config/config.php`. By default the PHP date function parameter `c` is used, and therefore the date/time is written in the format `2013-01-10T15:20:25+02:00`. By using the date format in the example below, the date/time format will be written in the format `January 10, 2013 15:20:25`.

```
"log_type" => "file",
"logfile" => "nextcloud.log",
"loglevel" => 3,
"logdateformat" => "F d, Y H:i:s",
```

Additional file-based logging parameters

The following optional parameters can also be set in `config/config.php`:

logfilemode
: Sets the file permissions (in octal notation) for the log file.

  Defaults to `0640`.

```
"logfilemode" => 0640,
```

logtimezone
: Sets the timezone used for log timestamps. See the PHP list of supported timezones.

  Defaults to `UTC`.

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

log_rotate_size
: Enables log rotation and limits the total size of log files. The value is specified in bytes. When the current log file reaches this size a new log file is created. If a rotated log file already exists it will be overwritten. Set to `0` to disable rotation.

  Defaults to `104857600` (100 MB).

```
"log_rotate_size" => 100 * 1024 * 1024,
```

log.backtrace
: When enabled, a backtrace is appended to every log line, not only to exceptions. This significantly increases log size and should only be used for debugging.

  Defaults to `false`.

```
"log.backtrace" => true,
```

log_query
: Appends all database queries and their parameters to the log file. Use this only for debugging as it produces very large log files.

  Defaults to `false`.

```
"log_query" => true,
```

loglevel_frontend
: Sets a separate log level for messages originating from the Nextcloud frontend (browser). Accepts the same values as `loglevel` (0–4).

  Defaults to `2` (WARN).

```
"loglevel_frontend" => 2,
```

loglevel_dirty_database_queries
: Sets the log level at which dirty database queries (queries executed after the response has already been sent) are logged.

  Defaults to `0` (DEBUG).

```
"loglevel_dirty_database_queries" => 0,
```

syslog

All log information will be sent to your default syslog daemon.

```
"log_type" => "syslog",
"syslog_tag" => "Nextcloud",
"logfile" => "",
"loglevel" => 3,
```

systemd

All log information will be sent to Systemd journal. Requires php-systemd extension.

```
"log_type" => "systemd",
"syslog_tag" => "Nextcloud",
```

8.15.3 Conditional logging (log.condition)

Nextcloud supports conditional overrides that temporarily increase the log level to **DEBUG** when certain criteria are met. This is useful for diagnosing problems in production without flooding the entire log with debug output.

The `log.condition` parameter is set in `config/config.php`.

Basic conditions

At the top level you can specify one or more of the following keys. When *any* of them match, the log level for that request is set to **DEBUG**:

shared_secret
: Match requests that pass the query parameter `log_secret` with this value.

users
: An array of user IDs. If the currently authenticated user is in the list, the condition is satisfied.

apps
: An array of app identifiers. Any log message whose app context matches one of these apps will be logged at DEBUG level.

Example – enable debug logging for user `jane` and the `files` app:

```
'log.condition' => [
    'users' => ['jane'],
    'apps' => ['files'],
],
```

Advanced compound conditions (matches)

The `matches` key accepts an array of condition groups. Each group can combine all of the keys above plus:

message
: A substring that must appear in the log message.

loglevel
: The log level to apply when this group matches (instead of the default DEBUG / `0`).

All keys within a single group must match for the group to apply (logical AND). Multiple groups are evaluated independently (logical OR).

Example – log all messages from the `files` app at INFO level, and log any message containing `"Lock"` for user `admin` at DEBUG level:

```
'log.condition' => [
    'matches' => [
        [
            'apps' => ['files'],
            'loglevel' => 1,
        ],
```

(continues on next page)

(continued from previous page)

```
        [
            'users' => ['admin'],
            'message' => 'Lock',
            'loglevel' => 0,
        ],
    ],
],
```

Using a shared secret for on-demand debugging

You can trigger debug logging for a single request by adding a `log_secret` query parameter. Set a secret in `config/config.php`:

```
'log.condition' => [
    'shared_secret' => '57b58edb6637fe3059b3595cf9c41b9',
],
```

Then call your Nextcloud URL with the secret appended:

```
https://cloud.example.com/index.php?log_secret=57b58edb6637fe3059b3595cf9c41b9
```

Warning

Keep the shared secret private. Anyone who knows it can enable debug-level logging on your instance.

8.15.4 Log fields explained

Example log entries


```
{
    "reqId":"TBsuA2uE86DiOD0S8f9j",
    "level":1,
    "time":"April 13, 2021 16:55:37",
    "remoteAddr":"192.168.56.1",
    "user":"admin",
    "app":"admin_audit",
    "method":"GET",
    "url":"/ocs/v1.php/cloud/users?disabled",
    "message":"Login successful: \"admin\"",
    "userAgent":"curl/7.68.0",
    "version":"21.0.1.1"
}

{
    "reqId":"ByeDVLuwkXKMfLpBgvxC",
    "level":2,
    "time":"April 14, 2021 09:03:29",
    "remoteAddr":"192.168.56.1",
    "user":"--",
    "app":"no app in context",
```


(continues on next page)

(continued from previous page)

```
    "method":"POST",
    "url":"/login",
    "message":"Login failed: asdf (Remote IP: 192.168.56.1)",
    "userAgent":"Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like␣
↪Gecko) Chrome/89.0.4389.114 Safari/537.36",
    "version":"21.0.1.1"
}
```

Log field breakdown

- **reqId** (request id): any log lines related to a single request have the same value
- **level**: logged incident's level, always 1 in audit.log
- **time**: date and time (format and timezone can be configured in config.php)
- **remoteAddr**: the IP address of the user (if applicable – empty for occ commands)
- **user**: acting user's id (if applicable)
- **app**: affected app (always admin_audit in audit.log)
- **method**: HTTP method, for example GET, POST, PROPFIND, etc. – empty on occ calls
- **url**: request path (if applicable – empty on occ calls)
- **scriptName**: the PHP script name that handled the request
- **message**: event information message
- **userAgent**: user agent (if applicable – empty on occ calls)
- **exception**: full exception with trace (if applicable)
- **data**: additional structured data (if applicable)
- **version**: Nextcloud version at the time of request
- **clientReqId**: value of the `X-Request-Id` HTTP header sent by the client (only present when the header is set)
- **occ_command**: the occ command that was executed, as an array of up to two arguments (only present in CLI mode)
- **backtrace**: full PHP backtrace (only present when `log.backtrace` is enabled in config.php)

Empty values are written as two dashes: `--`.

8.15.5 Admin audit log (Optional)

By enabling the **admin_audit** app, additional information about various events can be logged. The audit log supports all of the logging backends described above (file, syslog, systemd, errorlog).

Default audit log location

When using the default `file` backend, the audit log is written to **audit.log** inside the directory configured by `datadirectory` in `config/config.php` (e.g. `/var/www/html/data/audit.log`).

You can override the path with the `logfile_audit` system config key:

```
'logfile_audit' => '/var/log/nextcloud/audit.log',
```

Dedicated audit logging configuration

The audit log backend can be configured independently from the main Nextcloud log using three dedicated keys in `config/config.php`:

log_type_audit
: The logging backend for audit events. Accepts the same values as `log_type`: `file`, `syslog`, `systemd`, or `errorlog`.

  Defaults to `file`.

logfile_audit
: Path to the audit log file (only used when `log_type_audit` is `file`). When empty, the default `audit.log` in the data directory is used.

syslog_tag_audit
: The syslog identifier for audit messages (only used when `log_type_audit` is `syslog` or `systemd`). Defaults to the value of `syslog_tag`.

Example – send audit events to syslog instead of a file:

```
'log_type_audit' => 'syslog',
'syslog_tag_audit' => 'Nextcloud',
'logfile_audit' => '',
```

Log level interaction

The **admin_audit** app writes its messages at **INFO** level (`1`). Because the default system `loglevel` is **WARN** (`2`), audit messages are suppressed unless you explicitly lower the system log level or add a conditional override.

The recommended approach is to add a `log.condition` entry that forces DEBUG-level logging for the `admin_audit` app context, without affecting other apps:

```
'log.condition' => [
    'apps' => ['admin_audit'],
],
```

This ensures audit events are always written regardless of the global `loglevel` setting.

Integrating into the web interface

The built-in **logreader** app (which provides *Administration settings* → *Logging*) only reads the file-based `nextcloud.log`. By default the audit log is written to a separate `audit.log` file, so audit entries will not appear in the web interface.

If you want audit events to appear in the logreader, point `logfile_audit` at the same file as `nextcloud.log`:

```
'log.condition' => [
    'apps' => ['admin_audit'],
],
'logfile_audit' => '/var/www/html/data/nextcloud.log',
```

> **Note**
>
> Adjust the path above to match your actual `datadirectory` location. This merges audit entries into the main log file; the separate `audit.log` will no longer be written.

Configuring through admin_audit app settings (legacy)

Previously the audit log file was defined via app config. This setting is still read as a fallback when `logfile_audit` is not set in `config/config.php`, but it is considered a **legacy** parameter. The system config key should be preferred for new installations.

```
occ config:app:set admin_audit logfile --value=/var/log/nextcloud/audit.log
```

8.15.6 Workflow log

The **workflowengine** app records events related to Nextcloud Flow (automated workflows configured under *Administration settings → Flow*). By default, these events are written to `flow.log` inside the data directory.

The path of the workflow log can be changed with:

```
occ config:app:set workflowengine logfile --value=/var/log/nextcloud/flow.log
```

To disable workflow logging entirely, redirect the output to `/dev/null`:

```
occ config:app:set workflowengine logfile --value=/dev/null
```

8.15.7 Temporary overrides

You can override the config.php log level for `occ` commands as *documented here*.

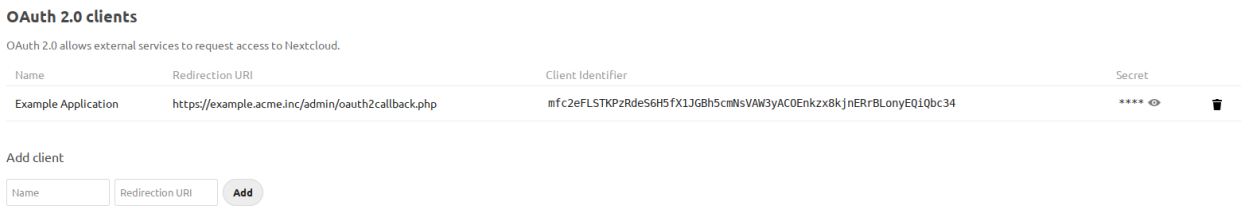
8.16 OAuth2

Nextcloud allows connecting external services (for example Moodle) to your Nextcloud. This is done via `OAuth2`. See RFC6749 for the OAuth2 specification.

> **Note**
>
> Nextcloud supports only confidential clients.

8.16.1 Add an OAuth2 Application

Head over to your Administrator Security Settings. Here you can add a new `OAuth2` client.



Enter the name of your application and provide a redirection url. You should now have a Client Identifier and Secret. Enter those into your `OAuth2` client.

Please provide the OAuth2 application the following details:

- Authorization endpoint: `https://cloud.example.org/apps/oauth2/authorize`

- Token endpoint: `https://cloud.example.org/apps/oauth2/api/v1/token`

Note that you must include `index.php` if pretty URL is not configured - i.e. `https://cloud.example.org/index.php/apps/oauth2/api/v1/token`.

8.16.2 The access token

The access token obtained is a so called Bearer token. Which means that for request to the Nextcloud server you will have to send the proper authorization header.

Authorization: Bearer <TOKEN>

Note that apache by default strips this. Make sure you have `mod_headers`, `mod_rewrite` and `mod_env` enabled.

8.16.3 Security considerations

Nextcloud `OAuth2` implementation currently does not support scoped access. This means that every token has full access to the complete account including read and write permission to the stored files. It is essential to store the `OAuth2` tokens in a safe way!

Without scopes and restrictable access it is not recommended to use a Nextcloud instance as a user authentication service.

8.16.4 Skipping pre-login warning

In Nextcloud default `OAuth2` flow, a confirmation step is shown before login if the user is not yet logged-in, and a second one is shown after login. To skip the pre-login one for a trusted application, the configuration option `skipAuthPickerApplications` can be set through occ:

```
sudo -E -u www-data php occ config:app:set oauth2 skipAuthPickerApplications --type␣
↪array --value '["myapplication"]'
```

8.17 Reverse proxy

Nextcloud can be run through a reverse proxy, which can cache static assets such as images, CSS or JS files, move the load of handling HTTPS to a different server or load balance between multiple servers.

8.17.1 Defining trusted proxies

For security, you must explicitly define the proxy servers that Nextcloud is to trust. Connections from trusted proxies will be specially treated to get the real client information, for use in access control and logging. Parameters are configured in `config/config.php`

Set the `trusted_proxies` parameter as an array of:

- IPv4 addresses
- IPv4 ranges in CIDR notation
- IPv6 addresses
- IPv6 ranges in CIDR notation

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.

8.17.2 Overwrite parameters

The automatic hostname, protocol or webroot detection of Nextcloud can fail in certain reverse proxy situations. This configuration allows the automatic detection to be manually overridden. If Nextcloud fails to automatically detect the hostname, protocol or webroot you can use the **overwrite** parameters inside the `config/config.php`.

- `overwritehost` set the hostname of the proxy. You can also specify a port.
- `overwriteprotocol` set the protocol of the proxy. You can choose between the two options **http** and **https**.
- `overwritewebroot` set the absolute web path of the proxy to the Nextcloud folder.
- `overwritecondaddr` overwrite the values dependent on the remote address. The value must be a **regular expression** of the IP addresses of the proxy. This is useful when you use a reverse SSL proxy only for https access and you want to use the automatic detection for http access.
- `overwrite.cli.url` the base URL for any URLs which are generated within Nextcloud using any kind of command line tools. For example, the value set here will be used by the notifications area.

Leave the value empty or omit the parameter to keep the automatic detection.

8.17.3 Service Discovery

The redirects for CalDAV or CardDAV does not work if Nextcloud is running behind a reverse proxy. The recommended solution is that your reverse proxy does the redirects.

Apache2

```
RewriteEngine On
RewriteRule ^/\.well-known/carddav https://%{SERVER_NAME}/remote.php/dav/ [R=301,L]
RewriteRule ^/\.well-known/caldav https://%{SERVER_NAME}/remote.php/dav/ [R=301,L]
```

Thanks to @ffried for apache2 example.

Traefik 1

Using Docker labels:

```
traefik.frontend.redirect.permanent: 'true'
traefik.frontend.redirect.regex: 'https://(.*)/.well-known/(?:card|cal)dav'
traefik.frontend.redirect.replacement: 'https://$$1/remote.php/dav'
```

Using traefik.toml:

```
[frontends.frontend1.redirect]
  regex = "https://(.*)/.well-known/(?:card|cal)dav"
  replacement = "https://$1/remote.php/dav
  permanent = true
```

Thanks to @pauvos and @mrtumnus for traefik examples.

Traefik 2

Using Docker labels:

```
- "traefik.http.routers.nextcloud.middlewares=nextcloud_redirectregex@docker"
- "traefik.http.middlewares.nextcloud_redirectregex.redirectregex.permanent=true"
- "traefik.http.middlewares.nextcloud_redirectregex.redirectregex.regex=https://(.*)/.
↪well-known/(?:card|cal)dav"
- "traefik.http.middlewares.nextcloud_redirectregex.redirectregex.replacement=https://
↪$${1}/remote.php/dav"
```

Using a TOML file:

```
[http.middlewares]
  [http.middlewares.nextcloud-redirectregex.redirectRegex]
    permanent = true
    regex = "https://(.*)/.well-known/(?:card|cal)dav"
    replacement = "https://${1}/remote.php/dav"
```

HAProxy

```
acl url_discovery path /.well-known/caldav /.well-known/carddav
http-request redirect location /remote.php/dav/ code 301 if url_discovery
```

NGINX

If using nginx as Nextcloud's webserver from behind another nginx reverse proxy, put this only in the reverse proxy's configuration.

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

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

location ^~ /.well-known {
    return 301 $scheme://$host/index.php$uri;
}
```

When using NGINX Proxy Manager, the entry `proxy_hide_header Upgrade;` must be added in the *"Advanced Settings"* of the proxy host under *"Custom Nginx Configuration"*, otherwise mobile devices (iPad, iPhone etc.) will simply receive the Error Message "Connection Closed".

Caddy

```
subdomain.example.com {
    redir /.well-known/carddav /remote.php/dav/ 301
    redir /.well-known/caldav /remote.php/dav/ 301

    reverse_proxy {$NEXTCLOUD_HOST:localhost}
}
```

Pomerium

```
- from: https://subdomain.example.com
  path: /.well-known/carddav
  redirect:
    path_redirect: /remote.php/dav/
- from: https://subdomain.example.com
  path: /.well-known/caldav
  redirect:
    path_redirect: /remote.php/dav/
```

Thanks to @JeffMatson for Pomerium example.

8.17.4 Examples

Nextcloud behind a reverse proxy (subdirectory)

If your Nextcloud is served at a subdirectory, for example **https://example.com/nextcloud**, behind a reverse proxy with IP address **10.0.0.1** that terminates TLS, set the following parameters inside `config/config.php`:

```
<?php
$CONFIG = array (
  'trusted_proxies'   => ['10.0.0.1'],
  'overwriteprotocol' => 'https',
  'overwritewebroot'  => '/nextcloud',
  'overwrite.cli.url' => 'https://example.com/nextcloud',
);
```

Note

`overwritehost` is not needed in most setups — Nextcloud will read the hostname from the `Host` header forwarded by the proxy. Only set it if you need to force a specific hostname regardless of the incoming request. Leave any parameter unset or empty to keep the automatic detection.

Nextcloud accessible via multiple domains / conditional overwrite

If Nextcloud is reachable both directly (HTTP) and through a reverse proxy (HTTPS), or through multiple proxies serving different public domains, use `overwritecondaddr` to apply the overwrite parameters only when requests arrive from a specific proxy IP address. Requests that do not originate from that proxy will use automatic detection.

In the example below, the overwrite parameters are applied only when requests come from the proxy at **10.0.0.1**, which serves Nextcloud as **https://public.example.com**:

```
<?php
$CONFIG = array (
  'trusted_proxies'   => ['10.0.0.1'],
  'overwritehost'     => 'public.example.com',
  'overwriteprotocol' => 'https',
  'overwritecondaddr' => '^10\.0\.0\.1$',
  'overwrite.cli.url' => 'https://public.example.com',
);
```

> **Note**
>
> `overwritecondaddr` takes a regular expression matching the remote address of the proxy. When set, the overwrite parameters are only applied if the remote address matches. This is useful when the same Nextcloud instance is accessible both with and without a reverse proxy, or when different proxies serve the instance under different hostnames.

Multiple trusted domains and share link URLs

When multiple entries are listed in `trusted_domains`, Nextcloud accepts requests on all of them and generates URLs based on the hostname of each incoming request. This means share links, download URLs, and notification links will reflect whichever hostname was used at the time they were copied.

A share link copied while accessing Nextcloud via an internal hostname (e.g. `cloud.local`) will contain that hostname and will not be reachable by users on a different network who only have access to the public hostname (e.g. `cloud.example.com`).

To ensure all generated URLs always use a single canonical hostname, set `overwritehost` and `overwrite.cli.url` unconditionally:

```
<?php
$CONFIG = array (
  'trusted_domains'   => ['cloud.local', 'cloud.example.com'],
  'overwritehost'     => 'cloud.example.com',
  'overwriteprotocol' => 'https',
  'overwrite.cli.url' => 'https://cloud.example.com',
);
```

> **Note**
>
> Unlike the `overwritecondaddr` pattern above, this applies the overwrite to every request regardless of origin. Use this when you want one authoritative public URL for all generated links, even when the instance is also reachable internally under a different hostname.

8.18 Text app

8.18.1 Disable rich workspaces globally

Rich workspaces can be disabled globally by the admin by setting the following config option to 0 (default is 1):

```
occ config:app:set text workspace_available --value=0
```

8.18.2 Default file extension

The default file extension can be changed to txt in order to always create plain text files (default is md):

```
occ config:app:set text default_file_extension --value=txt
```

8.18.3 Disable rich text editing

Rich text editing can be turned off globally to always open markdown files in their raw format, without rendering of the formatting (default is 1):

```
occ config:app:set text rich_editing_enabled --value=0
```

8.18.4 File encodings

Text can automatically detect encodings of files and will convert them to UTF-8 when saving. Due to the variety of encodings, not all of them can be detected, however you can configure a list of encodings and in which priority they should be detected using the php `mbstring.detect_order` setting in your php.ini:

```
mbstring.detect_order = ASCII,JIS,UTF-8,SJIS,EUC-JP
```

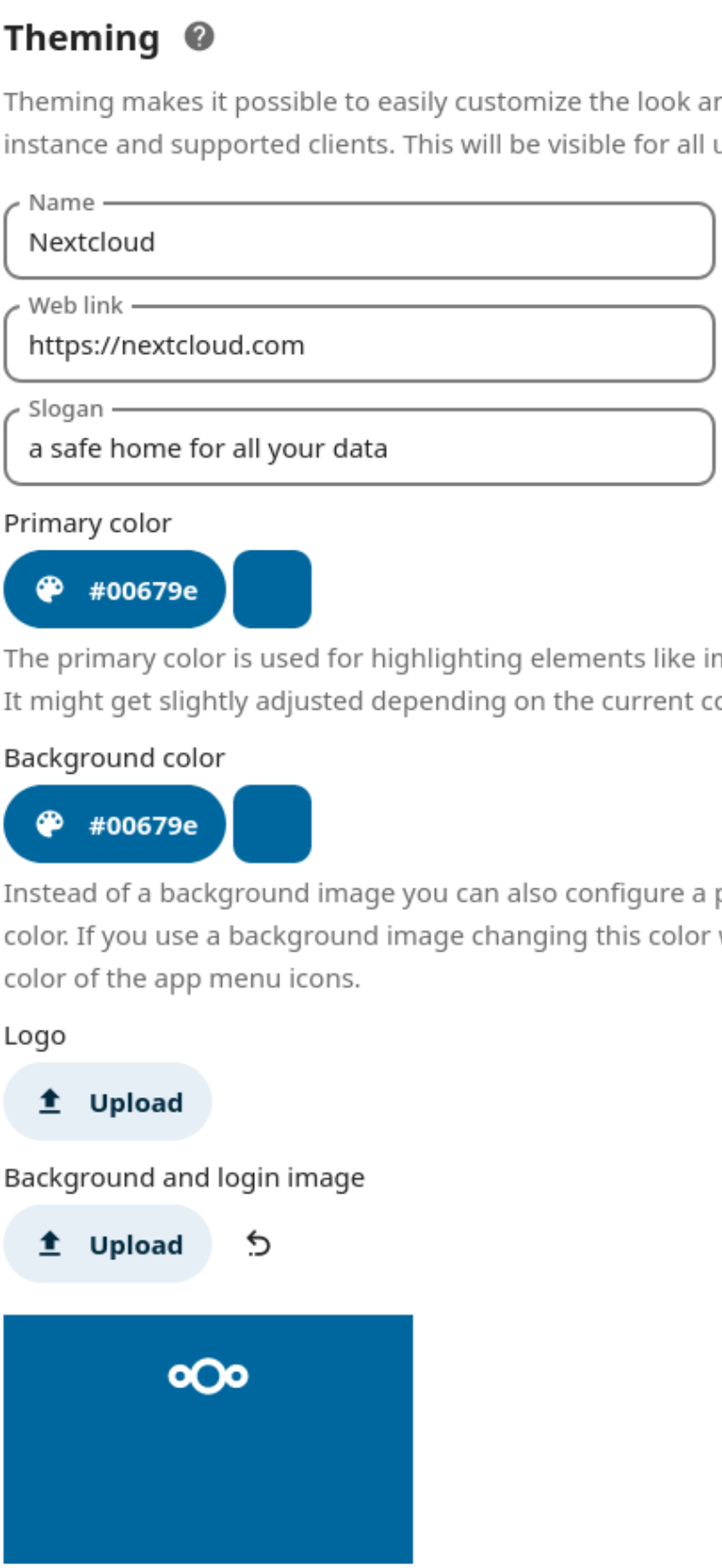
8.19 Theming

The theming feature lets you customize the look and feel of your Nextcloud instance to match your organization's design (or - if for personal use - your own style). You can replace the Nextcloud logo and background images with your own assets, customize colors and slogans, and add dedicated legal and privacy links.

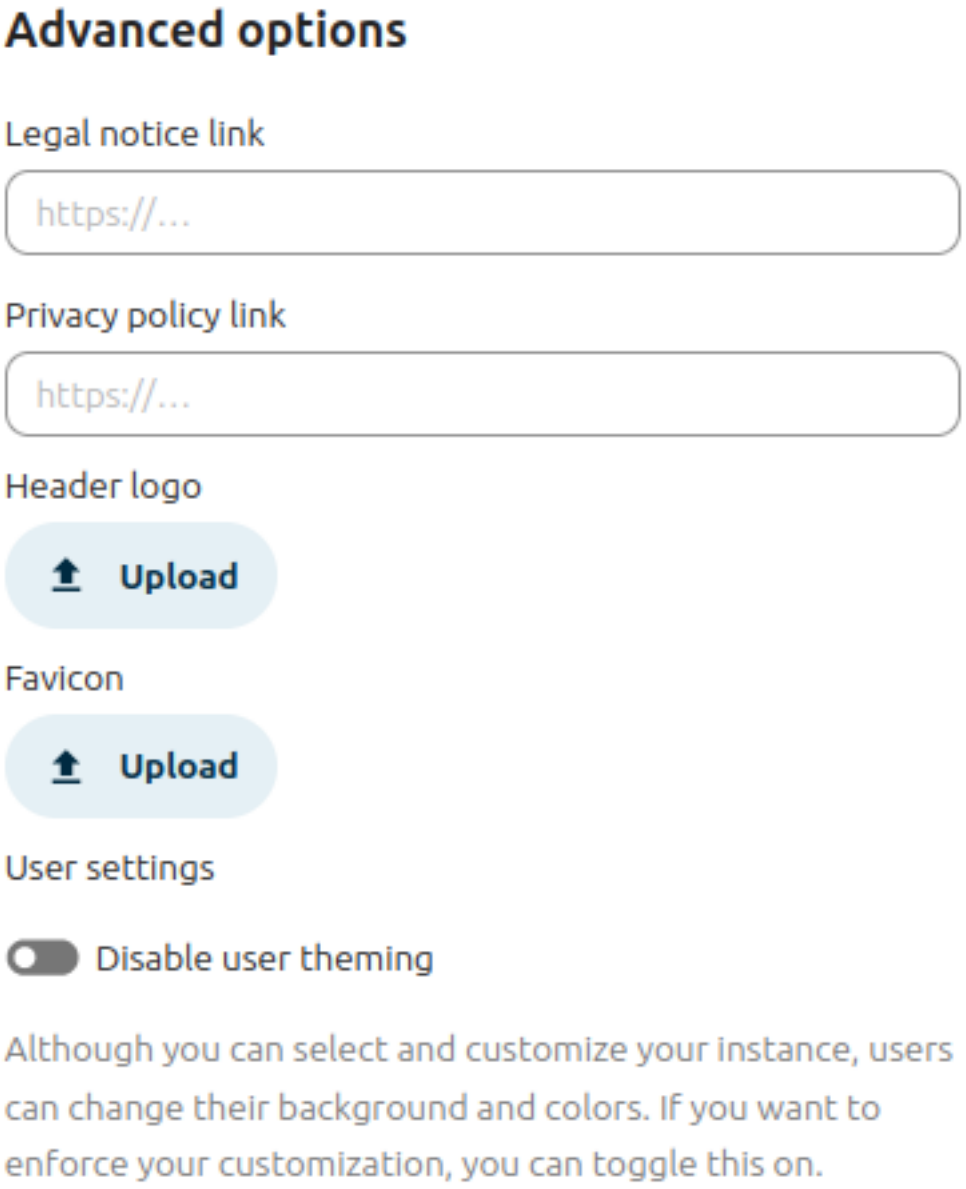
Theme customization is provided by the shipped and always-enabled `theming` app. Use the **Theming** section within the *Administration settings* menu to access all theming-related settings.

8.19.1 Customize the appearance of Nextcloud

You can change the following aspects of your instance's appearance:


Theming
Theming makes it possible to easily customize the look and feel of your instance and supported clients. This will be visible for all users.
Name
Nextcloud
Web link
https://nextcloud.com
Slogan
a safe home for all your data
Primary color
#00679e
The primary color is used for highlighting elements like important buttons. It might get slightly adjusted depending on the current color schema.
Background color
#00679e
Instead of a background image you can also configure a plain background color. If you use a background image changing this color will influence the color of the app menu icons.
Logo
Upload
Background and login image
Upload

- **Name** (e.g., ACME Inc. Cloud).
- **Web link** (e.g., https://acme.inc/).
- **Slogan**.
- **Primary color**: Used for important buttons, checkboxes, and folder icons.
- **Background color**: Used when no image is set; header bar icon color is also derived from this.
- **Logo**: Appears in the header and on the login page (default size: 62x34 px).
- **Background and login image**: Sets the page background.



- **Additional legal links**: Legal notice and privacy policy.
- **Custom header logo and favicon**: Optionally override the auto-generated favicon based on your logo.

- **Disable user theming**: Prevent users from changing backgrounds and colors. Toggle this to enforce your custom theme.

8.19.2 Configure theming via CLI

You can also configure theming using the `occ theming:config` command.

Available settings include:

- *name*, *url*, *imprintUrl*, *privacyUrl*, *slogan*, *background_color*, *primary_color*. Example: `occ theming:config name "My Example Cloud"`
- *background*, *logo*, *favicon*, *logoheader*. Example: `occ theming:config logo /tmp/mylogo.png`
- *disable-user-theming* (yes/no). Example: `occ theming:config disable-user-theming yes`

> **Note**
>
> Images must be read from a local file on the Nextcloud server.

To use a color (instead of an image) for the background:

```
occ theming:config background_color "#0082c9"
occ theming:config background backgroundColor
```

8.19.3 Icon theming

Based on your settings, Nextcloud will automatically generate favicons and a header logo using your logo and theme color.

This requires:

- The PHP `imagick` module.
- SVG support for imagick (e.g., `libmagickcore-7.q16-10-extra` on Debian 13 and `libmagickcore-6.q16-7-extra` on Ubuntu 24.04).

> **Tip**
>
> In the advanced options of the theming app, you can set a custom favicon if you do not want to use auto-generated resources or install the above dependencies.

8.19.4 Branded clients

> **Note**
>
> Nextcloud GmbH - the company that employs Nextcloud's core maintainers - offers branding services, providing sync clients (mobile and desktop) that use your corporate identity and are pre-configured for your users. For more information on advanced branding and enterprise support offerings, contact Nextcloud GmbH.

The theming app supports changing the URLs for the mobile apps (Android and iOS) that appear when users access the web interface from mobile devices. By default, these links point to the official Nextcloud apps, but you can set them to branded versions.

Set custom app links using the `occ` command:

```
occ config:app:set theming AndroidClientUrl --value "https://play.google.com/store/
↪apps/details?id=com.nextcloud.client"
occ config:app:set theming iTunesAppId --value "1125420102"
occ config:app:set theming iOSClientUrl --value "https://itunes.apple.com/us/app/
↪nextcloud/id1125420102?mt=8"
```

CHAPTER

NINE

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.

9.1 Running occ

You must run `occ` as your HTTP user so that the file ownership and permissions on your Nextcloud data directory stay consistent with the web server.

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.

Note

APCu is disabled by default for the command-line mode of PHP, which can cause issues with Nextcloud's `occ` command. Please make sure you set the `apc.enable_cli` parameter to `1` in your PHP CLI's `php.ini` configuration file or append `--define apc.enable_cli=1` each time you invoke `occ` - e.g.:

```
sudo -u www-data php --define apc.enable_cli=1 occ config:list system
```

If you fail to do this, you will receive output such as the following:

```
An unhandled exception has been thrown:
OCP\HintException: [0]: Memcache \OC\Memcache\APCu not available for local cache␣
↪(Is the matching PHP module installed and enabled?)
```

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

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

Note

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

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

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

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

Usage:
 command [options] [arguments]

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

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

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

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

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

Display your Nextcloud version:

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

Query your Nextcloud server status:

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

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

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

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

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

or `json_pretty`:

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

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

9.1.1 Environment variables

`sudo` does not forward environment variables by default. You can prepend the variable and use the `-E` switch to pass it through:

```
NC_debug=true sudo -E -u www-data php occ status
```

Alternatively, `export` the variable first:

```
export NC_debug=true
sudo -E -u www-data php occ status
```

9.2 Enabling autocompletion

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

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

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

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

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

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

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

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

9.3 Limitations in maintenance mode

In maintenance mode, apps are not loaded[1], so commands from apps are unavailable. Commands integrated into Nextcloud server are available in maintenance mode.

We discourage the use of maintenance mode unless the command explicitly prompts you to do so or unless the commands' documentation explicitly states that maintenance mode should be used.

A command may use events to communicate with other apps. An app can only react to an event when loaded. Example: The command user:delete deletes a user account. UserDeletedEvent is emitted. Calendar app implements an event listener to delete user data[2]. In maintenance mode, the Calendar app is not loaded, and hence the user data not deleted.

[1] Exception: The settings app is loaded
[2] Calendar app event listener for UserDeletedEvent

9.4 Debugging

Every `occ` command accepts the standard Symfony Console verbosity flags:

- `-v` — normal output (errors, warnings, and key messages)
- `-vv` — verbose output (additional progress detail)
- `-vvv` — debug output (full trace, useful for diagnosing command failures)

Example:

```
sudo -E -u www-data php occ files:scan --all -vv
```

To also enable debug-level log output from Nextcloud itself, set the `NC_loglevel` environment variable:

```
NC_loglevel=0 sudo -E -u www-data php occ status
```

See *Logging* for more on log levels.

9.5 Command reference

9.5.1 Apps, background jobs & config commands

Apps commands

The `app` commands list, enable, and disable apps:

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

Download and install an app:

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

Install but don't enable:

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

Install regardless of the Nextcloud version requirement:

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

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

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

List all of your installed and enabled (flag `--enabled`) or disabled (flag `--disabled`) apps:

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

List non-shipped installed apps only:

```
sudo -E -u www-data php occ app:list --shipped false
```

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

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

Enable an app regardless of the Nextcloud version requirement:

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

Enable an app for specific groups of users (i.e. restrict an app so only specific groups can see and use them):

```
sudo -E -u www-data php occ app:enable --groups admin --groups sales files_external
files_external enabled for groups: admin, sales
```

Enable multiple apps simultaneously:

```
sudo -E -u www-data php occ app:enable app1 app2 app3
app1 enabled
app2 enabled
app3 enabled
```

Disable an app:

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

Disable and remove an app:

```
sudo -E -u www-data php occ app:remove files_external
files_external disabled
files_external 1.21.0 removed
```

Remove an app, but keep the app data:

```
sudo -E -u www-data php occ app:remove --keep-data files_external
files_external 1.21.0 removed
```

You can get the full filepath to an app:

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

To update an app, for instance Contacts:

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

To update all apps:

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

To show available update(s) without updating:

```
sudo -E -u www-data php occ app:update --showonly
```

To update an app to an unstable release, for instance News:

```
sudo -E -u www-data php occ app:update --allow-unstable news
```

Background jobs selector

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

```
background
 background:cron       Set background jobs to cron mode
 background:ajax       Set background jobs to ajax mode
 background:webcron    Set background jobs to webcron mode
```

Example:

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

The other two commands are:

- `background:ajax`
- `background:webcron`

See *Background jobs* to learn more.

Config commands

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

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

While setting a configuration value, multiple options are available:

- `--value=VALUE` change the configuration value
- `--type=TYPE` change the type of the value. Use carefully; can break your instance
- `--lazy|--no-lazy` set value as *lazy*
- `--sensitive|--no-sensitive` set value as *sensitive*
- `--update-only` only updates if a value is already stored

> **Note**
>
> See Appconfig Concepts to learn more about *typed value*, *lazy* and *sensitive* flag.

You can list all configuration values with one command:

```
sudo -E -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 export of all configuration values the `--private` flag needs to be set:

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

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

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

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

Getting a single configuration value

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

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

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

sudo -E -u www-data php occ config:app:set files_sharing
incoming_server2server_share_enabled --value="yes"
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 -E -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`
- `float`
- `integer`
- `json`
- `null`
- `string` (default)

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

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

Setting an array configuration value

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

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

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

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

Alternatively, you can set the entry array at once by using the `json` type:

```
sudo -E -u www-data php occ config:system:set trusted_domains --type json --value '[
↪"nextcloud.local","example.com"]'
System config value trusted_domains set to json ["nextcloud.local","example.com"]

sudo -E -u www-data php occ config:system:get trusted_domains
```

(continues on next page)

(continued from previous page)

```
nextcloud.local
example.com
```

Setting a hierarchical configuration value

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

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

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

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

Alternatively, you can set the entry configuration at once by using the `json` type:

```
sudo -E -u www-data php occ config:system:set redis --type json --value '{"host":"/
↪var/run/redis/redis.sock","port":0,"dbindex":0,"password":"secret","timeout":1.5}'
```

Deleting a single configuration value

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

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

sudo -E -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 -E -u www-data php occ config:system:delete doesnotexist
--error-if-not-exists
System config value doesnotexist could not be deleted because it did not exist
```

9.5.2 DAV & database commands

DAV commands

The `dav` commands manage addressbooks, calendars, calendar subscriptions, absences, and related data:

```
dav
 dav:absence:get                 get the out-of-office absence settings for a user
 dav:absence:set                 set the out-of-office absence settings for a user
 dav:clear-calendar-unshares     clear calendar unshares for a user
 dav:clear-contacts-photo-cache  clear cached contact photos
 dav:create-addressbook          create a dav addressbook
 dav:create-calendar             create a dav calendar
 dav:create-subscription         create a dav subscription
 dav:delete-calendar             delete a dav calendar
 dav:delete-subscription         delete a calendar subscription for a user
 dav:fix-missing-caldav-changes  insert missing calendarchanges rows for existing␣
↪events
 dav:list-addressbooks           list all addressbooks of a user
 dav:list-calendar-shares        list all calendar shares for a user
 dav:list-calendars              list all calendars of a user
 dav:list-subscriptions          list all calendar subscriptions for a user
 dav:move-calendar               move a calendar from one user to another
 dav:remove-invalid-shares       remove invalid dav shares
 dav:retention:clean-up          delete CalDAV trash elements that are due for removal
 dav:send-event-reminders        send event reminder notifications
 dav:sync-birthday-calendar      synchronize the birthday calendar
 dav:sync-system-addressbook     synchronize users to the system addressbook
calendar
 calendar:export                 export a calendar of a user
 calendar:import                 import a calendar to a user
```

Manage addressbooks

dav:list-addressbooks

List all addressbooks for a user:

```
sudo -E -u www-data php occ dav:list-addressbooks layla
+-------------+----------+-----------------------------+-------------------+----------
↪+
| Database ID | URI      | Owner principal             | Owner displayname | Writable␣
↪|
+-------------+----------+-----------------------------+-------------------+----------
↪+
| contacts    | Contacts | principals/users/layla      | Layla Smith       | ✓       ␣
↪|
+-------------+----------+-----------------------------+-------------------+----------
↪+
```

dav:create-addressbook

Create an addressbook for a user:

```
sudo -E -u www-data php occ dav:create-addressbook layla work
```

Manage calendars

dav:list-calendars

List all calendars for a user:

```
sudo -E -u www-data php occ dav:list-calendars layla
+----------+-------------+------------------------------+-------------------+----------
↪+
| URI      | Displayname | Owner principal              | Owner displayname | Writable␣
↪|
+----------+-------------+------------------------------+-------------------+----------
↪+
| personal | Personal    | principals/users/layla       | Layla Smith       | ✓       ␣
↪|
+----------+-------------+------------------------------+-------------------+----------
↪+
```

dav:list-calendar-shares

List all calendar shares for a user:

```
sudo -E -u www-data php occ dav:list-calendar-shares layla
  User layla has no calendar shares
```

Use `--calendar-id` to filter results to a specific calendar.

dav:create-calendar

Create a calendar for a user:

```
sudo -E -u www-data php occ dav:create-calendar layla holidays
```

dav:delete-calendar

Delete a named calendar for a user:

```
sudo -E -u www-data php occ dav:delete-calendar layla holidays
```

Use `--birthday` to delete the birthday calendar instead of specifying a name. Use `-f` / `--force` to delete immediately instead of moving to the trashbin:

```
sudo -E -u www-data php occ dav:delete-calendar --force layla holidays
sudo -E -u www-data php occ dav:delete-calendar --birthday layla
```

dav:move-calendar

> **Note**
>
> Moving a calendar changes its existing share URLs.

Move a calendar from one user to another:

```
sudo -E -u www-data php occ dav:move-calendar personal layla fred
```

Without `--force`, the command fails if the destination user already has a calendar with the same name, or if the calendar is shared with a group the destination user is not a member of.

Use `-f` / `--force` to proceed anyway: conflicting group shares are dropped, and if the calendar name is already taken at the destination a new name is tried automatically (`personal-1`, `personal-2`, up to ten attempts). The command fails if no free name is found within those attempts.

dav:clear-calendar-unshares

Clear stale calendar unshare records for a user. Useful when a user's calendar sharing state has become inconsistent:

```
sudo -E -u www-data php occ dav:clear-calendar-unshares layla
  User layla has no calendar unshares
```

Manage calendar subscriptions

dav:list-subscriptions

List all calendar subscriptions for a user:

```
sudo -E -u www-data php occ dav:list-subscriptions layla
  User layla has no subscriptions
```

dav:create-subscription

Create a subscription calendar for a user:

```
sudo -E -u www-data php occ dav:create-subscription layla "Astronomy Calendar" \
  webcal://cantonbecker.com/astronomy-calendar/astrocal.ics
```

Optionally pass a HEX color code for the calendar:

```
sudo -E -u www-data php occ dav:create-subscription layla "Astronomy Calendar" \
  webcal://cantonbecker.com/astronomy-calendar/astrocal.ics "#ff5733"
```

If not set, the theming default color is used.

dav:delete-subscription

Delete a calendar subscription for a user:

```
sudo -E -u www-data php occ dav:delete-subscription layla "Astronomy Calendar"
```

Manage absences (out-of-office)

dav:absence:set

Set the out-of-office absence for a user. `first-day` and `last-day` are inclusive and formatted as `YYYY-MM-DD`.

- **short-message** — a brief one-liner displayed as the user's status next to their name across the UI while they are absent (e.g. in contacts, chat, and user lists).
- **message** — the full out-of-office text shown to users who try to reach the absent user.
- **replacement-user-id** — optional; a colleague to contact instead. Their display name is stored alongside the absence and shown to others.

```
sudo -E -u www-data php occ dav:absence:set layla 2026-05-01 2026-05-15 \
  "On leave" \
  "I am on leave until May 15. For urgent matters please contact my colleague."
```

To set an absence with a replacement user:

```
sudo -E -u www-data php occ dav:absence:set layla 2026-05-01 2026-05-15 \
  "On leave" \
  "I am on leave. Please contact fred for urgent matters." \
  fred
```

dav:absence:get

Show the current absence settings for a user:

```
sudo -E -u www-data php occ dav:absence:get layla
  Start day: 2026-05-01
  End day: 2026-05-15
  Short message: On leave
  Message: I am on leave until May 15. For urgent matters please contact my colleague.
  Replacement user: none
  Replacement display name: none
```

Sync

dav:sync-system-addressbook

Synchronize all users to the *system address book*:

```
sudo -E -u www-data php occ dav:sync-system-addressbook
  Syncing users ...
```

dav:sync-birthday-calendar

Add birthdays from shared addressbooks to a user's calendar:

```
sudo -E -u www-data php occ dav:sync-birthday-calendar layla
  Start birthday calendar sync for layla
```

Maintenance

dav:fix-missing-caldav-changes

Restore missing calendar sync change records. If the `calendarchanges` table has lost data, clients may not sync correctly. Run for a single user, or omit the user ID to fix all users (may take some time on large instances):

```
sudo -E -u www-data php occ dav:fix-missing-caldav-changes layla
sudo -E -u www-data php occ dav:fix-missing-caldav-changes
```

dav:remove-invalid-shares

Remove invalid DAV shares created by a bug in a previous version:

```
sudo -E -u www-data php occ dav:remove-invalid-shares
```

dav:retention:clean-up

Delete elements from the CalDAV trash that are past their retention period:

```
sudo -E -u www-data php occ dav:retention:clean-up
```

dav:send-event-reminders

Send pending calendar event reminder notifications. By default, Nextcloud sends reminders via the background job. To use this command instead, switch the mode first:

```
sudo -E -u www-data php occ config:app:set dav sendEventRemindersMode --value occ
```

Then call the command regularly via a dedicated cron job to ensure reminders are sent on time:

```
sudo -E -u www-data php occ dav:send-event-reminders
```

See *Calendar / CalDAV* for the recommended cron schedule.

dav:clear-contacts-photo-cache

Clear all cached contact photos. Useful after a migration or if contact avatars are not displaying correctly:

```
sudo -E -u www-data php occ dav:clear-contacts-photo-cache
  No cached contact photos found.
```

Export and import calendars

calendar:export

Export a calendar to a file or standard output:

```
sudo -E -u www-data php occ calendar:export layla personal --location /tmp/personal.
↪ics
```

`--format` selects the output format (default: `ical`):

- `ical` — iCalendar (RFC 5545)

- `xcal` — XML iCalendar (RFC 6321)
- `jcal` — JSON iCalendar (RFC 7265)

`--location` sets the output file path. If omitted, output is written to standard output:

```
sudo -E -u www-data php occ calendar:export layla personal --format xcal \
  --location /tmp/personal.xcal
sudo -E -u www-data php occ calendar:export layla personal --format jcal
```

calendar:import

Import calendar entries into a user's calendar:

```
sudo -E -u www-data php occ calendar:import layla personal /tmp/personal.ics
```

`--format` selects the input format (default: `ical`). Same values as `calendar:export`. If `location` is omitted, reads from standard input:

```
sudo -E -u www-data php occ calendar:import --format xcal layla personal \
  /tmp/personal.xcal
cat /tmp/personal.ics | sudo -E -u www-data php occ calendar:import layla personal
```

`--supersede` forces override of existing objects with the same UID.

Validation behaviour is controlled with `--validation`:

- `0` — no validation
- `1` — validate, skip invalid entries
- `2` — validate, fail on invalid entries

Error handling is controlled with `--errors`:

- `0` — continue on error (default)
- `1` — fail on first error

Use `--show-created`, `--show-updated`, `--show-skipped`, or `--show-errors` to see the UIDs of affected objects after import.

Disable creation of example events

Disable the automatic creation of example events for new calendar users:

```
sudo -E -u www-data php occ config:app:set dav create_example_event \
  --value=false --type=boolean
```

Database commands

The `db` commands manage database schema, indices, and data conversions:

```
db
 db:add-missing-columns          add missing optional columns to the database tables
 db:add-missing-indices          add missing indices to the database tables
 db:add-missing-primary-keys     add missing primary keys to the database tables
 db:convert-filecache-bigint     convert the ID columns of the filecache to BigInt
```

(continues on next page)

(continued from previous page)

```
 db:convert-mysql-charset        convert charset of MySQL/MariaDB to utf8mb4
 db:convert-type                 convert the Nextcloud database to a different type
 db:schema:check                 compare the live database schema with the expected␣
↪schema
 db:schema:expected              export the expected database schema for a fresh␣
↪installation
 db:schema:export                export the current database schema
```

Schema maintenance

db:add-missing-indices

Add any indices to the database that are defined in the schema but missing from the current installation. Run this after upgrades if performance has degraded or if prompted by the admin overview:

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

Use `--dry-run` to preview the SQL statements without executing them:

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

db:add-missing-columns

Add optional columns that are defined in the schema but missing from the current installation:

```
sudo -E -u www-data php occ db:add-missing-columns
  Done.
```

Use `--dry-run` to preview the SQL statements without executing them.

db:add-missing-primary-keys

Add primary keys that are defined in the schema but missing from the current installation:

```
sudo -E -u www-data php occ db:add-missing-primary-keys
  Done.
```

Use `--dry-run` to preview the SQL statements without executing them.

Schema inspection

db:schema:check

Added in version 35.0.

Compare the live database schema with the schema expected for the currently installed Nextcloud version:

```
sudo -E -u www-data php occ db:schema:check
  The live database schema matches the expected schema.
```

The expected schema is generated from the migrations provided by Nextcloud core, all enabled apps, and any disabled apps whose code is still present. The command reports missing or unexpected tables, columns, and indexes. It also reports column properties that differ, such as the type, length, default value, or whether the column is nullable.

The command only inspects the schema and does not modify the database. It exits with status `0` if there are no findings for core or an enabled app, and status `1` otherwise. This makes it suitable for use in monitoring scripts. Findings that only concern a disabled app never affect the exit status. They are excluded from the plain output unless you add `-v`:

```
sudo -E -u www-data php occ db:schema:check -v
  oc_example: column 'size' differs in: type, default
  Disabled apps (not affecting exit code):
  If the schema for a disabled app differs from what is expected, this might indicate␣
↪the app was
  updated since it was disabled. Missing migrations will be applied once the app is␣
↪enabled again.
    files_external:
      - oc_storages: missing column 'backend'
```

Specify a table name to limit the comparison to that table:

```
sudo -E -u www-data php occ db:schema:check oc_filecache
```

Use the complete table name, including the configured database table prefix. Use `--output=json` or `--output=json_pretty` to produce machine-readable output. Unlike plain output, JSON output always includes every finding, including those for disabled apps. For example:

```
$ sudo -E -u www-data php occ db:schema:check --output=json_pretty
[
    {
        "table": "oc_filecache",
        "type": "modified_column",
        "name": "size",
        "changes": [
            "type",
            "default"
        ],
        "app": null,
        "enabled": true
    },
    {
        "table": "oc_storages",
        "type": "missing_column",
        "name": "backend",
        "app": "files_external",
        "enabled": false
    },
    {
        "table": "oc_example",
        "type": "unexpected_table",
        "app": null,
        "enabled": true
    }
]
```

Each finding contains:

- `table` — the complete database table name.
- `type` — one of `missing_table`, `unexpected_table`, `missing_column`, `unexpected_column`, `modified_column`, `missing_index`, or `unexpected_index`.

- `name` — the affected column or index. This field is omitted for table findings.
- `changes` — the changed column properties. This field is only present when `type` is `modified_column`. Possible values include `type`, `length`, `precision`, `scale`, `nullable`, `default`, `autoincrement`, `unsigned`, `fixed`, and `comment`.
- `app` — the ID of the disabled app that owns the affected table, or `null` if the table belongs to core or an enabled app.
- `enabled` — `false` if the finding only concerns a disabled app and does not affect the exit status, `true` otherwise.

When the schemas match, JSON output contains an empty array.

Note

The expected schema does not account for all database-wide settings or defaults, so some reported differences may be valid. A table or other object still counts as unexpected if it belongs to an app that was disabled and then fully removed. Review each finding before changing the database.

The schema check also runs automatically after `occ upgrade`. Findings for core or an enabled app are displayed as warnings but do not cause the upgrade to fail; findings that only concern a disabled app are not shown here. For example, the following output may appear near the end of an upgrade:

```
The database schema does not match what is expected for the installed version:
  - oc_filecache: column 'size' differs in: type, default
  - oc_filecache: missing index 'fs_storage_path_hash'
  - unexpected table 'oc_example'
Run "occ db:schema:check -v" for details.
```

Run `occ db:schema:check` separately to inspect the differences in plain (add `-v` to include disabled-app findings) or machine-readable format.

db:schema:export

Export the current database schema. Useful for debugging or comparing against the expected schema:

```
sudo -E -u www-data php occ db:schema:export
  - oc_filecache:
    - columns:
      - checksum:
        - name: checksum
        - type: string
```

Limit output to a single table:

```
sudo -E -u www-data php occ db:schema:export oc_filecache
```

Use `--sql` to output SQL `CREATE TABLE` statements instead of the default structured format. Use `--output=json_pretty` for machine-readable output.

db:schema:expected

Export the expected schema for a fresh installation of the current version. Compare this against `db:schema:export` to identify divergences caused by incomplete upgrades or manual changes:

```
sudo -E -u www-data php occ db:schema:expected
sudo -E -u www-data php occ db:schema:expected oc_filecache
```

Supports the same `--sql` and `--output` options as `db:schema:export`.

Data conversions

db:convert-filecache-bigint

Convert the `filecache` table's ID columns from integer to BigInt. Required on large installations where file IDs have exceeded the integer limit. This command is non-destructive but may take time on large databases:

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

db:convert-mysql-charset

Convert all MySQL or MariaDB tables to use `utf8mb4` (full Unicode, including emoji). Required for proper Unicode support. Run this once after switching to or upgrading MySQL/MariaDB:

```
sudo -E -u www-data php occ db:convert-mysql-charset
```

db:convert-type

Convert the Nextcloud database from SQLite to MySQL, MariaDB, or PostgreSQL. SQLite is suitable for testing and single-user setups, but production servers with multiple users should use one of the other supported databases.

Requirements:

- The target database and its PHP connector must be installed.
- Login credentials for a database admin user.
- The database port number, if non-standard.

This example converts from SQLite to MySQL/MariaDB:

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

For a detailed walkthrough see *Convert to a different database type*.

9.5.3 Encryption commands

The `encryption` commands manage server-side encryption, encryption keys, and encryption modules. The core commands are always available. Commands that operate on the encryption module itself (master key, key repair, legacy migration) require the **Encryption** app (`encryption`) to be enabled.

> **Note**
>
> For a full guide to configuring server-side encryption, see *Server-side Encryption*.

```
encryption
 encryption:change-key-storage-root      change key storage root
 encryption:clean-orphaned-keys          scan the keys storage for orphaned keys and␣
```

(continues on next page)

(continued from previous page)

```
↪remove them
 encryption:decrypt-all                 disable server-side encryption and decrypt␣
↪all files
 encryption:disable                     disable encryption
 encryption:disable-master-key          disable the master key and use per-user keys␣
↪instead
 encryption:drop-legacy-filekey         scan the files for the legacy filekey format␣
↪using RC4 and get rid of it
 encryption:enable                      enable encryption
 encryption:enable-master-key           enable the master key
 encryption:encrypt-all                 encrypt all files for all users
 encryption:fix-encrypted-version       fix the encrypted version if the encrypted␣
↪file(s) are not downloadable
 encryption:fix-key-location            fix the location of encryption keys for␣
↪external storage
 encryption:list-modules                list all available encryption modules
 encryption:migrate-key-storage-format  migrate the format of the key storage to a␣
↪newer format
 encryption:recover-user                recover user data in case of password loss
 encryption:scan:legacy-format          scan the files for the legacy format
 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
```

Status and control

encryption:status

Show whether encryption is enabled and which module is active:

```
sudo -E -u www-data php occ encryption:status
  - enabled: false
  - defaultModule: OC_DEFAULT_MODULE
```

Use `--output=json_pretty` for machine-readable output.

encryption:enable

Enable server-side encryption. The **Encryption** app must be enabled first and a default module must be configured:

```
sudo -E -u www-data php occ app:enable encryption
sudo -E -u www-data php occ encryption:enable
  Encryption enabled

  Default module: OC_DEFAULT_MODULE
```

encryption:disable

Disable server-side encryption. This only disables the encryption flag — it does not decrypt any files:

```
sudo -E -u www-data php occ encryption:disable
  Encryption disabled
```

To also decrypt all existing files, run `encryption:decrypt-all` afterwards.

Encrypt and decrypt all data

encryption:encrypt-all

Encrypt all files for all users. Encryption must be enabled before running this command. The command manages maintenance mode internally — do **not** enable maintenance mode first, as the command will fail if it is already active:

```
sudo -E -u www-data php occ encryption:encrypt-all
  You are about to encrypt all files stored in your Nextcloud installation.
  Depending on the number of available files, and their size, this may take quite␣
↪some time.
  Please ensure that no user accesses their files during this time!
  Note: The encryption module you use determines which files get encrypted.

  Do you really want to continue? (y/n)
```

The command requires an interactive terminal (TTY). If running inside a Docker container, use `docker exec -it`.

encryption:decrypt-all

Decrypt all files for all users, or for a single user. The command manages maintenance mode internally — do **not** enable maintenance mode first:

```
sudo -E -u www-data php occ encryption:decrypt-all
sudo -E -u www-data php occ encryption:decrypt-all layla
```

When decrypting all users, server-side encryption is disabled automatically. When decrypting a single user, encryption remains enabled for others.

Depending on the encryption module in use, decryption may require:

- **Master key mode** — no extra credentials needed.
- **Per-user key mode** — the user must have enabled the recovery key on their personal settings page.

The command requires an interactive terminal (TTY). If running inside a Docker container, use `docker exec -it`.

Encryption modules

encryption:list-modules

List all available encryption modules. The active default module is marked with `[default*]`:

```
sudo -E -u www-data php occ encryption:list-modules
  - OC_DEFAULT_MODULE: Default encryption module [default*]
```

encryption:set-default-module

Set the default encryption module. Maintenance mode must be disabled:

```
sudo -E -u www-data php occ encryption:set-default-module OC_DEFAULT_MODULE
  Set default module to "OC_DEFAULT_MODULE"
```

Key storage

encryption:show-key-storage-root

Show where encryption keys are stored:

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

encryption:change-key-storage-root

Move encryption keys to a different directory. The target directory must exist and be writable by the web server user before running the command:

```
sudo -E -u www-data php occ encryption:change-key-storage-root /etc/nextcloud/keys
  Change key storage root from default storage location to /etc/nextcloud/keys
  Start to move keys:
  [============================]
  Key storage root successfully changed to /etc/nextcloud/keys
```

Omit the argument to reset the key storage root back to the default location (`data/`). The command will ask for confirmation:

```
sudo -E -u www-data php occ encryption:change-key-storage-root
  No storage root given, do you want to reset the key storage root to the default␣
↪location? (y/n)
```

encryption:migrate-key-storage-format

Migrate the key storage to the current format (JSON-wrapped, re-encrypted with the server secret). Run this once after upgrading from an installation that used the legacy key format:

```
sudo -E -u www-data php occ encryption:migrate-key-storage-format
  Updating key storage format
  Start to update the keys:
  [============================]
  Key storage format successfully updated
```

Master key

The master key encrypts all user data with a single server-managed key. It is the recommended setup for new installations because it simplifies key management and enables admin-side decryption without per-user passwords. The master key is enabled by default when the Encryption app is first enabled.

Both commands are only available when the **Encryption** app is enabled.

> **Warning**
>
> Switching between master key and per-user key mode is only safe on a **fresh installation with no existing encrypted data**. If you switch modes on an instance that already has encrypted files, those files will become permanently inaccessible: the new mode looks for keys that were never created for the existing data. **There is no recovery path.** Always run `encryption:decrypt-all` first if you need to change modes on an existing installation.

> **Note**
>
> Despite the warning shown by both commands ("no way to enable/disable it again"), the switch is technically reversible as long as no encrypted data exists yet. The warning is intended to convey that you cannot safely switch modes once users have data.

encryption:enable-master-key

Enable the master key. Only use this on a **fresh installation with no existing encrypted data**. The command prompts for confirmation:

```
sudo -E -u www-data php occ encryption:enable-master-key
  Master key successfully enabled.
```

encryption:disable-master-key

Disable the master key and revert to per-user keys. Only use this on a **fresh installation with no existing encrypted data**. The command prompts for confirmation:

```
sudo -E -u www-data php occ encryption:disable-master-key
  Master key successfully disabled.
```

Switching to per-user keys has the following consequences:

- **Performance** — per-user key operations are slower. Each user's key must be derived individually on every file access.
- **Password loss means data loss** — without the master key there is no admin-side decryption path. Each user must enable the recovery key on their personal settings page *before* losing their password. Without it, `encryption:recover-user` cannot help and the files cannot be recovered.
- **Admin decryption is no longer possible** — admins cannot decrypt or access a user's files without that user's password or a pre-set recovery key.

Maintenance and repair

encryption:clean-orphaned-keys

Scan the key storage for keys that no longer have a corresponding file and remove them. Optionally limit the scan to a single user:

```
sudo -E -u www-data php occ encryption:clean-orphaned-keys
sudo -E -u www-data php occ encryption:clean-orphaned-keys layla
```

When orphaned keys are found, the command lists them and asks interactively whether to delete all, specific ones, or none:

```
sudo -E -u www-data php occ encryption:clean-orphaned-keys layla
  Key storage scan for layla
  ==========================

  Orphaned key found: /layla/files_encryption/keys/files/old-doc.pdf/OC_DEFAULT_
↪MODULE/fileKey
  Do you want to delete all orphaned keys? (y/n)
```

If no orphaned keys are found, the command exits silently.

encryption:fix-encrypted-version

Fix the encrypted version counter in the file cache when encrypted files cannot be downloaded. Requires **master key** encryption.

Run for a single user:

```
sudo -E -u www-data php occ encryption:fix-encrypted-version layla
  Verifying the content of file "/layla/files/Documents/report.pdf"
  The file "/layla/files/Documents/report.pdf" is: OK
```

Run for all users:

```
sudo -E -u www-data php occ encryption:fix-encrypted-version --all
  Processing files for layla
  Verifying the content of file "/layla/files/Documents/report.pdf"
  The file "/layla/files/Documents/report.pdf" is: OK
```

When a broken file is found, the command tries decrementing and then incrementing the encrypted version counter until the file can be read:

```
Attempting to fix the path: "/layla/files/Documents/broken.pdf"
Decrement the encrypted version to 2
Fixed the file: "/layla/files/Documents/broken.pdf" with version 2
```

Use `-p` / `--path` to limit the scan to a specific directory:

```
sudo -E -u www-data php occ encryption:fix-encrypted-version layla --path="/Documents"
```

encryption:fix-key-location

Fix the location of encryption keys for files on external storage mounts. Run this when users cannot access files on external storage after a migration:

```
sudo -E -u www-data php occ encryption:fix-key-location layla
  Migrating key for /layla/files/ExternalDrive/report.pdf ✓
```

Use `--dry-run` to list files that need migration without making any changes:

```
sudo -E -u www-data php occ encryption:fix-key-location --dry-run layla
  /layla/files/ExternalDrive/report.pdf needs migration
```

encryption:recover-user

Recover a user's files after a password loss. Only available in **per-user key** mode (not applicable when master key is enabled). The user must have enabled the recovery key in their personal settings before the password was lost.

The command prompts for the recovery key password and the new login password:

```
sudo -E -u www-data php occ encryption:recover-user layla
  Please enter the recovery key password:
  Please enter the new login password for the user:
  Start to recover users files... This can take some time...Done.
```

Legacy migration

encryption:drop-legacy-filekey

Scan all files for the legacy RC4 filekey format and migrate them to the current format. Requires **master key** encryption. Files not migrated by this command will be migrated automatically on their next write. Running this command upfront also converts old base64-encoded files to binary format, saving approximately 33% disk space:

```
sudo -E -u www-data php occ encryption:drop-legacy-filekey
  Scanning all files for legacy filekey
  Scanning all files for layla
  All scanned files are properly encrypted.
```

encryption:scan:legacy-format

Scan all files for the legacy encryption format. Use this to determine whether any files still need migration before disabling legacy format compatibility:

```
sudo -E -u www-data php occ encryption:scan:legacy-format
  Scanning all files for legacy encryption
  Scanning all files for layla
  All scanned files are properly encrypted. You can disable the legacy compatibility␣
↪mode.
```

If files using the legacy format are found, their paths are listed. Run `encryption:drop-legacy-filekey` to migrate them.

9.5.4 Files commands

- *File operations*
- *Object store*
- *Preview*
- *Trash bin*
- *File versions*
- *File sharing*
- *Federation sync*
- *Files external*
- *Trashbin*
- *Versions*
- *Integrity check*

The `occ` commands in this section cover day-to-day file management, storage maintenance, and sharing administration. They let you scan and repair the file cache, move or delete files on behalf of users, manage trash and version history, configure external storage mounts, inspect object store contents, and verify app integrity from the command line.

Several sections require specific apps to be enabled. All of the following are shipped with Nextcloud and can be enabled in the Apps menu:

- **Trash bin** commands require the **Deleted files** app (`files_trashbin`) — enabled by default
- **File versions** commands require the **Versions** app (`files_versions`) — enabled by default
- **Federation sync** requires the **Federation** app (`federation`) — enabled by default
- **External storage** commands require the **External storage support** app (`files_external`) — not enabled by default

File operations

```
files
 files:cleanup                          remove file cache entries with no matching␣
↪storage entry
 files:copy                             copy a file or folder
 files:delete                           delete a file or folder
 files:get                              get the contents of a file
 files:mount:list                       list all mounts for a user
 files:mount:refresh                    refresh the list of mounts for a user
 files:move                             move a file or folder
 files:put                              write content to a file
 files:mkdir                            create a new folder
 files:touch                            update the modified date of a file or folder
 files:reminders                        list file reminders
 files:repair-tree                      repair malformed filesystem tree structures
 files:sanitize-filenames               rename files that do not match the current␣
↪filename constraints
 files:scan                             rescan the filesystem
 files:scan-app-data                    rescan the AppData folder
 files:transfer-ownership               transfer all files and shares from one user to␣
↪another
 files:windows-compatible-filenames     toggle enforcement of Windows-compatible␣
↪filenames
```

files:cleanup

Remove all file cache entries that have no matching entry in the storage table. Run this after removing orphaned storage records or after a migration that left the file cache in an inconsistent state:

```
sudo -E -u www-data php occ files:cleanup
```

files:copy

Copy a file or folder within Nextcloud. Source and target accept either a Nextcloud path or a numeric file ID:

```
sudo -E -u www-data php occ files:copy /layla/files/Photos \
  /layla/files/Photos-backup
```

If the target path already exists and is a folder, the source is copied into it (standard cp behaviour). Use `--no-target-directory` / `-T` to overwrite the target folder instead:

```
sudo -E -u www-data php occ files:copy -T /layla/files/Photos \
  /layla/files/Photos-backup
```

If the target exists the command asks for confirmation. Use `--force` / `-f` to skip the prompt and suppress type-mismatch warnings.

files:delete

Delete a file or folder:

```
sudo -E -u www-data php occ files:delete /layla/files/Documents/old-draft.pdf
```

The command asks for confirmation before deleting. Use `--force` / `-f` to skip the prompt.

If the path points to the root of a share received by a user, the command asks whether to unshare (remove the user's access) rather than delete the underlying file:

```
sudo -E -u www-data php occ files:delete /layla/files/Shared-Folder
  /layla/files/Shared-Folder in a shared file, do you want to unshare
  the file from layla instead of deleting the source file? [Y/n]
```

When the file is accessible by multiple users, the command lists all affected users and asks for confirmation before proceeding.

If the trashbin is enabled, you can use `--skip-trash` to bypass the trashbin and permanently delete the files directly.

files:get

Download a file from Nextcloud to a local path. The source argument accepts either a Nextcloud path or a numeric file ID:

```
sudo -E -u www-data php occ files:get /layla/files/Documents/report.pdf \
  /tmp/report.pdf
```

Omit the output path (or pass –) to write to standard output:

```
sudo -E -u www-data php occ files:get /layla/files/Documents/report.pdf -
```

Warning

Writing binary files to a terminal can corrupt your terminal session. Nextcloud will refuse to write binary content to STDOUT unless you explicitly pass – as the output path.

files:mount:list

List all registered mounts for a user, including mount point, storage ID, provider, and whether the mount is still actively provided:

```
sudo -E -u www-data php occ files:mount:list layla
/layla/: local::home/nextcloud/data/layla/files
        - provider: OC\Files\Mount\LocalRootMountProvider
        - storage id: 1
        - root id: 1
```

Stale mounts (registered in the cache but no longer provided) are marked with `registered but no longer provided`.

files:mount:refresh

Re-register all currently active mounts for a user in the mount cache. Useful after mount configuration changes that were not automatically picked up:

```
sudo -E -u www-data php occ files:mount:refresh layla
  Registered 3 mounts
```

files:move

Move a file or folder within Nextcloud:

```
sudo -E -u www-data php occ files:move \
  /layla/files/Documents/draft.pdf \
  /layla/files/Documents/Archive/draft.pdf
```

If the target already exists, the command asks for confirmation. Use `--force` / `-f` to skip the prompt. If source and target are different types (file vs. folder), the target is deleted before the move; the command refuses if the target is a mount root or is not deletable.

files:put

Upload a local file to Nextcloud. The target argument accepts either a Nextcloud path or a numeric file ID of an existing file:

```
sudo -E -u www-data php occ files:put /tmp/report.pdf \
  /layla/files/Documents/report.pdf
```

Read from standard input by passing - as the source:

```
cat /tmp/report.pdf | sudo -E -u www-data php occ files:put - \
  /layla/files/Documents/report.pdf
```

Any missing parent folder for the specified destination will be created automatically.

files:mkdir

Create a new empty folder, parent folders will automatically be created if needed:

```
sudo -E -u www-data php occ files:mkdir /layla/files/Documents/Reports
```

files:touch

Update the last modified date of a file or folder. The target argument accepts either a Nextcloud path or a numeric file ID of an existing file:

```
sudo -E -u www-data php occ files:touch  /layla/files/Documents/report.pdf
```

Instead of using the current time, you can specify the modified date to use with the `--date` option. Accepted formats are: ISO8601, "YYYY-MM-DD" and Unix time in seconds.:

```
sudo -E -u www-data php occ files:touch  /layla/files/Documents/report.pdf \
  --date 2026-07-15
```

files:reminders

List all file reminders across all users, or for a specific user:

```
sudo -E -u www-data php occ files:reminders
+-------+---------+--------------------------------------------+----------------------
↪----+
| User  | File Id | Path                                       | Due Date (UTC)       ␣
↪    |
+-------+---------+--------------------------------------------+----------------------
↪----+
| layla | 42      | /layla/files/Documents/report.pdf          | 2026-05-
↪01T09:00:00+00:00 |
+-------+---------+--------------------------------------------+----------------------
↪----+

sudo -E -u www-data php occ files:reminders layla
```

Use `--output=json_pretty` for machine-readable output.

files:repair-tree

Repair entries in the file cache whose stored path does not match the path derived from their parent. This can cause files to appear in the wrong location: listing a folder shows the file, but accessing it by path fails. The command resets each affected entry to its correct path:

```
sudo -E -u www-data php occ files:repair-tree
```

files:sanitize-filenames

Rename files that do not comply with the current filename constraints (for example, after enabling *Windows-compatible filenames*). Run for all users or a specific user:

```
sudo -E -u www-data php occ files:sanitize-filenames
sudo -E -u www-data php occ files:sanitize-filenames layla
```

Invalid characters are replaced with a space, underscore, or dash — whichever is allowed by the current constraints. Use `--char-replacement` to specify the replacement character when none of those defaults are permitted:

```
sudo -E -u www-data php occ files:sanitize-filenames \
  --char-replacement=_ layla
```

Use `--dry-run` to preview the renames without making changes:

```
sudo -E -u www-data php occ files:sanitize-filenames --dry-run
```

files:scan

Scan for new or changed files and update the file cache. Run for all users, a specific user, or a specific path:

```
sudo -E -u www-data php occ files:scan --all
sudo -E -u www-data php occ files:scan layla
sudo -E -u www-data php occ files:scan layla fred
```

Use `--path` to limit the scan to a specific directory. The path must include the user ID and `files/`:

```
sudo -E -u www-data php occ files:scan --path="/layla/files/Photos"
```

The `--path`, `--all`, and `[user_id]` options are mutually exclusive; only one may be given at a time.

Additional options:

- `--generate-metadata` — generate metadata (e.g. EXIF data) for scanned files
- `--unscanned` — only scan files marked as not yet fully scanned (useful when triggering the scan that the background job would otherwise run)
- `--shallow` — do not scan folders recursively
- `--home-only` — skip external storages and shares, scan only the home storage
- `--quiet` — suppress output; without this option, statistics are shown after the scan
- `--output=json_pretty` — machine-readable output

Use `-v` to see each file as it is processed. Verbosity levels `-vv` and `-vvv` are silently capped to `-v`.

A background job runs every 10 minutes to scan files with an unknown size in the file cache, so routine rescans are handled automatically. Use this command when you need an immediate rescan — for example after copying files directly into the data directory, after a migration, or to investigate file cache inconsistencies. The background scan can be disabled by setting `'files_no_background_scan' => true` in `config.php`.

files:scan-app-data

Rescan the `appdata` directory and update the file cache for files shared between users (avatar images, file previews, cached CSS, etc.):

```
sudo -E -u www-data php occ files:scan-app-data
```

Limit the scan to a specific appdata subdirectory:

```
sudo -E -u www-data php occ files:scan-app-data preview
```

Unlike `files:scan`, there is no background job that automatically rescans appdata. Run this command manually when the appdata file cache may have become inconsistent; for example, after restoring appdata from a backup, after manually moving or copying files into the appdata directory, or when previews or avatars are not displaying correctly despite the underlying files being present on disk.

files:transfer-ownership

Transfer all files and shares from one user to another. Useful before removing a user account:

```
sudo -E -u www-data php occ files:transfer-ownership layla fred
```

The transferred files appear in a subdirectory inside the destination user's home folder.

> **Note**
>
> Unless server-side encryption is enabled, the command initialises the destination user's file system if they have not yet logged in. If the destination user's data directory cannot be written to, the command reports: `unable to rename, destination directory is not writable`.

If the destination user's home is empty, use `--move` to move files directly into the root without creating a subdirectory:

```
sudo -E -u www-data php occ files:transfer-ownership --move layla fred
```

Transfer only a specific folder with `--path`:

```
sudo -E -u www-data php occ files:transfer-ownership \
  --path="Documents/Project" layla fred
```

Incoming shares (files shared with the source user) are not transferred by default because ownership remains with the original sharer. Transfer them with `--transfer-incoming-shares=1`:

```
sudo -E -u www-data php occ files:transfer-ownership \
  --transfer-incoming-shares=1 layla fred
```

Set `'transferIncomingShares' => true` in `config.php` to always transfer incoming shares. The command-line option takes precedence:

```
sudo -E -u www-data php occ files:transfer-ownership \
  --transfer-incoming-shares=0 layla fred
```

Users may also transfer files selectively via the web interface. See user documentation for details.

files:windows-compatible-filenames

Toggle enforcement of *Windows-compatible filenames*:

```
sudo -E -u www-data php occ files:windows-compatible-filenames --enable
sudo -E -u www-data php occ files:windows-compatible-filenames --disable
```

After enabling, run `files:sanitize-filenames` to rename any existing files that do not comply.

Object store

```
files
 files:object:delete                  delete an object from the object store
 files:object:get                     get the contents of an object
 files:object:info                    get the metadata of an object
 files:object:list                    list all objects in the object store
 files:object:orphans                 list objects in the object store with no␣
↪matching database entry
 files:object:put                     write a file to the object store
```

These commands operate directly on the underlying object storage (S3, Swift, etc.) used as Nextcloud's primary storage. They bypass the file cache and normal access controls.

Warning

These commands manipulate objects directly in the object store. Writing or deleting objects that belong to existing files will corrupt those files. Use them only for debugging or recovery, never for routine file management.

files:object:list

List all objects in the object store. Requires the configured object store to support metadata listing:

```
sudo -E -u www-data php occ files:object:list
+------------+--------+-------------------------------+
| URN        | Size   | Last modified                 |
+------------+--------+-------------------------------+
| urn:oid:1  | 512    | 2026-01-01T00:00:00+00:00     |
| urn:oid:42 | 204800 | 2026-04-01T12:00:00+00:00     |
+------------+--------+-------------------------------+
```

Use `--bucket` / `-b` if the bucket cannot be determined from the config. Use `--output=json_pretty` for machine-readable output.

files:object:get

Download an object from the object store to a local file. Pass `-` as the output path to write to standard output:

```
sudo -E -u www-data php occ files:object:get urn:oid:42 /tmp/recovered.pdf
sudo -E -u www-data php occ files:object:get urn:oid:42 -
```

files:object:info

Show metadata (size, MIME type, last-modified date) for an object:

```
sudo -E -u www-data php occ files:object:info urn:oid:42
  - size: 200 KB
  - mimetype: application/pdf
  - mtime: 2026-04-01T12:00:00+00:00
```

Use `--output=json_pretty` for machine-readable output.

files:object:put

Upload a local file to the object store under the given object name. Read from standard input by passing `-`:

```
sudo -E -u www-data php occ files:object:put /tmp/data.bin urn:oid:42
sudo -E -u www-data php occ files:object:put - urn:oid:42
```

If the object already corresponds to a file in the database, the command warns and asks for confirmation before overwriting. To update a file safely, use `files:put` with the file ID instead.

files:object:delete

Delete an object from the object store:

```
sudo -E -u www-data php occ files:object:delete urn:oid:42
```

If the object belongs to a file in the database, the command warns that deleting it will corrupt the file and shows the file ID so you can use `files:delete` instead for a clean removal.

files:object:orphans

List objects in the object store that have no matching entry in the file cache database. These are objects that Nextcloud no longer tracks and may be safe to remove after investigation:

```
sudo -E -u www-data php occ files:object:orphans
```

Use `--output=json_pretty` for machine-readable output. Requires the object store to support metadata listing.

Preview

```
preview
 preview:cleanup                      remove all generated preview files
```

preview:cleanup

Remove all generated preview files. Useful after changing preview configuration (sizes, quality, or supported file types), or on installations using object storage as primary storage where the preview folder cannot be deleted manually:

```
sudo -E -u www-data php occ preview:cleanup
  Previews removed
```

After running this command, Nextcloud regenerates previews on demand as users access files.

See *Previews configuration* for preview settings.

Trash bin

```
trashbin
 trashbin:cleanup                     permanently delete all files in the trash for a␣
↪user
 trashbin:expire                      expire trash items that exceed the configured␣
↪retention period
 trashbin:size                        show or set the target trash bin size
```

trashbin:size

Show the current trash bin size target:

```
sudo -E -u www-data php occ trashbin:size
  Default size: default (50% of available space)
```

Show the effective size for a specific user:

```
sudo -E -u www-data php occ trashbin:size --user layla
  default (50% of available space)
```

Set the global default target size. Accepts human-readable sizes:

```
sudo -E -u www-data php occ trashbin:size 10GB
```

> **Note**
>
> Changing the global default immediately triggers a cleanup of existing trash bins. A user's trash may temporarily exceed the configured size until the user next moves a file to trash.

Set a per-user target size:

```
sudo -E -u www-data php occ trashbin:size --user layla 2GB
```

trashbin:cleanup

Permanently delete all files in the trash bin for one or more users, or for all users:

```
sudo -E -u www-data php occ trashbin:cleanup layla
  Remove deleted files of   layla

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

Use `--verbose` to see the amount of data freed per user. Either a user ID or `--all-users` must be given; the two cannot be combined.

trashbin:expire

Expire trash items that exceed the configured retention period, as defined by `trashbin_retention_obligation` in `config.php`:

```
sudo -E -u www-data php occ trashbin:expire layla
sudo -E -u www-data php occ trashbin:expire
```

> **Note**
>
> This command only runs when a custom retention period is configured. If Nextcloud is set to auto-expiration (the default), the command exits with an informational message and does nothing; auto-expiration is handled by the background job instead.

See *Controlling file versions and aging* for retention configuration.

File versions

```
versions
 versions:cleanup                    delete stored file versions
 versions:expire                     expire file versions that exceed the configured␣
↪retention period
```

versions:cleanup

Delete stored file versions for one or more users, all users, or a specific path:

```
sudo -E -u www-data php occ versions:cleanup layla
  Delete versions of   layla

sudo -E -u www-data php occ versions:cleanup layla fred
sudo -E -u www-data php occ versions:cleanup
```

Use `--path` to limit deletion to a specific directory. The path must include the user ID and `files/`:

```
sudo -E -u www-data php occ versions:cleanup \
  --path="/layla/files/Documents" layla
```

versions:expire

Expire file versions that exceed the configured retention period, as defined by `versions_retention_obligation` in `config.php`:

```
sudo -E -u www-data php occ versions:expire layla
sudo -E -u www-data php occ versions:expire
```

> **Note**
>
> This command only runs when a custom retention period is configured. If Nextcloud is set to auto-expiration (the default), the command exits and does nothing; auto-expiration is handled by the background job instead.

See *Controlling file versions and aging* for retention configuration.

File sharing

```
sharing
 sharing:cleanup-remote-storages      clean up remote storage entries with no␣
↪matching federated share
 sharing:delete-orphan-shares         delete shares where the owner no longer has␣
↪file access
 sharing:expiration-notification      notify share initiators when a share expires␣
↪the next day
 sharing:fix-share-owners             fix share owner after a legacy transfer-
↪ownership operation
share
 share:list                           list available shares
```

share:list

List shares across the instance. Without options, lists all shares:

```
sudo -E -u www-data php occ share:list
+----+-------+--------------------------------+------+-------+-----------+-------+
| id | file  | source-path                    | type | owner | recipient | by    |
+----+-------+--------------------------------+------+-------+-----------+-------+
| 1  | 42    | /layla/files/Documents/rep... | user | layla | fred      | layla |
+----+-------+--------------------------------+------+-------+-----------+-------+
```

Filter options:

- `--owner` — only shares owned by a specific user
- `--recipient` — only shares with a specific recipient
- `--by` — only shares initiated by a specific user
- `--file` — only shares of a specific file (path or file ID)
- `--parent` — only shares of files inside a specific folder
- `--recursive` — combine with `--parent` to include nested shares
- `--type` — filter by share type: `user`, `group`, `link`, `email`, `remote`, `room`, `deck`
- `--status` — only shares with a specific status

Use `--output=json_pretty` for machine-readable output.

sharing:cleanup-remote-storages

Remove `shared::` storage entries from the database that have no matching entry in the `shares_external` table. These orphaned entries are left behind when a federated share is removed without proper cleanup:

```
sudo -E -u www-data php occ sharing:cleanup-remote-storages
  5 remote storage(s) need(s) to be checked
  3 remote share(s) exist
  deleting shared::abc123 [14] ... deleted 1 storage
```

Use `--dry-run` to preview what would be deleted without making changes:

```
sudo -E -u www-data php occ sharing:cleanup-remote-storages --dry-run
```

sharing:delete-orphan-shares

Delete shares where the owner has lost access to the shared file or the file no longer exists:

```
sudo -E -u www-data php occ sharing:delete-orphan-shares
  /layla/files/Documents/report.pdf owned by layla
    file still exists but the share owner lost access to it,
    run occ info:file 42 for more information about the file
  Delete 1 orphan shares? [y/N]
```

Use `--force` / `-f` to delete without prompting. Filter to shares by a specific user with `--owner` or to shares with a specific user with `--with`:

```
sudo -E -u www-data php occ sharing:delete-orphan-shares --owner layla
sudo -E -u www-data php occ sharing:delete-orphan-shares --with fred
```

sharing:expiration-notification

Send in-app notifications to share initiators whose shares expire the following day. Run this daily via a cron job to ensure timely notifications:

```
sudo -E -u www-data php occ sharing:expiration-notification
```

sharing:fix-share-owners

Fix the recorded owner of shares that were broken by a `transfer-ownership` operation performed on an older Nextcloud version. Use `--dry-run` to preview changes:

```
sudo -E -u www-data php occ sharing:fix-share-owners --dry-run
  Share with id 7 (target: /fred/files/report.pdf) can be
  updated to owner layla

sudo -E -u www-data php occ sharing:fix-share-owners
  Share with id 7 (target: /fred/files/report.pdf) updated to owner layla
  No broken shares detected
```

Federation sync

```
federation
 federation:sync-addressbooks         synchronize address books of all federated␣
↪clouds
```

federation:sync-addressbooks

Synchronize the shared address books of all federated Nextcloud servers. Federated servers share user address books so that usernames auto-complete in share dialogs. Run this command to trigger an immediate sync rather than waiting for the background job:

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

Files external

```
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:dependencies          check for missing dependencies needed for␣
↪mounting external storages
 files_external:export                export mount configurations
 files_external:import                import mount configurations
 files_external:list                  list configured admin or personal mounts
 files_external:notify                listen for active update notifications for a␣
↪configured external mount
 files_external:option                manage mount options for a mount
 files_external:scan                  scan an external storage for changed files
 files_external:verify                verify mount configuration
```

Manage Nextcloud's external storage mounts. Commands that read or write mount configuration operate on the same data as the **External Storages** admin settings page.

files_external:list

List configured mounts. Without arguments, lists admin-level mounts:

```
sudo -E -u www-data php occ files_external:list
+----+------------------+-----------+--------------+---------+--------+-------+
| ID | Mount Point      | Storage   | Auth. Type   | Config  | Status | Users |
+----+------------------+-----------+--------------+---------+--------+-------+
| 1  | /shared/data     | amazons3  | builtin      | valid   | ok     | all   |
+----+------------------+-----------+--------------+---------+--------+-------+
```

List personal mounts for a specific user:

```
sudo -E -u www-data php occ files_external:list layla
```

Use `--output=json_pretty` for machine-readable output. Add `--show-password` to include credentials in the output.

files_external:create

Create a new mount configuration:

```
sudo -E -u www-data php occ files_external:create \
  /shared/photos amazons3 builtin::builtin \
  --config bucket=my-nextcloud-photos \
  --config region=eu-central-1 \
  --config key=AKIAIOSFODNN7EXAMPLE \
  --config secret=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

Arguments:

- **mount-point** — the path inside Nextcloud (e.g. `/shared/photos`)
- **storage-backend** — the storage type identifier (e.g. `amazons3`, `sftp`, `smb`, `owncloud`)
- **auth-backend** — the authentication method (e.g. `builtin::builtin`, `password::sessioncredentials`)

Use `files_external:backends` to list all available storage and authentication backends.

For a personal mount, specify the user with `--user`:

```
sudo -E -u www-data php occ files_external:create \
  /my-s3 amazons3 builtin::builtin \
  --config bucket=layla-bucket \
  --user layla
```

files_external:config

Read or write individual backend configuration options for a mount. Use the mount ID from `files_external:list`:

```
sudo -E -u www-data php occ files_external:config 1 get bucket
  my-nextcloud-photos

sudo -E -u www-data php occ files_external:config 1 set bucket new-bucket-name
```

files_external:applicable

Manage which users and groups have access to a mount:

```
sudo -E -u www-data php occ files_external:applicable 1 --add-user layla
sudo -E -u www-data php occ files_external:applicable 1 --add-group milliways
sudo -E -u www-data php occ files_external:applicable 1 --remove-user layla
sudo -E -u www-data php occ files_external:applicable 1 --remove-group milliways
```

A mount with no users or groups assigned is available to all users.

files_external:option

Read or write mount options such as `enable_sharing` or `filesystem_check_changes`:

```
sudo -E -u www-data php occ files_external:option 1 get enable_sharing
sudo -E -u www-data php occ files_external:option 1 set enable_sharing true
```

files_external:verify

Test the configuration of a mount and report whether a connection can be established:

```
sudo -E -u www-data php occ files_external:verify 1
  +----------+---------+
  | Result   | Message |
  +----------+---------+
  | success  |         |
  +----------+---------+
```

Pass additional configuration key-value pairs (e.g. credentials) with `--config` when the stored configuration is incomplete:

```
sudo -E -u www-data php occ files_external:verify 1 \
  --config user=layla --config password=secret
```

files_external:delete

Delete a mount configuration:

```
sudo -E -u www-data php occ files_external:delete 1
```

The command asks for confirmation. Use `--yes` to skip the prompt.

files_external:scan

Scan an external storage for changed files and update the file cache. Useful for storage backends that do not support active change notifications:

```
sudo -E -u www-data php occ files_external:scan 1
```

For mounts that use session credentials, pass them with `--user` and `--password`:

```
sudo -E -u www-data php occ files_external:scan 1 \
  --user layla --password secret
```

files_external:export

Export all admin mounts to JSON:

```
sudo -E -u www-data php occ files_external:export > mounts.json
```

Export personal mounts for a specific user:

```
sudo -E -u www-data php occ files_external:export layla > layla-mounts.json
```

Use the exported JSON with `files_external:import` to replicate the configuration on another Nextcloud instance.

files_external:import

Import mount configurations from a JSON file produced by `files_external:export`:

```
sudo -E -u www-data php occ files_external:import mounts.json
```

Import as personal mounts for a specific user:

```
sudo -E -u www-data php occ files_external:import \
  --user layla layla-mounts.json
```

files_external:backends

List all available storage and authentication backends:

```
sudo -E -u www-data php occ files_external:backends
```

Filter by type:

```
sudo -E -u www-data php occ files_external:backends storage
sudo -E -u www-data php occ files_external:backends authentication
```

Use `--output=json_pretty` to inspect the full capability and configuration schema for each backend.

files_external:notify

Listen for active change notifications from an external mount that supports push-based update events (e.g. SMB with inotify):

```
sudo -E -u www-data php occ files_external:notify 1
```

This is a long-running process; run it under a process supervisor or in a dedicated systemd service.

files_external:dependencies

Check for any missing PHP extensions or system packages required to use the configured external storage backends:

```
sudo -E -u www-data php occ files_external:dependencies
```

Trashbin

> **Note**
>
> These commands are only available when the "Deleted files" app (`files_trashbin`) is enabled.

```
trashbin
 trashbin:cleanup  remove deleted files
 trashbin:expire   expire deleted files according to the configured retention policy
 trashbin:restore  restore all deleted files according to the given filters
 trashbin:size     configure or show the target trashbin size
```

trashbin:cleanup

Remove all deleted files for all users, or for specific users:

```
sudo -E -u www-data php occ trashbin:cleanup
sudo -E -u www-data php occ trashbin:cleanup layla fred
```

trashbin:expire

Apply the configured trashbin retention policy and remove files that have exceeded the maximum retention age. Runs for all users by default, or for specific users:

```
sudo -E -u www-data php occ trashbin:expire
sudo -E -u www-data php occ trashbin:expire layla
```

trashbin:restore

Restore deleted files according to the given filters. Restore all deleted files for all users:

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

Restore deleted files for specific users:

```
sudo -E -u www-data php occ trashbin:restore layla
```

Use `--scope` to limit the restore to a specific scope; one of `user`, `groupfolders`, or `all` (default: `user`):

```
sudo -E -u www-data php occ trashbin:restore --scope groupfolders layla
```

Use `--since` and `--until` to limit the restore to files deleted within a given time window. Both options accept any format supported by PHP `strtotime`:

```
sudo -E -u www-data php occ trashbin:restore --scope all \
  --since "2026-08-01 11:55:22" --until "2026-08-02 01:33:00" layla
```

Use `--dry-run` to simulate the restore without making any changes:

```
sudo -E -u www-data php occ trashbin:restore --dry-run --all-users
```

> **Note**
>
> Use `-v` or `-vv` to see more detail about the restore process and why some files may be skipped.

trashbin:size

Show or configure the target trashbin size. When called without arguments, shows the current configured size:

```
sudo -E -u www-data php occ trashbin:size
```

Set the default trashbin size for all users:

```
sudo -E -u www-data php occ trashbin:size 10GB
```

Set the trashbin size for a specific user:

```
sudo -E -u www-data php occ trashbin:size --user layla 5GB
```

Versions

> **Note**
>
> These commands are only available when the "Versions" app (`files_versions`) is enabled.

```
versions
 versions:cleanup  delete file versions
 versions:expire   expire file versions according to the configured retention policy
```

versions:cleanup

Delete all file versions for all users, or for specific users. Use `--path` to limit deletion to a specific path:

```
sudo -E -u www-data php occ versions:cleanup
sudo -E -u www-data php occ versions:cleanup layla
sudo -E -u www-data php occ versions:cleanup layla --path="/files/Documents"
```

versions:expire

Apply the configured version retention policy and remove versions that have exceeded the maximum retention age. Runs for all users by default, or for specific users:

```
sudo -E -u www-data php occ versions:expire
sudo -E -u www-data php occ versions:expire layla
```

Integrity check

```
integrity
 integrity:check-app                 check integrity of an app using a signature
 integrity:check-core                check core integrity using a signature
 integrity:sign-app                  sign an app using a private key
```

Apps that carry a `Featured` tag must be code-signed by Nextcloud. Unsigned featured apps cannot be installed.

integrity:check-app

Check the integrity of an app against its signature:

```
sudo -E -u www-data php occ integrity:check-app contacts
```

Check all installed apps at once:

```
sudo -E -u www-data php occ integrity:check-app --all
```

When the app passes, the command produces no output (use `-v` to confirm). When it fails, each integrity error is listed with details. Use `--path` to point to a non-standard app location:

```
sudo -E -u www-data php occ integrity:check-app \
  --path=/var/www/nextcloud/apps/myapp myapp
```

Apps without a `signature.json` file are skipped with an informational message.

integrity:check-core

Check the integrity of Nextcloud core against its signature:

```
sudo -E -u www-data php occ integrity:check-core
```

integrity:sign-app

Sign an app with a private key before distribution. Requires the key and certificate obtained through the Nextcloud signing process:

```
sudo -E -u www-data php occ integrity:sign-app \
  --path=/path/to/app \
  --privateKey=/path/to/myapp.key \
  --certificate=/path/to/myapp.crt
```

See Code Signing in the Developer manual for the full signing process.

9.5.5 LDAP commands

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:check-group              checks whether a group exists on LDAP.
 ldap:create-empty-config      creates an empty LDAP configuration
```

(continues on next page)

(continued from previous page)

```
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
ldap:test-user-settings       runs tests and show information about user
                              related LDAP settings
```

ldap:search

Search for an LDAP user, using this syntax

> sudo -E -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 -E -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 -E -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.

ldap:check-user

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

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

Use `--update` to update the account fields from LDAP:

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

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

You can also use `--all-seen-users` to run the check on all users that logged into Nextcloud at least once. You may use `--limit` and `--offset` to do so by batch. You can combine this with `--update` to update the information of seen users.

ldap:check-group

Checks whether a group still exists in the LDAP directory. Use with `--update` to update the group membership cache on the Nextcloud side:

```
sudo -E -u www-data php occ ldap:check-group --update mygroup
```

ldap:create-empty-config

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

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

Then you can list and view your configurations:

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

And view the configuration for a single configID:

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

ldap:delete-config

Deletes an existing LDAP configuration:

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

ldap:set-config

This command is for manipulating configurations, like this example that sets search attributes:

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

ldap:test-user-settings

Tests user-related LDAP settings:

```
sudo -E -u www-data php occ ldap:test-user-settings "cn=philip j. fry,ou=people,
↪dc=planetexpress,dc=com" --group "Everyone"

User cn=philip j. fry,ou=people,dc=planetexpress,dc=com is mapped with account name␣
```

(continues on next page)

(continued from previous page)

```
↪fry.
Known UUID is ce6cd914-71d5-103f-95a8-ad2dab17b2f9.
Configuration prefix is s01

Attributes set in configuration:
- ldapExpertUsernameAttr: uid
- ldapUuidUserAttribute: auto
- ldapEmailAttribute: mail
- ldapUserDisplayName: cn

Attributes fetched from LDAP using filter (|(objectclass=inetOrgPerson)):
- entryuuid: ["ce6cd914-71d5-103f-95a8-ad2dab17b2f9"]
- uid: ["fry"]
- mail: ["fry@planetexpress.com"]
- cn: ["Philip J. Fry"]

Detected UUID attribute: entryuuid

UUID for cn=philip j. fry,ou=people,dc=planetexpress,dc=com: ce6cd914-71d5-103f-95a8-
↪ad2dab17b2f9

Group information:
Configuration:
- ldapGroupFilter: (|(objectclass=groupOfNames))
- ldapGroupMemberAssocAttr: member

Primary group:
Group from gidNumber:
All known groups: ["Ship crew", "Everyone"]
MemberOf usage: off (0,1)

Group Everyone:
Group cn=everyone,ou=groups,dc=planetexpress,dc=com is mapped with name Everyone.
Known UUID is ce8b61c2-71d5-103f-95af-ad2dab17b2f9.
Members: ["bender", "fry", "leela"]
```

ldap:show-remnants

Used to clean up the LDAP mappings table, and is documented in *LDAP user cleanup*.

9.5.6 OCM commands

The `ocm` commands manage the signing keys that Nextcloud uses for Open Cloud Mesh (OCM) federation. Outbound OCM requests are signed with HTTP message signatures (RFC 9421), and the matching public keys are published as a JSON Web Key Set (JWKS) so that federated peers can verify them.

These commands are part of the server core and are always available. They are used to inspect and rotate the JWKS signing keys; you do not need them for normal operation, as a key is generated automatically on the first OCM request.

> **Note**
>
> For more on federation and OCM, see *Configuring Federation Sharing*.

```
ocm
 ocm:keys:list       list JWKS-published signing keys
 ocm:keys:stage      generate a new JWKS key and advertise it via JWKS without using␣
↪it for signing yet
 ocm:keys:activate   promote the staged JWKS key to active; the previous active key␣
↪moves to retiring
 ocm:keys:retire     delete the retiring JWKS key; signatures that referenced its kid␣
↪can no longer be verified
```

Key slots

The signing keys live in three slots, all of which are published in the JWKS endpoint while they are populated:

- **active** — the key currently used to sign outbound OCM requests.
- **pending** — a newly staged key that is advertised in the JWKS but not yet used for signing. This gives federated peers time to pick it up before it becomes active.
- **retiring** — the previous active key, kept published so that signatures created with it can still be verified until it is removed.

Listing keys

ocm:keys:list

Show the current JWKS signing keys and the slot each one occupies:

```
sudo -E -u www-data php occ ocm:keys:list
+------+----------+---------------------------------+
| Pool | Slot     | Key ID                          |
+------+----------+---------------------------------+
| 1    | active   | ecdsa-p256-sha256-...           |
| 2    | pending  | ecdsa-p256-sha256-...           |
+------+----------+---------------------------------+
```

If no keys exist yet, the command reports that one will be generated on the first OCM request. Use `--output=json` or `--output=json_pretty` for machine-readable output.

Rotating keys

Key rotation is a three-step process: stage a new key, activate it, then retire the old one. Allow time between the steps so that federated peers can refresh their cached copy of your JWKS (the cache lifetime is one hour).

ocm:keys:stage

Generate a new key into the **pending** slot. It is advertised in the JWKS but is not yet used for signing:

```
sudo -E -u www-data php occ ocm:keys:stage
  Staged new JWKS key: ecdsa-p256-sha256-...
  Wait for federated peers to refresh their JWKS cache before activating.
```

An active key is required first; if none exists, one is generated automatically. The command fails if a pending key already exists — activate or retire it before staging another.

ocm:keys:activate

Promote the staged (pending) key to **active**. The previous active key moves to the **retiring** slot, where it stays published in the JWKS so that in-flight signatures can still be verified:

```
sudo -E -u www-data php occ ocm:keys:activate
  Staged key promoted to active.
  Run occ ocm:keys:retire once any in-flight signatures using the previous key have␣
↪been verified.
```

The command fails if there is no pending key to activate, or if the retiring slot is still occupied — retire the previous key first.

ocm:keys:retire

Delete the **retiring** key. Once removed, signatures that referenced its key ID can no longer be verified, so only run this after you are confident that no peer still needs it (for example, after at least one JWKS cache lifetime has passed since activation):

```
sudo -E -u www-data php occ ocm:keys:retire
  Retiring key deleted.
```

The command fails if there is no retiring key to remove.

9.5.7 User & group commands

User commands

The `user` commands create and manage user accounts, reset passwords, manage authentication tokens, and report on user activity:

```
user
 user:add                              adds a user
 user:add-app-password                 (deprecated) alias for user:auth-tokens:add
 user:auth-tokens:add                  add an app password for an account
 user:auth-tokens:delete               delete an authentication token
 user:auth-tokens:list                 list authentication tokens for an account
 user:clear-avatar-cache               clear avatar cache
 user:delete                           deletes the specified user
 user:disable                          disables the specified user
 user:enable                           enables the specified user
 user:info                             show information about a user
 user:keys:verify                      verify the stored public key matches the stored␣
↪private key
 user:lastseen                         show when a user was last logged in
 user:list                             list all registered users
 user:profile                          read and modify user profile data
 user:report                           show how many users have access
 user:resetpassword                    reset the password for a user
 user:setting                          read and modify user settings
 user:sync-account-data                sync user backend data to the accounts table
 user:welcome                          send the welcome email to a user
```

user:add

Create a new user with a display name, login name, and optional group memberships:

```
user:add [--password-from-env] [--generate-password] [--display-name[="..."]] [-g|--
↪group[="..."]] [--email EMAIL] uid
```

The `display-name` corresponds to the **Full Name** on the Users page in your Nextcloud web interface, and the `uid` is their **Username** (login name). Any groups that do not exist are created automatically:

```
sudo -E -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"
```

`--password-from-env` reads the password from the `OC_PASS` environment variable. This keeps the password out of the process list and allows scripting the creation of multiple users. Note that `sudo` strips environment variables by default; the `-E` flag preserves them, as shown in the example below:

```
export OC_PASS=newpassword
sudo -E -u www-data php occ user:add --password-from-env \
  --display-name="Layla Smith" --group="users" layla
The user "layla" was created successfully
Display name set to "Layla Smith"
User "layla" added to group "users"
```

`--generate-password` sets a securely generated password that is never shown in the output. Combined with `--email`, this creates a user with a temporary password and sends a welcome email:

```
sudo -E -u www-data php occ user:add layla --generate-password --email layla@example.
↪tld
  The account "layla" was created successfully
  Welcome email sent to layla@example.tld
```

`--email` sets the user's email address and sends a welcome email if `newUser.sendEmail` is set to `yes` in the `core` app config, or if it is not set at all (`yes` is the default):

```
sudo -E -u www-data php occ user:add layla --email layla@example.tld
  Enter password:
  Confirm password:
  The account "layla" was created successfully
  Welcome email sent to layla@example.tld
```

user:resetpassword

Reset any user's password, including administrators (see *Resetting a lost admin password*):

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

Clear a user's password with `--no-password`:

```
sudo -E -u www-data php occ user:resetpassword --no-password layla
  Are you sure you want to clear the password for layla?
  Successfully reset password for layla
```

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

```
export OC_PASS=newpassword
sudo -E -u www-data php occ user:resetpassword --password-from-env layla
  Successfully reset password for layla
```

user:delete

Delete a user:

```
sudo -E -u www-data php occ user:delete layla
```

user:disable and user:enable

Disable a user. Their active sessions will be invalidated within 5 minutes. To invalidate sessions immediately, use `user:auth-tokens:delete` before or after disabling the account:

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

Re-enable a disabled user:

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

user:lastseen

Show a specific user's most recent login:

```
sudo -E -u www-data php occ user:lastseen layla
  layla's last login: 2024-03-20 17:18
```

Show all users' most recent logins:

```
sudo -E -u www-data php occ user:lastseen --all
  albert's last login: 2024-03-18 10:30
  bob has never logged in.
  layla's last login: 2024-03-20 17:18
  stephanie's last login: 2024-01-11 13:26
```

user:list

List all registered users. By default, output is limited to 500 users; use `--limit` and `--offset` to page through larger sets:

```
sudo -E -u www-data php occ user:list
  - admin: admin
  - layla: Layla Smith
  - fred: Fred Jones
```

Use `--disabled` to list only disabled users, and `--info` to include additional backend details.

user:info

Show account details for a user, including display name, email, groups, quota, and storage usage:

```
sudo -E -u www-data php occ user:info layla
  - user_id: layla
  - display_name: Layla Smith
  - email: layla@example.tld
  - cloud_id: layla@cloud.example.tld
  - enabled: true
  - groups:
    - users
    - db-admins
  - quota: none
  - storage:
    - free: 162409623552
    - used: 1110
    - total: 162409624662
    - relative: 0
    - quota: -3
  - first_seen: 2024-03-01T08:44:46+00:00
  - last_seen: 2024-03-20T17:18:00+00:00
  - user_directory: /var/www/nextcloud/data/layla
  - backend: Database
```

user:profile

Read user profile properties:

```
sudo -E -u www-data php occ user:profile layla
  - displayname: Layla Smith
  - address: Berlin
  - email: layla@example.tld
  - profile_enabled: 1
  - pronouns: they/them
```

Get a single profile property:

```
sudo -E -u www-data php occ user:profile layla address
  Berlin
```

Set a profile property:

```
sudo -E -u www-data php occ user:profile layla address Stuttgart
```

Delete a profile property:

```
sudo -E -u www-data php occ user:profile layla address --delete
```

user:setting

Read user settings:

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

Filter by app:

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

Get a single setting:

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

Set a setting:

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

Delete a setting:

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

user:report

Show a count of all users, including users on external authentication backends such as LDAP:

```
sudo -E -u www-data php occ user:report
+------------------+----+
| User Report      |    |
+------------------+----+
| Database         | 12 |
| LDAP             | 86 |
|                  |    |
| total users      | 98 |
|                  |    |
| user directories | 2  |
| active users     | 15 |
| disabled users   | 0  |
+------------------+----+
```

Active users are those who have logged in at least once. Users who have never logged in are not counted as active or disabled. Some backends do not support a user count and may show as zero.

user:auth-tokens:list

List all active authentication tokens (sessions and app passwords) for a user:

```
sudo -E -u www-data php occ user:auth-tokens:list layla
+----+-----------------+---------------------------+-----------+------------+
| id | name            | lastActivity              | type      | scope      |
+----+-----------------+---------------------------+-----------+------------+
| 42 | Firefox on Linux | 2024-03-20T17:18:00+00:00 | temporary | filesystem |
| 47 | Backup script    | 2024-03-19T08:00:00+00:00 | permanent | filesystem |
+----+-----------------+---------------------------+-----------+------------+
```

Use `--output=json` or `--output=json_pretty` for machine-readable output.

user:auth-tokens:add

Create an app password for a user. If no login password is provided, the generated token will have limited capabilities (operations that require the login password will fail):

```
sudo -E -u www-data php occ user:auth-tokens:add --name="Backup script" layla
  Enter password:
  app password: kFrH9-TXk4s-gUoOQ-KOVH8
```

Use `--password-from-env` to read the login password from `NC_PASS` non-interactively:

```
export NC_PASS=userpassword
sudo -E -u www-data php occ user:auth-tokens:add --name="CI runner" \
  --password-from-env layla
```

user:auth-tokens:delete

Delete a specific token by its ID (from `user:auth-tokens:list`):

```
sudo -E -u www-data php occ user:auth-tokens:delete layla 47
```

Delete all tokens for a user that have not been used since a given date:

```
sudo -E -u www-data php occ user:auth-tokens:delete layla \
  --last-used-before="2024-01-01"
```

user:clear-avatar-cache

Clear the cached avatar images for all users. Useful after changing avatar storage settings or after migrating user data:

```
sudo -E -u www-data php occ user:clear-avatar-cache
```

user:keys:verify

Verify that the stored public key for a user matches their stored private key. Returns a confirmation or mismatch notice. Useful for diagnosing end-to-end encryption issues:

```
sudo -E -u www-data php occ user:keys:verify layla
  Stored public key matches stored private key
```

user:sync-account-data

Sync user data from the configured user backends (LDAP, SAML, etc.) to the Nextcloud accounts table. Useful after backend changes to ensure profile data, email addresses, and display names are up to date:

```
sudo -E -u www-data php occ user:sync-account-data
  layla - updated
```

Use `--limit` and `--offset` to process users in batches.

user:welcome

Send the welcome email to a user. The instance must have a working email configuration:

```
sudo -E -u www-data php occ user:welcome layla
```

Add `--reset-password` to include a password reset link in the email:

```
sudo -E -u www-data php occ user:welcome --reset-password layla
```

Group commands

The `group` commands create and manage groups and their memberships:

```
group
 group:add                          add a group
 group:adduser                      add a user to a group
 group:delete                       remove a group
 group:info                         show information about a group
 group:list                         list configured groups
 group:removeuser                   remove a user from a group
```

group:add

Create a new group:

```
sudo -E -u www-data php occ group:add milliways
```

group:adduser and group:removeuser

Add one or more existing users to the specified group with the `group:adduser` command. The syntax is:

```
group:adduser <gid> <uid1> [uid2 ... uidN]
```

This example adds the users “denis”, “dora” and “daisy” to the existing group “milliways”:

```
sudo -E -u www-data php occ group:adduser milliways denis dora daisy
```

You can remove one or more users from the group with the `group:removeuser` command. This example removes the existing users “denis”, “dora” and “daisy” from the existing group “milliways”:

```
sudo -E -u www-data php occ group:removeuser milliways denis dora daisy
```

group:delete

Remove a group. This does not delete the users in the group. The `admin` group cannot be removed:

```
sudo -E -u www-data php occ group:delete milliways
```

group:list

List configured groups. Optionally filter by a search string:

```
sudo -E -u www-data php occ group:list
  - admin:
    - admin
  - milliways:
    - layla
  - users:
    - layla

sudo -E -u www-data php occ group:list milli
  - milliways:
    - layla
```

Use `--limit` and `--offset` to page through large numbers of groups. Use `--info` to include the backend for each group. Use `--output=json` or `--output=json_pretty` for machine-readable output.

group:info

Show details about a group, including its members and backend:

```
sudo -E -u www-data php occ group:info admin
  - groupID: admin
  - displayName: admin
  - backends:
    - Database
```

Use `--output=json_pretty` for machine-readable output.

Two-factor authentication

The `twofactorauth` commands manage two-factor authentication (2FA) enforcement and provider state:

```
twofactorauth
 twofactorauth:cleanup               clean up provider associations for a removed␣
↪provider
 twofactorauth:disable               disable 2FA for a user (provider-specific)
 twofactorauth:enable                enable 2FA for a user (provider-specific)
 twofactorauth:enforce               enforce or disable mandatory 2FA globally or per␣
↪group
 twofactorauth:state                 show the 2FA state for a user
```

twofactorauth:disable and twofactorauth:enable

If a user loses access to their second factor (for example, a lost phone), an admin can disable 2FA for that user for a specific provider:

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

Re-enable 2FA for the user:

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

> **Note**
>
> Not all 2FA providers support per-user enable/disable via occ.

twofactorauth:enforce

Enforce 2FA for all users:

```
sudo -E -u www-data php occ twofactorauth:enforce --on
  Two-factor authentication is enforced for all users
```

Enforce 2FA only for specific groups, optionally excluding others:

```
sudo -E -u www-data php occ twofactorauth:enforce --on \
  --group=admin --group=finance --exclude=service-accounts
  Two-factor authentication is enforced for members of the group(s) admin, finance
```

Disable enforcement:

```
sudo -E -u www-data php occ twofactorauth:enforce --off
  Two-factor authentication is not enforced
```

twofactorauth:state

Show whether 2FA is enabled, enforced, and which providers are active for a user:

```
sudo -E -u www-data php occ twofactorauth:state layla
  Two-factor authentication is not enabled for user layla

  Disabled providers:
  - backup_codes
  - totp
```

twofactorauth:cleanup

Remove stored 2FA provider associations for a provider that has been uninstalled. This cleans up stale data after removing a 2FA app:

```
sudo -E -u www-data php occ twofactorauth:cleanup <provider_id>
```

System Tags

System tags are admin-managed labels that can be assigned to files for use in workflows and automated actions.

Tags have three access levels:

Level	Visible[1]	Assignable[2]
public	Yes	Yes
restricted	Yes	No
invisible	No	No

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

See *Automated tagging of files* for typical use cases for restricted and invisible tags, such as retention and access control.

tag:list

List all system tags:

```
sudo -E -u www-data php occ tag:list
  - 1:
    - name: confidential
    - access: restricted
  - 2:
    - name: needs-review
    - access: public
```

tag:add

Create a new system tag:

```
sudo -E -u www-data php occ tag:add confidential restricted
  - id: 1
  - name: confidential
  - access: restricted
```

tag:edit

Rename a tag or change its access level. Use the tag ID from `tag:list`:

```
sudo -E -u www-data php occ tag:edit --name "reviewed" --color="" 2
  Tag updated ("reviewed", true, true, "")
```

`--name` and `--access` are optional. `--color=""` must be passed explicitly due to a known issue with the command.

tag:delete

Delete a tag by its ID:

```
sudo -E -u www-data php occ tag:delete 1
  The specified tag was deleted
```

Assigning tags to files

Add one or more tags to a file or directory (specified by file ID or path). The `access` argument identifies which tag to apply — because two tags can share the same name but have different access levels, both name and access level are required to uniquely identify a tag. If no matching tag exists, it is created automatically:

```
sudo -E -u www-data php occ tag:files:add /layla/files/report.pdf confidential␣
↪restricted
  restricted tag named confidential added.
```

Multiple tags can be specified as a comma-separated list. All tags in a single call must share the same access level.

Remove specific tags from a file:

```
sudo -E -u www-data php occ tag:files:delete /layla/files/report.pdf confidential␣
↪restricted
```

Remove all tags from a file:

```
sudo -E -u www-data php occ tag:files:delete-all /layla/files/report.pdf
```

9.5.8 System & maintenance commands

The `occ` commands in this section cover server administration, background job management, and operational tasks. They let you configure logging and theming, manage background jobs and AI task processing, administer OAuth2 clients, delegation rules and workflows, check server status, and perform installation and upgrade procedures from the command line.

- *Admin delegation*
- *Background jobs*
- *Broadcast*
- *Info*
- *Logging commands*
- *Maintenance commands*
- *OAuth2*
- *Security*
- *Setup checks*
- *Share operations*
- *Snowflake IDs*
- *Status*

- *Task processing*
- *Theming*
- *Webhook listeners*
- *Workflows*
- *Command line installation*
- *Command line upgrade*
- *Antivirus*

Admin delegation

The `admin-delegation` commands allow granting non-admin groups access to specific admin settings panels, without giving them full administrator privileges:

```
admin-delegation
 admin-delegation:add     add setting delegation to a group
 admin-delegation:remove  remove settings delegation from a group
 admin-delegation:show    show delegated settings
```

admin-delegation:add

Delegate an admin settings class to a group:

```
sudo -E -u www-data php occ admin-delegation:add \
  'OCA\Settings\Settings\Admin\Sharing' milliways
```

The `settingClass` argument must be the fully qualified PHP class name of a settings class that implements `IDelegatedSettings`. Common examples:

- `OCA\Settings\Settings\Admin\Sharing` — sharing administration
- `OCA\Settings\Settings\Admin\Users` — user management
- `OCA\Settings\Settings\Admin\Mail` — email configuration
- `OCA\Theming\Settings\Admin` — theming settings

admin-delegation:remove

Remove a settings delegation from a group:

```
sudo -E -u www-data php occ admin-delegation:remove \
  'OCA\Settings\Settings\Admin\Sharing' milliways
```

admin-delegation:show

Show all currently configured settings delegations:

```
sudo -E -u www-data php occ admin-delegation:show
```

Use `--output=json_pretty` for machine-readable output.

Background jobs

The `background-job` commands let you inspect and execute individual background jobs. The separate `background:cron`, `background:ajax`, and `background:webcron` commands configure how the background job scheduler is triggered:

```
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
background-job
 background-job:delete   Remove a background job from database
 background-job:execute  Execute a single background job manually
 background-job:history  Show past jobs
 background-job:list     List background jobs
 background-job:running  Show currently running jobs
 background-job:worker   Run a background job worker
```

background:cron

Set the background job execution mode to cron. Nextcloud recommends the system cron mode for production instances:

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

Use `background:ajax` to switch to the in-browser AJAX scheduler or `background:webcron` for an external webcron service instead.

See *Background jobs* for a full guide to configuring the background job scheduler.

background-job:delete

Remove a background job from the database. The command shows the job details and asks for confirmation:

```
sudo -E -u www-data php occ background-job:delete 42
  Job class: OCA\Files\BackgroundJob\ScanFiles
  Arguments: []

  Do you really want to delete this background job? It could create some␣
↪misbehaviours in Nextcloud. (y/N)
```

> **Warning**
>
> Deleting a background job can cause Nextcloud features to stop working correctly. Only delete jobs that you know are safe to remove, such as duplicate or orphaned entries.

background-job:execute

Execute a single background job by its database ID:

```
sudo -E -u www-data php occ background-job:execute 42
```

The command shows the job's class, type, last run time, and next scheduled execution before running it. If the job was recently run and is not yet due, it may be skipped; use `--force-execute` to run it regardless:

```
sudo -E -u www-data php occ background-job:execute --force-execute 42
```

background-job:history

Show job history with their duration and peak memory usage:

```
sudo -E -u www-data php occ background-job:history -l 5
+------------------+-----------+-----------------------------------------+---------
↪------------+-----------+---------+-----------+--------------+
| Run ID           | Status    | Class                                   | Started␣
↪at          | Server ID | PID     | Duration  | Memory usage |
+------------------+-----------+-----------------------------------------+---------
↪------------+-----------+---------+-----------+--------------+
| 98109593040965632 | Succeeded | OC\Command\CommandJob                   | 2026-06-
↪22 09:15:19 | 30        | 3847229 | 284 ms    | 251.6 MB     |
| 98109592927719424 | Succeeded | OC\FilesMetadata\Job\UpdateSingleMetadata | 2026-06-
↪22 09:15:19 | 30        | 3847229 | 16 ms     | 222 MB       |
| 98109592852221952 | Succeeded | OC\FilesMetadata\Job\UpdateSingleMetadata | 2026-06-
↪22 09:15:19 | 30        | 3847229 | 8 ms      | 221.4 MB     |
| 98109588267847680 | Succeeded | OC\Command\CommandJob                   | 2026-06-
↪22 09:15:18 | 30        | 3847229 | 909 ms    | 293 MB       |
| 97985889682313216 | Crashed   | OCA\Text\Cron\Cleanup                   | 2026-06-
↪22 01:15:17 | 30        | 3733554 | 594627 ms | 0 B          |
+------------------+-----------+-----------------------------------------+---------
↪------------+-----------+---------+-----------+--------------+
```

Use `-l`/`--limit` to control how many jobs are shown (default: 200), one or multiple `-c`/`--class` to filter by job class, and one or multiple `-s`/`--status` to filter by status (0: running, 1: succeeded, 2: failed, 3: crashed).

background-job:list

List all background jobs registered in the database:

```
sudo -E -u www-data php occ background-job:list
+----+-------------------------------------------------------+-------------------------
↪--+------------+
| id | class                                                 | last_run                ␣
↪  | argument   |
+----+-------------------------------------------------------+-------------------------
↪--+------------+
| 1  | OCA\Files\BackgroundJob\ScanFiles                     | 2025-06-
↪23T10:00:00+00:00 | []         |
| 2  | OCA\Activity\BackgroundJob\DigestMail                 | 2025-06-
↪23T08:00:00+00:00 | []         |
| 3  | OC\Share20\SharesReminderJob                          | 2025-06-
↪23T07:00:00+00:00 | []         |
+----+-------------------------------------------------------+-------------------------
↪--+------------+
```

Use `-c` / `--class` to filter by job class, `-l` / `--limit` to control how many jobs are shown (default: 500), and `-o` / `--offset` to page through results.

background-job:running

List all currently running jobs:

```
sudo -E -u www-data php occ background-job:running
+------------------+----------------------+--------------------+-----------+-------
↪--+---------------+
| Run ID           | Class                | Started at         | Server ID | PID  ␣
↪  | Running since |
+------------------+----------------------+--------------------+-----------+-------
↪--+---------------+
| 98108257692012544 | OC\Command\CommandJob | 2026-06-22 09:10:08 | 30        |␣
↪3845056 | 1 seconds     |
+------------------+----------------------+--------------------+-----------+-------
↪--+---------------+
```

Use `-l` / `--limit` to limit the number of job shown (default: 200).

background-job:worker

Run a continuous background job worker. This is an alternative to system cron for environments where you want to run background jobs from a persistent process rather than a periodic cron invocation:

```
sudo -E -u www-data php occ background-job:worker
```

By default the worker runs indefinitely, polling for new jobs every second.

Use `--once` to process at most one job then exit (equivalent to a single cron run):

```
sudo -E -u www-data php occ background-job:worker --once
```

Use `--stop_after` to set a maximum runtime before the worker exits cleanly (the currently running job is allowed to finish). Accepts seconds, or values like `30s`, `10m`, `2h`:

```
sudo -E -u www-data php occ background-job:worker --stop_after 1h
```

Use `--interval` to set the poll interval in seconds (default: 1). Set to `0` to process jobs once and exit if none are found:

```
sudo -E -u www-data php occ background-job:worker --interval 5
```

Pass one or more job class names to process only jobs of those types:

```
sudo -E -u www-data php occ background-job:worker \
  'OCA\Files\BackgroundJob\ScanFiles'
```

Broadcast

Send a test Server-Sent Events (SSE) broadcast to verify that real-time push notifications are working:

```
sudo -E -u www-data php occ broadcast:test layla
```

Pass a custom event name as the second argument (default: `test`):

```
sudo -E -u www-data php occ broadcast:test layla my-event
```

Info

The `info` commands display detailed information about files and storages. They are useful for debugging file access problems, investigating storage layout, and identifying space usage:

```
info
 info:file         get information for a file
 info:file:space   summarize space usage of a folder
 info:storage      get information for a single storage
 info:storages     list storages ordered by file count
```

info:file

Show detailed information for a file or folder identified by file ID or path:

```
sudo -E -u www-data php occ info:file /layla/files/Documents/report.pdf
  FileId: 12345
  MimeType: application/pdf
  Modified: 2025-06-01T14:30:00+00:00
  Encrypted: false
  Size: 204800
  ETag: abc123
  Permissions: 27
  Users with access: layla
```

Use `-c` / `--children` to list the immediate contents of a folder. Use `--storage-tree` to display the storage and cache wrapping tree.

info:file:space

Show how much space a folder uses, with a ranked breakdown of the largest items:

```
sudo -E -u www-data php occ info:file:space /layla/files/Videos
  Total: 14.7 GB
  +------------------+--------+
  | Path             | Size   |
  +------------------+--------+
  | lecture-2025.mp4 | 8.2 GB |
  | demo-video.mp4   | 4.1 GB |
  +------------------+--------+
```

Use `-c` / `--count` to change the number of items shown (default: 25), or `-a` / `--all` to show all items.

info:storage

Show information for a single storage by its numeric storage ID:

```
sudo -E -u www-data php occ info:storage 5
```

info:storages

List all storages ordered by the number of files they contain:

```
sudo -E -u www-data php occ info:storages
```

Use `-c` / `--count` to limit the number of storages shown (default: 25), or `-a` / `--all` to list all storages.

Logging commands

These commands view and configure your Nextcloud logging preferences:

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

log:file

Show the current logging status:

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

Use `--enable` to turn on the Nextcloud log backend, `--file` to set a different log file path, and `--rotate-size` to rotate the log at a given file size in bytes (`0` disables rotation).

log:manage

Set the logging backend, log level, and timezone. The defaults are `file`, `warning`, and `UTC`:

```
sudo -E -u www-data php occ log:manage --backend file --level warning --timezone UTC
  Log backend: file
  Log level: Warning (2)
  Log timezone: UTC
```

Options:

- `--backend` — one of: `file`, `syslog`, `errorlog`, `systemd`
- `--level` — one of: `debug`, `info`, `warning`, `error`, `fatal`
- `--timezone` — a PHP timezone name; see https://www.php.net/manual/en/timezones.php

Maintenance commands

Use these commands when you upgrade Nextcloud, perform backups, or carry out other tasks that require locking users out temporarily:

```
maintenance
 maintenance:data-fingerprint    update the systems data-fingerprint after a backup␣
↪is restored
 maintenance:mimetype:update-db  update database mimetypes and update filecache
 maintenance:mimetype:update-js  update mimetypelist.js
 maintenance:mode                set maintenance mode
```

(continues on next page)

(continued from previous page)

```
 maintenance:repair              repair this installation
 maintenance:repair-share-owner  fix some shares owner if it fell out of sync
 maintenance:theme:update        apply custom theme changes
 maintenance:update:htaccess     update the .htaccess file
```

maintenance:data-fingerprint

After restoring a backup of your data directory or the database, run this command once. It updates the ETag for all files, allowing sync clients to detect that files were modified:

```
sudo -E -u www-data php occ maintenance:data-fingerprint
```

maintenance:mimetype:update-db and maintenance:mimetype:update-js

Update the Nextcloud database and file cache with changed mimetypes from `config/mimetypemapping.json`. Run after modifying that file. Pass `--repair-filecache` to apply the change to existing files:

```
sudo -E -u www-data php occ maintenance:mimetype:update-db --repair-filecache
```

`maintenance:mimetype:update-js` regenerates the client-side mimetype list:

```
sudo -E -u www-data php occ maintenance:mimetype:update-js
```

maintenance:mode

Lock the sessions of all logged-in users, including administrators, and display 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 the server comes out of maintenance mode, logged-in users must refresh their browser to continue working:

```
sudo -E -u www-data php occ maintenance:mode --on
  Maintenance mode enabled

sudo -E -u www-data php occ maintenance:mode --off
  Maintenance mode disabled
```

maintenance:repair

Runs automatically during upgrades to clean up the database. You can also run it manually if needed:

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

maintenance:repair-share-owner

Fix share owner records that have fallen out of sync with the actual file ownership:

```
sudo -E -u www-data php occ maintenance:repair-share-owner
```

maintenance:theme:update

Run when icons in a custom theme are not updating correctly. This rebuilds the mimetype list and clears the image cache:

```
sudo -E -u www-data php occ maintenance:theme:update
```

maintenance:update:htaccess

Regenerate the `.htaccess` file from the current configuration. Run this after changing URL rewriting settings or after an upgrade if the file appears outdated:

```
sudo -E -u www-data php occ maintenance:update:htaccess
```

OAuth2

> **Note**
>
> This command is only available when `'oauth2.enable_oc_clients' => true` is set in `config/config.php`. It is intended for ownCloud-to-Nextcloud migrations only.

Import an OAuth2 client record from an ownCloud database during a migration. The `client-id` and `client-secret` arguments come directly from the `oc_oauth2_clients` table in the ownCloud database:

```
sudo -E -u www-data php occ oauth2:import-legacy-oc-client \
  <client-id> <client-secret>
```

Security

Use these commands to manage server-wide security parameters, including *Brute force protection* and trusted SSL certificates. Trusted certificates are useful when creating federation connections with servers that use self-signed certificates:

```
security
 security:bruteforce:attempts  show bruteforce attempts status for a given IP address
 security:bruteforce:reset     reset bruteforce attempts for a given IP address
 security:certificates         list trusted certificates
 security:certificates:export  export the certificate bundle
 security:certificates:import  import trusted certificate
 security:certificates:remove  remove trusted certificate
```

security:bruteforce:attempts

Show the current bruteforce throttle status for an IP address, including the number of recorded attempts and the current delay being applied. Optionally filter by action name:

```
sudo -E -u www-data php occ security:bruteforce:attempts 192.168.1.100
sudo -E -u www-data php occ security:bruteforce:attempts 192.168.1.100 login
```

Use `--output=json_pretty` for machine-readable output.

security:bruteforce:reset

Clear all recorded bruteforce attempts for an IP address, removing any login delay immediately:

```
sudo -E -u www-data php occ security:bruteforce:reset 192.168.1.100
```

security:certificates

List all trusted certificates installed on the server:

```
sudo -E -u www-data php occ security:certificates
+--------------------+-------------+-------------------+-------------+-----------+
| File Name          | Common Name | Organization      | Valid Until | Issued By |
+--------------------+-------------+-------------------+-------------+-----------+
| myserver.crt       | myserver    | My Org            | 2026-01-01  | My CA     |
+--------------------+-------------+-------------------+-------------+-----------+
```

Use `--output=json_pretty` for machine-readable output.

security:certificates:export

Export the full certificate bundle to stdout. Useful for backup or inspection:

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

security:certificates:import

Install a trusted certificate from a file. The certificate name is derived from the filename:

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

security:certificates:remove

Remove a trusted certificate by its file name (as shown in `security:certificates`):

```
sudo -E -u www-data php occ security:certificates:remove my-certificate
```

Setup checks

Run the setup checks from the command line to verify your installation configuration:

```
sudo -E -u www-data php occ setupchecks
```

Example output:

```
dav:
  ✓ DAV system address book: No outstanding DAV system address book sync.
network:
  ✓ WebDAV endpoint: Your web server is properly set up to allow file synchronization␣
↪over WebDAV.
  ✓ Data directory protected
  ✓ Internet connectivity
  ...
```

Use `--output=json_pretty` for machine-readable output suitable for automated monitoring.

Use `--class=CLASS_NAME`" to run only the check for the setup check with the given class name.

Use `--category=CATEGORY` to run only the checks for the given category, e.g. `network` or `system`.

Share operations

Available `occ` commands for the `share` namespace:

```
share
 share:list  list available shares
```

share:list

List all shares on the system:

```
sudo -E -u www-data php occ share:list
```

Filter by owner, recipient, or the user who created the share:

```
sudo -E -u www-data php occ share:list --owner layla
sudo -E -u www-data php occ share:list --recipient fred
sudo -E -u www-data php occ share:list --by layla
```

Filter by a specific file path:

```
sudo -E -u www-data php occ share:list --file "/layla/files/Documents/report.pdf"
```

Filter by a folder; use `--recursive` to include shares nested anywhere inside it:

```
sudo -E -u www-data php occ share:list --parent "/layla/files/Projects" --recursive
```

Filter by share type (one of `user`, `group`, `link`, `email`, `remote`, `room`, `deck`) or by share status:

```
sudo -E -u www-data php occ share:list --type link
sudo -E -u www-data php occ share:list --status 0
```

Snowflake IDs

Nextcloud uses Snowflake IDs for unique identifiers in several subsystems. Decode a Snowflake ID to inspect its embedded timestamp and metadata:

```
sudo -E -u www-data php occ snowflake:decode 6768789079123765868
```

Example output:

```
+--------------------+-------------------------+
| Snowflake ID       | 6768789079123765868     |
| Seconds            | 1575981518              |
| Milliseconds       | 50                      |
| Created from CLI   | no                      |
| Server ID          | 441                     |
| Sequence ID        | 2668                    |
| Creation timestamp | 1575981518.050          |
```

(continues on next page)

(continued from previous page)

```
| Creation date      | 2019-12-10 11:18:38.050 |
+--------------------+-------------------------+
```

Status

Use the `status` command to retrieve information about the current installation:

```
sudo -E -u www-data php occ status
  - installed: true
  - version: 30.0.0.0
  - versionstring: 30.0.0
  - edition:
  - maintenance: false
  - needsDbUpgrade: false
  - productname: Nextcloud
  - extendedSupport: false
```

Use `--output=json_pretty` for machine-readable output:

```
sudo -E -u www-data php occ status --output=json_pretty
{
    "installed": true,
    "version": "30.0.0.0",
    "versionstring": "30.0.0",
    "edition": "",
    "maintenance": false,
    "needsDbUpgrade": false,
    "productname": "Nextcloud",
    "extendedSupport": false
}
```

Status return code

Use the `-e` flag to get a machine-readable exit code reflecting the installation state. There is no output by default, making it suitable for scripts, monitoring checks, and systemd units:

```
sudo -E -u www-data php occ status -e
echo $?
0
sudo -E -u www-data php occ maintenance:mode --on
Maintenance mode enabled
sudo -E -u www-data php occ status -e
echo $?
1
sudo -E -u www-data php occ maintenance:mode --off
Maintenance mode disabled
sudo -E -u www-data php occ status -e
echo $?
0
```

Return code	Description
0	normal operation
1	maintenance mode is enabled; the instance is currently unavailable to users
2	`occ upgrade` is required

Task processing

The `taskprocessing` commands manage AI and background task processing jobs. Task processing provides infrastructure for AI features such as text generation, image classification, and speech-to-text:

```
taskprocessing
 taskprocessing:task-type:set-enabled enable or disable a task type
 taskprocessing:task:cleanup          cleanup old tasks
 taskprocessing:task:get              display all information for a specific task
 taskprocessing:task:list             list tasks
 taskprocessing:task:stats            get statistics for tasks
 taskprocessing:worker                run a dedicated worker for synchronous␣
↪TaskProcessing providers
```

taskprocessing:task-type:set-enabled

Enable or disable a task type:

```
sudo -E -u www-data php occ taskprocessing:task-type:set-enabled core:text2text 1
sudo -E -u www-data php occ taskprocessing:task-type:set-enabled core:text2text 0
```

taskprocessing:task:cleanup

Delete old task records and their associated output files. By default, tasks older than four months are removed:

```
sudo -E -u www-data php occ taskprocessing:task:cleanup
```

Pass a maximum age in seconds to override the default:

```
sudo -E -u www-data php occ taskprocessing:task:cleanup 2592000
```

taskprocessing:task:get

Display all information for a specific task by its numeric ID:

```
sudo -E -u www-data php occ taskprocessing:task:get 42
```

taskprocessing:task:list

List task processing tasks:

```
sudo -E -u www-data php occ taskprocessing:task:list
```

Filter by user, type, app, or custom ID:

```
sudo -E -u www-data php occ taskprocessing:task:list --userIdFilter layla
sudo -E -u www-data php occ taskprocessing:task:list --type core:text2text
```

Filter by status (0=UNKNOWN, 1=SCHEDULED, 2=RUNNING, 3=SUCCESSFUL, 4=FAILED, 5=CANCELLED):

```
sudo -E -u www-data php occ taskprocessing:task:list --status 4
```

Other available filters: `--appId`, `--customId`, `--scheduledAfter`, `--endedBefore`.

taskprocessing:task:stats

Show statistics for tasks (max/average running time, queuing time, and input/output sizes). Accepts the same filter options as `taskprocessing:task:list`:

```
sudo -E -u www-data php occ taskprocessing:task:stats
```

taskprocessing:worker

Run a dedicated worker that processes tasks from synchronous TaskProcessing providers. Use this when you want to offload AI task execution to a separate process:

```
sudo -E -u www-data php occ taskprocessing:worker
```

Use `--once` to process at most one task then exit. Use `--timeout` to set a maximum runtime in seconds (default: `0` = run indefinitely). Use `--taskTypes` (repeatable) to restrict the worker to specific task type IDs:

```
sudo -E -u www-data php occ taskprocessing:worker \
  --taskTypes core:text2text --once
```

Theming

The Theming app (`theming`) is always enabled. The `theming:config` command lets you view and update theming settings without logging into the web interface:

```
theming
 theming:config  set theming app config values
```

theming:config

With no arguments, show all current theming values:

```
sudo -E -u www-data php occ theming:config
  Current theming config:
  name: Nextcloud
  url: https://nextcloud.com
  slogan: a safe home for all your data
  ...
```

Show the current value of a single key:

```
sudo -E -u www-data php occ theming:config name
```

Set a value:

```
sudo -E -u www-data php occ theming:config name "Acme Cloud"
sudo -E -u www-data php occ theming:config url "https://acme.example.com"
sudo -E -u www-data php occ theming:config slogan "Secure file sync for Acme"
sudo -E -u www-data php occ theming:config primary_color "#0082c9"
sudo -E -u www-data php occ theming:config background_color "#00679e"
sudo -E -u www-data php occ theming:config disable-user-theming true
```

To set an image, pass the path to the image file as the value:

```
sudo -E -u www-data php occ theming:config logo /path/to/logo.png
sudo -E -u www-data php occ theming:config favicon /path/to/favicon.ico
sudo -E -u www-data php occ theming:config background /path/to/background.jpg
```

Reset a key to its default:

```
sudo -E -u www-data php occ theming:config --reset name
```

Supported text keys: `name`, `url`, `imprintUrl`, `privacyUrl`, `slogan`, `primary_color`, `background_color`, `disable-user-theming`.

Supported image keys: `background`, `logo`, `logoheader`, `favicon`.

Webhook listeners

Note

The Webhook listeners app (`webhook_listeners`) is shipped with Nextcloud but not enabled by default. Enable it in the Apps menu before using these commands.

List all webhook listener configurations registered on the server. Each entry shows the listener ID, the user or app that registered it, the HTTP method and target URI, the Nextcloud event it listens for, any event filter, and the configured authentication method (`none` or `header`):

```
sudo -E -u www-data php occ webhook_listeners:list
```

Use `--output=json_pretty` for machine-readable output, which also includes the full header and authentication data fields:

```
sudo -E -u www-data php occ webhook_listeners:list --output=json_pretty
```

Webhook listeners are configured through the Admin settings UI or the REST API. This command is read-only — it provides an audit view of what listeners are active without needing to access the web interface.

Workflows

The Workflow engine (`workflowengine`) is always enabled:

```
workflows
 workflows:list  list configured workflows
```

workflows:list

List configured workflow rules. Use the optional `scope` argument to filter by scope; `admin` (default) or `user`:

```
sudo -E -u www-data php occ workflows:list
sudo -E -u www-data php occ workflows:list user
sudo -E -u www-data php occ workflows:list user layla
```

The output is a JSON representation of the configured workflow operations.

Command line installation

These commands are only available before Nextcloud has been installed, after you have unpacked the archive and copied Nextcloud into the appropriate directories.

Display the available installation options:

```
$ sudo -E -u www-data php /var/www/nextcloud/occ maintenance:install --help
Nextcloud is not installed - only a limited number of commands are available

Usage:
  maintenance:install [options]

Options:
      --database[=DATABASE]                  Supported database type [default: "sqlite
↪"]
      --database-name[=DATABASE-NAME]        Name of the database
      --database-host[=DATABASE-HOST]        Hostname of the database [default:
↪"localhost"]
      --database-port[=DATABASE-PORT]        Port the database is listening on
      --database-user[=DATABASE-USER]        Login for database connection
      --database-pass[=DATABASE-PASS]        Password of the database login
      --database-table-space[=DATABASE-TABLE-SPACE]
                                                Table space of the database (``oci``␣
↪only)
      --database-ssl-mode[=DATABASE-SSL-MODE]
                                                Encryption mode for the database␣
↪connection,
                                                e.g. "require" or "verify-full"␣
↪(``pgsql`` only)
      --database-ssl-ca[=DATABASE-SSL-CA]    Path to the CA certificate the database␣
↪server
                                                is verified against (``mysql`` and␣
↪``pgsql`` only)
      --database-ssl-cert[=DATABASE-SSL-CERT]
                                                Path to the client certificate used to
                                                authenticate against the database
                                                (``mysql`` and ``pgsql`` only)
      --database-ssl-key[=DATABASE-SSL-KEY]  Path to the private key of the client␣
↪certificate
                                                (``mysql`` and ``pgsql`` only)
      --database-ssl-crl[=DATABASE-SSL-CRL]  Path to the certificate revocation list
                                                (``pgsql`` only)
      --database-ssl-no-verify               Do not verify that the database server␣
↪certificate
```

(continues on next page)

(continued from previous page)

```
                                                matches the hostname used to connect
                                                (``mysql`` only)
      --disable-admin-user                      Disable the creation of an administrator␣
↪login
      --admin-user[=ADMIN-USER]                 Login for initial administrator account␣
↪[default: "admin"]
      --admin-pass[=ADMIN-PASS]                 Password for initial administrator login
      --admin-email[=ADMIN-EMAIL]               E-Mail to associate with the initial␣
↪administrator login
      --data-dir[=DATA-DIR]                     Path to data directory
      --password-salt[=PASSWORD-SALT]           Password salt; generated if not provided␣
↪(ADVANCED)
      --server-secret[=SERVER-SECRET]           Server secret; generated if not provided␣
↪(ADVANCED)
```

This example installs Nextcloud with a MySQL database. `occ` prompts for the database and administrator passwords:

```
$ sudo -E -u www-data php /var/www/nextcloud/occ maintenance:install \
    --database mysql \
    --database-name nextcloud \
    --database-host 127.0.0.1 \
    --database-user nextcloud \
    --admin-user admin
What is the password to access the database with user <nextcloud>?
What is the password you like to use for the admin account <admin>?
Nextcloud was successfully installed
```

Supported databases:

- `sqlite` — SQLite (not recommended for production usage)
- `mysql` — MySQL or MariaDB
- `pgsql` — PostgreSQL
- `oci` — Oracle; contact Nextcloud GmbH for enterprise support

Added in version 35: The `--database-ssl-*` options set up an SSL/TLS encrypted connection to the database, see *Encrypted database connection*.

Command line upgrade

These commands are available after downloading an upgraded package or tarball, and before completing the upgrade.

Important

`occ upgrade` only runs the **migration phase**: database schema updates and app upgrades. It does **not** download or replace Nextcloud's code files.

Before running `occ upgrade` you must first replace the code using one of:

- The built-in updater: `sudo -E -u www-data php /var/www/nextcloud/updater/updater.phar`
- A manually extracted tarball (see *Upgrade manually*)

If `occ upgrade` reports *"Nextcloud is up to date"* when you expect an upgrade to run, the code files have not been replaced yet — run the updater or extract the new tarball first.

See *How to upgrade* for the full upgrade process.

When performing an upgrade, use `occ upgrade` instead of the web-based upgrader to avoid PHP timeout limits (the web interface enforces a 3600-second limit). On large installations this may not be sufficient, leaving the system in an inconsistent state.

After completing all preliminary steps (see *How to upgrade*), run the upgrade:

```
sudo -E -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
```

Use `-v` to display timestamps on each step:

```
sudo -E -u www-data php occ upgrade -v
  2025-06-23T09:06:15+0000 Turned on maintenance mode
  2025-06-23T09:06:15+0000 Checked database schema update
  2025-06-23T09:06:15+0000 Checked database schema update for apps
  2025-06-23T09:06:15+0000 Updated database
  2025-06-23T09:06:15+0000 Updated <files_sharing> to 0.6.6
  2025-06-23T09:06:15+0000 Update successful
  2025-06-23T09:06:15+0000 Turned off maintenance mode
```

If an error occurs, the exception is logged in the Nextcloud log file:

```
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
```

Antivirus

Note

These commands require the files_antivirus app to be installed and enabled.

```
files_antivirus
 files_antivirus:background-scan  run the background scan
 files_antivirus:mark             mark a file as scanned or unscanned
 files_antivirus:scan             scan a file
 files_antivirus:status           show antivirus scanner status
```

files_antivirus:background-scan

Trigger the background antivirus scan manually. By default all pending files are processed. Use `--max` to limit the number of files scanned in a single run:

```
sudo -E -u www-data php occ files_antivirus:background-scan
sudo -E -u www-data php occ files_antivirus:background-scan --max 500
```

files_antivirus:mark

Mark a file as scanned or unscanned. The `file` argument is the path to the file and `mode` is either `scanned` or `unscanned`:

```
sudo -E -u www-data php occ files_antivirus:mark /layla/files/report.pdf scanned
sudo -E -u www-data php occ files_antivirus:mark /layla/files/report.pdf unscanned
```

Use `--forever` (`-f`) when marking a file as scanned to permanently exclude it from future rescans:

```
sudo -E -u www-data php occ files_antivirus:mark --forever /layla/files/report.pdf␣
↪scanned
```

files_antivirus:scan

Scan a single file immediately and report the result:

```
sudo -E -u www-data php occ files_antivirus:scan /layla/files/report.pdf
```

Use `--debug` to enable verbose backend output, useful when diagnosing scanner connectivity issues:

```
sudo -E -u www-data php occ files_antivirus:scan --debug /layla/files/report.pdf
```

files_antivirus:status

Show the current antivirus scanner status, including the configured backend and whether the scanner is reachable:

```
sudo -E -u www-data php occ files_antivirus:status
```

Use `-v` for additional backend detail.

CHAPTER

TEN

REFERENCE MANAGEMENT

The reference management system brings 2 features in Nextcloud:

- *The link previews* (also called reference widgets)
- *The Smart Picker*

Both those features are generic and need to be extended by the Nextcloud apps.

Apps can add support for some HTTP links so previews are rendered in various places like Text documents and Talk messages.

The Smart Picker is a frontend component which allows users to search or generate links or text.

Apps can register Smart Picker providers to extend its capabilities. Administrators can choose which Smart Picker providers they want to make available to the users by choosing which apps they install. All the Smart Picker providers shipped in the recommended apps do **not** send any data to 3rd party services. Some community apps Smart Picker providers might rely on 3rd party services.

10.1 Link previews

Link previews are available in some places in Nextcloud. There are 3 types of link preview:

- **The ones for links that are supported by a reference provider**
  - Without custom reference widget (uses a default generic style, image + title + description)
  - With custom reference widget (implemented by the app which supports the link)
- Default ones from OpenGraph information. This is the fallback for every unsupported link

10.1.1 Where do they appear?

The link previews provided by the Nextcloud reference system appear in the following places:

- **Text (and Collectives pages, Notes, Deck card comments, Files comments etc…)**
  - Directly in the document content, next to the links
  - Only one link preview per paragraph is rendered
  - Custom widgets can be rendered
- **Talk**
  - In the messages
  - Only one link preview per message is rendered
  - Custom widgets can be rendered

- **Nextcloud Office**
  - In the document content when hovering on links
  - Custom widgets are not rendered

10.1.2 How does it work?

The Nextcloud frontend asks the server to resolve the links via an API request. A rich object is returned as a response and is used by the frontend to render the preview.

The apps can optionally register a custom reference widget to render a specific rich object type (on the links it supports). Therefore the apps have complete freedom over how some previews look like.

10.1.3 Known link preview providers

- Collectives: Links to Collective pages
- Tables: Links to tables
- Deck: Links to boards, cards and comments
- Talk: Links to conversations
- GitHub integration: Links to GitHub issues, pull requests, comments and repositories
- GitLab integration: Links to Gitlab issues, merge requests, comments and repositories
- Zammad integration: Links to Zammad tickets
- Reddit integration: Links to subreddits, publications and comments
- Mastodon integration: Links to members and toots
- The Movie Database integration: Links to people, movies and series
- OpenStreetMap integration: Location links from OpenStreetMap, Google maps, Bing maps, Here maps and Duck-duckgo maps
- Giphy integration: Links to GIFs
- Notion integration: Links to Notion documents
- Peertube integration: Links to videos

10.2 The Smart Picker

Every Smart Picker provider can be enabled by installing and configuring the corresponding app.

10.2.1 Where can it be used?

The Smart Picker can be used in:

- Text (and everywhere Text is used like Collectives pages, Deck card comments, Files comments…): by pressing the "/" key or using a top menu entry
- Talk: by pressing the "/" key in the message composition input
- Nextcloud Office: with a top menu entry (*Insert* → *Pick Link* or *Smart Picker*, depending on Collabora version)
- Mail: in the email composition area with a context menu entry

10.2.2 Known Smart Picker providers

- **Accessing internal data**
  - Collectives: To get links to Collective pages
  - Tables: To get links to tables
  - Deck: To get links to boards and comments
  - Talk: To get links to conversations
  - Files: To get internal links to files (not share links yet)
  - Text templates: To get personal and global text templates
- **Relying on 3rd party services**
  - GitHub integration: To get links to GitHub issues, pull requests, and repositories
  - GitLab integration: To get links to Gitlab issues, merge requests, and repositories
  - Zammad integration: To get links to Zammad tickets
  - Reddit integration: To get links to subreddits and publications
  - Mastodon integration: To get links to members, toots and hashtags
  - The Movie Database integration: To get links to people, movies and series
  - OpenStreetMap integration: To get location links from OpenStreetMap
  - Giphy integration: To get links to GIFs
  - Notion integration: To get links to Notion documents
  - Peertube integration: To get links to videos
  - OpenAI integration: To generate images with Dall-e, text with GPT and transcribe/translate with Whisper (speech-to-text)
  - Replicate integration: To generate images with stable diffusion, and transcribe/translate with Whisper (speech-to-text)

CHAPTER

ELEVEN

WEBHOOK LISTENERS

11.1 Introduction

Nextcloud supports sending notifications to external services whenever something important happens, such as when files are changed or updated.

11.2 Overview

The Webhook Listeners app enables your Nextcloud server to automatically notify external services whenever important events - such as file changes, uploads, or deletions - occur in your instance. By configuring webhook listeners, administrators can set up custom HTTP notifications (webhooks) that are triggered by specific internal events, allowing seamless integration with other platforms and automation of workflows without manual intervention.

The app works by monitoring Nextcloud's event system and dispatching HTTP requests to defined endpoints whenever a matching event takes place. Management and configuration of webhook listeners are handled via the Nextcloud OCS API and command-line tools. The app is ideal for scenarios where you want to connect Nextcloud with notification systems, external automation platforms, or custom integrations - without requiring manual polling.

11.3 Installation

Enable the `webhook_listeners` app that comes bundled with Nextcloud - e.g.

```
occ app:enable webhook_listeners
```

11.4 Listening to events

You can use the OCS API to add webhooks for specific events. See: Register a new webhook.

Note: When authenticating with the OCS API to register webhooks, the account you use must have administrator rights or delegated administrator rights.

11.4.1 Listing registered webhooks

You can list all currently registered webhook listeners from the command line:

```
occ webhook_listeners:list
```

By default this prints a table. Use the `--output` option to change the format. Each row contains the full configuration of a registered webhook

11.4.2 Filters

When registering a webhook listener, you can specify a filter parameter (`eventFilter`). The filter is evaluated against the **complete webhook payload envelope**, which has this structure:

```
{
  "event": { "class": "…", "…event-specific fields…" },
  "user": { "uid": "…", "displayName": "…" },
  "time": 1700100000
}
```

Filter keys use **dot-notation** to traverse into nested objects. For example:

- `time` — matches the top-level Unix timestamp
- `user.uid` — matches the `uid` field inside the `user` object
- `event.class` — matches the event's fully-qualified class name inside `event`
- `event.node.path` — matches the `path` field inside `event` → `node` (for node events)
- `event.calendarId` — matches the `calendarId` field inside `event` (for calendar events)

The value of the filter parameter must be a JSON object whose properties represent filter conditions. The `{}` object is an empty query, meaning no specific criteria are set, so all events are matched.

If you want to match events triggered by a specific user, you can pass `{ "user.uid": "bob" }` to match all events associated with the user `bob`.

To enforce multiple criteria, simply pass multiple properties: `{ "event.tableId": 42, "event.rowId": 3 }`.

If you want to match values partially, you can use regular expressions: `{ "user.uid": "/admin_.*/" }` will match any user whose user ID starts with `admin_`. This can be especially useful for filesystem events when filtering by path: `{ "event.node.path": "/^\\/.*\\/files\\/Special folder\\//" }` will match files inside the `Special folder` of any user. Note that the slashes in the path need to be escaped with two backslashes: once because you are inside a JSON string, and once because you are inside a regular expression.

You can also use comparison operators (`$e`, `$ne`, `$gt`, `$gte`, `$lt`, `$lte`, `$in`, `$nin`, `$all`, `$exists`, `$mod`) as well as logical operators (`$and`, `$or`, `$not`, `$nor`).

For example, use `{ "time": { "$lt": 1711971024 } }` to accept only events prior to April 1st, 2024, and `{ "time": { "$not": { "$lt": 1711971024 } } }` to accept events after April 1st, 2024.

Use `{ "event.node.id": { "$exists": true } }` to match only events where the node still has an ID (i.e. is not a `NonExistingFile`/`NonExistingFolder`).

Use `{ "event.class": "OCP\\Files\\Events\\Node\\NodeCreatedEvent" }` to match only `NodeCreatedEvent` events (backslashes must be escaped in JSON strings).

11.4.3 Speeding up webhook dispatch

This app uses background jobs to trigger registered webhooks. By default, webhooks are triggered every 5 minutes, as the default cron interval is set to 5 minutes. To trigger webhooks sooner, you can set up a background job worker.

The following command will launch a worker for the webhook call background job:

Screen or tmux session

Run the following `occ` command inside a screen or tmux session, preferably four or more times, to enable parallel processing of multiple requests by different users or the same user. It is best to run one command per screen session or tmux window/pane to keep logs visible and make each worker easy to restart.

```
set -e; while true; do sudo -E -u www-data php occ background-job:worker -v -t 60
↪"OCA\WebhookListeners\BackgroundJobs\WebhookCall"; done
```

For Nextcloud-AIO you should use this command on the host server.

```
set -e; while true; do sudo docker exec -it nextcloud-aio-nextcloud docker exec -it␣
↪nextcloud-aio-nextcloud sudo -E -u www-data php occ background-job:worker -v -t 60
↪"OCA\WebhookListeners\BackgroundJobs\WebhookCall"; done
```

You may want to adjust the number of workers and the timeout (in seconds) to your needs. The logs of the worker can be checked by attaching to the screen or tmux session.

Systemd service

1. Create a systemd service file in `/etc/systemd/system/nextcloud-webhook-worker@.service` with the following content:

```
[Unit]
Description=Nextcloud Webhook worker %i
After=network.target

[Service]
ExecStart=/opt/nextcloud-webhook-worker/taskprocessing.sh %i
Restart=always
StartLimitInterval=60
StartLimitBurst=10

[Install]
WantedBy=multi-user.target
```

2. Create a shell script in `/opt/nextcloud-webhook-worker/taskprocessing.sh` with the following content and make sure to make it executable:

```
#!/bin/sh
echo "Starting Nextcloud Webhook Worker $1"
cd /path/to/nextcloud
sudo -E -u www-data php occ background-job:worker -t 60 'OCA\WebhookListeners\
↪BackgroundJobs\WebhookCall'
```

You may want to adjust the timeout to your needs (in seconds).

3. Enable and start the service 4 or more times:

```
for i in {1..4}; do systemctl enable --now nextcloud-webhook-worker@$i.service; done
```

The status of the workers can be checked with (replace 1 with the worker number):

```
systemctl status nextcloud-webhook-worker@1.service
```

The list of workers can be checked with:

```
systemctl list-units --type=service | grep nextcloud-webhook-worker
```

The complete logs of the workers can be checked with (replace 1 with the worker number):

```
sudo journalctl -xeu nextcloud-webhook-worker@1.service -f
```

It is recommended to restart this worker at least once a day to make sure code changes are effective and avoid memory leaks, in this example the service restarts every 60 seconds.

11.5 Nextcloud Webhook Events

This is list of typically available events. It features the event ID and the available variables for filtering.

> **Note**
>
> In addition to the events listed below (which are provided by Nextcloud server and included apps), optional apps may register their own webhook-compatible events. The exact set of events available on your server will depend on your Nextcloud version and installed apps. If you need to discover available events, consult your app documentation or refer to your app source code or the Nextcloud developer documentation.

11.5.1 Payload envelope

Every webhook HTTP POST body shares the same top-level structure:

```
{
  "event": {},
  "user": { "uid": "alice", "displayName": "Alice" },
  "time": 1700100000
}
```

Where:

- `"event"` is an object containing the event-specific data **plus** a `"class"` field holding the fully-qualified PHP class name (e.g. `"OCP\\Files\\Events\\Node\\NodeCreatedEvent"`).
- `"user"` is the user who triggered the event (`null` when no user session exists). Contains `"uid"` (string) and `"displayName"` (string).
- `"time"` is an integer Unix timestamp of when the event was captured.

Note: Example payloads below are based on actual webhook HTTP POST payloads, using real JSON and data types. Field types are indicated by their value type: for example, `"id": 123` is an integer, not a string.

11.5.2 Node (Files / Folders) Events

Node events include those that act on a single node as well as those that act on two nodes (such as copy, rename, and restore). In single-node events, the `event` object includes a `node` object. In two-node events, it includes both a `source` and a `target` object. The examples below show the **complete** payloads.

1. Single-node events

Applies to:

- `OCP\Files\Events\Node\BeforeNodeCreatedEvent`
- `OCP\Files\Events\Node\BeforeNodeTouchedEvent`
- `OCP\Files\Events\Node\BeforeNodeWrittenEvent`
- `OCP\Files\Events\Node\BeforeNodeReadEvent`

- `OCP\Files\Events\Node\BeforeNodeDeletedEvent`
- `OCP\Files\Events\Node\NodeCreatedEvent`
- `OCP\Files\Events\Node\NodeTouchedEvent`
- `OCP\Files\Events\Node\NodeWrittenEvent`
- `OCP\Files\Events\Node\NodeDeletedEvent`

```
{
  "event": {
    "class": "OCP\\Files\\Events\\Node\\NodeCreatedEvent",
    "node": {
      "id": 437,
      "path": "/admin/files/test-webhook.txt"
    }
  },
  "user": {
    "uid": "admin",
    "displayName": "Admin"
  },
  "time": 1700100000
}
```

Where:

- `"event.class"` is the fully-qualified event class name (string)
- `"event.node.id"` is an integer (unique node ID) — may be absent, see note below
- `"event.node.path"` is a string (file/folder path)

Note

In some events (such as `NodeDeletedEvent` or when the node refers to a non-existent file or folder), the `node` object may not have an `id` field. This happens when the node no longer exists in the filesystem/database, and is represented in Nextcloud by an internal `NonExistingFile` or `NonExistingFolder`. In such cases, only the `path` field will be present. Always check for the presence of the `id` field in these payloads. Usually, the corresponding `Before*` event (such as `BeforeNodeDeletedEvent`) can be used if you require the `id` of the node before deletion.

Example of a deleted node webhook payload:

```
{
  "event": {
    "class": "OCP\\Files\\Events\\Node\\NodeDeletedEvent",
    "node": {
      "path": "/user/files/oldfile.txt"
    }
  },
  "user": {
    "uid": "user",
    "displayName": "User"
  },
  "time": 1700100500
}
```

2. Two-node events

Applies to:

- `OCP\Files\Events\Node\NodeCopiedEvent`
- `OCP\Files\Events\Node\NodeRenamedEvent`
- `OCP\Files\Events\Node\NodeRestoredEvent`
- `OCP\Files\Events\Node\BeforeNodeCopiedEvent`
- `OCP\Files\Events\Node\BeforeNodeRestoredEvent`
- `OCP\Files\Events\Node\BeforeNodeRenamedEvent`

```
{
  "event": {
    "class": "OCP\\Files\\Events\\Node\\NodeRestoredEvent",
    "source": {
      "id": 399,
      "path": "/admin/files_trashbin/files/myfile.txt"
    },
    "target": {
      "id": 437,
      "path": "/admin/files/myfile.txt"
    }
  },
  "user": {
    "uid": "admin",
    "displayName": "Admin"
  },
  "time": 1700100000
}
```

Where:

- `"source"` is an object representing the original node, with `"id"` (integer) and `"path"` (string)
- `"target"` is an object representing the resulting/copied/restored/renamed node, with `"id"` (integer) and `"path"` (string)

Note

For some two-node events, the `source` or `target` node may not have an `id` field, for instance if the node was deleted or is otherwise missing from the filesystem. Always check for the presence of the `id` field in these objects. Typically, if you need the `id` of a node just before deletion or change, the respective `Before*` event (such as `BeforeNodeRenamedEvent`) will include it.

Example of a two-node event (NodeRenamedEvent) where the source node is missing:

```
{
  "event": {
     "class": "OCP\\Files\\Events\\Node\\NodeRenamedEvent",
     "source": {
         "path": "/user/files/previousname.txt"
     },
```

(continues on next page)

(continued from previous page)

```
    "target": {
        "id": 599,
        "path": "/user/files/newname.txt"
     }
   },
   "user": {
        "uid": "user",
        "displayName": "Joe User"
    },
    "time": 1700100000
}
```

Note

Only these fields are guaranteed by Nextcloud core. Additional fields may appear if added by custom plugins or future versions.

11.5.3 System Tag Events

- `OCP\SystemTag\TagAssignedEvent`
- `OCP\SystemTag\TagUnassignedEvent`

```
{
    "event": {
        "class": "OCP\\SystemTag\\TagAssignedEvent",
        "objectType": "files",
        "objectIds": ["437", "438"],
        "tagIds": [3, 17]
    },
    "user": {
        "uid": "admin",
        "displayName": "Admin"
    },
    "time": 1700100000
}
```

11.5.4 Calendar Object Events

Calendar object events use two distinct payload formats, depending on the event.

Standard payload

Applies to:

- `OCP\Calendar\Events\CalendarObjectCreatedEvent`
- `OCP\Calendar\Events\CalendarObjectDeletedEvent`
- `OCP\Calendar\Events\CalendarObjectMovedToTrashEvent`
- `OCP\Calendar\Events\CalendarObjectRestoredEvent`
- `OCP\Calendar\Events\CalendarObjectUpdatedEvent`

```
{
  "event": {
    "class": "OCP\\Calendar\\Events\\CalendarObjectCreatedEvent",
    "calendarId": 9,
    "calendarData": {
      "id": 9,
      "uri": "work",
      "{http://calendarserver.org/ns/}getctag": "1736283",
      "{http://sabredav.org/ns}sync-token": 9651,
      "{urn:ietf:params:xml:ns:caldav}supported-calendar-component-set": "VEVENT,VTODO
↪",
      "{urn:ietf:params:xml:ns:caldav}schedule-calendar-transp": "opaque"
    },
    "shares": [
      {
        "href": "mailto:alice@example.com",
        "commonName": "Alice",
        "status": 2,
        "readOnly": false,
        "{http://owncloud.org/ns}principal": "principal:users/alice",
        "{http://owncloud.org/ns}group-share": false
      }
    ],
    "objectData": {
      "id": 22,
      "uri": "event-20251111T100000Z.ics",
      "lastmodified": 1700099500,
      "etag": "19fa45b394",
      "calendarid": 9,
      "size": 4096,
      "component": "VEVENT",
      "classification": 0
    }
  },
  "user": {
    "uid": "david",
    "displayName": "David"
  },
  "time": 1700100000
}
```


Distinct payload for two-calendar events

Applies to:

- `OCP\Calendar\Events\CalendarObjectMovedEvent`


```
{
  "event": {
    "class": "OCP\\Calendar\\Events\\CalendarObjectMovedEvent",
    "sourceCalendarId": 9,
    "sourceCalendarData": {
      "id": 9,
      "uri": "work",
```


(continues on next page)

(continued from previous page)


```
    },
    "targetCalendarId": 11,
    "targetCalendarData": {
      "id": 11,
      "uri": "meetings",
    },
    "sourceShares": [
      {
        "href": "mailto:alice@example.com",
        "commonName": "Alice"
      }
    ],
    "targetShares": [
      {
       "href": "mailto:bob@example.com",
       "commonName": "Bob"
      }
    ],
    "objectData": {
      "id": 22,
      "uri": "event-20251111T100000Z.ics"
    },
  },
  "user": {
    "uid": "david",
    "displayName": "David"
  },
  "time": 1700100000
}
```


11.5.5 Forms App Events

When the optional `forms` app is installed:

- `OCA\Forms\Events\FormSubmittedEvent`


```
{
  "event": {
    "class": "OCA\\Forms\\Events\\FormSubmittedEvent",
    "form": {
      "id": 51,
      "hash": "abc123def456",
      "title": "Employee Feedback",
      "description": "Annual employee feedback form.",
      "ownerId": "alice",
      "fileId": 1002,
      "fileFormat": "pdf",
      "created": 1700001000,
      "access": 0,
      "expires": 1702606600,
      "isAnonymous": false,
      "submitMultiple": false,
```


(continues on next page)

(continued from previous page)


```
      "showExpiration": true,
      "lastUpdated": 1700001200,
      "submissionMessage": null,
      "state": 1
    },
    "submission": {
      "id": 220,
      "formId": 51,
      "userId": "bob",
      "timestamp": 1700001234
    }
  },
  "user": {
    "uid": "bob",
    "displayName": "Bob"
  },
  "time": 1700001234,
}
```


11.5.6 Tables App Events

When the optional `tables` app is installed:

- `OCA\Tables\Event\RowAddedEvent`
- `OCA\Tables\Event\RowDeletedEvent`
- `OCA\Tables\Event\RowUpdatedEvent`


```
{
  "event": {
    "class": "OCA\\Tables\\Event\\RowAddedEvent",
    "tableId": 34,
    "rowId": 7,
    "previousValues": null,
    "values": {
      "0": "Project X",
      "1": 2025,
      "2": "active"
    }
  },
  "user": {
    "uid": "carol",
    "displayName": "Carol"
  },
  "time": 1700054321,
}
```


> **Note**
>
> For filtering or automation, always check the actual payload you receive, as it matches the JSON examples above, not PHPDoc or internal PHP array type style.

CHAPTER

TWELVE

WINDMILL WORKFLOWS

Nextcloud integrates the Windmill workflow engine to allow advanced custom workflows interacting with your Nextcloud instance.

12.1 Installation

- Set up a Windmill instance
  - This should be a stand-alone instance. The External App "Flow" (see *External Apps*) made it possible to have a packaged instance installed in Nextcloud, but this is deprecated since Nextcloud 33. For more information, please check out the admin manual for Nextcloud 32.
- Enable the `webhook_listeners` app that comes with Nextcloud

```
occ app:enable webhook_listeners
```

- Install the Windmill integration
- Enable *Pretty URLs* in your Nextcloud instance
- *Recommended but optional:* start *background job workers for the webhook listeners*

12.2 Setting up the workspace connection

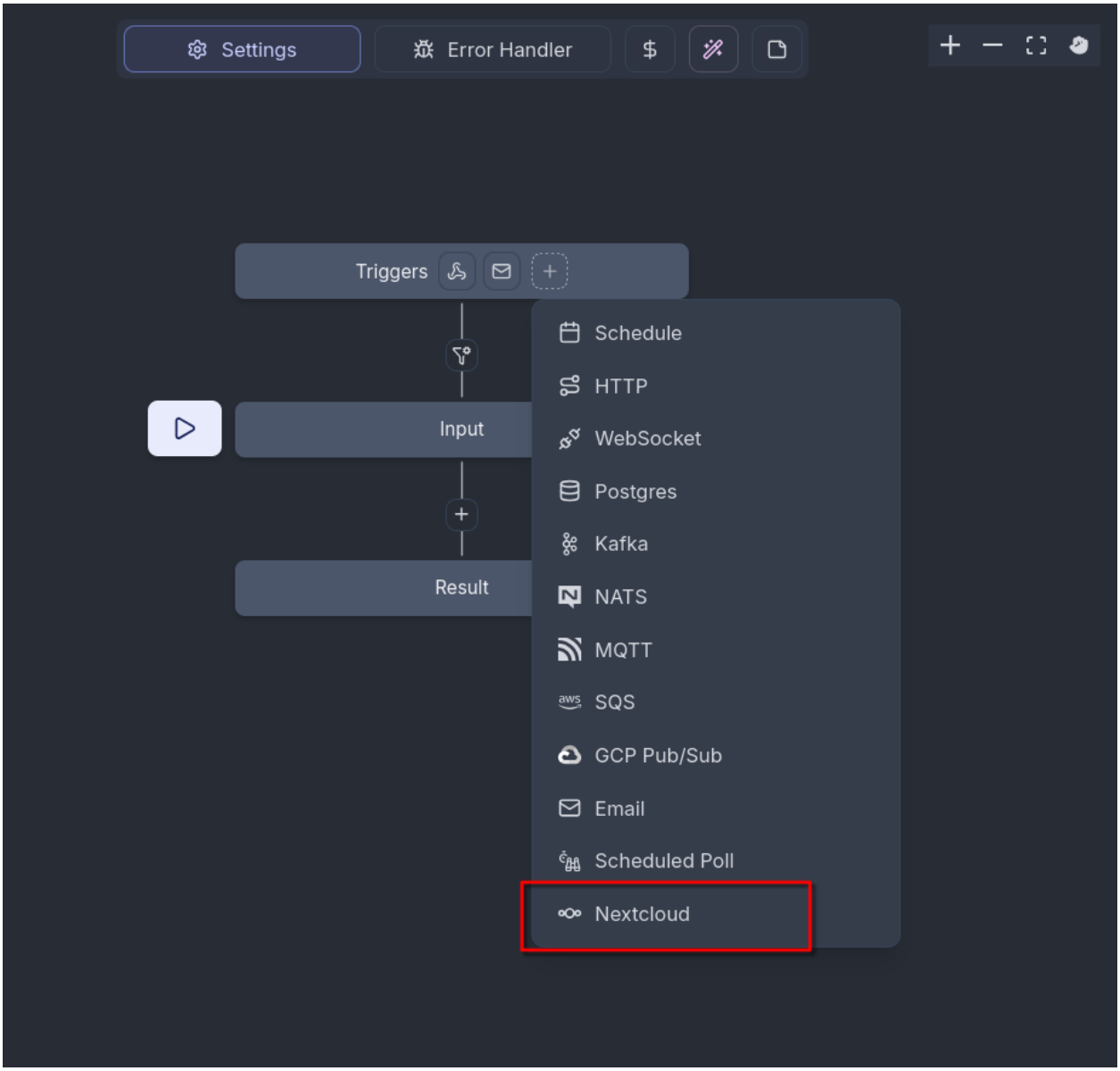
In Windmill, you have separate workspaces available. To enable Windmill to react to events happening on your Nextcloud instance, you need to connect a workspace to your Nextcloud connection. This is only possible for workspace admins.

- In Windmill, open the workspace settings of the workspace you want to connect to Nextcloud via Settings -> Workspace
- Go to the Native Triggers tab and choose Nextcloud -> Configure OAuth
- Follow the instructions to set up the OAuth connection
- Click on "Connect" and accept the OAuth connection **with a Nextcloud admin account** or an account that has webhook admin privileges
- If it shows "Connected", the workspace connection is successfully set up.

Every Flow configured in this workspace can now add Nextcloud triggers and scripts based on the credentials of the account you accepted the OAuth connection with.

12.3 Building a workflow

To make a workflow react to a Nextcloud Webhook Event, you need to add a trigger. To do this, click on the Plus in the Triggers box of the Flow and select “Nextcloud”



Now you can choose an event out of a drop-down list of events that your flow should react to. You can additionally fill in some parameters:

- *Event filters* allows more fine grained filtering for which events should be used. The filter condition as well as the available events with their payloads is documented in the *webhook_listeners documentation*.
- The *User ID filter* allows to define the user that can trigger a flow with their actions in Nextcloud. The webhook will only be called by requests from this user. Empty or null means no filtering.
- The *Headers* field allows to define an array of headers to be sent in a webhook call, which will mostly not be needed.

You can add more than one trigger to a flow.

12.4 Nextcloud Scripts

Nextcloud makes available a variety of scripts to be used in Windmill for interfacing with Nextcloud apps. You can find them at https://hub.windmill.dev/integrations/nextcloud and https://hub.windmill.dev/integrations/nextcloud/approvals or in your windmill instance when selecting existing scripts for creating a new workflow.

If you want to use a function that is not already represented there, you can easily write your own scripts using a skeleton script and our OCS API that provides lots of endpoints. You can check out the available endpoints in the Nextcloud OCS API documentation or you can install our OCS API Viewer app and check it out directly in your Nextcloud.

This is a skeleton for a script written in TypeScript (Bun). To use it, you have to add an action in Windmill and choose the correct script language. It includes everything necessary to use the Nextcloud OCS API, the parts marked like `$THIS` have to be filled in with your data.

```
import createClient, { type Middleware } from "openapi-fetch";

export async function main(
   nextcloud: RT.Nextcloud,
   $PARAMETER: $TYPE,
   // add any input parameters you need here
) {

   // this part is the same for any script and should  not be changed

   const client = createClient<paths>({ baseUrl: nextcloud.baseUrl });
   const authMiddleware: Middleware = {
      async onRequest({ request, options }) {
         // fetch token, if it doesn't exist
         // add Authorization header to every request
         request.headers.set("Authorization", `Basic ${btoa(nextcloud.userId + ':' +␣
↪nextcloud.token)}`);
         return request;
      },
   };
   client.use(authMiddleware);

   //starting here you can adapt the script

   data = await client.GET("/ocs/v2.php/apps/$APP/$PATH/{$PATHPARAMETER}", {
      params: {
         header: {
            "OCS-APIRequest": true,
         },
         query: {
            format: "json",
         },
         path: {
            $PATHPARAMETER: "$VALUE",
         },

      },
      body: {
         $BODYPARAMETER: $VALUE,
      },
```

(continues on next page)

(continued from previous page)

```
    });

return data;
}
```

The endpoint path you have to fill in is provided in the API documentation.

OCS uses two kinds of parameters: Path parameters that are part of the endpoint URL, and body parameters that are transmitted in the request body. In the example script, you can provide both kinds in the function parameters.

Remember to also adapt the HTTP method (GET, POST, PUT etc.) if needed.

The `data` variable receives the response of the HTTP request, so any data or error messages coming from the Nextcloud instance is available in there.

12.4.1 Authentication

All the scripts we provide have one input parameter in common: *nextcloud* needs to be an object of the type "Nextcloud" and contain what is necessary to authenticate against Nextcloud:

- baseUrl: the URL your instance is reachable at, e.g. `https://example.cloud`
- userId: the user id of the user the script should authenticate with
- token: a password or token for that user

We advise to either add these credentials as a resource of the Nextcloud type to your workspace and refer to the resource in your script, or use the authentication credentials that are provided in the webhook callback. For every flow that is triggered by a Nextcloud event, there are 2 sets of temporary credentials included:

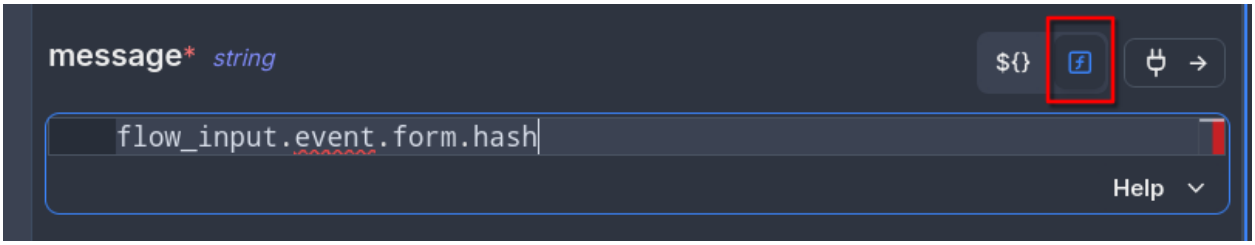
- the credentials for the user account that is saved in the initial OAuth connection, available as `flow_input.authentication.owner`
- the credentials for the user account that triggered the event with their action, available as `flow_input.authentication.trigger`

These temporary authentication credentials are valid for one hour after triggering the event.

12.4.2 Passing values between blocks

When specifying script inputs you can either fill the parameters with static values or make references to the workflow input and the previous workflow steps.

In order to reference the workflow input (details about the event that triggered the webhook), use the `flow_input` variable. For example, `flow_input.event.form.hash` will reference the hash of a form from a nextcloud Forms event. As it is a JavaScript expression and not a static value, you have to switch the parameter input with the button next to it.



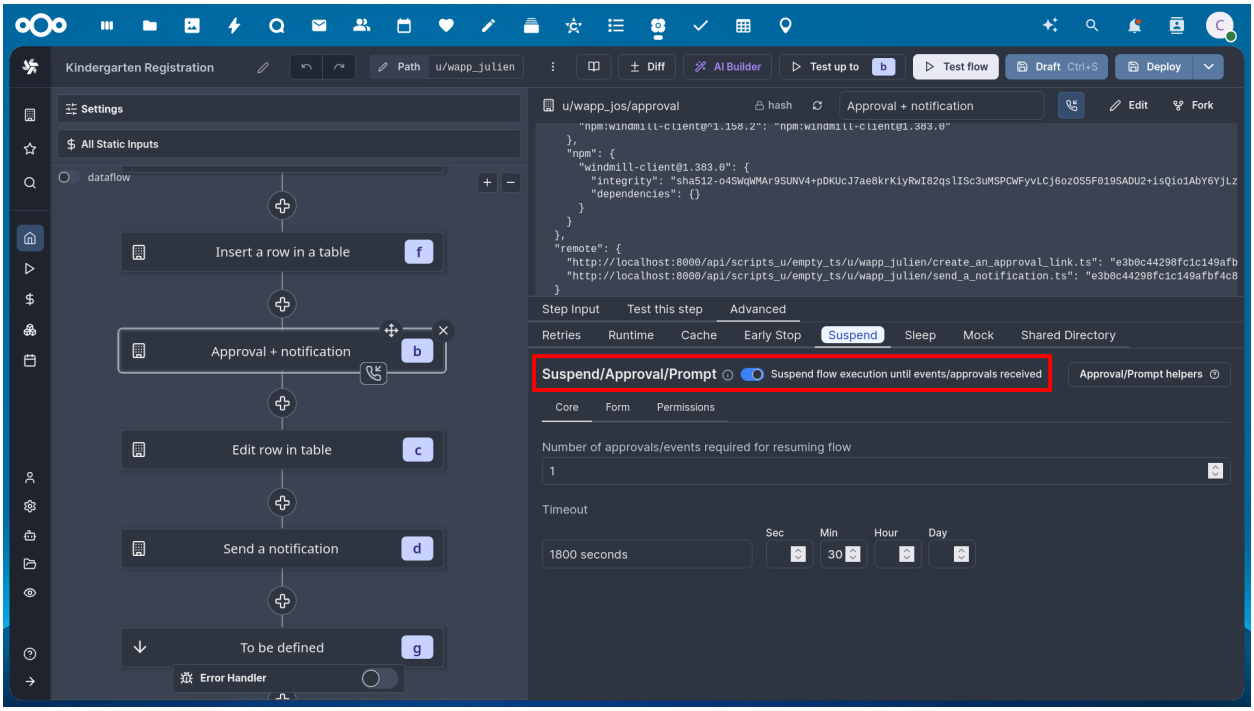
The fields available for each event are listed in the *webhook_listeners documentation*.

Each step in a workflow is automatically assigned a letter identifier. In order to reference results from previous steps in your parameters, use the `results` variable with the id of the step to reference as a sub property. For example, use `results.e.submission.answers` to use the answers of of a form submission retrieved via the "Get form submission from Nextcloud Forms" script identified with the letter "e". You can identify the letters in the flow overview diagram.

12.4.3 Approval/Suspend steps

Windmill allows using so-called approval steps, which are essentially asynchronous scripts that wait for the call to an additional webhook URL. The most prominent use case for this are approval workflows where you get automated input from somewhere which needs to be approved by a human. Once the human approves or disapproves by triggering the webhook URL the workflow will resume.

In order to turn a newly added step into an approval step, the workflow edit screen, select the script and in the step panel, go in the "Advanced" tab, "Suspend" sub tab and check "Suspend/Approval/Prompt".



Using the scripts provided for Nextcloud, you can send approval links to the humans in charge of approving via Nextcloud Talk or a simple notification in Nextcloud. Of course, you may also use any of the other scripts for sending messages available in the Windmill hub.

Windmill has a default approval user interface at a specific URL, but it looks very technical. We recommend using the approve_links app which allows creating a beautiful temporary approval page with a custom message and approve and disapprove buttons.

12.4.4 Example workflows

We provide some example workflows at https://hub.windmill.dev/integrations/nextcloud/flows showcasing the possibilities of combining Windmill workflows with Nextcloud AI.

12.5 FAQ

12.5.1 Can I create a script?

If the Windmill Hub does not contain any script to perform the action you have in mind, you can take an existing Nextcloud script as example and create your own. Your custom scripts can make requests to any endpoint of the Nextcloud OCS API .

CHAPTER

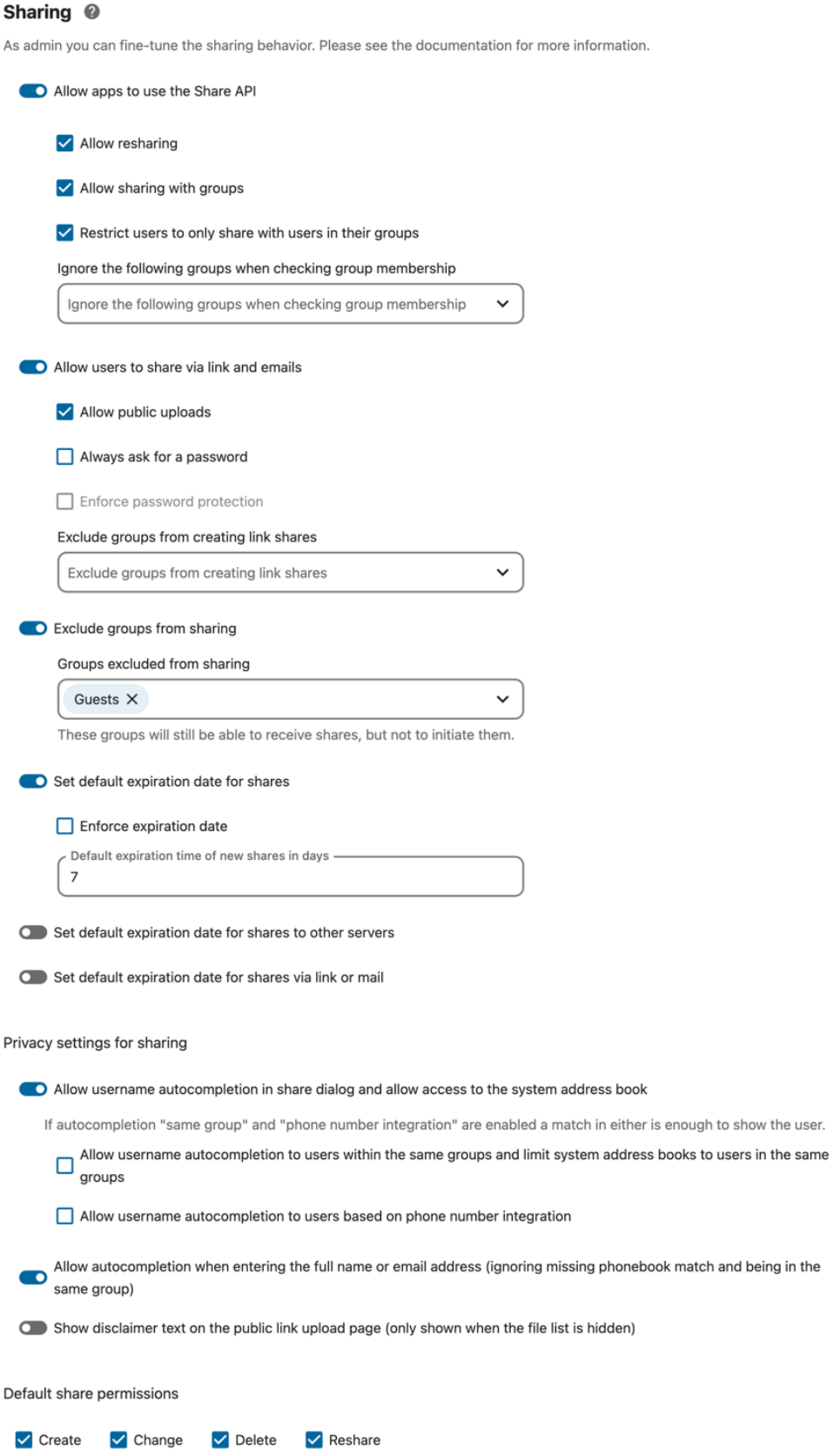
THIRTEEN

FILE SHARING AND MANAGEMENT

13.1 File Sharing

Nextcloud users can share files with their Nextcloud groups and other users on the same Nextcloud server, with Nextcloud users on *other Nextcloud servers*, and create public shares for people who are not Nextcloud users. You have control of a number of user permissions on file shares.

Configure your sharing policy on your Admin page in the Sharing section.


Sharing
As admin you can fine-tune the sharing behavior. Please see the documentation for more information.
Allow apps to use the Share API
Allow resharing
Allow sharing with groups
Restrict users to only share with users in their groups
Ignore the following groups when checking group membership
Ignore the following groups when checking group membership
Allow users to share via link and emails
Allow public uploads
Always ask for a password
Enforce password protection
Exclude groups from creating link shares
Exclude groups from creating link shares
Exclude groups from sharing
Groups excluded from sharing
Guests
These groups will still be able to receive shares, but not to initiate them.
Set default expiration date for shares
Enforce expiration date
Default expiration time of new shares in days
7
Set default expiration date for shares to other servers
Set default expiration date for shares via link or mail
Privacy settings for sharing
Allow username autocompletion in share dialog and allow access to the system address book
If autocompletion "same group" and "phone number integration" are enabled a match in either is enough to show the user.
Allow username autocompletion to users within the same groups and limit system address books to users in the same groups
Allow username autocompletion to users based on phone number integration
Allow autocompletion when entering the full name or email address (ignoring missing phonebook match and being in the same group)
Show disclaimer text on the public link upload page (only shown when the file list is hidden)
Default share permissions
Create
Change
Delete
Reshare

- 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 resharing` to enable users to re-share files shared with them.
  - Check `Allow sharing with groups` to enable users to share with groups.
  - Check `Restrict users to only share with users in their groups` to confine sharing within group memberships. When you check this, you'll get an optional dropdown list of ignored groups when checking group membership. Type any group name to search for.
    * Groups added to `Ignore the following groups when checking group membership` won't be taken in account to determine if users are in same groups and may share with each others.

> **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 share via link and email` to enable creating public shares for people who are not Nextcloud users via hyperlink.
  - Check `Allow public uploads` to allow anyone to upload files to public shares.
  - Check `Always ask for a password` to proactively ask a user to set a password for a share link.
  - Check `Enforce password protection` to force users to set a password on all public share links. This does not apply to local user and group shares.
  - Add groups to `Exclude groups from creating link shares` to no apply the settings for that groups.
- 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. Type any group name to search for. Members of excluded groups can still receive shares, but not create any.
- Check `Set default expiration date for shares` to set a default expiration date on local user and group shares.
  - Check `Enforce expiration date` to always enforce the configured expiration date on local user and group shares.

> **Note**
>
> Users will not be able to set the expiration date further in the future than the enforced expiration date, although they will be able to set a more recent date. Also note that users will be able to update the expiration date again at a later point. The expiration date is based on the current date and not on the share creation date. The user will be able to extend the expiration date again whenever a previous expiration date is close to be reached.

- Check `Set default expiration date for shares via link or email` to set a default expiration date on public shares.
  - Check `Enforce expiration date` to always enforce the configured expiration date on public shares.

> **Note**
>
> Users will not be able to set the expiration date further in the future than the enforced expiration date, although they will be able to set a more recent date. Also note that users will be able to update the expiration date again at a later point. The expiration date is based on the current date and not on the share creation date. The user will be able to extend the expiration date again whenever a previous expiration date is close to be reached.

- Check `Allow username autocompletion in share dialog and allow access to the system address book` to enable auto-completion of Nextcloud usernames and list the system address book as resource when syncing contacts with CardDAV.
  - Check `Allow username autocompletion to users within the same groups and limit system address books to users in the same groups` to limit username autocompletion to users from within the same groups as the share owner.
  - Check `Allow username autocompletion to users based on phone number integration` to limit username autocompletion to users when the share owner has synced their phone address book via the Nextcloud Talk mobile clients and it contained the phone number the user configured in their profile.
- Check `Allow autocompletion when entering the full name or email address (ignoring missing phonebook match and being in the same group)` to show despite of the previous restrictions a user suggestion, when the complete display name or user id was typed.
- Check `Show disclaimer text on the public link upload page` to set and show a disclaimer text on public links with hidden file lists. If you enable this feature a text input will be shown to input the disclaimer text.

With `Default share permissions` you are able to set the default permissions for user-shares (`Create`, `Change`, `Delete` and `Reshare`) without forcing them.

> **Note**
>
> Nextcloud does not preserve the mtime (modification time) of directories, though it does update the mtimes on files. See Wrong folder date when syncing for discussion of this.

> **Note**
>
> There are more sharing options on config.php level available: Configuration Parameters

13.1.1 Advanced settings

Here are some edge case settings which are not editable from the web interface, because they are only useful to small subset of administrators.

You can use the `occ` command to update those, for example:

```
occ config:app:set core shareapi_restrict_user_enumeration_full_match_email --value␣
↪yes
```

- **`core.shareapi_restrict_user_enumeration_full_match_ignore_second_display_name`**
  - When full match is activated, ignore the appended second display name.
  - Default: `no`

  - Examples:

Setting value	Search query	User name	Will match
`yes`	User 1	User 1 (Second display name)	yes
`no`	User 1	User 1 (Second display name)	no

- **`core.shareapi_restrict_user_enumeration_full_match_userid`**
  - When full match is activated, do not match user ID
  - Default: `yes`
- **`core.shareapi_restrict_user_enumeration_full_match_email`**
  - When full match is activated, do not match user email
  - Default: `yes`

13.1.2 Distinguish between max expiration date and default expiration date

The expiration date which can be set and enforced in the settings above are the hard limit and the default value at the same time. Sometimes admins want to have a moderate default expire date, for example 7 days but make sure that the user can't extend it to more than 14 days.

In order to do so, set a enforced expiration date in the settings as described above and set the default value to something below the maximal possible expiration date with the following OCC commands:

```
occ config:app:set --value <DAYS> core internal_defaultExpDays
occ config:app:set --value <DAYS> core link_defaultExpDays
```

13.1.3 Get a notification before a share expires

Users can get a notification before a share expires. In order to do so a cronjob need to be configured which calls the following OCC command once a day:

```
occ sharing:expiration-notification
```

A notification will be send for all shares which expire within the next 24 hours.

13.1.4 Transferring files to another user

You may transfer files from one user to another with `occ`. This is useful when you have to remove a user. Be sure to transfer the files before you delete the user! This transfers all files from user1 to user2, and the shares and metadata info associated with those files (shares, tags, comments, etc). Trashbin contents are not transferred:

```
occ files:transfer-ownership user1 user2
```

(See *Using the occ command* for a complete `occ` reference.)

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

13.1.5 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.

13.1.6 Using File Drop Share links

Using a File Drop Share allows users to upload files to Nextcloud through an unauthenticated session. File Drop Share links will only work when `Allow public uploads` is checked in the Sharing section of the Administration Settings page.

> **Note**
>
> File Drop Shares currently have a limitation in that any files uploaded through an unauthenticated session will not be chunked. Therefore the maximum file size that can be uploaded through File Drop Shares depends entirely on settings set within your environment.

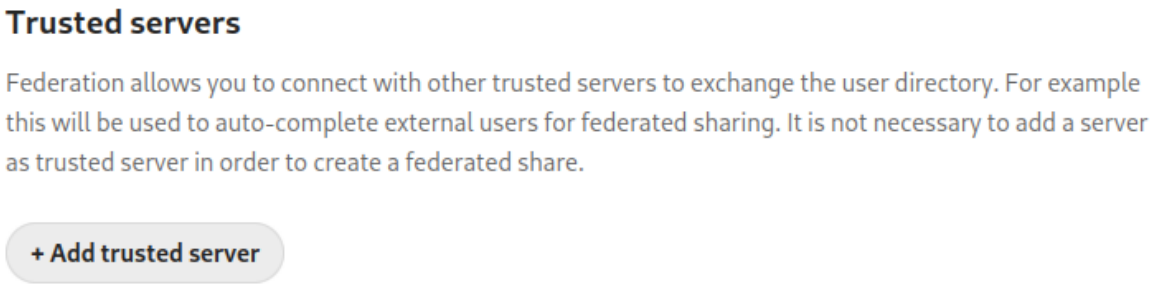
13.2 Configuring Federation Sharing

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.

13.2.1 Creating a new Federation Share

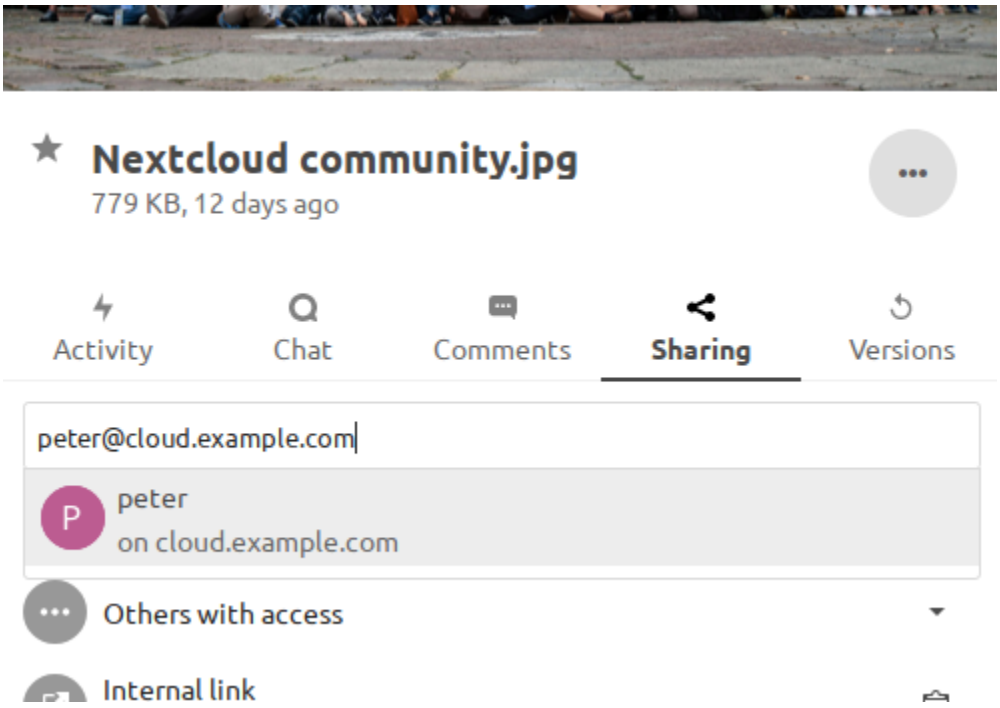
Follow these steps to create a new Federation share between two Nextcloud servers. This requires no action by the user on the remote server; all it takes is a few steps on the originating server.

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. 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.

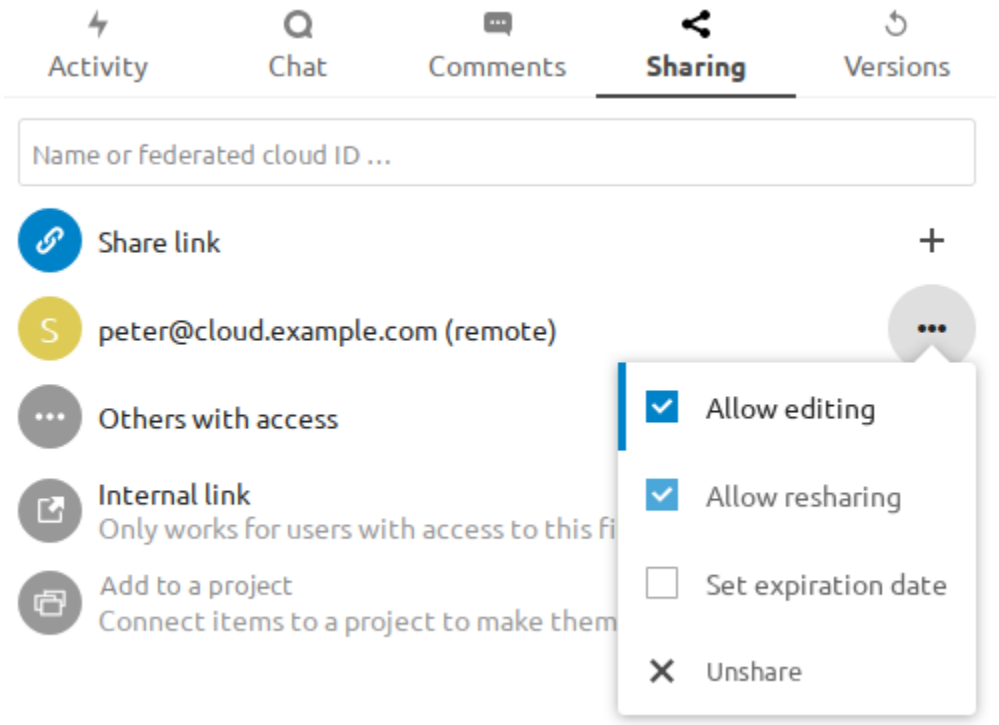


4. Now go to your Files page and select a folder to share. Click the share icon, and then enter the username and URL of the user on the remote Nextcloud server. In this example, that is `freda@https://example.com/nextcloud`.

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.

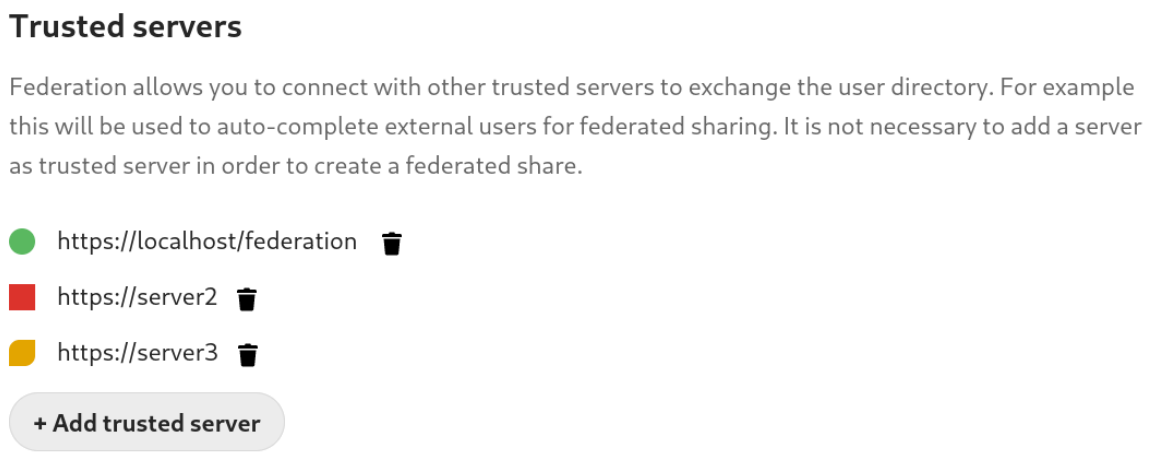
13.2.2 Configuring trusted Nextcloud servers

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.

You may also enter Nextcloud server URLs in the **Add Nextcloud Server** field.

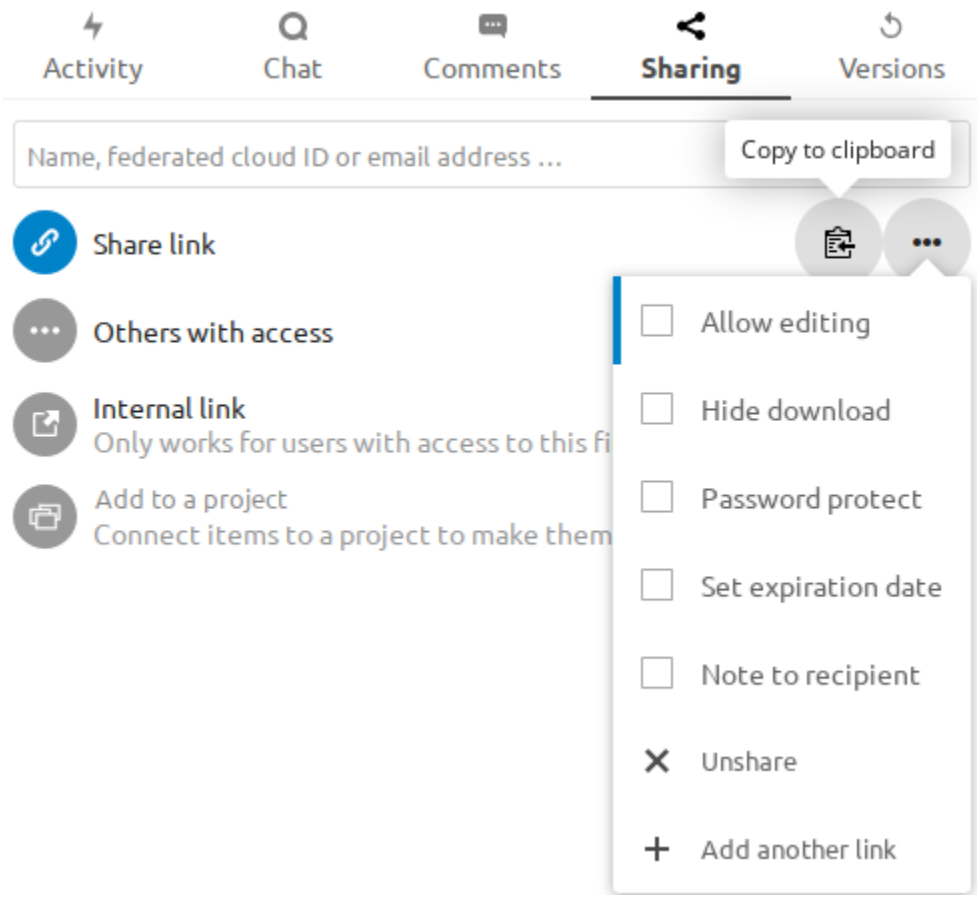
A red light means the connection failed. The yellow light indicates a successful connection, with no user names exchanged. The green light indicates a successful connection with user names exchanged.

The prerequisiste for a green status is that the trusted servers were maintained in both interacting Nextcloud servers. Additionally `occ federation:sync-addressbooks` must have been executed (part of cron job list). The delay to execute the cron is based on local configuration of the cron frequency.

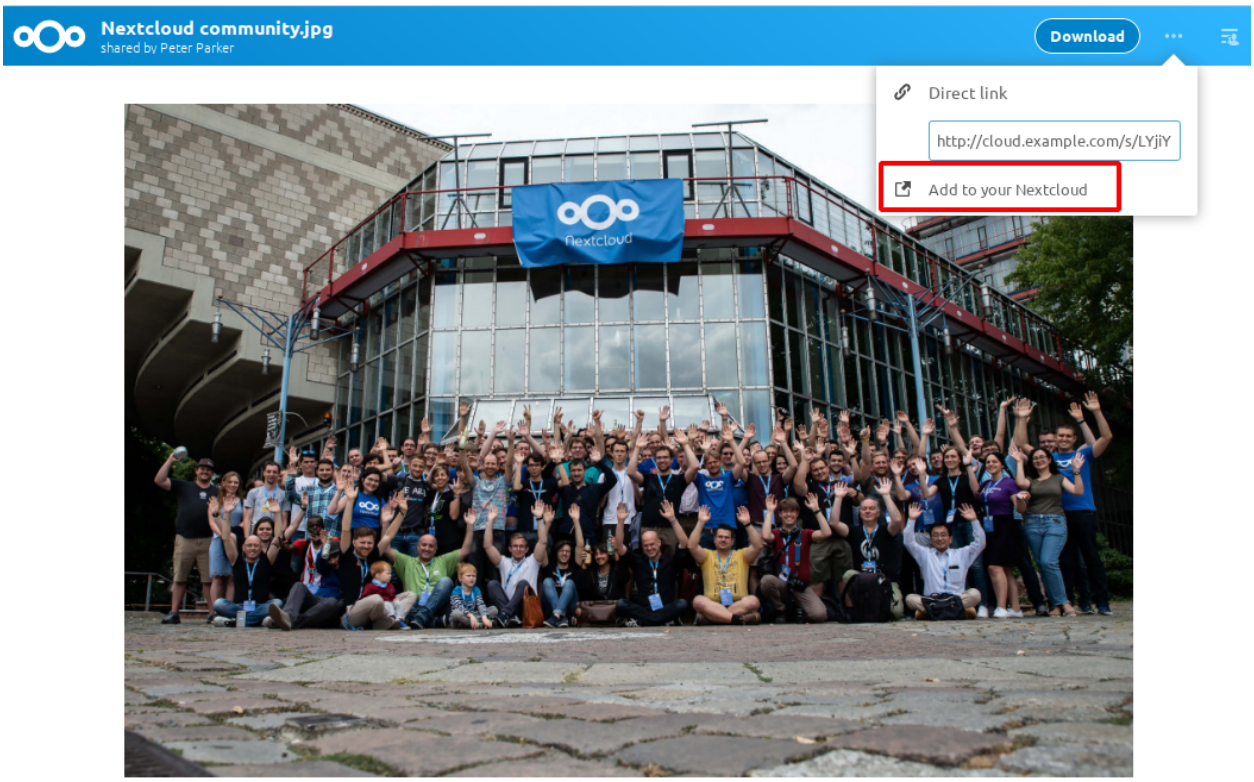


13.2.3 Creating Federation Shares via public Link Share

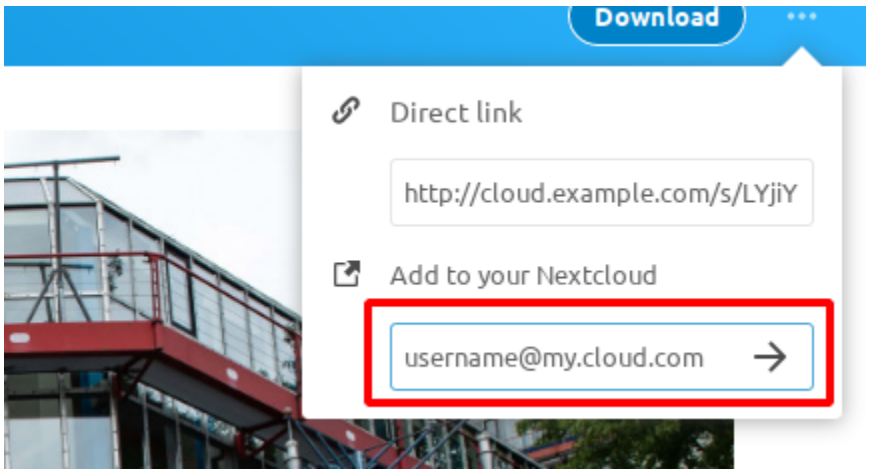
Check the `Share link` entry to expose more sharing options (which are described more fully in *File Sharing*). You may create a Federation share by allowing Nextcloud to create a public link for you, and then email it to the person you want to create the share with.



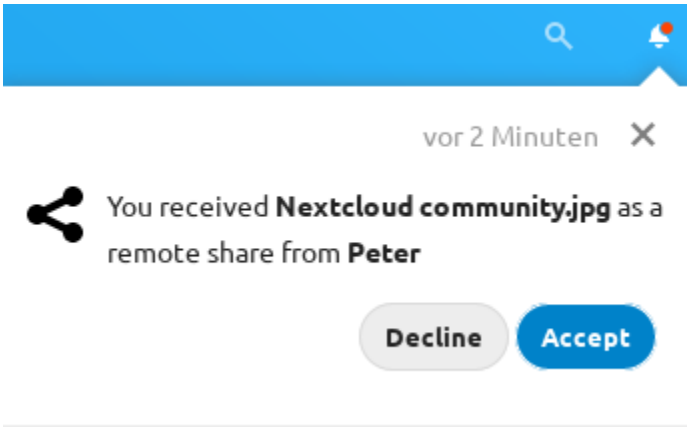
You may optionally set a password and expiration date on it. When your recipient receives your email they must click the link, or copy it to a Web browser. They will see a page displaying a thumbnail of the file, with a button to **Add to your Nextcloud**.



Your recipient should click the **Add to your Nextcloud** button. On the next screen your recipient needs to enter the URL to their Nextcloud server, and then press the return key.



Your recipient has to take one more step, and that is to confirm creating the federated cloud share link by clicking the **Accept** button.



Un-check the `Share link` checkbox to disable any federated cloud share created this way.

13.2.4 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.
- If you encounter timeouts for downloading or uploading large files, you can use the option `davstorage.request_timeout` in your `config.php` to increase the timeout. The default value is 30 seconds.

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`.

13.2.5 Changing the display of federated shares

By default, federated shares are displayed in a separate section in the Nextcloud interface. It is possible to change this behavior and display them in the same section as internal shares. This can be controlled with a `occ` command:

```
occ config:app:set --value false --type boolean files_sharing show_federated_shares_
↪as_internal
```

Set the value to `true` to display federated shares mixed with internal shares, or `false` to keep them in a separate section (default).

13.3 Uploading big files > 512MB

The default maximum file size for uploads is 512MB. You can increase this limit up to what your filesystem and operating system allows. There are certain hard limits that cannot be exceeded:

- < 2GB on 32Bit OS-architecture
- < 2GB with IE6 - IE8
- < 4GB with IE9 - IE11

64-bit filesystems have much higher limits; consult the documentation for your filesystem.

Note

The Nextcloud sync client is not affected by these upload limits as it is uploading files in smaller chunks. See Client documentation for more information on configuration options.

13.3.1 System configuration

- Make sure that the latest version of PHP is installed
- Disable user quotas, which makes them unlimited
- Your temp file or partition has to be big enough to hold multiple parallel uploads from multiple users; e.g. if the max upload size is 10GB and the average number of users uploading at the same time is 100: temp space has to hold at least 10x100 GB

13.3.2 Configuring your Web server

Note

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

Set the following two parameters inside the corresponding php.ini file (see the **Loaded Configuration File** section of *PHP version and information* to find your relevant php.ini files)

```
php_value upload_max_filesize 16G
php_value post_max_size 16G
```

The `upload_max_filesize` and `post_max_size` settings may not apply to file uploads through WebDAV single file PUT requests or Chunked file uploads For those, PHP and webserver timeouts are the limiting factor on the upload size.

Adjust these values for your needs. If you see PHP timeouts in your logfiles, increase the timeout values, which are in seconds:

```
php_value max_input_time 3600
php_value max_execution_time 3600
```

The mod_reqtimeout Apache module could also stop large uploads from completing. If you're using this module and getting failed uploads of large files either disable it in your Apache config or raise the configured `RequestReadTimeout` timeouts.

There are also several other configuration options in your Web server config which could prevent the upload of larger files. Please see the manual of your Web server for how to configure those values correctly:

Apache

- LimitRequestBody (In Apache HTTP Server <=2.4.53 this defaulted to unlimited, but now defaults to 1 GiB. The new default limits uploads from non-chunking clients to 1 GiB. If this is a concern in your environment, override the new default by either manually setting it to `0` or to a value similar to that used for your local environment's PHP `upload_max_filesize / post_max_size / memory_limit` parameters.)
- SSLRenegBufferSize
- Timeout

Apache with mod_fcgid

- FcgidMaxRequestInMem
- FcgidMaxRequestLen

Note

If you are using Apache/2.4 with mod_fcgid, as of February/March 2016, `FcgidMaxRequestInMem` still needs to be significantly increased from its default value to avoid the occurrence of segmentation faults when uploading big files. This is not a regular setting but serves as a workaround for Apache with mod_fcgid bug #51747.

Setting `FcgidMaxRequestInMem` significantly higher than normal may no longer be necessary, once bug #51747 is fixed.

Apache with mod_proxy_fcgi

- ProxyTimeout

nginx

- client_max_body_size
- fastcgi_read_timeout [often the solution to 504 timeouts during `MOVE` transactions that occur even when using chunking]
- client_body_temp_path

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.

13.3.3 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`

13.3.4 Configuring Nextcloud

As an alternative to the `upload_tmp_dir` of PHP (e.g. if you don't have access to your `php.ini`) you can also configure a temporary location for uploaded files by using the `tempdirectory` setting in your `config.php` (See *Configuration Parameters*).

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

13.3.5 Adjust chunk size on Nextcloud side

For upload performance improvements in environments with high upload bandwidth, the server's upload chunk size may be adjusted:

```
sudo -E -u www-data php occ config:system:set --type int --value 20971520 files.
↪chunked_upload.max_size
```

Put in a value in bytes (in this example, 20MB). Set `--value 0` for no chunking at all.

Default is `104857600` (100 MiB).

13.3.6 Large file upload on object storage

Chunked file uploads do have a larger space consumption on the temporary folder when processing those uploads on object storage as the individual chunks get downloaded from the storage and will be assembled to the actual file on the Nextcloud servers temporary directory. It is recommended to increase the size of your temp directory accordingly and also ensure that request timeouts are high enough for PHP, webservers or any load balancers involved.

Tip

In more recent versions of Nextcloud Server, when uploading to S3 in *Primary Storage* mode, we use S3 *MultipartUpload*. This allows chunked upload streaming of the chunks directly to S3 so that the final MOVE request no longer needs to assemble the final file on the Nextcloud server. This requires your `memcache.distributed` to be set to

use Redis (or Memcached), otherwise we fall back on the prior behavior which consumes space on the Nextcloud Server for file assembly (as described above).

13.3.7 Federated Cloud Sharing

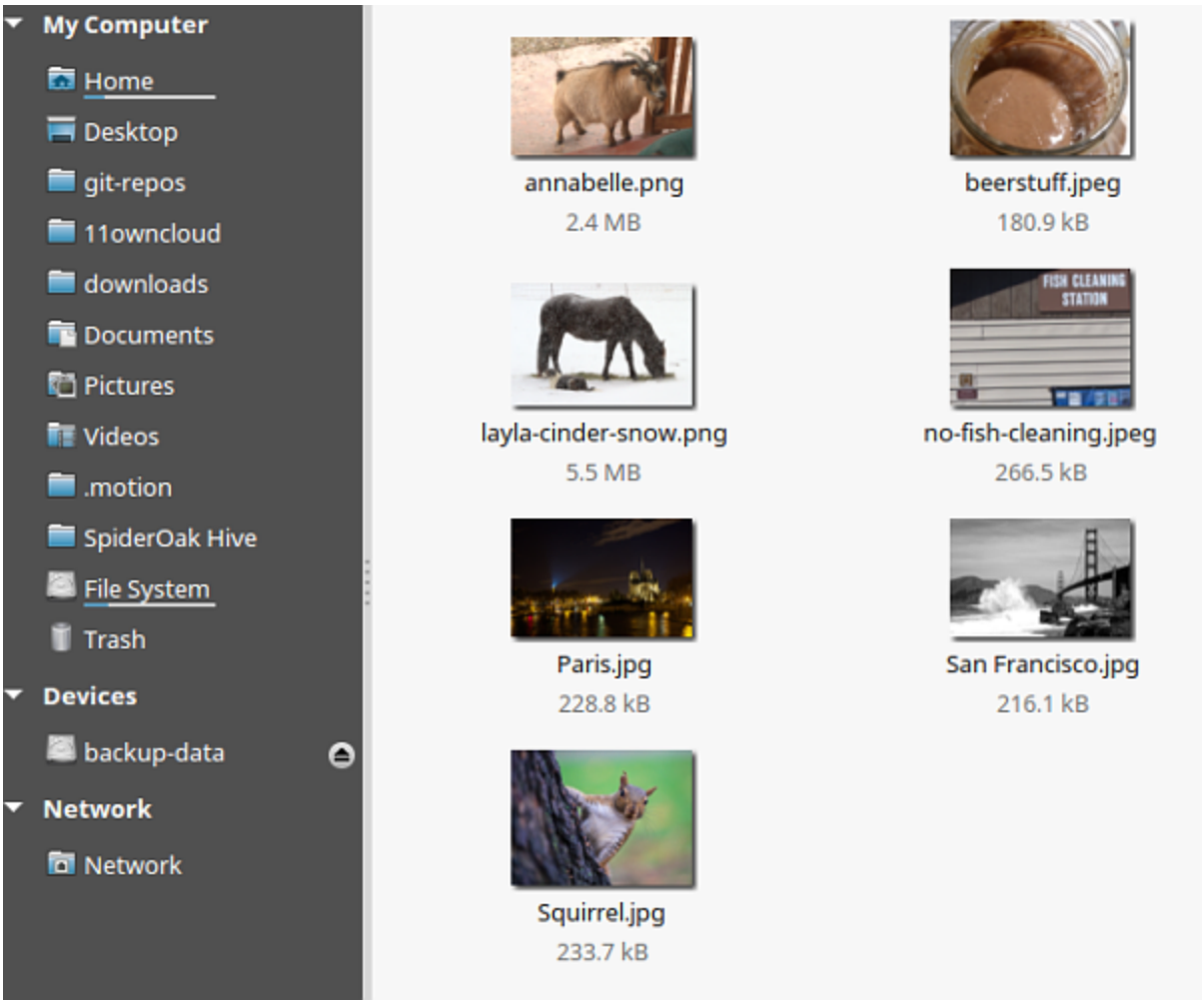
If you are using *Federated Cloud Sharing* and want to share large files, you can increase the timeout values for requests to the federated servers. Therefore, you can set `davstorage.request_timeout` in your `config.php`. The default value is 30 seconds.

13.4 Providing default files

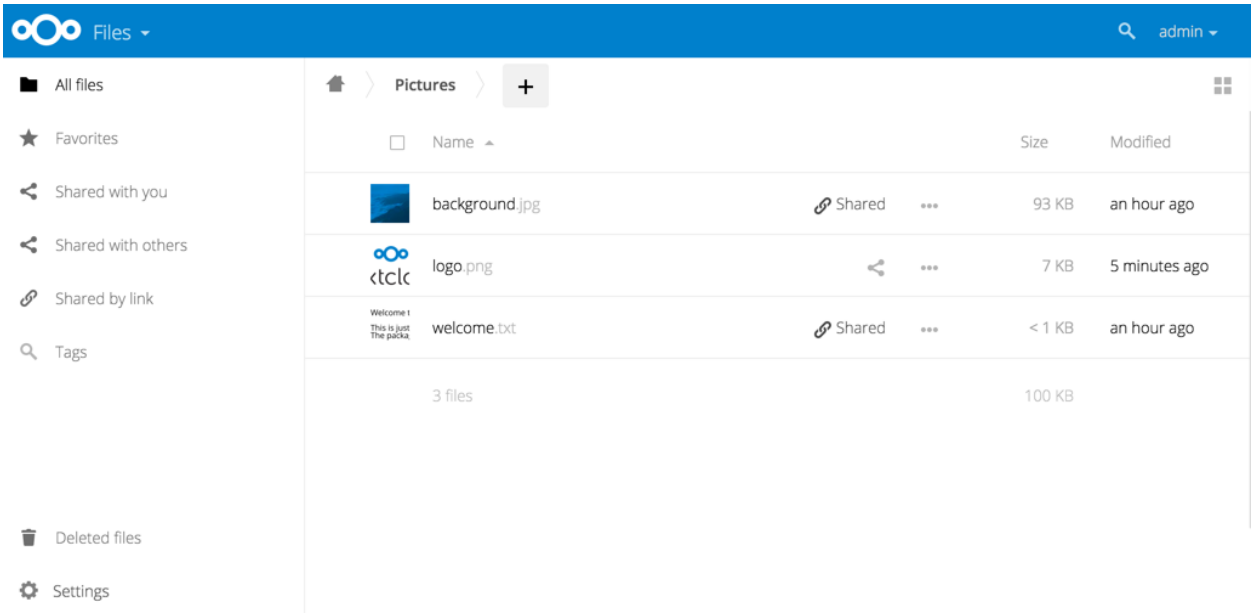
You may distribute a set of default files and folders to all users by placing them in directory that is readable by the webserver user. This allows you to overwrite the files that are shipped by default with Nextcloud in `core/skeleton`. That custom directory should then be configured in the `config.php` via the configuration option `skeletondirectory` (see *Configuration Parameters*). Leave empty to not copy any skeleton files.

These files will be copied only to new users after their initial login, and existing users will not see files that are added to this directory after their first login. The files in the `skeleton` directory are copied into the users data directories, so they may change and delete the files without affecting the originals.

This screenshot shows a set of photos in the `skeleton` directory.



They appear on the user's Nextcloud Files page just like any other files.



Note

Overwriting the files in `core/skeleton` is not recommended, because those changes will be overwritten on the next update of the Nextcloud server.

13.4.1 Default file templates

The default path for user templates is at `/Templates` (translated in the user's language). If you need to override this path for all users, you can set

```
occ config:app:set core defaultTemplateDirectory --value="CustomPath"
```

This will only apply to new users.

Note

To create their own template directory, users should click the `+ New` button and then `+ Create templates folder`.

13.5 Configuring Object Storage as Primary Storage

Nextcloud allows to configure object storages like OpenStack Swift or Amazon Simple Storage Service (S3) or any compatible S3-implementation (e.g. Minio or Ceph Object Gateway) as primary storage replacing the default storage of files.

By default, files are stored in `nextcloud/data` or another directory configured in the `config.php` of your Nextcloud instance. This data directory might still be used for compatibility reasons)

13.5.1 Differences from External Storage

When an object store is used as Primary Storage, Nextcloud requires exclusive access over the bucket being used. All metadata (filenames, directory structures, etc) is stored in Nextcloud and not in the object store. The metadata is only stored in the database and the object store only holds the file content by unique identifier.

Performance Implications

Because of this, object stores configured as Primary Storage usually perform better than when using the same object store via the External Storage support application, but the downside is being unable to access the files from outside of Nextcloud. This makes using an object store as Primary Storage distinct from using an object store via External Storage.

Data Backup and Recovery Implications

One impact of using an object store as Primary Storage is that your data backup strategy needs to incorporate this. **Your data is no longer stored on your Nextcloud server, but your files are also no longer accessible by simply bypassing your Nextcloud server and accessing your object store directly.**

13.5.2 Configuration

Primary object stores need to be configured in `config.php` by specifying the objectstore backend and any backend specific configuration.

> **Note**
>
> Configuring a primary object store on an existing Nextcloud instance will make all existing files on the instance inaccessible.

The configuration has the following structure:

```
'objectstore' => [
    'class' => 'Object\\Storage\\Backend\\Class',
    'arguments' => [
        ...
    ],
],
```

OpenStack Swift

The OpenStack Swift backend mounts a container on an OpenStack Object Storage server into the virtual filesystem.

The class to be used is `\OC\Files\ObjectStore\Swift`

Both openstack v2 and v3 authentication are supported,

V2 Authentication:

```
'objectstore' => [
    'class' => '\\OC\\Files\\ObjectStore\\Swift',
    'arguments' => [
        'username' => 'username',
        'password' => 'Secr3tPaSSWoRdt7',
        // the container to store the data in
        'bucket' => 'nextcloud',
        'autocreate' => true,
```

(continues on next page)

(continued from previous page)

```
        'region' => 'RegionOne',
        // The Identity / Keystone endpoint
        'url' => 'http://example.com/v2.0',
        // optional on some swift implementations
        'tenantName' => 'username',
        'serviceName' => 'swift',
        // The Interface / url Type, optional
        'urlType' => 'internal'
    ],
],
```

V3 Authentication:

```
'objectstore' => [
    'class' => 'OC\\Files\\ObjectStore\\Swift',
    'arguments' => [
        'autocreate' => true,
        'user' => [
            'name' => 'UserName',
            'password' => 'Secr3tPaSSWoRdt7',
            'domain' => [
                'name' => 'Default',
            ],
        ],
        'scope' => [
            'project' => [
                'name' => 'TenantName',
                'domain' => [
                    'name' => 'Default',
                ],
            ],
        ],
        'serviceName' => 'swift',
        'region' => 'regionOne',
        'url' => 'http://example.com/v3',
        'bucket' => 'nextcloud',
    ],
],
```

Simple Storage Service (S3)

The Simple Storage Service (S3) backend mounts a bucket on an Amazon S3 object storage or compatible implementation (e.g. Minio or Ceph Object Gateway) into the virtual filesystem.

The class to be used is `\OC\Files\ObjectStore\S3`

Amazon-hosted S3:

```
'objectstore' => [
    'class' => '\\OC\\Files\\ObjectStore\\S3',
    'arguments' => [
        'bucket' => 'my-nextcloud-store',
        'region' => 'us-east-1',
```

(continues on next page)

(continued from previous page)

```
        'key' => 'EJ39ITYZEUH5BGWDRUFY',
        'secret' => 'M5MrXTRjkyMaxXPe2FRXMTfTfbKEnZCu+7uRTVSj',
    ],
],
```

Non-Amazon hosted S3:

```
'objectstore' => [
    'class' => '\\OC\\Files\\ObjectStore\\S3',
    'arguments' => [
        'bucket' => 'my-nextcloud-store',
        'hostname' => 's3.example.com',
        'key' => 'EJ39ITYZEUH5BGWDRUFY',
        'secret' => 'M5MrXTRjkyMaxXPe2FRXMTfTfbKEnZCu+7uRTVSj',
        'port' => 8443,
        // required for some non-Amazon S3 implementations
        'use_path_style' => true,
    ],
],
```

Minimum required parameters are:

- `bucket` [Note: Even if non-Amazon hosted, bucket names must meet AWS S3 naming requirements regardless of what your S3 provider/platform considers acceptable - i.e. no underscores]
- `key`
- `secret`

Note

You will *probably* need to specify additional parameters beyond these, unless the default values (see below) exactly match your situation. In particular, your `region` (if Amazon hosted) or `hostname` (if non-Amazon hosted).

Optional parameters most commonly needing adjustment (and their defaults values if left unconfigured):

- `region` defaults to `eu-west-1`
- `storageClass` defaults to `STANDARD`
- `hostname` defaults to `s3.REGION.amazonaws.com` [Note: If using this parameter (non-Amazon), specify the generic S3 endpoint hostname, **not** the hostname that contains your bucket name]
- `use_ssl` defaults to `true`

Optional parameters sometimes needing adjustment:

- `use_path_style` defaults to `false`
- `port` defaults to `443`
- `sse_c_key` has no default

Optional parameters less commonly needing adjustment:

- `concurrency` defaults to `5` [Note: This defines the maximum number of concurrent multipart uploads]
- `proxy` defaults to `false`

- `connect_timeout` defaults to `5` [Note: the connection timeout is set in seconds, but decimal precision can be used for subsecond accuracy (for example, 4.2 for 4200 milliseconds)]
- `timeout` defaults to `15`
- `uploadPartSize` defaults to `524288000`
- `putSizeLimit` defaults to `104857600`
- `useMultipartCopy` defaults to `true`
- `copySizeLimit` defaults to `5242880000`
- `legacy_auth` has no default
- `version` defaults to `latest`
- `verify_bucket_exists` defaults to `true` [Note: Setting this to `false` *after* confirming the bucket has been created may provide a performance benefit, but may not be possible in multibucket scenarios.]

If you are using Amazon S3: the `region` parameter is required unless you're happy with the default of `eu-west-1`. There is no need to override the `hostname` or `port`. And `storageClass` only needs to be modified if you're using a different configuration at AWS. Lastly, `use_path_style` is rarely required with Amazon, but some legacy Amazon datacenters may require it.

If you using a non-Amazon hosted S3 store: you will need to set the `hostname` parameter (and can ignore the `region` parameter). You may need to use `use_path_style` if your non-Amazon S3 store does *not* support requests like `https://bucket.hostname.domain/`. Setting `use_path_style` to true configures the S3 client to make requests like `https://hostname.domain/bucket` instead.

Microsoft Azure Blob Storage

The Azure Blob Storage backend mounts a container on Microsoft's Azure Blob Storage into the virtual filesystem.

The class to be used is `\OC\Files\ObjectStore\Azure`

```
'objectstore' => [
    'class' => '\\OC\\Files\\ObjectStore\\Azure',
    'arguments' => [
        'container' => 'nextcloud',
        'autocreate' => true,
        'account_name' => 'account_name',
        'account_key' => 'xxxxxxxxxx'
    ],
],
```

13.5.3 Multibucket Object Store

It's possible to configure Nextcloud to distribute the data over multiple buckets for scalability purposes.

To setup multiple buckets, set `'multibucket => true'` in the object store configuration in `config.php`:

```
'objectstore' => [
    'class' => 'Object\\Storage\\Backend\\Class',
    'arguments' => [
        'multibucket' => true,
        // optional, defaults to 64
        'num_buckets' => 64,
        // will be postfixed by an integer in the range from 0 to (num_nuckets-1)
```

(continues on next page)

(continued from previous page)

```
        'bucket' => 'nextcloud_',
        ...
    ],
],
```

Multibucket object store backend maps every user to a range of buckets and saves all files for that user in their corresponding bucket.

Note

While it is possible to change the number of buckets used by an existing Nextcloud instance, the user-to-buckets mapping is only created once, so only newly created users will be mapped to the updated range of buckets.

You can find out more information about upscaling with object storage and Nextcloud in the Nextcloud customer portal.

13.5.4 Multibucket Object Store with per Bucket configuration overrides

When using an Object Store with `'multibucket => true'` it is possible to configure overrides for all config options per bucket:

```
'objectstore' => [
    'class' => 'Object\\Storage\\Backend\\Class',
    'arguments' => [
        'multibucket' => true,
        'bucket' => 'nextcloud_',
        'perBucket' => [
            'nextcloud_1' => [
                'port' => 9999,
            ],
        ],
    ],
],
```

This can be useful for example if you want to configure credentials per bucket that is used by a Team folder. A script for provisioning new Team folders this way could look like this (first make sure the bucket exists with those credentials):

```
occ config:system:set --type=string --value=KEYVALUE objectstore arguments perBucket␣
↪BUCKETNAME key
occ config:system:set --type=string --value=SECRETVALUE objectstore arguments␣
↪perBucket BUCKETNAME secret
occ groupfolders:create --bucket BUCKETNAME TEAMFOLDERNAME
```

The credentials must be set before the new Team folder is created.

13.5.5 Multi-instance Object Store

It's possible to configure Nextcloud to distribute the data over multiple object store instances for further scaling and gradual migration.

To setup multiple buckets, set `'objectstore'` to an array of named configurations configuration in `config.php` and set the `'default'` to the name of the configuration to use for newly created users:

```
'objectstore' => [
    'default' => 'server2',
    'root' => 'server1',
    'server1' => [
        'class' => 'Object\\Storage\\Backend\\Class',
        'arguments' => [
            'hostname' => 's3-server1.example.com',
            'bucket' => 's1_nextcloud',
            ...
        ],
    ],
    'server2' => [
    'class' => 'Object\\Storage\\Backend\\Class',
        'arguments' => [
            'multibucket' => true,
            'hostname' => 's3-server2.example.com',
            'bucket' => 's2_nextcloud_',
            ...
        ],
    ],
],
```

Note

Bucket names must be unique between all configured object store instances.

Newly created users will be mapped to the object store instance set in `default`. Files that are not part of the users storage are put in the `root` instance, or in the `default` instance if no `root` instance is configured.

In the above example, if `server2` is starting to run low on capacity, an admin can setup and configure a new `server3` and change the `default` to `server3`. Than any newly created user will have their files put on `server3`.

Note

As with multibucket object store, the user-to-instance mapping is only created once, so only newly created users will be mapped to the new default instance.

It is possible to mix different object store backends and multibucket and non-multibucket in a multi-instance configuration.

13.5.6 S3 SSE-C encryption support

Nextcloud supports server side encryption, also known as SSE-C, with compatible S3 bucket provider. The encryption and decryption happens on the S3 bucket side with a key provided by the Nextcloud server.

The key can be specified with the `sse_c_key` parameter which needs to be provided as a base64 encoded string with a maximum length of 32 bytes. A random key could be generated using the the following command:

```
openssl rand 32 | base64
```

The following example shows how to configure the S3 object store with SSE-C encryption support in the objectstore section of the Nextcloud config.php file:

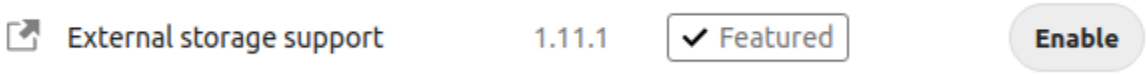
```
'objectstore' => [
    array (
        'class' => 'OC\\Files\\ObjectStore\\S3',
        'arguments' =>
        array (
            'bucket' => 'nextcloud',
            'key' => 'nextcloud',
            'secret' => 'nextcloud',
            'hostname' => 's3',
            'port' => '443',
            'use_ssl' => true,
            'use_path_style' => true,
            'autocreate' => true,
            'verify_bucket_exists' => true,
            'sse_c_key' => 'o9d3Q9tHcPMv6TIpH53MSXaUmY91YheZRwuIhwCFRSs=',
        ),
    );
],
```

13.6 External Storage

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.

13.6.1 Enabling

External Storage Support is provided by a bundled (automatically installed) app. It is disabled by default, so to use this feature you simply need to enable it under **Apps**.

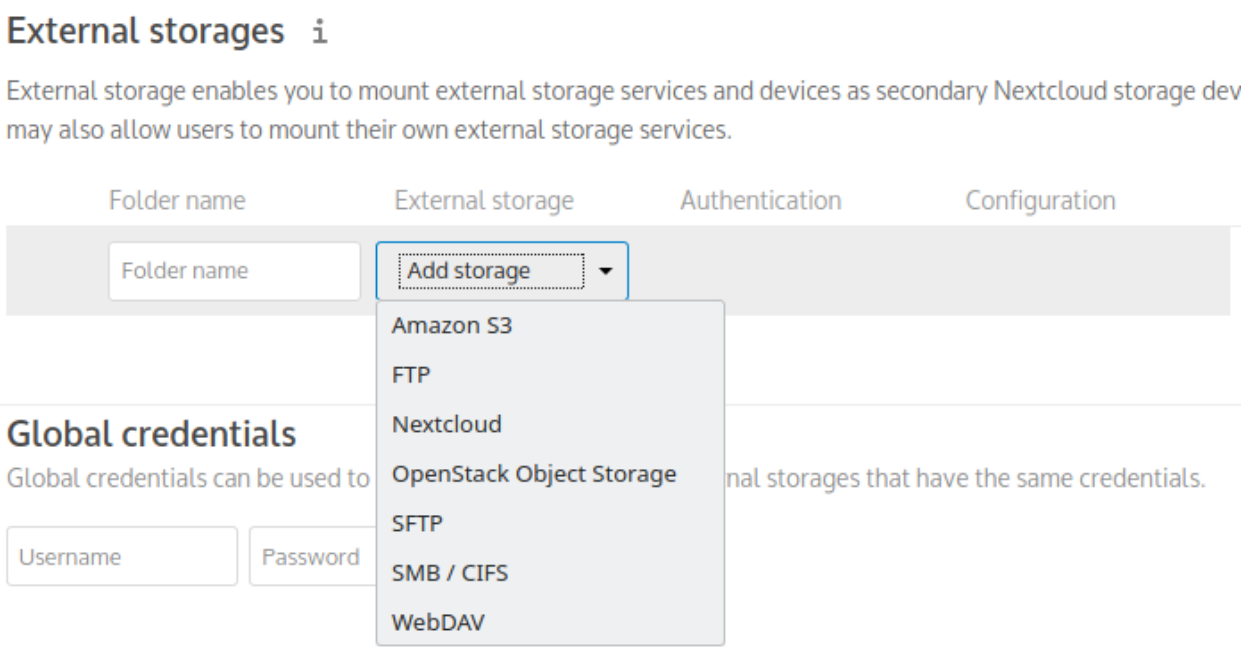


13.6.2 Configuring

To access the settings for configuring external storage mounts, click your **Profile** icon and select **Settings** from the dropdown. On the left side, under **Administration**, select **External Storage**.

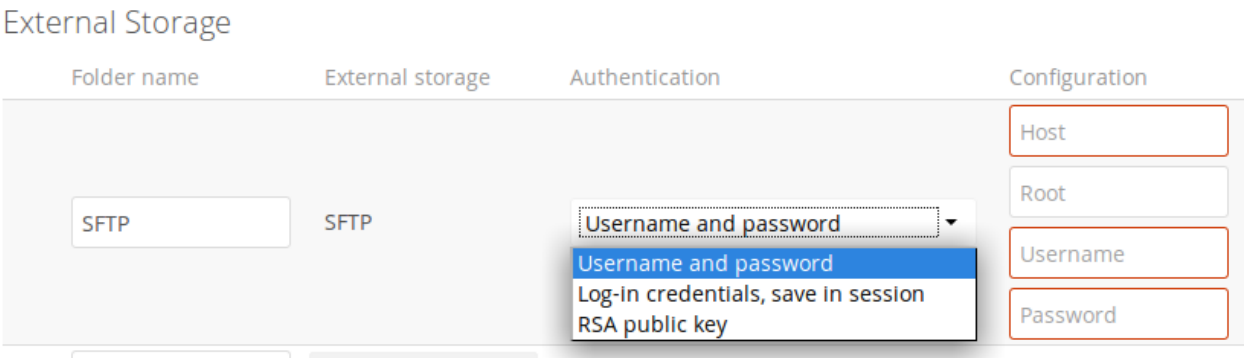
> **Note**
>
> External storage can also be configured via the occ command. See *occ documentation*.

To create a new external storage mount, select an available backend from the **Add storage** dropdown. 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 are specific to the backend, while 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 chosen mechanism. For example, the SFTP backend 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.

13.6.3 Folder name

The `Folder name` is the name the folder will have within Nextcloud - the name that will be visible to NextCloud users.

Note that the folder name cannot include a path or subdirectory - do not include slashes in your `Folder name`.

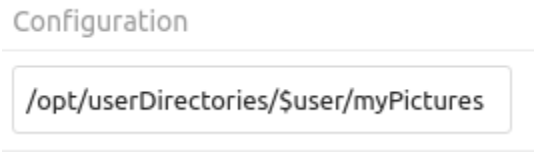
13.6.4 Usage of variables for mount paths

The external storage mounting mechanism accepts variables in the mount path.

Use `$user` for automatic substitution with the logged-in user's username.

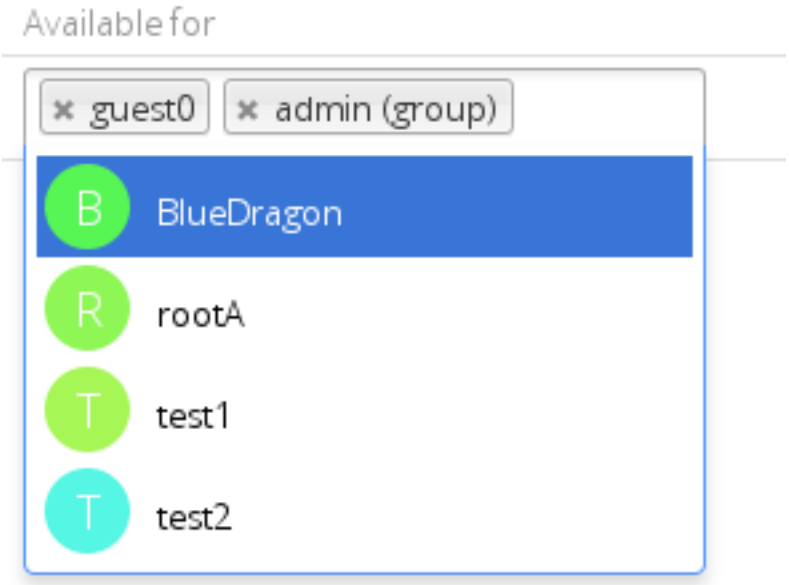
Use `$home` for automatic substitution with a configurable home directory variable (requires LDAP; see *Special attributes* in the LDAP configuration documentation for details).

In the following example, the mount point for a logged-in user "alice" would resolve to `/opt/userDirectories/alice/myPictures`.



13.6.5 User and Group Permissions

A storage configured in a user's personal settings is available only to the user who created it. A storage configured in the Admin settings is available to all users by default, but it can be restricted to specific users and groups in the **Available for** field.



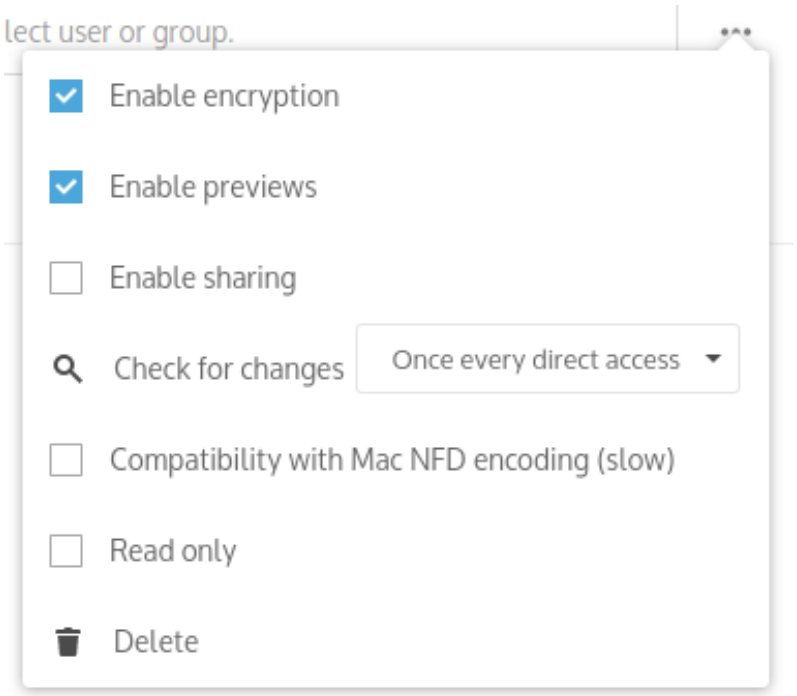
13.6.6 Mount Options

The overflow menu (three dots) exposes the settings and trashcan icons. Click the trashcan to delete the mount point. The settings button allows you to configure each storage mount individually with the following options:

- Encryption
- Previews
- Enable Sharing
- **Filesystem check frequency** — controls how the server rescans an external storage path when a WebDAV `PROPFIND` request arrives for it. **Once per direct access** makes a shallow rescans on each `PROPFIND` request; **Never** skips the rescan entirely. This setting does *not* cause Nextcloud or desktop clients to poll automatically. See *Detecting changes made outside Nextcloud* for details.
- Mac NFD Compatibility
- Read Only

The **Encryption** checkbox is visible only when the Encryption app is enabled. Note that server-side encryption is not available for other Nextcloud servers used as external storage.

Enable Sharing allows the Nextcloud admin to enable or disable sharing on individual mount points. 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.



13.6.7 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.

13.6.8 Available Storage Backends

The following backends are provided by the external storages app.

Amazon S3

To connect an Amazon S3 (or compatible) bucket to Nextcloud you will need to know your:

- S3 bucket name
- S3 access key ID
- S3 secret access key
- S3 region (if Amazon hosted) or S3 hostname (if non-Amazon hosted) [Note: If specifying a hostname, use the generic S3 endpoint hostname, **not** the hostname that contains your bucket name]

> **Attention**
>
> Some S3-compatible storages, including Amazon S3, allow repeating delimiters `/` (`e.g. Photos//cat.png`). This is unsupported and prefixes with repeating delimiters and its contents are ignored.

In the **Folder name** field enter a folder name to use as the local mountpoint for this external storage. If this does not exist it will be created.

In the **External storage** field select **Amazon S3**.

In the **Authentication** field select **Access key**.

In the **Bucket** field enter your *S3 bucket name*. [Note: Even if non-Amazon hosted, bucket names must meet AWS S3 naming requirements regardless of what your S3 provider/platform considers acceptable - i.e. no underscores]

In the **Access key** field enter your *S3 access key ID*.

In the **Secret key** field enter your *S3 access key*.

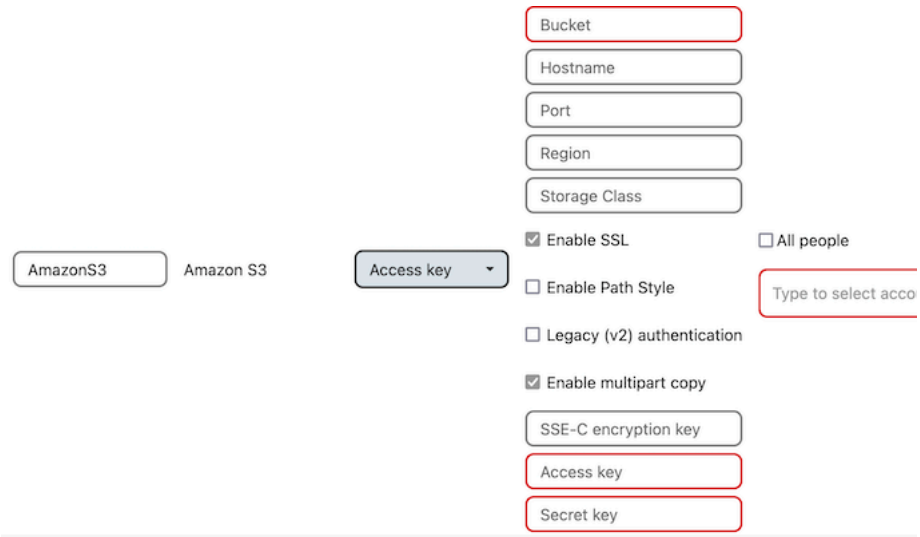
If you are using Amazon S3: the `Region` parameter is required unless you're happy with the default of `eu-west-1` (which will be used if you don't specify anything). There is no need to override the `Hostname` or `Port`. And `Storage Class` only needs to be modified if you're using a different configuration at AWS. Lastly, `Enable Path Style` is rarely required with Amazon, but some legacy Amazon datacenters may require it. Leave `Legacy (v2) authentication` unselected.

If you using a non-Amazon hosted S3 store: you will need to set the `Hostname` parameter (and can ignore the `Region` parameter). You may need to enable `Enable Path Style` if your non-Amazon S3 store does *not* support requests like `https://bucket.hostname.domain/`. Setting `Enable Path Style` to true configures the S3 client to make requests like `https://hostname.domain/bucket` instead. It's rare to need `Legacy (v2) authentication`, but enable it if your in-house object store or service provider requires it over the default (v4) authentication.

In the **Available for** field enter the users or groups who you want to give give access to your S3 mount.

The `Enable SSL` checkbox enables HTTPS connections and generally preferred. It is the default unless you disable it here.

Optionally, a 32-byte base64 encoded SSE-C key can be provided for server side encryption. See *Configuring Object Storage as Primary Storage* and the SSE-C AWS documentation for more information how to generate a key.



See *External Storage* 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 *External Storage* for additional mount options and information.

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

Local

Local storages provide access to any directory on the Nextcloud server. Since this is a significant security risk, Local storage can only be configured in the Nextcloud admin settings. Non-admin users cannot create Local storage mounts.

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

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

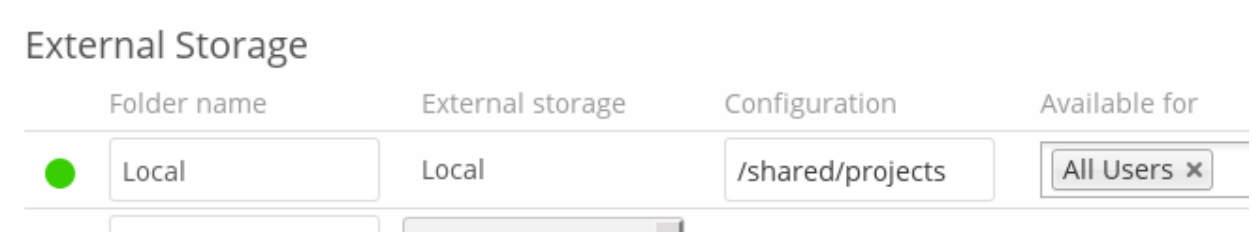
Important: If you use consecutive commands, make sure, you are user `www-data`:

```
sudo -E -u www-data bash
cd /path/to/localdir
mkdir data
```

In the **Folder name** field enter the folder name that you want to appear on your Nextcloud Files page.

In the **Configuration** field enter the full filepath of the directory you want to mount.

In the **Available for** field enter the users or groups who have permission to access the mount. By default all users have access.



See *External Storage* for additional mount options and information.

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

Nextcloud

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

When filling in the **URL** field, use the path to the root of the Nextcloud installation, rather than the path to the WebDAV endpoint. So, for a server at `https://example.com/nextcloud`, use `https://example.com/nextcloud` and not `https://example.com/nextcloud/remote.php/dav`.

See *External Storage* for additional mount options and information.

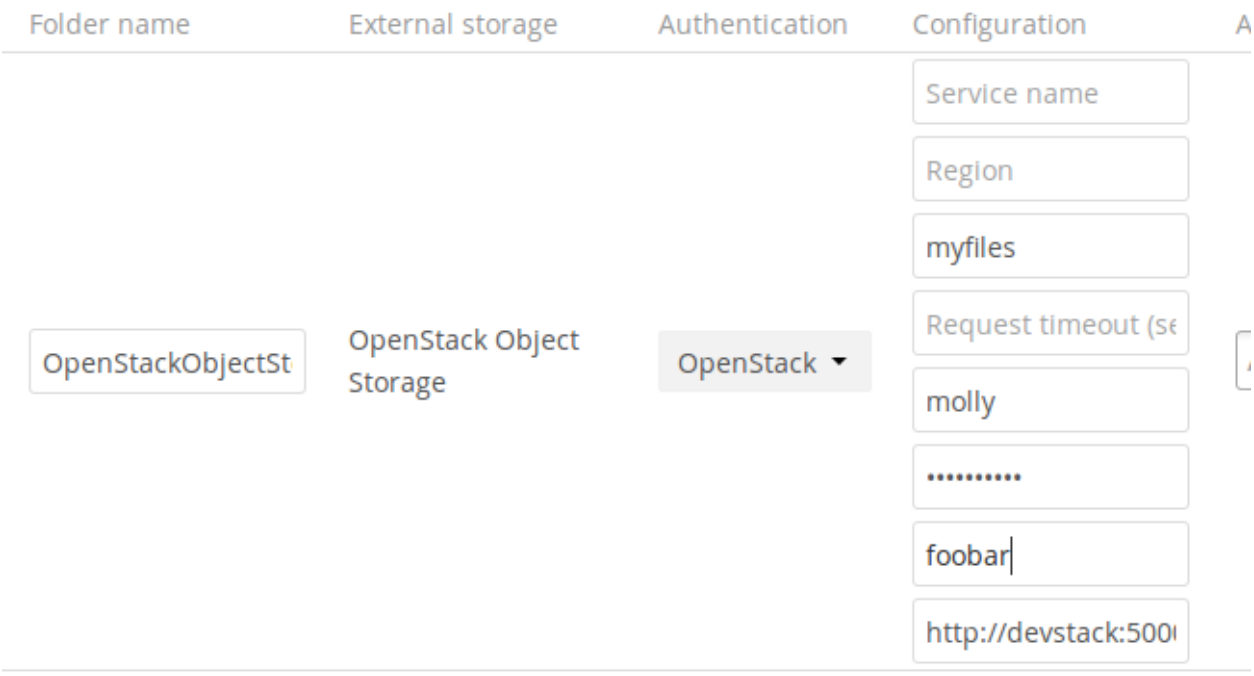
See *External Storage authentication mechanisms* for more information on authentication schemes.

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:

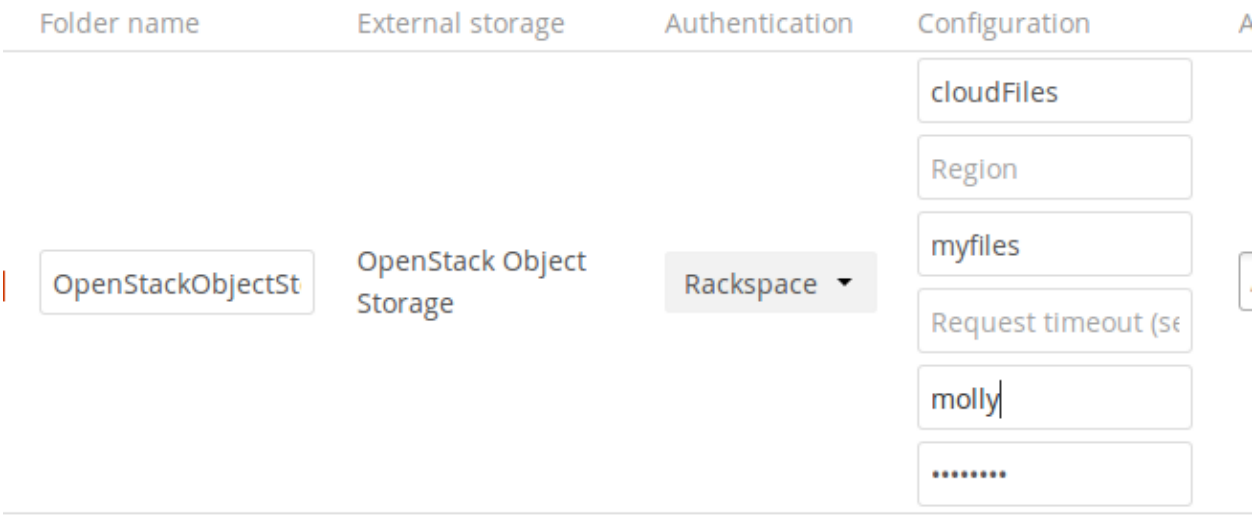
- **Bucket**. This is user-defined; think of it as a subdirectory of your total storage. The bucket will be created if it does not exist.
- **Username** of your account.
- **Password** of your account.
- **Tenant name** of your account. (A tenant is similar to a user group.)
- **Identity Endpoint URL**, 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 *External Storage* for additional mount options and information.

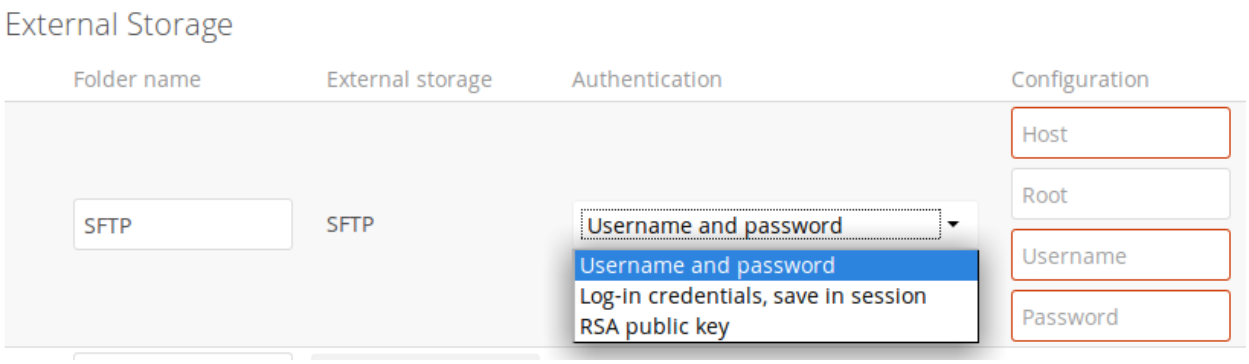
See *External Storage authentication mechanisms* for more information on authentication schemes.

SFTP

Nextcloud's SFTP (SSH File Transfer Protocol) backend supports both password and public key authentication.

The **Host** field is required. 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 *External Storage* 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.

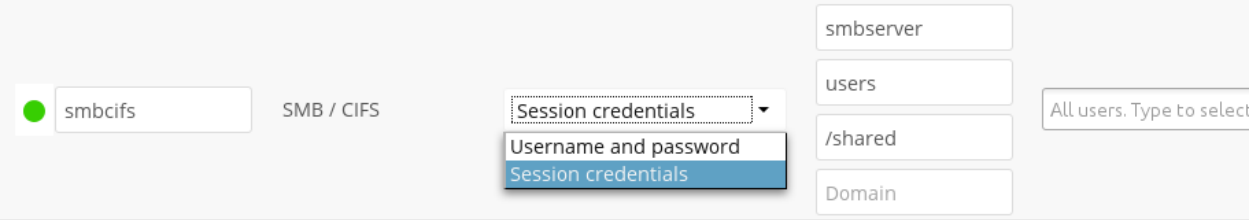
> **Note**
>
> The SMB/CIFS backend requires `smbclient` or the PHP smbclient module (`php-smbclient` / `libsmbclient-php`) to be installed on the Nextcloud server. The PHP smbclient module is **strongly preferred** — without it, the backend falls back to spawning the `smbclient` binary, which has known limitations (notably, downloads of files larger than ~512 MB from external SMB shares can fail; see nextcloud/server#31308). These should be included in any Linux distribution. (See PECL smbclient if your distro does not include them.)

You need the following information:

- Folder name for your local mountpoint.
- Host: The URL of the Samba server.
- Username: The username or `domain\username` (see below) used to login to the Samba server.
- Password: the password to login to the Samba server.
- Share: The share on the Samba server to mount.
- Remote Subfolder: The remote subfolder inside the Samba share to mount (optional, defaults to /). To assign the Nextcloud logon username automatically to the subfolder, use `$user` instead of a particular subfolder name.
- And finally, the Nextcloud users and groups who get access to the share.

Optionally, you can specify a `Domain`. This is useful in cases where the SMB server requires a domain and a username, and an advanced authentication mechanism like session credentials is used so that the username cannot be modified. This is concatenated with the username, so the backend gets `domain\username`

> **Note**
>
> For improved reliability and performance, we recommended installing `libsmbclient-php`, a native PHP module for connecting to SMB servers.



See *External Storage* for additional mount options and information.

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

Case sensitive file system

This option tells Nextcloud whether the SMB share is backed by a case-sensitive filesystem. It does not change how the SMB server stores files, it only affects how Nextcloud performs lookups and validates paths.

- **Enabled (default)**: Nextcloud assumes the share is case-sensitive. File existence checks use exact case and case-only renames are allowed.
- **Disabled**: Nextcloud assumes the share is case-insensitive and compensates by scanning directory entries to find the exact name used on the server. It also blocks renames that only change letter case. This can be slower on large directories.

If the setting does not match the SMB server's filesystem, you can run into confusing behavior:

- Case-insensitive SMB (for example NTFS or FAT) with the option enabled can lead to cache inconsistencies or conflicts, because Nextcloud may treat `File.txt` and `file.txt` as separate while the server treats them as the same file.
- Case-sensitive SMB (for example ext4) with the option disabled can cause unnecessary directory scans and case-only renames being rejected, even though the server itself would allow them.

SMB update notifications

See also

Detecting changes made outside Nextcloud explains why external changes to a mounted storage are not detected automatically and how update notifications fit into that picture.

Nextcloud can use smb update notifications to listen for changes made to a configured SMB/CIFS storage and detect external changes made to the storage in near real-time.

Note

Due to limitations of linux based SMB servers, this feature only works reliably on Windows SMB servers.

Note

Using update notifications requires `smbclient` 4.x or newer. Due to limitations with the smbclient PHP module, the `smbclient` binary is required even when using the PHP module.

To start listening to update notifications, start the `occ` command like this:

```
occ files_external:notify <mount_id>
```

You can find the mount id for a specific storage using `occ files_external:list`

On default this command shows no output, can you see the list of detected changes by passing the `-v` option to the command.

SMB authentication

Update notifications are not supported when using 'Login credentials, save in session' authentication. Using update notifications is only supported with 'Login credentials, save in database'.

Even when using 'Login credentials, save in database' or 'User entered, stored in database' authentication the notify process can not use the credentials saved to attach to the smb shares because the notify process does not run in the context of a specific user in those cases you can provide the username and password using the `--username` and `--password` arguments.

Decrease sync delay

Any updates detected by the notify command will only be synced to the client after the Nextcloud cron job has been executed (usually every 15 minutes). If this interval is too high for your use case, you can decrease it by running `occ files:scan --unscanned --all` at the desired interval. Note that this might increase the server load and you'll need to ensure that there is no overlap between runs.

Hidden files upload failure or not shown

If you have the configuration `hide dot files = Yes`, you will not be able to upload a hidden file (dot file) nor will you be able to show hidden files on your filelist (even if the 'show hidden file' option is checked on the nextcloud settings. Make sure you have the following option in your configuration: `hide dot files = No`

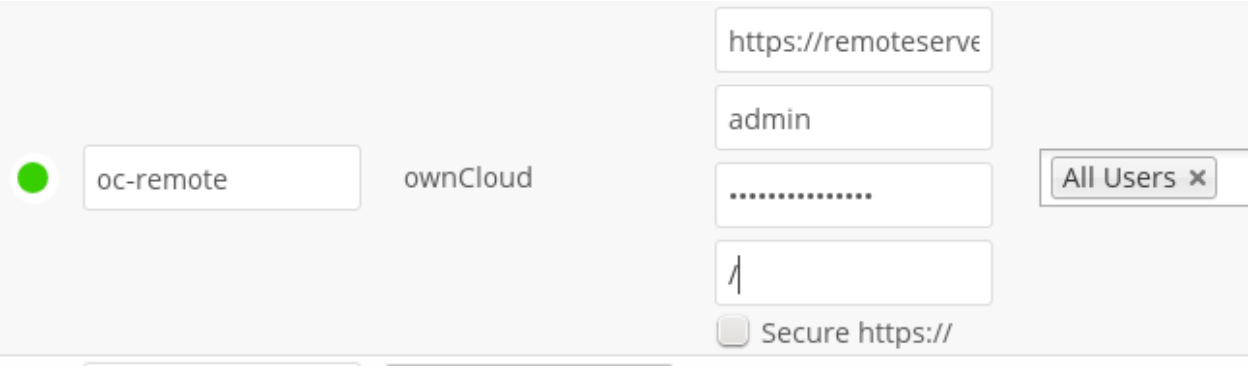
WebDAV

Use this backend to mount a directory from any WebDAV server, or another Nextcloud server.

You need the following information:

- Folder name: The name of your local mountpoint.
- The URL of the WebDAV or Nextcloud server.
- Username and password for the remote server
- Secure `https://`: We always recommend `https://` for security, though you can leave this unchecked for `http://`.

Optionally, a `Remote Subfolder` can be specified to change the destination directory. The default is to use the whole root.



Note

CPanel users should install Web Disk to enable WebDAV functionality.

See *External Storage* for additional mount options and information.

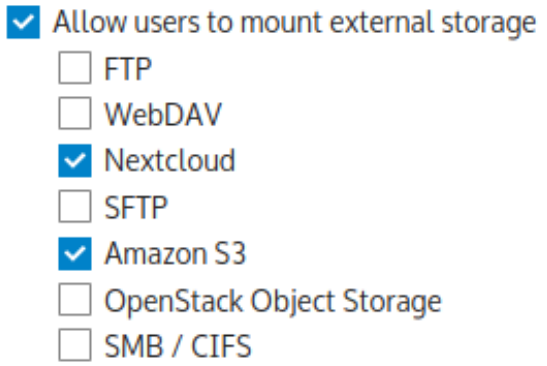
See *External Storage authentication mechanisms* for more information on authentication schemes.

Note

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

13.6.9 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!



13.6.10 Detecting changes made outside Nextcloud

When files are added, modified, or deleted on an external storage **directly** (without going through Nextcloud), Nextcloud will not know about those changes immediately. Nextcloud maintains an internal index (the *filecache*) that is only updated when Nextcloud itself performs a scan. Until that scan happens, the filecache is stale and the changes are invisible to Nextcloud, desktop clients, and mobile apps.

How the "Filesystem check frequency" option works

The **Filesystem check frequency** mount option (see *Mount Options*) controls what happens when a WebDAV `PROPFIND` request reaches the Nextcloud server for a path inside the external storage:

- **Once per direct access** — during `PROPFIND` requests made to a given directory, Nextcloud rescans that directory level then updates the filecache with what it finds there.
- **Never** — the server never rescans the external storage during `PROPFIND` requests and always returns the content stored in its internal index.

This option is *only* about the server's behavior during a `PROPFIND`. It does not trigger any background polling and does not change how desktop clients or mobile apps behave.

Because this rescan is driven by user actions (opening a folder in the web UI, browsing in the mobile app, or a client issuing a `PROPFIND`), change detection remains random and unreliable for deep folder hierarchies.

Why desktop clients miss deep changes

The desktop sync client only issues a `PROPFIND` on the root of the sync folder on every sync cycle because scanning the entire tree would be too expensive. It relies on **etag propagation**: when a file changes, the etag of every parent folder up to the root must also change, giving the client a trail to follow down to the changed file.

When a change is made outside Nextcloud, no etag is updated. The client sees no trail from the root and never follows it down to the changed file.

How to make external changes reliably detectable

There are two approaches, depending on the storage type:

A) SMB/CIFS storages — use update notifications

The `files_external:notify` command listens for change events sent by the SMB server itself. When a change event arrives, Nextcloud rescans the affected path and propagates the etag changes up to the root, so desktop clients pick up the change on their next sync cycle.

```
occ files_external:notify <mount_id>
```

See *SMB/CIFS* for setup details, including authentication requirements and how to reduce sync delay.

B) All other storages — periodic rescan via cron

For storage types that do not support push notifications, set up a cron job that rescans the external storage periodically using its mount ID:

```
sudo -E -u www-data php occ files_external:scan <mount_id>
```

You can find the mount ID with `occ files_external:list`. See *Using the occ command* for the full `files_external:scan` reference. A typical interval is every 15 minutes; adjust to balance freshness against server load.

Note

If you are running Nextcloud AIO, the equivalent command is:

```
sudo docker exec --user www-data -i nextcloud-aio-nextcloud php occ files_
↪external:scan <mount_id>
```

Limitation: renames are not detected reliably

Whenever Nextcloud rescans an external storage — whether via periodic cron (option B) or via the **Once per direct access** filesystem check frequency setting — the scanner cannot reliably detect that an entry was **renamed**. It sees the old name as deleted and the new name as a newly created entry. This causes **metadata loss**: all shares, tags, comments, and activity history associated with the old entry are permanently deleted from Nextcloud's database.

Warning

If users rename files or folders directly on the external storage (outside Nextcloud), that metadata loss is unavoidable. To preserve metadata, all renames should be done through Nextcloud itself, or — for SMB — use update notifications (option A) which handles renames via the SMB change event stream.

13.6.11 Troubleshooting File Name Encoding

When using external storage, it can happen that some files with special characters will not appear in the file listing, or they will appear but not be accessible.

When this happens, please run the *files scanner*, for example:

```
sudo -E -u www-data php occ files:scan --all
```

If the scanner reports an encoding issue on the affected file, please enable Mac encoding compatibility in the *mount options* and then *rescan the external storage*.

Note

This mode comes with a performance impact because Nextcloud will always try both encodings when detecting files on external storages.

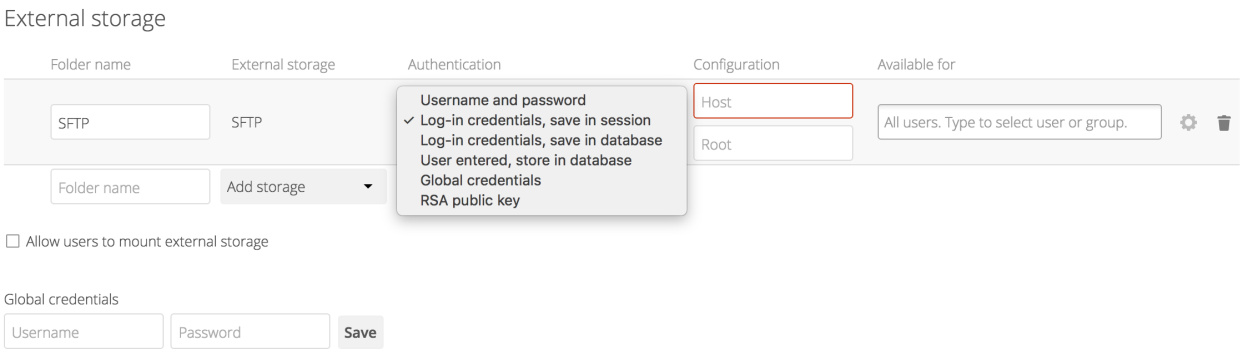
Mac computers use the NFD Unicode normalization for file names, which is different from NFC, the one used by other operating systems. Mac users might upload files directly to the external storage using NFD-normalized file names. When uploading through Nextcloud, file names will always be normalized to the NFC standard for consistency.

It is recommended to let Nextcloud use external storages exclusively to avoid such issues.

See also the technical explanation about NFC vs NFD normalizations.

13.7 External Storage authentication mechanisms

Nextcloud storage backends accept one or more authentication schemes such as passwords, OAuth, or token-based, to name a few examples. Each authentication scheme may be implemented by combining multiple authentication mechanisms. Different mechanisms require different configuration parameters, depending on their behavior.



13.7.1 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.

13.7.2 Password-based mechanisms

The **Username and password** mechanism requires a manually-defined username and password. These get passed directly to the backend and are specified during the setup of the mount point.

The **Log-in credentials, save in session** mechanism uses the Nextcloud login credentials of the user to connect to the storage. These are not stored anywhere on the server, but rather in the user session, giving increased security. This method has some important drawbacks, since Nextcloud has no access to the storage credentials and therefore cannot perform any background tasks on the storage:

- Sharing is disabled
- Background file scanning does not work
- Background versions expiration does not work

- Desktop and mobile clients that use tokens to authenticate can not access those shares
- Other services that might request the file through a different request like Collabora Online or OnlyOffice will not be able to open files from that storage
- The method cannot be used with SAML/SSO authentication, because Nextcloud does not get a hold of any credentials whatsoever

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

- The method cannot be used with SAML/SSO authentication, because Nextcloud does not get a hold of any credentials whatsoever

The **User entered, store in database** mechanism work in the same way as the "Username and password" mechanism but the credentials need to be specified by each user individually. Before the first access to that mount point the user will be prompted to enter the credentials.

The **Global credentials** mechanism uses the general input field for "Global credentials" in the external storage settings section as source for the credentials instead of individual credentials for a mount point.

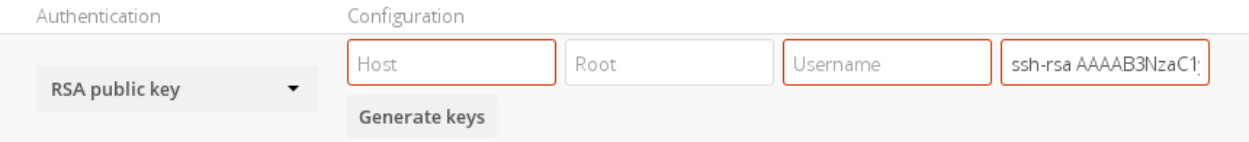
Considerations for shared storage

13.7.3 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.

After generating your keys, you need to copy your new public key to the destination server to `.ssh/authorized_keys`.

See *SFTP* for additional information on how to set up certificate based authentication on SFTP.



13.7.4 Considerations for shared storage

Every external storage, which is using user specific authentication, is connected individually. Nextcloud cannot recognise shared file locks across individual connections, even when accessing the same file.

This has an influence on e.g. file locking as a locked individual file is not shown as locked to other users or users cannot collaboratively edit documents.

If collaborative working on external storage is required, the authentication "Global credentials" has to be used.

13.8 Server-side Encryption

13.8.1 Overview

Nextcloud provides several encryption methods to protect your data. These work at different layers of the system and address different security needs. This guide focuses on Nextcloud's built-in Server-Side Encryption (SSE).

Note

Encryption and risk management is a complex and nuanced topic. Unless you are already an expert, it is recommended to consult a professional or perform a deep dive to ensure your approach targets your specific concerns. You may also find this Nextcloud Blog post helpful: Data encryption methods in Nextcloud. It provides a solid high-level overview of the different approaches commonly considered when using Nextcloud.

13.8.2 Definitions

- **Server-Side Encryption (SSE):** Performed by the Nextcloud server, protecting files at rest on local and external storage. Encryption keys are stored on the server.
- **End-to-End Encryption (E2EE):** Performed by Nextcloud desktop or mobile clients before uploading files. Only the client can decrypt, making data inaccessible to server administrators and external storage providers.
- **Master Key:** A central key controlled by the server, used to encrypt all files.
- **User Keys:** Each user has their own key, protected by their password, to encrypt their files.
- **Recovery Key:** An admin-defined key to recover files if users lose their passwords.
- **Disk/Block Device Encryption:** A method of securing all data stored on a physical storage device by encrypting it at the hardware or filesystem level - typically using tools such as LUKS on Linux - so that data is only accessible after the device is unlocked with the correct key or password.

13.8.3 Encryption Method Comparison

Method	Encryption Location	Who Can Decrypt?	Protects Against
SSE (Master Key)	Server	Admins & users	External storage providers
SSE (User Keys)	Server	Users & malicious admins	External storage providers
SSE (User Keys w/ Recovery)	Server	Users & admins with recovery key	External storage providers
E2EE	Client	Users only	Admins, external storage providers
Disk/Block Encryption	Server	OS admin	Physical tampering, theft

13.8.4 Key Points & Limitations

- Encryption methods are not interchangeable; each is designed for specific risks.
- **Server-Side Encryption (SSE)** is mainly for protecting files on external, third-party storage.
- **End-to-End Encryption (E2EE)** is for scenarios where server administrators must not access data.
- SSE does **not** encrypt filenames or folder structures, only file contents.
- SSE does not protect data from a compromised Nextcloud server or malicious administrator. Use E2EE for this threat.
- SSE cannot be reversed via the Nextcloud Web interface.
- Troubleshooting SSE generally requires `occ` command access. Make sure you have this before enabling SSE!
- Losing encryption keys or your instance secret results in permanent data loss.
- Nextcloud quotas are based on unencrypted file size; files encrypted with SSE may be ~1% larger (was 35% before Nextcloud 25).

- SSL/TLS (HTTPS) terminates before files are encrypted, so files may be exposed in memory between SSL/TLS and Nextcloud's SSE encryption code.
- When files on external storage are encrypted with SSE, you cannot share them directly from the external storage provider; sharing is only possible via Nextcloud, since the decryption key never leaves the Nextcloud server.
- For local storage, it may be better to use other encryption tools, such as disk/block device encryption (e.g., LUKS) provided by your operating system. This protects against other concerns, such as theft of your physical server, which is not SSE's goal.

Warning

SSE does **not** encrypt filenames or folder structures, only file contents.

Note

Don't confuse Nextcloud's SSE with S3 SSE-C (also supported).

Changed in version 9.0.0: Nextcloud (since v9.0.0) supports Authenticated Encryption for all newly encrypted files. See https://hackerone.com/reports/108082 for technical details.

Tip

For maximum security, configure external storage with "Check for changes: Never". This causes Nextcloud to ignore new files not added via Nextcloud, preventing unauthorized additions by external storage admins. Do not use this if your storage is subject to legitimate external changes.

13.8.5 Before You Enable Encryption

1. Read this guide fully and understand the risks.
2. Back up your instance configuration and all encryption keys in a safe location before proceeding.
3. Decide which key management mode suits your needs (see below).

13.8.6 Key Management Modes

Master Key (default):

- All files are encrypted with a central server-controlled key.
- Admins can decrypt any user's files.
- **Recovery keys are not available in master key mode.** Files remain accessible if a user forgets their password, as they are encrypted by the master key, not the user password.
- Recommended for most deployments.

User Keys:

- Each user's files are encrypted with a password-protected key.
- Admins cannot (readily) decrypt files without the user's password, unless a recovery key is defined.
- If a user forgets their password and no recovery key exists, their files are lost.

- This mode does not work with all authentication methods (e.g., app passwords, single sign-on) and is only recommended for compatibility with older setups.

How to choose:

- If you trust your server administrators, use master key mode.
- If you need to prevent admins from accessing files, use E2EE.
- User key mode offers some protection against malicious server administrators, but has limitations.

To select user key mode:

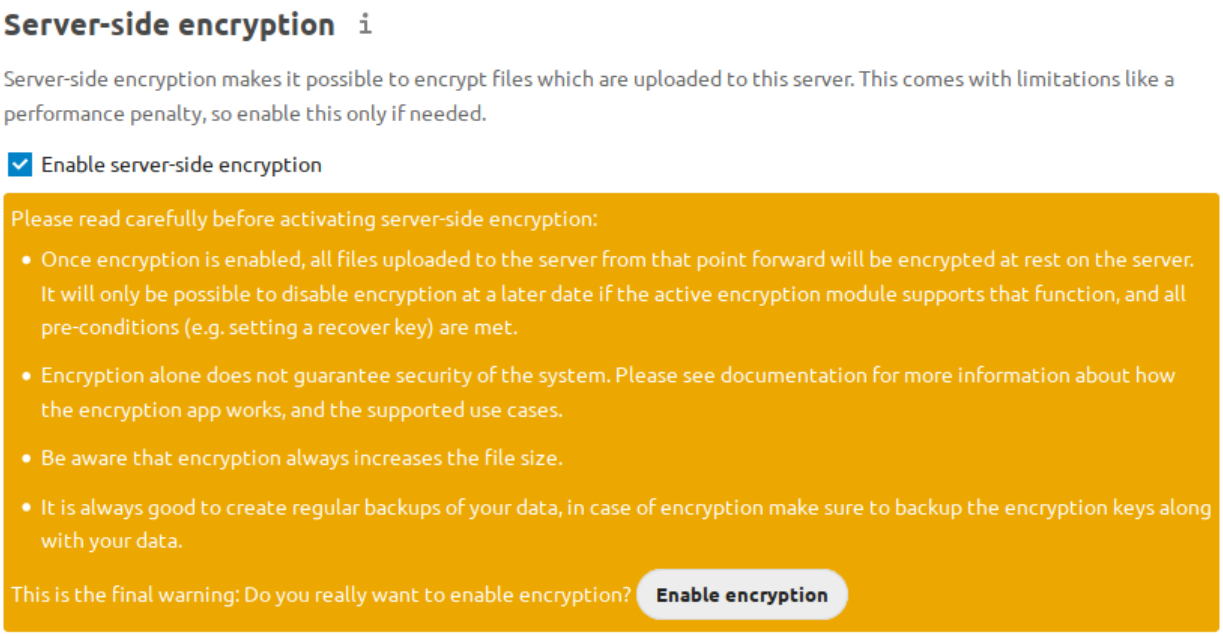
Run:

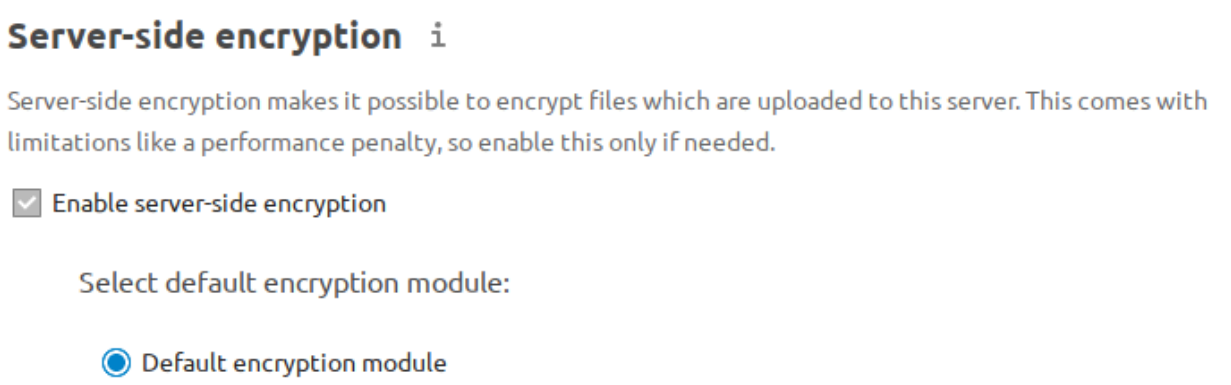
```
occ encryption:disable-master-key
```

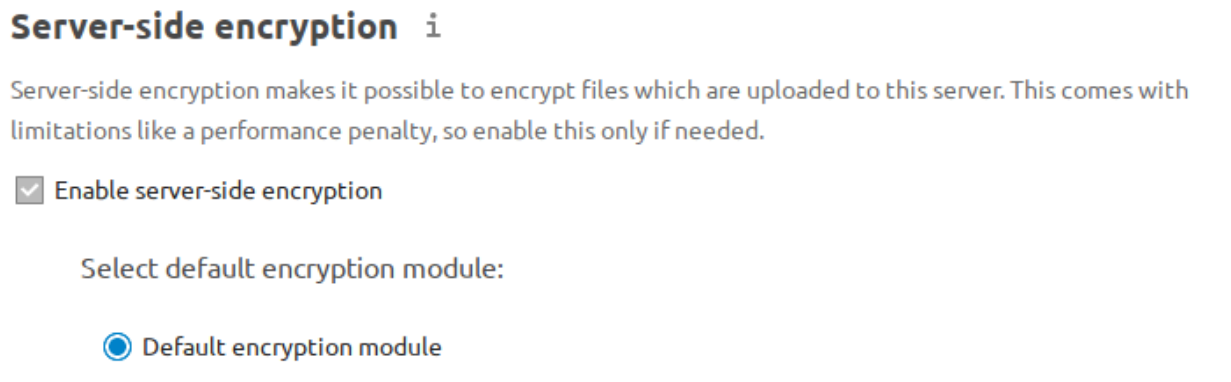
before enabling encryption.

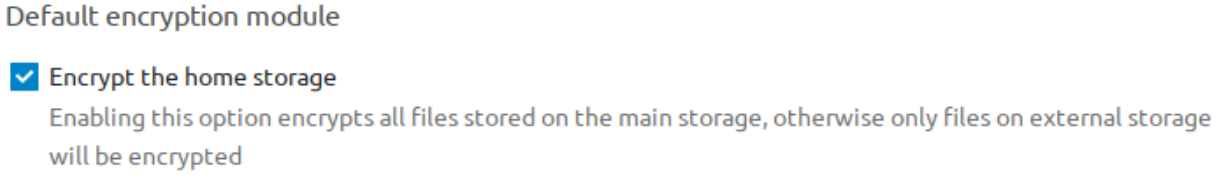
13.8.7 Enabling Encryption (Step-by-Step)

1. Go to the **Server-side encryption** section of your Admin page.
2. Check **Enable server-side encryption**.
3. You'll see a message: "No encryption module loaded." Go to your Apps page and enable the Nextcloud Default Encryption Module.
4. Return to your Admin page. The module will appear and be auto-selected.
5. Log out and log back in to initialize your encryption keys.
6. Optional: Un-check the box for encrypting home storage if you wish to keep local files unencrypted.









13.8.8 Backups

Encryption keys are stored in:

- `data/<user>/files_encryption` (per-user keys)
- `data/files_encryption` (system-wide/external storage keys)

13.8.9 Encrypting External Mountpoints and Team Folders

- You and your users can encrypt external mountpoints.
  - Set encryption options in the mount configuration for each external storage.
  - See *Mount Options* in *External Storage*.
- To encrypt Team Folders, run:

```
occ config:app:set groupfolders enable_encryption --value=true
```

Note

Only new or updated files in team folders will be encrypted.

13.8.10 Managing Encryption via occ Commands

Here is a reference table for common occ commands:

Command	Description
occ encryption:status	Show encryption status and module
occ encryption:enable	Enable server-side encryption
occ encryption:list-modules	List available encryption modules
occ encryption:set-default-module [Module ID]	Select default encryption module
occ encryption:encrypt-all	Encrypt all files for all users
occ encryption:decrypt-all [user]	Decrypt all files (or for one user)
occ encryption:show-key-storage-root	Show key storage location
occ encryption:change-key-storage-root [dir]	Move key storage directory
occ encryption:enable-master-key	Enable master key mode
occ encryption:disable-master-key	Disable master key mode
occ encryption:fix-encrypted-version	Fix bad signature errors
occ encryption:fix-key-location [user]	Fix key not found errors

Tip

See the *Encryption commands* section of the `occ` reference guide for further `encryption` command examples and details.

Example: Move keys to a new directory (Ubuntu Linux):

```
cd /your/nextcloud/data
mkdir keys
chown -R root:www-data keys
chmod -R 0770 keys
occ encryption:change-key-storage-root keys
```

13.8.11 Encrypting All Files

By default, only new and changed files are encrypted when you enable SSE. To encrypt all files for all users run:

```
occ encryption:encrypt-all
```

- **Make sure you have backups before running.**
- The command creates a key pair for each user and encrypts their files.
- Progress is displayed until all files are encrypted.

- **Make sure no users access files during this process.**

13.8.12 Decrypting Files / Disabling Encryption

- Only possible via occ.
- First, decrypt all files:

```
occ encryption:decrypt-all
```

- **Make sure you have backups before running.**
- Server enters maintenance mode. If interrupted, rerun until complete.
- If some files remain encrypted, rerun the command after resolving issues.
- **Warning:** Disabling encryption without decrypting all files will cause unpredictable errors.

You can decrypt for individual users:

```
occ encryption:decrypt-all <user-id>
```

13.8.13 Data Not Encrypted

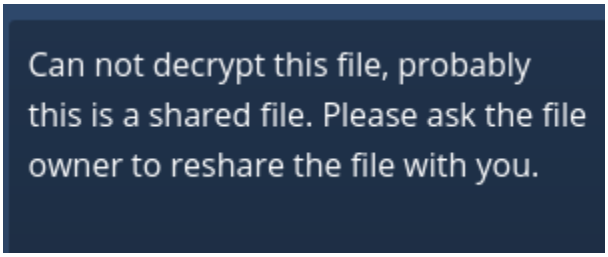
Only file contents are encrypted. The following are **not** encrypted:

Not Encrypted
Filenames and folder structures
Existing trash bin files
Existing historical file versions
Image thumbnails
Image previews
Full text search index
Application data that isn't file-based (e.g., Deck, Tables)

13.8.14 User Keys: Sharing & Recovery

Sharing encrypted files:

- After enabling user key mode, users must log out and log in to generate keys.
- Users who see "Encryption App is enabled but your keys are not initialized…" must log out and back in.
- **Shared files may need to be re-shared after encryption is enabled.**
  - For individual shares: un-share and re-share the file.
  - For group shares: share with any individuals who cannot access the share, then remove their individual shares.



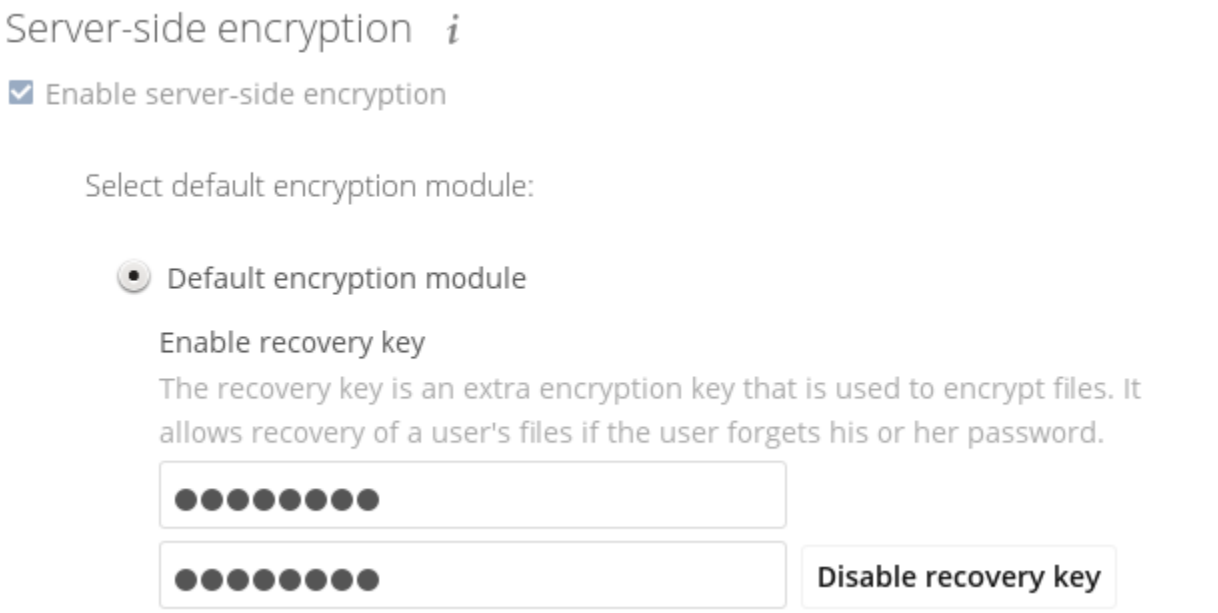
Enabling file recovery keys:

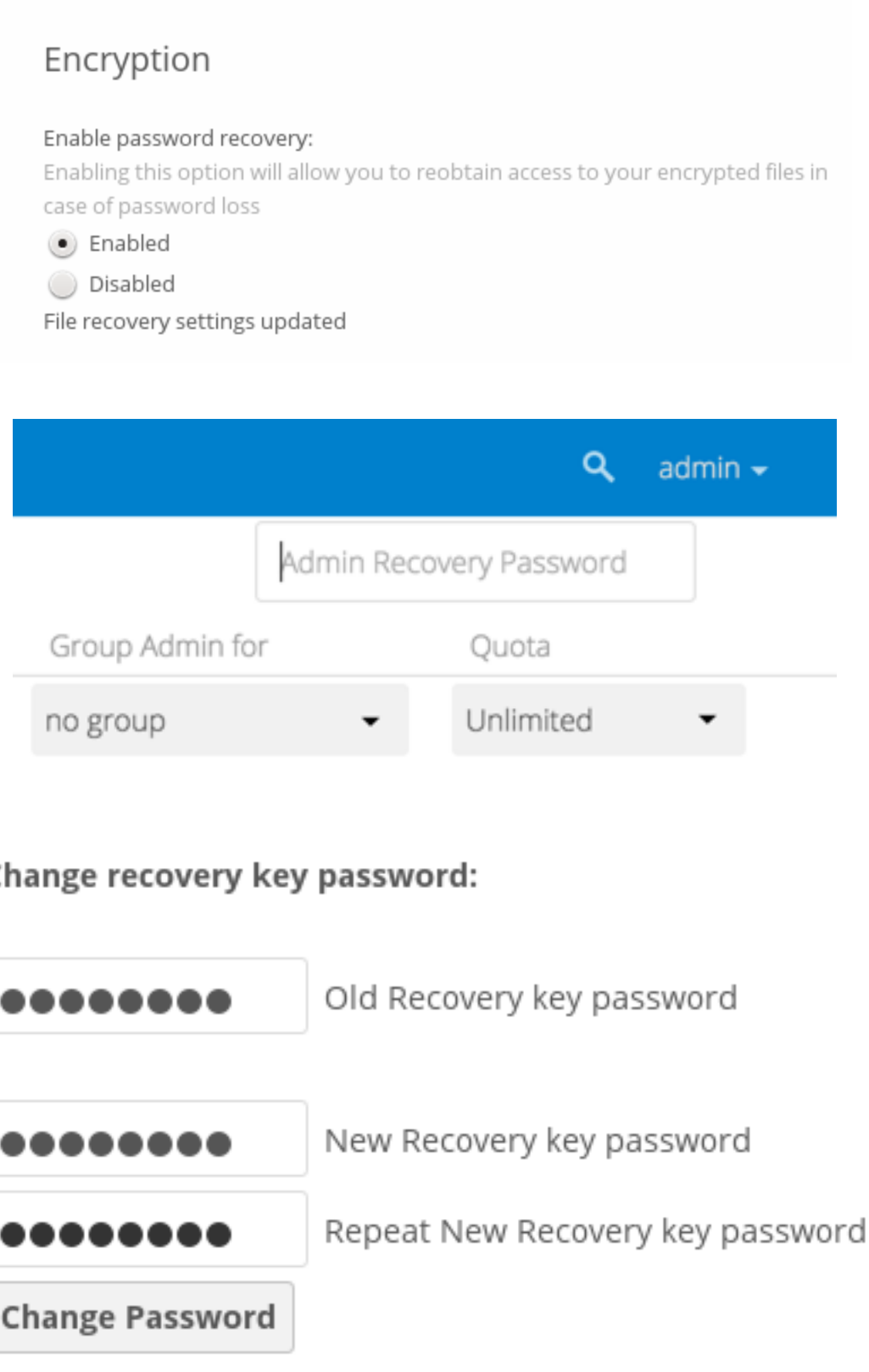
Recovery keys are only available in per-user key mode (not the default master key mode).

- If you lose your Nextcloud password, you lose access to your encrypted files.
- If a user loses their password, their files are unrecoverable unless a recovery key is enabled (per-user key mode only).
- To enable recovery (in per-user key mode), go to Encryption in Admin page and set a recovery key password.
- Users must enable password recovery in their Personal settings for the Recovery Key to work.
- For users who have enabled password recovery, admins can reset passwords and recover files using the Recovery Key.

Warning

The recovery process can be slow and resource-intensive, especially for instances with large amounts of encrypted data. Test recovery procedures before relying on them in production.




Encryption
Enable password recovery:
Enabling this option will allow you to reobtain access to your encrypted files in case of password loss
Enabled
Disabled
File recovery settings updated
admin
Admin Recovery Password
Group Admin for
Quota
no group
Unlimited
Change recovery key password:
Old Recovery key password
New Recovery key password
Repeat New Recovery key password
Change Password

13.8.15 LDAP and External User Backends

- If using LDAP/Samba and changing passwords on the backend, users will need both their old and new passwords on next login.
- If recovery key is enabled, admins can reset the password via Nextcloud and notify users.

13.8.16 Troubleshooting

Why don't I see the recovery key option in the Encryption settings?

Recovery keys are only available in per-user key mode. Since Nextcloud 13, the default encryption mode uses master keys (system-wide encryption). Master key mode does not expose recovery key options in the Admin settings because recovery keys are not needed—admins can reset user passwords and files remain accessible.

If you are using master key mode (the default and recommended mode), you do not need recovery keys. Recovery keys are only relevant for per-user key setups, which are maintained for compatibility with older deployments.

See *Key Management Modes* for guidance on the differences between master key and per-user key modes, and GitHub Issue #8283 for technical context on this design decision.

Invalid private key for encryption app

See GitHub Issue #8546 and workaround.

Bad signature error

In some rare cases, encrypted files cannot be downloaded and return a "500 Internal Server Error." If the Nextcloud log contains an error about "Bad Signature," run the following command to repair affected files:

```
occ encryption:fix-encrypted-version userId --path=/path/to/broken/file.txt
```

Replace "userId" and the path accordingly. The command will perform a test decryption for all files and automatically repair those with a signature error.

Encryption key cannot be found

If the logs contain an error stating that the encryption key cannot be found, you can manually search the data directory for a folder that has the same name as the file name. For example, if a file "example.md" cannot be decrypted, run:

```
find path/to/datadir -name example.md -type d
```

Then check the results located in the `files_encryption` folder. If the key folder is in the wrong location, move it to the correct folder and try again.

The `data/files_encryption` folder contains encryption keys for group folders and system-wide external storages, while `data/$userid/files_encryption` contains the keys for specific user storage files.

> **Note**
>
> This can happen if encryption was disabled at some point but the *occ command for decrypt-all* was not run. If someone then moved the files to another location, the keys did not get moved.

Encryption key cannot be found with external storage or group folders

To resolve this issue, run the following command:

```
sudo -E -u www-data php occ encryption:fix-key-location <user-id>
```

This will attempt to recover keys that were not moved properly.

If this doesn't resolve the problem, refer to the section *Encryption key cannot be found* for a manual procedure.

Note

There were two known issues where:

- moving files between an encrypted and non-encrypted storage like external storage or group folder would not move the keys with the files.
- putting files on system-wide external storage would store the keys in the wrong location.

13.8.17 Further Reading

- *occ Command Reference: Encryption*
- How Nextcloud uses encryption to protect your data
- Technical impact of Authenticated Encryption
- *Nextcloud SSE Implementation Details*
- Nextcloud Encryption (SSE & E2EE) Recovery Tools
- Nextcloud E2EE Server API App (required for E2EE usage)

13.9 Server-side encryption details

This document describes the server-side encryption scheme implemented by Nextcloud's default encryption module. This includes:

- the encryption and signature of files with a master key.
- the encryption and signature of files with a public sharing key.
- the encryption and signature of files with a recovery key.
- the encryption and signature of files with a user key.

These conventions apply throughout this document:

- Given file paths in this document are relative to the Nextcloud data directory that can be retrieved as `datadirectory` from the `config.php`.
- Placeholders are denoted as `$variable`. The variable has to be replaced with the appropriate information.
- Static strings are denoted as `"some string"`.
- The concatenation of strings is denoted as `$variable."some string"`.

Note

However, files that have been encrypted in Nextcloud versions 15 and lower may have slightly different structures.

13.9.1 Key type: master key

This is the default encryption mode in Nextcloud. With master key encryption enabled there is one central key that is used to secure the files handled by Nextcloud. The master key is protected by the instance *secret* that is generated at installation time. The advantage of the master key encryption is that the encryption is transparent to the users but has the disadvantage that the server administrator is able to decrypt user files without knowing any user password.

13.9.2 Key type: public sharing key

The public sharing key is used to secure files that have been publicly shared. The advantage of the public sharing key is that it is independent of the selected encryption mode so that Nextcloud is able to provide publicly shared files to outside parties.

13.9.3 Key type: recovery key

The recovery key is used to provide a restore mechanism in cases where the user key encryption is enabled, where the administrator has enabled the recovery key feature and the user has opted into using the recovery key feature. The recovery key can then be used to restore files when users have lost their passwords. The recovery key is protected by a recovery password that the server administrator should store securely. The advantage of the recovery key is that files can be recovered but has the disadvantage that the server administrator is able to decrypt user files without knowing any user password.

13.9.4 Key type: user key

User key encryption needs to be explicitly activated by calling `./occ encryption:disable-master-key`. In older versions of Nextcloud this had been enabled by default. With user key encryption enabled all users have their own user keys that are used to secure the files handled by Nextcloud. The user keys are protected by the user passwords. The advantage is that the server administrator is not able to decrypt user files without knowing any user password - unless the file is publicly shared or a recovery key is defined - but has the disadvantage that files are permanently lost if the users forget their user passwords - unless the files are (publicly) shared or a recovery key is defined.

> **Note**
>
> This method cannot be used with SAML authentication, because Nextcloud does not get a hold of any credentials whatsoever and therefore cannot use any users' passwords for encryption.

13.9.5 File type: public key file

Public key files contain RSA public keys that are used to encrypt/seal the share key files.

File format

Public key files are stored in PEM format.

File locations

The locations of public key files depend on their key type:

- master public key: `"files_encryption/OC_DEFAULT_MODULE/master_".$random.".publicKey"`
- public sharing public key: `"files_encryption/OC_DEFAULT_MODULE/pubShare_".$random.".publicKey"`
- recovery public key: `"files_encryption/OC_DEFAULT_MODULE/recoveryKey_".$random.".publicKey"`

- user public key: `$username."/files_encryption/OC_DEFAULT_MODULE/".$username."publicKey"`

13.9.6 File type: private key file

Private key files contain RSA private keys that are used to decrypt/unseal the share key files. The RSA private key is encrypted and signed with a password and stored in a format that is specific to the Nextcloud encryption module.

File format

The RSA private key that is represented in PEM format is encrypted and Base64 encoded (denoted as `$encryption`). For the encryption an initialization vector of 16 bytes is selected (denoted as `$iv`). Furthermore a hexadecimally encoded message authentication code of 64 bytes is calculated (denoted as `$signature`). The resulting file contains:

```
"HBEGIN:cipher:AES-256-CTR:keyFormat:hash:HEND".
$encrypted."00iv00".$iv."00sig00".$signature."xxx"
```

File locations

The locations of private key files depend on their key type:

- master private key: `"files_encryption/OC_DEFAULT_MODULE/master_".$random.".privateKey"`
- public sharing private key: `"files_encryption/OC_DEFAULT_MODULE/pubShare_".$random."privateKey"`
- recovery private key: `"files_encryption/OC_DEFAULT_MODULE/recoveryKey_".$random."privateKey"`
- user private key: `$username."/files_encryption/OC_DEFAULT_MODULE/".$username."privateKey"`

13.9.7 File type: share key file

Share key files contain so-called envelope keys that are needed to decrypt the file key files. The envelope keys are created by `openssl_seal()` during the encryption and are needed for `openssl_open()` during the decryption. The envelope keys are encrypted with the public keys of the recipients that are allowed to read the actual files.

File format

The envelope keys are stored in binary format.

File locations

The locations of share key files depend on the type of the encrypted file:

- regular file: `$username."/files_encryption/keys/files/".$filename."/OC_DEFAULT_MODULE/".$recipient.".shareKey"`
- version file: *version files use the same location for the share key file as their regular file*
- trashed file: `$username."/files_encryption/keys/files_trashbin/files/".$filename.".d".$timestamp."/OC_DEFAULT_MODULE/".$recipient.".shareKey"`
- trashed version file: *trashed version files use the same location for the share key file as their trashed file*

13.9.8 File type: file key file

File key files contain symmetric keys used to encrypt the actual files. The file keys consist of 32 random bytes and are encrypted/sealed with the envelope keys stored in the share key files.

File format

The file keys are stored in binary format.

File locations

The locations of the file key files depend on the type of the encrypted file:

- regular file: `$username."/files_encryption/keys/files/".$filename."/OC_DEFAULT_MODULE/fileKey"`
- version file: *version files use the same location for the file key file as their regular file*
- trashed file: `$username."/files_encryption/keys/files_trashbin/files/".$filename.".d".$delete_timestamp."/OC_DEFAULT_MODULE/fileKey"`
- trashed version file: *trashed version files use the same location for the file key file as their trashed file*

13.9.9 File type: file

Files contain the actual file content. The file content is encrypted and signed with a password and stored in a format that is specific to the Nextcloud encryption module.

File format

The file content is split into blocks of 6072 bytes. Each block is encrypted and Base64 encoded (denoted as `$encryption[0..$n]`). For the encryption an initialization vector of 16 bytes is selected for each block (denoted as `$iv[0..$n]`). Furthermore a hexadecimally encoded message authentication code of 64 bytes is calculated of each block (denoted as `$signature[0..$n]`). An encrypted block has a total size of 8192 bytes (8096 bytes for `$encrypted[]`, 6 bytes for `"00iv00"`, 16 bytes for `$iv[]`, 7 bytes for `"00sig00"`, 64 bytes for `$signature[]` and 3 bytes for `"xxx"`). Only the last encrypted block may be shorter. The header of the encrypted file is padded with 8147 bytes of `"-"` (denoted as `$padding`) to a total of 8192 bytes. The resulting file contains:

```
"HBEGIN:cipher:AES-256-CTR:keyFormat:hash:HEND".$padding.
$encrypted[0]."00iv00".$iv[0]."00sig00".$signature[0]."xxx".
$encrypted[1]."00iv00".$iv[1]."00sig00".$signature[1]."xxx".
$encrypted[2]."00iv00".$iv[2]."00sig00".$signature[2]."xxx".
[...]
$encrypted[$n]."00iv00".$iv[$n]."00sig00".$signature[$n]."xxx"
```

File locations

The locations of the files depend on the type of the encrypted file:

- regular file: `$username."/files/".$filename`
- version file: `$username."/files_versions/".$filename.".v".$version_timestamp`
- trashed file: `$username."/files_trashbin/files/".$filename.".d".$delete_timestamp`
- trashed version file: `$username."/files_trashbin/versions/".$filename.".v".$version_timestamp.".d".$delete_timestamp`

13.9.10 Key generation: generate the key pair

The key pair has to be generated with the `openssl_pkey_new()` function. Then the private key and public key are extracted from the the key resource with the `openssl_pkey_export()` function.

13.9.11 Key generation: store the public key

The public key is written to the `$username."publicKey"` file as documented in *File type: public key file*.

13.9.12 Key generation: store the private key

Derive the encryption key

The salt for the encryption key is derived by creating a raw SHA256 hash of `$uid.$instanceId.$instanceSecret` with the `hash()` function. `$instanceId` can be retrieved as `instanceid` from the `config.php`. `$instanceSecret` can be retrieved as `secret` from the `config.php`.

The encryption key is then derived by creating a raw SHA256-PBKDF2 hash of the password with the salt, 100.000 rounds and (by default) with a target size of 32 bytes (as required for AES-256-CTR) with the `hash_hmac()` function (denoted as `$passphrase`).

The used password depends on the key type:

- master private key: use `secret` from the `config.php`
- public sharing private key: use an empty password
- recovery private key: use the recovery password
- user private key: use the user password

Encrypt the private key

The initialization vector is generated as a random string of 16 bytes with the `random_bytes()` function (denoted as `$iv`). The private key is (by default) AES-256-CTR encrypted with the `$iv` and the `$passphrase` with the `openssl_encrypt()` function and returned as Base64 encoded without zero-padding (denoted as `$encrypted`).

Sign the private key

The message authentication key is derived by creating a raw SHA512 hash of `$passphrase.$version.$position."a"` with the `hash()` function.

- `$version` is always `"0"`.
- `$position` is always `"0"`.

The signature is then derived by creating a hexadecimally encoded SHA256-HMAC of `$encrypted` and the message authentication key with the `hash_hmac()` function (denoted as `$signature`).

Store the private key

The private key is written to the `$username."privateKey"` file with the derived `$encrypted`, `$iv` and `$signature` as documented in *File type: private key file*.

13.9.13 Encryption: generate the file key

Generate the file key

The file key is generated as a random string of 32 bytes with the `random_bytes()` function (denoted as `$filekey`).

Read the public key

The public keys of the recipients are read from the `$username."publicKey"` files as documented in *File type: public key file*.

Encrypt/seal the file key

The file key is encrypted/sealed with the `openssl_seal()` function with the public keys. This returns the encrypted file key and the encrypted envelope keys for the recipients.

Store the file key

The encrypted file key is stored in the `"fileKey"` file as documented in *File type: file key file*.

Store the envelope keys

The encrypted envelope keys for the recipients are stored in the `$username."shareKey"` files as documented in *File type: share key file*.

13.9.14 Encryption: encrypt the file

Split the file

The file is split into 6072 bytes sized blocks. Only the last encrypted block may be shorter. Each block is referenced by its zero-based index within the file (denoted as `$position`).

Encrypt the blocks

For each block the initialization vector is generated as a random string of 16 bytes with the `random_bytes()` function (denoted as `$iv[$position]`). The block is (by default) AES-256-CTR encrypted with the `$iv[$position]` and the `$filekey` with the `openssl_encrypt()` function and returned as Base64 encoded without zero-padding (denoted as `$encrypted[$position]`).

Sign the blocks

The message authentication key is derived by creating a raw SHA512 hash of `$filekey.$version.$position."a"` with the `hash()` function.

- `$version` is the `encrypted` value that can be retrieved from the `oc_filecache` table in the database and must not be zero. Take into account that a file in the `oc_filecache` table is identified by its `path` value as well as its `storage` value which references the `numeric_id` field in the `oc_storages` table. Including `$version` into the message authentication key prevents blocks from being swapped between different versions of the same file.
- `$position` is the index of the current block starting at `"0"` and is appended with `"end"` for the last block of the file. Including `$position` into the message authentication key prevents blocks from being swapped within the same file. Furthermore, adding `"end"` to the message authentication key of the last block prevents file truncation attacks.

The signature is then derived by creating a hexadecimally encoded SHA256-HMAC of `$encrypted[$position]` and the message authentication key with the `hash_hmac()` function (denoted as `$signature[$position]`).

Store the file

The encrypted file is written to the file with the derived `$encrypted[0..$n]`, `$iv[0..$n]` and `$signature[0..$n]` as documented in *File type: file*.

13.9.15 Decryption: read the private key

Read the private key file

The private key is read from the `$username."privateKey"` file and the values `$encrypted`, `$iv` and `$signature` are parsed as documented in *File type: private key file*.

Derive the decryption key

The salt for the decryption key is derived by creating a raw SHA256 hash of `$uid.$instanceId.$instanceSecret` with the `hash()` function. `$instanceId` can be retrieved as `instanceid` from the `config.php`. `$instanceSecret` can be retrieved as `secret` from the `config.php`.

The decryption key is then derived by creating a raw SHA256-PBKDF2 hash of the password with the salt, 100.000 rounds and (by default) with a target size of 32 bytes (as required for AES-256-CTR) with the `hash_hmac()` function (denoted as `$passphrase`).

The used password depends on the key type:

- master private key: use `secret` from the `config.php`
- public sharing private key: use an empty password
- recovery private key: use the recovery password
- user private key: use the user password

Check the signature

The message authentication key is derived by creating a raw SHA512 hash of `$passphrase.$version.$position."a"` with the `hash()` function.

- `$version` is always `"0"`.
- `$position` is always `"0"`.

The signature is then derived by creating a hexadecimally encoded SHA256-HMAC of `$encrypted` and the message authentication key with the `hash_hmac()` function. Only proceed when the derived signature is equal to *$signature* which is checked with the `hash_equals()` function.

Decrypt the private key

The private key is (by default) AES-256-CTR decrypted with the `$iv` and the `$passphrase` with the `openssl_decrypt()` function.

13.9.16 Decryption: read the file key

Read the file key

The encrypted file key is read from the `"fileKey"` file as documented in *File type: file key file*.

Read the envelope key

The encrypted envelope key for the recipient is read from the `$username."shareKey"` file as documented in *File type: share key file*.

Decrypt/unseal the file key

The encrypted file key is decrypted/unsealed with the `openssl_open()` function with the private key and encrypted envelope key for the recipient (denoted as `$filekey`).

13.9.17 Decryption: decrypt the file

Split the file

The encrypted file is split into a 8192 bytes sized header and one or more 8192 bytes sized blocks. Only the last encrypted block may be shorter. Each block is referenced by its zero-based index within the file (denoted as `$position`). The values `$encrypted[0..$n]`, `$iv[0..$n]` and `$signature[0..$n]` are parsed as documented in *File type: file*.

Check the block signatures

The message authentication key is derived by creating a raw SHA512 hash of `$filekey.$version.$position."a"` with the `hash()` function.

- `$version` is the `encrypted` value that can be retrieved from the `oc_filecache` table in the database and must not be zero. Take into account that a file in the `oc_filecache` table is identified by its `path` value as well as its `storage` value which references the `numeric_id` field in the `oc_storages` table. Including `$version` into the message authentication key prevents blocks from being swapped between different versions of the same file.
- `$position` is the index of the current block starting at `"0"` and is appended with `"end"` for the last block of the file. Including `$position` into the message authentication key prevents blocks from being swapped within the same file. Furthermore, adding `"end"` to the message authentication key of the last block prevents file truncation attacks.

The signature is then derived by creating a hexadecimally encoded SHA256-HMAC of `$encrypted[$position]` and the message authentication key with the `hash_hmac()` function. Only proceed when the derived signature is equal to `$signature[$position]` which is checked with the `hash_equals()` function.

Decrypt the blocks

Each block is (by default) AES-256-CTR decrypted with the `$iv[$position]` and the `$filekey` with the `openssl_decrypt()` function.

13.9.18 Sources

- encryption-recovery-tools repository on GitHub
- *Nextcloud Encryption Configuration documentation*
- Nextcloud Help response concerning the usage of version information
- Sourcecode: Creation of the Message Authentication Code
- Sourcecode: Derivation of the Encryption Key
- Sourcecode: Encryption of the File
- Sourcecode: Encryption/Sealing of the File Key
- Sourcecode: Extraction of the Private and Public Key
- Sourcecode: Generation of the File Key
- Sourcecode: Generation of the Initialization Vector
- Sourcecode: Generation of a Key Pair

13.10 Server-side encryption migration

13.10.1 Encryption format

Nextcloud still supports the legacy encryption scheme used for server side encryption where the encrypted files did not contain header information. This may still be used for installations that still have encrypted files from <= ownCloud 6. Files will be updated to the new encryption format once they are written again. However it is recommended to check if you have to still support this scheme.

Starting with version 20 for new installations the legacy encryption will be off by default. However if you are upgrading there is a migration path to check if you can disable legacy encryption.

Checking for old files

On the command line run:

```
occ encryption:scan:legacy-format
```

The command will tell you if you can remove the legacy encryption mode. If so set the *encryption.legacy_format_support* in your config.php to 'false'.

13.11 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.

Transactional File locking will use the database locking backend by default. This places a significant load on your database. Setting `memcache.locking` relieves the database load and improves performance. Admins of Nextcloud servers with heavy workloads should install a memcache. (See *Memory caching*.)

To use a memcache with Transactional File Locking, you must install a key-value store server like Valkey or Redis. Two cache backends are available for it:

- The Nextcloud `keyvalue` cache, which works with Valkey and Redis and does not require the installation of a PHP module (see *Redis / Valkey using the Nextcloud KeyValueCache*)
- The `Redis` cache, which requires the phpredis PHP module

13.11.1 Using the Nextcloud `KeyValueCache` cache

The `KeyValueCache` cache is built on a library bundled with Nextcloud, so no PHP module needs to be installed — you only need a running Valkey or Redis compatible server. Enter a configuration in your `config.php` file like this example:

```
'memcache.locking' => '\OC\Memcache\KeyValueCache',
'memcache.kvstore' => [
     'server' => [
          'host' => 'localhost',
          'port' => 6379,
     ],
     'password' => '', // Optional, if not defined no password will be used.
],
```

See *Redis / Valkey using the Nextcloud KeyValueCache* for all available options, including Unix sockets, TLS, Sentinel and cluster setups.

13.11.2 Using the Redis cache with the phpredis PHP module

To use the `Redis` cache instead, 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:

```
'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 your key-value store server to require a password. See http://redis.io/topics/security or https://valkey.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:

```
'memcache.locking' => '\OC\Memcache\Redis',
'redis' => array(
     'host' => '/run/redis/redis-server.sock',
     'port' => 0,
     'timeout' => 0.0,
      ),
```

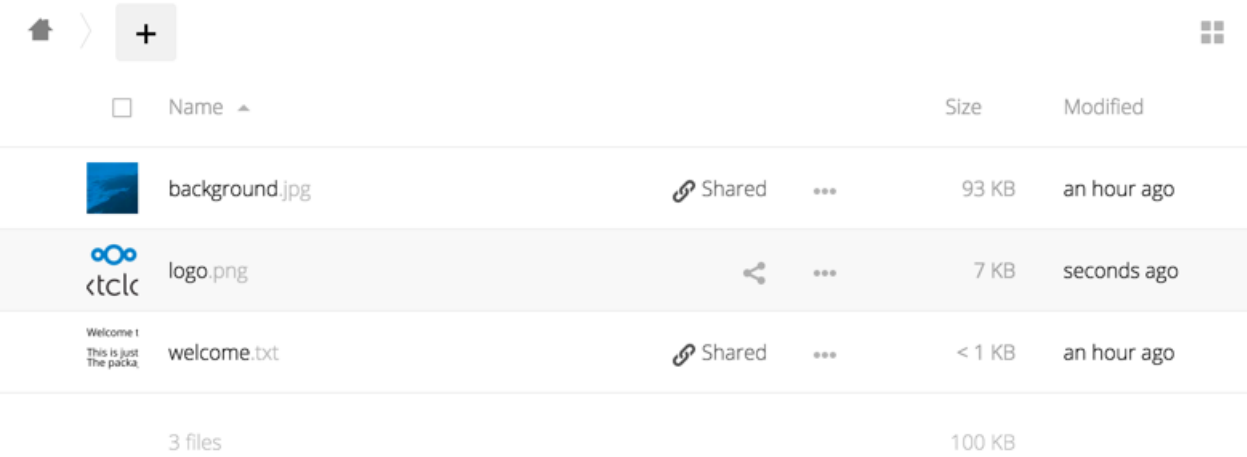
See `config.sample.php` to see configuration examples for Redis, and for all supported memcaches.

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

13.12 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
- Text documents

Note

Nextcloud can also generate previews of other file types (such as PDF, SVG, various Office document formats, and various video formats). Due to security and performance concerns those providers are disabled by default. While those providers are still available, we discourage enabling them and they are considered unsupported. The full list of the preview providers that are enabled by default (as well as those disabled by default) can be found under the `enabledPreviewProviders` *configuration parameter*.

Some preview providers depend on external binaries on the server. Video previews such as `OC\Preview\Movie` and `OC\Preview\MKV` require `ffmpeg`. Office document previews such as `OC\Preview\OpenDocument` and `OC\Preview\MSOfficeDoc` require LibreOffice. Image formats handled through ImageMagick, such as `OC\Preview\Illustrator`, `OC\Preview\SVG` and `OC\Preview\TIFF`, require the corresponding ImageMagick support.

13.12.1 Parameters

Please notice that the Nextcloud preview system comes already with sensible defaults, and therefore it is usually unnecessary to adjust those configuration values.

But deemed necessary, following changes have to be made in `config/config.php` file. As a best practice, take a backup of this config file before making a lot of changes.

After changing one or more of the following parameters, you might want to run the `preview:cleanup` occ command to get rid of the previews with obsolete settings. See *Preview* to learn more.

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` to `false`:

```
<?php
  'enable_previews' => false,
```

Maximum preview size:

There are two configuration options to set the maximum size of a preview.

```
<?php
  'preview_max_x' => null,
  'preview_max_y' => null,
```

By default, both options are set to null. 'Null' is equal to no limit. Numeric values represent the size in pixels. The following code limits previews to a maximum size of 100×100px:

```
<?php
  'preview_max_x' => 100,
  'preview_max_y' => 100,
```

'preview_max_x' represents the x-axis and 'preview_max_y' represents the y-axis.

Maximum scale factor:

If a lot of small pictures are stored on the Nextcloud instance and the preview system generates blurry previews, you might want to consider setting a maximum scale factor. By default, pictures are upscaled to 10 times the original size:

```
<?php
  'preview_max_scale_factor' => 10,
```

If you want to disable scaling at all, you can set the config value to '1':

```
<?php
  'preview_max_scale_factor' => 1,
```

If you want to disable the maximum scaling factor, you can set the config value to 'null':

```
<?php
  'preview_max_scale_factor' => null,
```

JPEG quality setting:

Default JPEG quality setting for preview images is '80'. Change this with:

```
occ config:app:set preview jpeg_quality --value="60"
```

Maximum memory for image generation:

By default, Nextcloud generates image previews using the GD Graphics Library. This configuration option limits the amount of memory that is allowed for preview generation. If creating the preview image would allocate more memory than the limit, preview generation will be disabled and the default mimetype icon is shown.

Default limit is 256 MB. Set to `-1` for no limit.

```
<?php
  'preview_max_memory' => 256,
```

13.13 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. Nextcloud will always keep the latest version in each of the time windows.

The Versions app never uses more than 50% of the user's currently available free space. If the stored versions exceed this limit, Nextcloud deletes the oldest file versions until it meets the disk space limit again.

Nextcloud manages file versions using a combination of on-save pruning and scheduled cleanup. This ensures that versions are retained while respecting storage quotas.

13.13.1 During Version Creation

Nextcloud automatically creates new file versions whenever a file is modified, allowing users to restore previous states when needed. After each new version is stored, the system automatically checks storage limits and retention rules. Versions are filtered according to the above pattern to keep representative versions and remove redundant ones. If the user's quota is exceeded, auto-expiry is triggered. When storage space runs low, Nextcloud sorts all versions from oldest to newest and removes the oldest ones first, while always preserving at least the two most recent versions to free up space.

13.13.2 During the Regular Background Job

Nextcloud runs a background cleanup task that automatically removes old file versions for each user. During this process, the system checks the user's version storage folder and identifies versions that are older than the configured maximum retention period or whose original files no longer exist. When an outdated or orphaned version is found, it is safely deleted from both the filesystem and the version database to reclaim storage space and maintain consistency.

> **Note**
>
> Versions named by a user will never be deleted.

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 automatic expiration (pruning) of file versions; file versions will continue to be created, but no old file versions will be deleted automatically.

13.13.3 Background job

To delete expired versions a background jobs runs every 30 minutes. It's possible to deactivate the background job and setup a (system) cron to expire the versions via occ.

Deactivate background job: `occ config:app:set --value=no files_versions background_job_expire_versions`

Activate background job: `occ config:app:delete files_versions background_job_expire_versions`

Expire versions: `occ versions:expire` or `occ versions:expire --quiet` (without the progress bar)

13.14 Deleted Files

13.14.1 Introduction

When a file or folder is deleted, Nextcloud normally moves it to the trash bin instead of deleting it immediately. This allows the item to be restored later.

Items in the trash bin are permanently deleted when one of the following occurs:

- The account owner selects *Delete permanently*.
- The configured retention policy expires the item.
- The effective trash-bin size limit is exceeded and the item is eligible for removal to free space.

Logged-in accounts can access their trash bin by selecting *Deleted files* in the **Files** area of the Nextcloud Web interface. Items in the trash bin can be restored, downloaded, or permanently deleted.

For information about using the trash bin as an account holder, see the Deleted files section of the User Manual.

> **Note**
>
> If the *Versions* app is enabled, versions associated with a deleted file are moved to the trash bin and are restored when the file is restored.

When a file is restored, Nextcloud attempts to restore it to its original location. If that location no longer exists or is not writable, the file is restored to the account's files root. If a file with the same name already exists, the restored file receives a unique name.

This functionality is provided by the *Deleted files* app, whose application ID is `files_trashbin`. The app provides:

- The web interface for viewing, restoring, downloading, and permanently deleting items.
- The trash-bin filesystem integration.
- Automatic expiration through a background job.
- The `trashbin:size`, `trashbin:expire`, `trashbin:cleanup`, and `trashbin:restore` commands.

13.14.2 Policy Configuration

The `files_trashbin` app uses two related mechanisms to determine when items are permanently deleted:

- A time-based retention policy configured with `trashbin_retention_obligation`.
- An effective trash-bin size limit, configured explicitly or calculated from the account's available space.

The default retention policy is `auto`. In the default configuration, items are retained for at least 30 days. After that minimum retention period, eligible items may be permanently deleted when space is needed.

The default effective trash-bin size is calculated as follows:

- For an account with a quota, the trash bin may use up to 50% of the account's remaining quota space.
- For an account without a quota, the trash bin may use the available filesystem space.
- If an explicit global or per-account trash-bin size is configured, that value overrides the calculated default.

The 50% calculation is based on the account's **remaining** quota space, not the account's total quota. For example, if an account has a 10 GB quota and is already using 8 GB, the default trash-bin allowance is calculated from the remaining 2 GB.

When the effective trash-bin limit is exceeded, Nextcloud processes the oldest items first. An item can be removed to free space only when it is eligible under the configured retention policy.

The global retention policy is configured with the `trashbin_retention_obligation` setting in `config/config.php`:

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

Available retention policy values are described below. `D1` and `D2` are numbers of days.

`auto`
: The default policy. This is equivalent to `auto, auto`.

  Files and folders are retained for at least 30 days. After 30 days, they may be permanently deleted when space is needed.

`D1, auto`
: Files and folders are retained for at least `D1` days. After that, they may be permanently deleted when space is needed.

`auto, D2`
: Files and folders may be permanently deleted when space is needed, regardless of age. In addition, all items older than `D2` days are permanently deleted automatically, even when space is not currently needed.

`D1, D2`
: Files and folders are retained for at least `D1` days. Items older than `D2` days are permanently deleted automatically.

  If the effective trash-bin size is exceeded, items older than `D1` days may also be deleted early to free space, oldest first. Items younger than `D1` days are protected from size-based cleanup.

  Items older than `D2` days are deleted automatically even when space is not currently needed. If `D2` is smaller than `D1`, Nextcloud adjusts the effective maximum retention period to `D1`.

`disabled`
: Automatic trash-bin expiration is disabled. Files and folders remain in the trash bin until they are manually deleted or another operation removes them.

A single numeric value is also accepted. The value is used as the minimum retention period, while the default maximum retention period is 30 days. If the resulting maximum period would be shorter than the minimum period, the maximum is raised to match the minimum.

For example, `10` is equivalent to `10, 30`: items are retained for at least 10 days and are automatically deleted once they exceed 30 days. By contrast, `45` is equivalent to `45, 45`, because the default maximum of 30 days is shorter than the 45-day minimum and is therefore raised to match it.

> **Note**
>
> Unlike `auto`, `D1, auto`, and `auto, D2`, this form (including a bare numeric value) disables purely space-triggered cleanup of the trash bin. Items may still be removed early if the trash-bin size limit is exceeded, but only once they are older than the minimum retention period (`D1`).

For clarity, use the explicit `D1, D2` form when configuring a numeric retention policy.

13.14.3 Trash-bin Size Limits

The `trashbin:size` command configures or displays the effective trash-bin size.

To display the global default and any per-account values:

```
sudo -E -u www-data php occ trashbin:size
```

To configure a global trash-bin size:

```
sudo -E -u www-data php occ trashbin:size 10G
```

To configure a size for a specific account:

```
sudo -E -u www-data php occ trashbin:size --user USER_ID 10G
```

The `size` argument accepts values understood by Nextcloud's file-size parser, such as `500M` or `10G`.

A per-account value takes precedence over the global value. If no per-account value is configured, the account uses the global value. If neither value is configured, Nextcloud uses the calculated default based on the account's remaining quota space or available filesystem space.

To display the configured value for one account:

```
sudo -E -u www-data php occ trashbin:size --user USER_ID
```

Changing a per-account size schedules cleanup for that account. Changing the global size causes existing trash bins to be cleaned up, although an account's trash bin can temporarily exceed the newly configured size until cleanup is performed or another item is moved to the trash bin.

Interaction with Quotas

An explicit trash-bin size is independent of the account quota. It can therefore be used for accounts both with and without quotas.

For accounts with quotas, the following order applies:

1. A per-account `trashbin_size` value is used when configured.
2. Otherwise, the global `files_trashbin trashbin_size` value is used when configured.
3. Otherwise, the default limit is calculated from 50% of the account's remaining quota space.

When no explicit trash-bin size is configured for an account without a quota, the available filesystem space is used to determine whether the trash bin needs cleanup.

Trash-bin cleanup does not necessarily remove all deleted files when an account exceeds its quota. Items are removed according to the retention policy and in oldest-first order until the effective trash-bin space requirement is satisfied.

13.14.4 Background Jobs

The `files_trashbin` app registers a background job for policies with a configured maximum retention age, such as `auto, D2` and `D1, D2`. The job runs at a 30-minute interval, processes accounts in batches of 10, and runs for no more than 30 minutes per invocation.

For the default `auto` policy and other policies without a configured maximum retention age, the background job does not perform an expiration pass. For `disabled`, automatic expiration is disabled.

Space-based cleanup can instead be scheduled when trash-bin usage or available space changes.

The background job is controlled by the `files_trashbin` app configuration value `background_job_expire_trash`. It is enabled by default.

To disable the background job:

```
sudo -E -u www-data php occ config:app:set \
   --value=no files_trashbin background_job_expire_trash
```

To re-enable it, remove the override:

```
sudo -E -u www-data php occ config:app:delete \
   files_trashbin background_job_expire_trash
```

When the background job is disabled, administrators can run expiration manually with `occ trashbin:expire` or configure an external system task to run that command.

13.14.5 Other Commands

The below operations are useful under certain circumstances, but do not typically need to be used directly.

Expiring Items Manually

To expire eligible items for all accounts:

```
sudo -E -u www-data php occ trashbin:expire
```

To expire eligible items for one or more accounts:

```
sudo -E -u www-data php occ trashbin:expire USER_ID
```

```
sudo -E -u www-data php occ trashbin:expire USER_ID_1 USER_ID_2
```

When no account IDs are provided, the command automatically discovers accounts that have previously logged in. This normally covers all accounts that can have trash-bin contents. For an account with imported or otherwise provisioned trash-bin data but no recorded login, provide its account ID explicitly.

The command displays a progress bar when processing all accounts. Use `--quiet` to suppress the progress bar:

```
sudo -E -u www-data php occ trashbin:expire --quiet
```

For any active retention policy, including the default `auto` policy, `trashbin:expire` performs a full expiration pass: it deletes items that exceed the configured maximum age, and it also deletes items to satisfy the effective trash-bin size limit when needed, subject to the retention policy's minimum age.

If trash-bin expiration is disabled (`trashbin_retention_obligation` set to `disabled`), `trashbin:expire` does not delete any items and exits with a non-zero status.

Space-based cleanup can also be triggered automatically outside of `trashbin:expire` and the background job — for example, when a file is moved to the trash bin or when a change to the account's available space causes the effective trash-bin limit to be exceeded.

The trash-bin expiration command is provided by the `files_trashbin` app:

```
sudo -E -u www-data php occ list files_trashbin
```

Removing All Trashed Items

To immediately and permanently delete **all** trashed items for all users from all configured user backends, bypassing the retention policy entirely, use `trashbin:cleanup --all-users`:

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

To clean up the trash bin for one or more specific accounts:

```
sudo -E -u www-data php occ trashbin:cleanup USER_ID_1 USER_ID_2
```

The command requires either one or more account IDs or `--all-users`. These forms are mutually exclusive.

Danger

This command deletes trashed items unconditionally and does not honor the configured retention policy. Deleted items cannot be recovered.

Restoring All Trashed Items

To restore all trashed items for one or more accounts:

```
sudo -E -u www-data php occ trashbin:restore USER_ID_1 USER_ID_2
```

To restore trashed items for all users from all configured user backends:

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

Account IDs and `--all-users` are mutually exclusive. The command requires one or the other.

The command supports additional filters, including `--since`, `--until`, and `--scope` (`user`, `groupfolders`, or `all`), as well as `--dry-run` to preview which items would be restored without performing any action. See `occ trashbin:restore --help` for details.

13.15 File conversion

In the majority of cases, it is recommended that either the Pandoc or *Nextcloud Office* apps are enabled, as they provide the majority of file conversions available. Other apps may also support their own file conversions for other file types if more specific conversions are required.

13.16 Windows compatible filenames

> **Note**
>
> This feature was introduced in Nextcloud 31.

By default Nextcloud supports all filenames which are valid on the underlying server. As Nextcloud runs only on POSIX compatible operating systems (Linux), this means that Nextcloud supports also filenames not valid on Microsoft Windows systems.

If your users use Windows and use the Nextcloud Desktop clients to synchronize their work to their computer they might encounter files created in the web interface, or on a Linux machine, which cannot be synchronized as the filename is not valid.

To solve this issue it is possible to enforce filenames only valid on Windows, this for example forbids characters like `*` from filenames or filenames like `AUX.txt` (on Windows `AUX` is a reserved name and cannot be used).

> **Note**
>
> Enabling this setting will not enforce case-insensitivity as modern Windows systems support case-sensitive filenames.

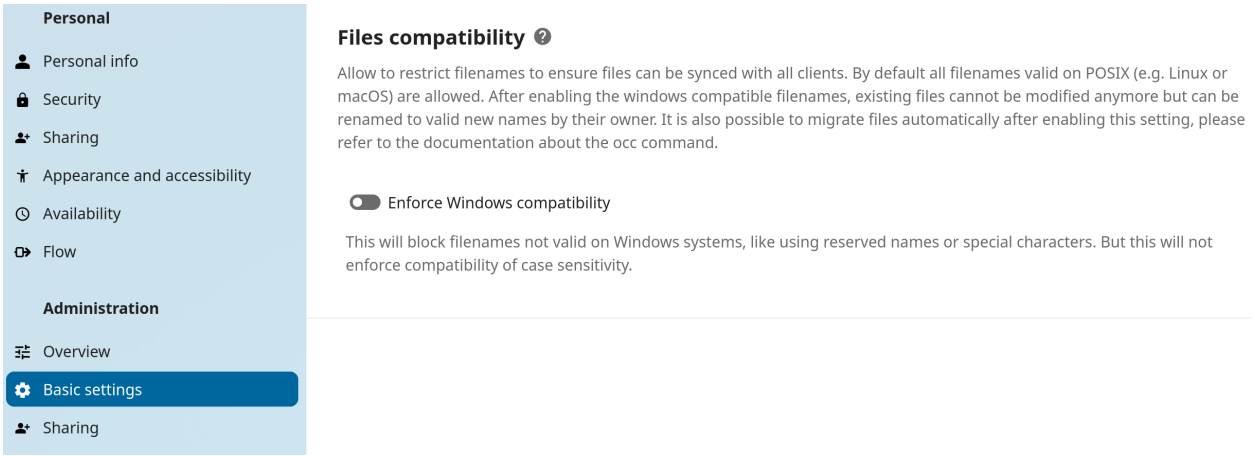
13.16.1 Enabling Windows compatible filenames

This feature can be enabled either by using the web interface or by using an `occ` command.

> **Note**
>
> This feature works by setting a predefined set of system configuration settings. So after enabling this the `config.php` will be adjusted, which also means enabling this feature requires a writable configuration.

Using the web interface

The setting is provided in the **Administration settings** under **Basic settings**. Within the **Files compatibility** section the Windows compatibility can be enabled.



Using the occ command

> **Note**
>
> This command was introduced in Nextcloud 32.

To quickly enable or disable the feature an *occ command* is provided.

13.16.2 Consequences

After enabling Windows compatible filenames users cannot create or modify files with invalid filenames. But they can still delete or rename those files (to valid names).

This works by setting a pre-defined set of configuration settings:

- `forbidden_filename_basenames` will be set to names reserved on Windows.
- `forbidden_filename_characters` will be set to characters not valid for filenames on Windows.
- `forbidden_filename_extensions` will be set to strings not allowed as trailing parts, like a trailing dot or spaces.

13.16.3 Sanitizing invalid filenames

After enabling the feature the users have to manually adjust all invalid filenames to be able to keep working with them. As an alternative Nextcloud provides the *occ files:sanitize-filenames* command to automatically rename all invalid files.

CHAPTER

FOURTEEN

FLOW

14.1 Flow configuration

Administrators can disable user flows since they can have an impact on the performance of a system and you might not want to give users the ability to define their own flows rules. They can be disabled through the following command:

```
occ config:app:set workflowengine user_scope_disabled --value yes
```

14.2 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*.

Note

In case you are using the *Context Chat App*, please keep in mind, that it is not affected by the File Access Control rules and will respond with indexed information, even when the file is not accessible by the user due to access control rules.

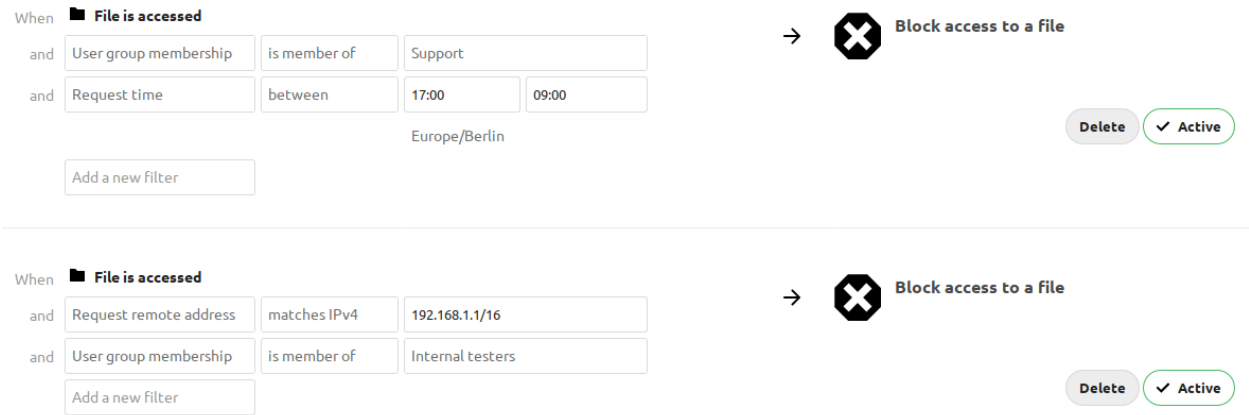
14.2.1 Denied access

If access to a file has been denied for a user, the user can not:

- Create/upload the file
- Modify the files
- Delete the file
- Download the file
- Synchronize the file with clients, such as the Nextcloud desktop and mobile clients

14.2.2 Examples

After installing the File Access Control app as described in *Apps management* navigate to the configuration and locate the settings for the Flow application.



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

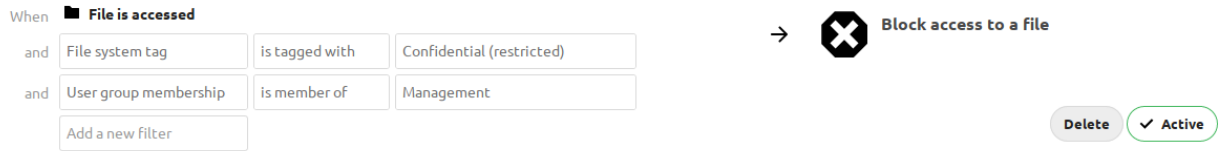
The second rule group `Internal testing` prevents users of the Internal testers group to access files from outside of the local network.

14.2.3 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. Therefore restricted and invisible tags are recommended, otherwise a user could remove and reassign the tag.

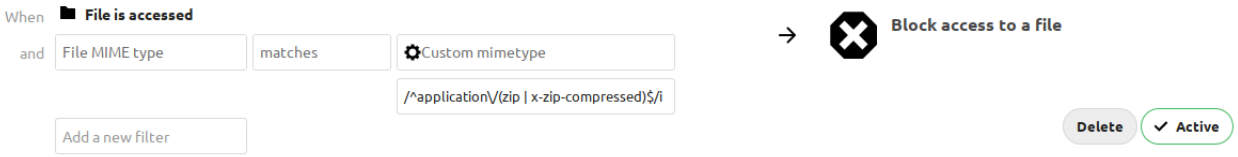
This example blocks access to any folder with the tag `Confidential`.



14.2.4 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`



14.2.5 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 therefore 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. Therefore it is recommended to disable the `Allow users to mount external storage` option, when trying to to completely lock out users.

14.2.6 Available rules

All rules can also be inverted (from `is` to `is not`) using the operator option.

- **File collaborative tag:** Either the file itself, or any of the file owner's parent folders needs to be tagged with the tag.

  > **Note**
  >
  > Tags used in access control rules should be restricted tags, otherwise any user can remove the tag to access the file again. The best way to do this is with the *Automated tagging of files*.

- **File MIME type:** The MIME type of the file, e.g. `text/plain` for a text file or `httpd/unix-directory` for a folder.

  > **Note**
  >
  > see mimetypealiases.dist.json for a full list of possible MIME types.

- **File name:** The name of the file (`is` and `is not` are case-insensitive)
- **File size:** The size of the file (*Only available on upload*)
- **Request remote address:** An IP range (either v4 or v6) for the accessing user
- **Request time:** Time span and timezone when the request happens

- **Request URL:** The URL which requests the file. (*This is the URL the file is served from, not the URL the user is currently looking at.*)
- **Request user agent:** The user agent of the users browser or client. Nextcloud desktop, Android and iOS clients are available as preconfigured options.
- **User group membership:** Whether the user is a member of the given group.

14.3 Automated tagging of files

Nextcloud's Files Automated Tagging app allows to assign collaborative tags to files and folders based on rules, similar to *Files access control*.

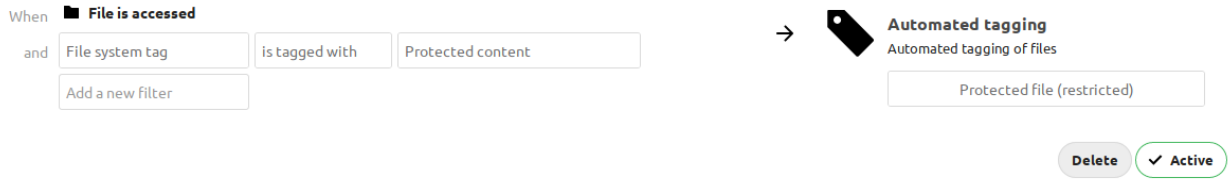
14.3.1 Assigning restricted and invisible tags

The main functionality of this app is to allow users to indirectly assign restricted and invisible tags to files they upload.

This is especially useful for retention and *Files access control*, so people that got the files shared can not remove the tag to stop the retention or allow access against the owner's will.

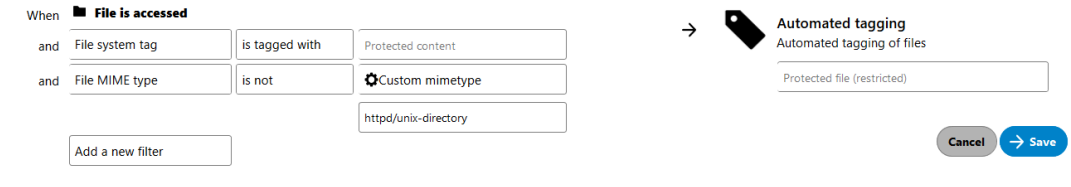
14.3.2 Example

After installing the Files automated tagging app as described in *Apps management* navigate to the configuration and locate the Workflow settings.



In the example you can see a simple rule with only one condition. It will tag all files with the restricted tag `Protected file` that are uploaded into a folder that is tagged with `Protect content`. No user can remove the tag `Protected file` and therefore access control and retention both work fine without users being able to work around them.

In this case the folder will also be tagged with tag `Protected file`, to avoid this, simply modify the rule to exclude Directory `httpd/unix-directory` from it.



14.3.3 Available rules

The available rules can be seen in the access control section: *Available rules*.

> **Note**
>
> Please note that the rules do not apply when creating external storages and groupfolders. The root folders of those need to be tagged manually with the desired initial tags. Items created inside later on apply the rules as defined.

14.3.4 Executing actions

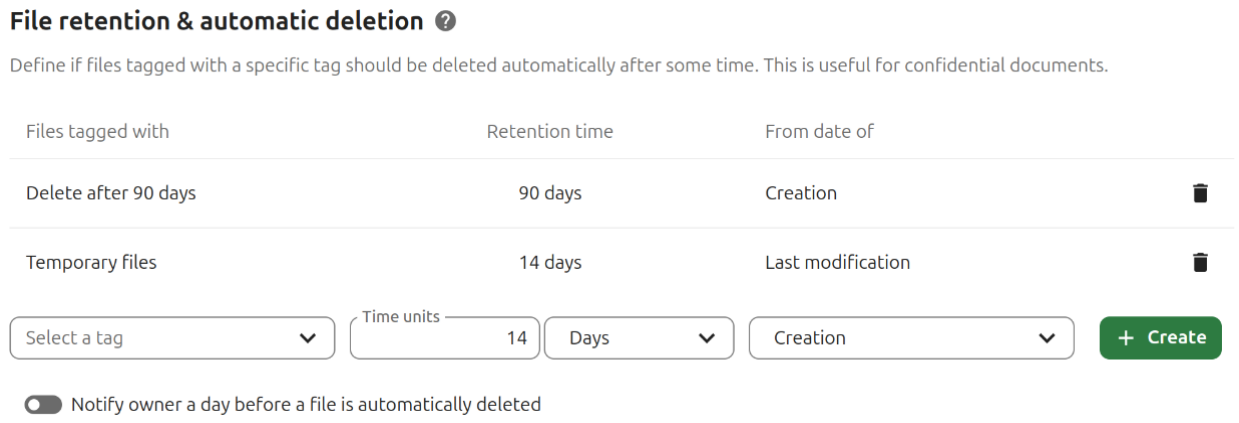
It is possible to execute actions like `convert to PDF` based on assigned tags. Nextcloud GmbH assists customers in this with hands-on help and documentation on our customer portal.

14.4 Retention of files

Nextcloud's Files Retention app allows to automatically delete files that are tagged with a collaborative tag and have a certain age.

14.4.1 Example

After installing the Retention app as described in *Apps management* navigate to Administration settings and then to Flow.



The rule from the example will delete all files tagged with `Temporary file` 14 days after the creation.

You can also use the "Notify owner a day before a file is automatically deleted" option to make sure the file owner will get a notification before a file will be deleted.

14.4.2 File age

There are 2 options available that can be used to decide when to delete a file:

- **Creation:** Time the file was created on the Nextcloud Server or uploaded to it.
- **Last modification:** Time when the file was last modified. Uploading also counts as a modification, so files that have not been modified since a long time before uploading are not deleted shortly after the upload.

14.4.3 Common misconfigurations

Public collaborative tag

Similar to *Files access control* retention should use `restricted` or `invisible` tags. Otherwise any user can remove the tag and the file is not removed after the given period. Use *Automated tagging of files* to assign such tags to newly uploaded files.

CHAPTER

FIFTEEN

MIMETYPES MANAGEMENT

15.1 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 -E -u www-data php occ maintenance:mimetype:update-js

# you may also need to update the mimetype for existing files, see nextcloud/server
↪#30566
$ sudo -E -u www-data php occ maintenance:mimetype:update-db --repair-filecache
```

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

15.2 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.

15.3 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

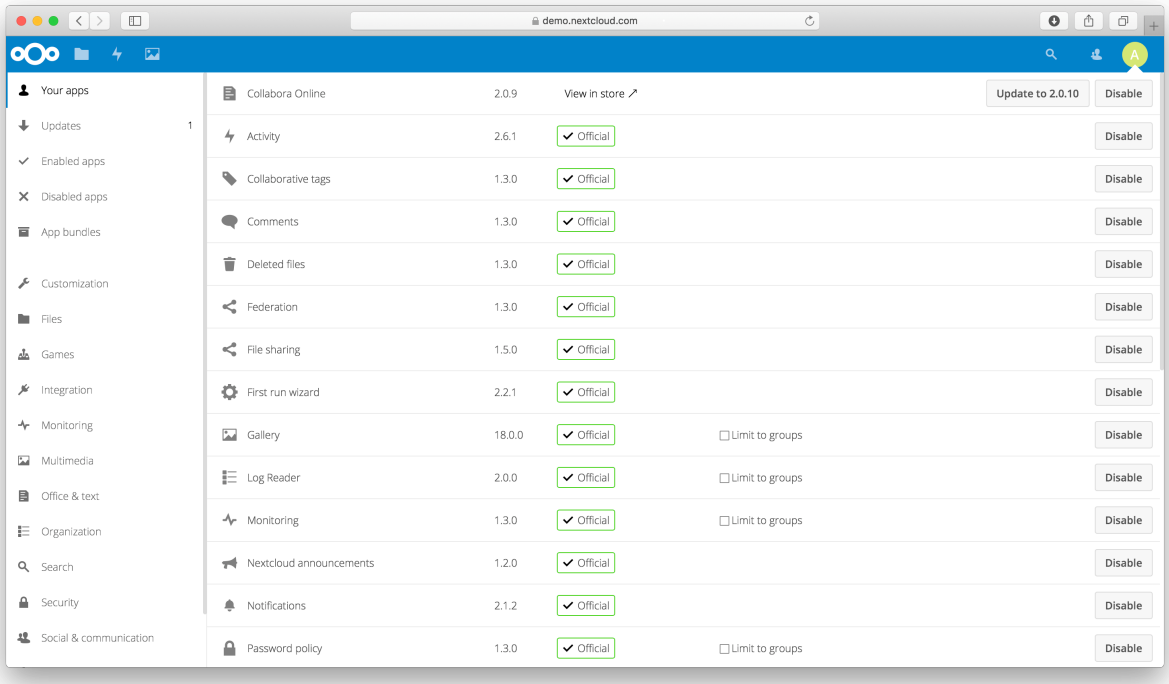
SIXTEEN

APPS MANAGEMENT

Nextcloud apps can enhance, customize, or restrict the features and experience available to you and your users on the Nextcloud server. In addition to built-in apps such as Files, Activity, and Photos, other apps such as Calendar, Contacts, and Talk can further expand its functionality.

After installing the Nextcloud server, you might want to consider enabling, disabling, or restricting some apps for specific groups, depending on your needs and those of your users.

16.1 Apps



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

Those apps are supported and developed by Nextcloud GmbH directly and have a **Featured** tag.

Note

Your Nextcloud server needs to be able to communicate with `https://apps.nextcloud.com`, `https://ltd[1-3].nextcloud.com`, `https://garm[1-5].nextcloud.com` to list and download apps. Please make sure to allow these hosts in your firewall or proxy if necessary.

Note

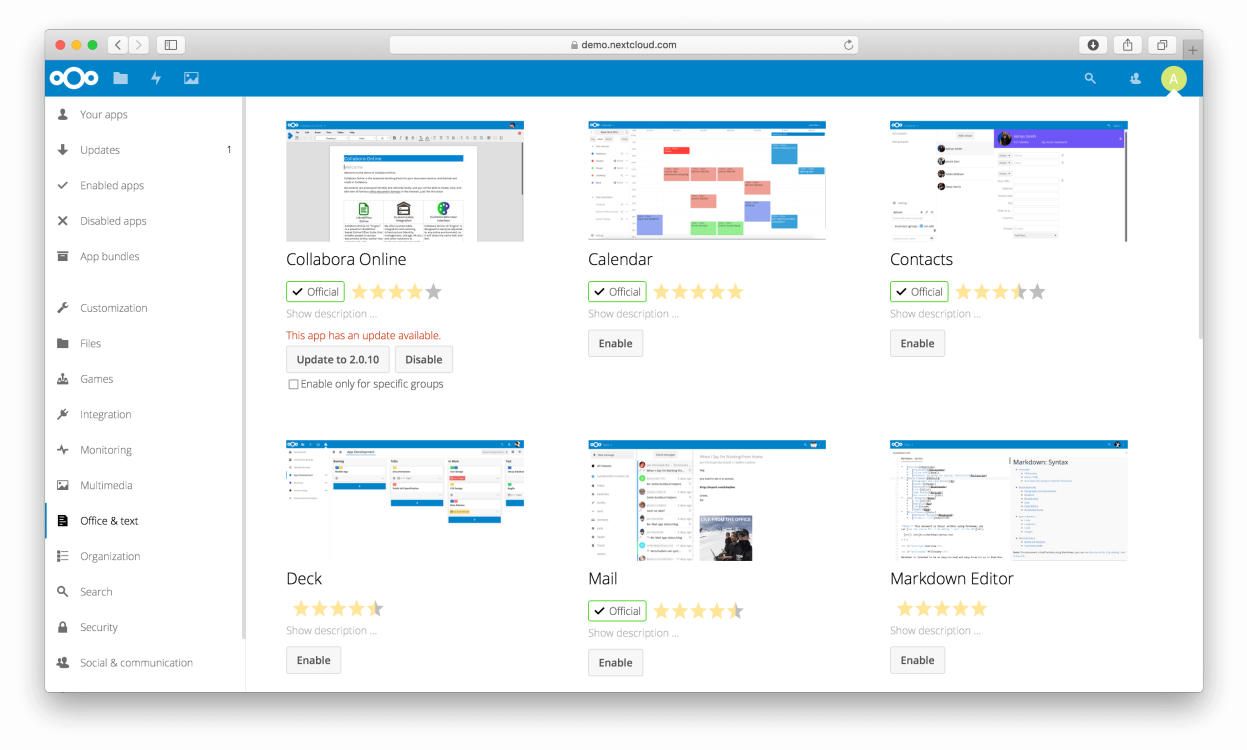
To get access to work-arounds, long-term-support, priority bug fixing and custom consulting for supported apps, contact Nextcloud GmbH.

Note

If you would like to develop your own Nextcloud app, you can find out more information in our developer manual.

All apps must be licensed under AGPLv3+ or any compatible license.

16.2 Managing apps



You will see which apps are enabled, disabled and available. You'll also see additional app bundles and filters, such as Customization, Security and Monitoring for finding more apps quickly.

On the Apps page, you can enable or disable applications. Some apps have configurable options on the Apps page, such as **Enable only for specific groups**, but mainly they are enabled or disabled here, and are configured in your Nextcloud

settings (admin and/or user-settings) or in the `config.php`.

Select an app to view its description and available configuration options. Clicking the **Enable** button will enable the app. If the app is not part of the Nextcloud installation, it will be downloaded from the app store, installed and enabled.

App updates will also be offered to you on this page. Simply click on the **Update** button to update a specific app or use the **Update all** button on top of the page to update all apps.

> **Note**
>
> **Beta releases**: You can also install beta releases of apps directly from here by switching your Nextcloud to the beta channel in the admin overview.

16.2.1 Update notifications

The default `updatenotification` app sends administrators notifications about available app and Nextcloud updates. Moreover, since Nextcloud 29, this app also allows to notify users about updated apps and the changes that are included in the update. This notification is enabled by default if the app provides a changelog.

To disable user notifications use:

```
occ config:app:set --type boolean --value="false" updatenotification app_updated.
↪enabled
```

By default. when using the `guests` app, guest users are not notified. To enable notifications for them use:

```
occ config:app:set --type boolean --value="true" updatenotification app_updated.
↪notify_guests
```

16.2.2 Managing apps with `occ`

In addition to managing apps via the web interface, administrators can enable or disable apps with the `occ` command.

To enable an app, use the following command:

```
occ app:enable <app-id>
```

For example, to enable the “files” app, run:

```
occ app:enable files
```

To enable the app for specific groups, use the `--groups` option:

```
occ app:enable files --groups=admin
```

This command enables the “files” app only for the “admin” group.

To disable an app, use:

```
occ app:disable <app-id>
```

16.2.3 Using private API

If a third-party app uses private APIs instead of public APIs, installation fails when `'appcodechecker' => true,` is set in `config.php`.

16.2.4 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.

Example: To ensure that the default `/apps/` folder only contains apps shipped with Nextcloud, follow this example to set up an `/extra-apps/` folder which will be used to store any additional apps you install:

```
"apps_paths" => [
    [
            "path"     => OC::$SERVERROOT . "/apps",
            "url"      => "/apps",
            "writable" => false,
    ],
    [
            "path"     => OC::$SERVERROOT . "/extra-apps",
            "url"      => "/extra-apps",
            "writable" => true,
    ],
],
```

Danger

Make sure that the values you choose for `path` and `url` for any custom apps directories do not conflict with directories which already exist in your Nextcloud Server root (installation directory).

Tip

Apps paths can be located outside the server root. However, for any **path** outside the server root, you need to create a symbolic link in the server root that points **url** to **path**. For instance, if **path** is `/var/local/lib/nextcloud/extra-apps`, and **url** is `/extra-apps`, then you would use the command `ln` to create the symbolic link like this: `ln -sf /var/local/lib/nextcloud/extra-apps ./extra-apps`

16.2.5 Configuring the app store timeout

The timeout for requests to fetch app store metadata can be configured with the `appstore-timeout` app setting. The value is specified in seconds. This setting is configured with `occ` and is not a `config.php` parameter.

For example, to set the timeout to 180 seconds:

```
occ config:app:set settings appstore-timeout --value=180
```

The default timeout is 120 seconds. To restore the default, remove the custom setting:

```
occ config:app:delete settings appstore-timeout
```

Changed in version 33.0.0: The default app store request timeout was increased from 60 to 120 seconds and can be configured with `occ`.

16.2.6 Using a self-hosted app store

This section explains how to enable installation of apps from a self-hosted app store. At least one of the configured apps directories must be writable.

To enable a self-hosted app store:

1. Set the `appstoreenabled` parameter to `true`.

   This parameter is used to enable the app store in Nextcloud.

2. Set the `appstoreurl` to the URL of your Nextcloud app store.

   This parameter is used to set the HTTP path to your self-hosted Nextcloud app store.

```
"appstoreenabled" => true,
"appstoreurl" => "https://my.appstore.instance/v1",
```

By default the app store is enabled and configured to use `https://apps.nextcloud.com/api/v1` as the app store URL. Nextcloud will fetch `apps.json` and `categories.json` from there. To use the defaults again, remove the `appstoreenabled` and `appstoreurl` parameters from the configuration.

Example: If `categories.json` is available at `https://apps.nextcloud.com/api/v1/categories.json` the app store URL is `https://apps.nextcloud.com/api/v1`.

CHAPTER

SEVENTEEN

APPS MANAGEMENT API

17.1 Get list of apps

Returns a list of apps installed on the Nextcloud server. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/apps/

- HTTP method: GET
- url argument: filter, string - optional (`enabled` or `disabled`)

Status codes:

- 100 - successful
- 101 - invalid input data

17.1.1 Example

```
$ curl -X GET http://admin:secret@example.com/ocs/v1.php/cloud/apps?filter=enabled -H
↪"OCS-APIRequest: true"
```

- Gets enabled apps

17.1.2 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>
```

17.2 Get app info

Provides information on a specific application. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/apps/{appid}

- HTTP method: GET

Status codes:

- 100 - successful

17.2.1 Example

```
$ curl -X GET http://admin:secret@example.com/ocs/v1.php/cloud/apps/files -H "OCS-
↪APIRequest: true"
```

- Get app info for the `files` app

17.2.2 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-3.0-or-later</licence>
    <author>Robin Appelman</author>
    <require>4.9</require>
    <shipped>true</shipped>
    <active>true</active>
    <standalone></standalone>
    <default_enable></default_enable>
    <types>
      <element>filesystem</element>
    </types>
  </data>
</ocs>
```

17.3 Enable an app

Enable an app. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/apps/{appid}

- HTTP method: POST

Status codes:

- 100 - successful

17.3.1 Example

```
$ curl -X POST http://admin:secret@example.com/ocs/v1.php/cloud/apps/files_texteditor␣
↪-H "OCS-APIRequest: true"
```

- Enable the `files_texteditor` app

17.3.2 XML output

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

17.4 Disable an app

Disables the specified app. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/apps/{appid}

- HTTP method: DELETE

Status codes:

- 100 - successful

17.4.1 Example

```
$ curl -X DELETE http://admin:secret@example.com/ocs/v1.php/cloud/apps/files_
↪texteditor -H "OCS-APIRequest: true"
```

- Disable the `files_texteditor` app

17.4.2 XML output

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

(continues on next page)

(continued from previous page)

```
  </meta>
</ocs>
```

CHAPTER

EIGHTEEN

EXAPPS MANAGEMENT

18.1 AppAPI and External Apps

Previously, Nextcloud only supported applications written in the PHP programming language. In order to support a wider range of use cases, an ecosystem for **ExApps** (short for "External Apps") was introduced, allowing for the installation of apps as Docker containers.

Most of our *Artificial Intelligence* (AI) apps are developed as ExApps and thus may require some preparation of your Nextcloud instance before you can install them.

Tip

AppAPI and ExApps are optional functionality. If you do not use any non-PHP applications then you may safely disable AppAPI by navigating to the Apps management page in your Nextcloud instance, going to "Your Apps" and clicking the "Disable" button next to AppAPI.

18.1.1 Installing AppAPI

All ExApps require the AppAPI Nextcloud app as a dependency. As of Nextcloud version 30.0.1, AppAPI is automatically installed by default. If AppAPI is not installed, you can still install it by simply navigating to the Apps management page in your Nextcloud instance and search for AppAPI from the Tools category.

18.1.2 Setup deploy daemon

A Deploy Daemon is the way for Nextcloud to install, communicate with, and control ExApps.

Note

If you are using Nextcloud AIO with the "HaRP" or "Docker Socket Proxy" container enabled, a Deploy Daemon will be automatically created and configured to work out-of-the-box. Otherwise, follow the steps below to set up a Deploy Daemon from the AppAPI admin settings.

Tip

After registering a Deploy Daemon, use the **Test Deploy** action to verify it is reachable and working. In the list of Deploy Daemons, click the … (three-dots) menu beside the daemon you want to verify and choose Test Deploy. For details on what this check does and how to interpret the results, see *Test Deploy*.

HaRP

This is the newer and the **recommended** way to install ExApps.
It requires changes in the proxy of your Nextcloud instance. If you don't have access to the proxy, you can use the usual method *described below*.

1. Setup a Docker container called HaRP that proxies access to Docker and to the ExApps for your Nextcloud instance. Be mindful of changing the values of `HP_SHARED_KEY` and `NC_INSTANCE_URL`.
2. Go to AppAPI admin settings.
3. Click on the "Register Daemon" button.
4. A filled form should appear. This default configuration `HaRP Proxy (Host)` should work for most setups. For Nextcloud AIO, use `HaRP All-in-One`.
   If you are using Nextcloud in a custom docker network and would want the HaRP container to be limited to it, use the `HaRP Proxy (Docker)` option to have the fields pre-filled with the common options or change them manually.
   Here, you should ensure the HaRP container itself is launched with the same network as your Nextcloud instance optionally with no ports exposed to the host in step 1, and the same docker network is mentioned in the `Network` field in the deploy config.
5. Ensure the same shared key is used in the HaRP container and in the AppAPI settings.
6. Click "Check connection" to verify that the configuration is correct.
7. Click "Register" to save the Deploy Daemon configuration.
8. Set up a location redirect in your Nextcloud's main proxy configuration to redirect requests to the HaRP container. Some examples for popular reverse proxies can be found in Configuring Your Reverse Proxy in the HaRP readme.
9. Test the whole setup with "Test deploy" in the 3-dots menu of the Deploy Daemon.

This is suitable for local setups where the Nextcloud server and the ExApps are on the same machine or in the same docker network. The ExApps in this configuration or the ExApp server need not expose any ExApp related port (23000-23999) necessarily to the host, nor do they need to be reachable from the host. They should be able to reach the HaRP container at the FRP port and the Nextcloud instance. For different/remote setups, see deployment configuration examples *here*.

> **Note**
>
> The existing ExApps can be migrated to use the new HaRP proxy following this guide.

Docker Socket Proxy

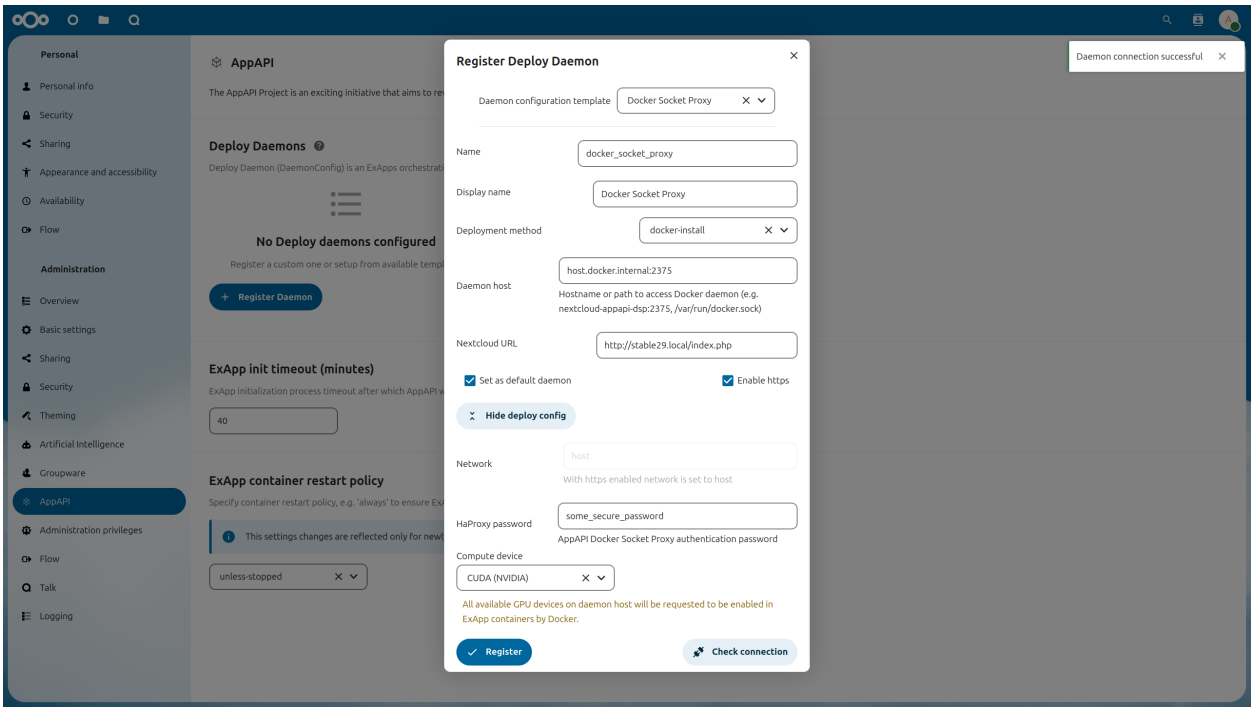
1. Setup a Docker container called docker-socket-proxy that proxies access to Docker for your Nextcloud instance.
2. Go to the AppAPI admin settings.
3. Click on the "Register Daemon" button.
4. **Fill in the required fields:**
   - `Name`: unique name of the Deploy daemon
   - `Display name`: the name that will be displayed in the UI
   - `Deployment method`: by default, you will need to choose `docker_install` (`manual_install` is for development or custom use case of manual ExApp installation)
   - `Daemon Host`: hostname/IP address + port of the Deploy daemon

- `Nextcloud URL`: autofilled with current domain, you might need to change the protocol to http/https depending on your setup
- `Set as default daemon`: check if you want set new Deploy daemon as default
- `Enable https`: check if your Deploy daemon (Docker Socket Proxy) is configured with TLS
- **Deploy Config:**
  - `Network`: Docker network name, depends on your networking setup, enforces to "host" if "Enable https" is checked
  - `HaProxy password`: password for Docker Socket Proxy, if it is configured with TLS
  - `Compute Device`: CPU, CUDA or ROCm, depending on your hardware config on Deploy daemon host machine
  - `Add additional option` (see *Additional options*): setup additional KEY + VALUE deploy config options

5. Click "Check connection" to verify that the configuration is correct.
6. Click "Register" to save the Deploy Daemon configuration.

Note

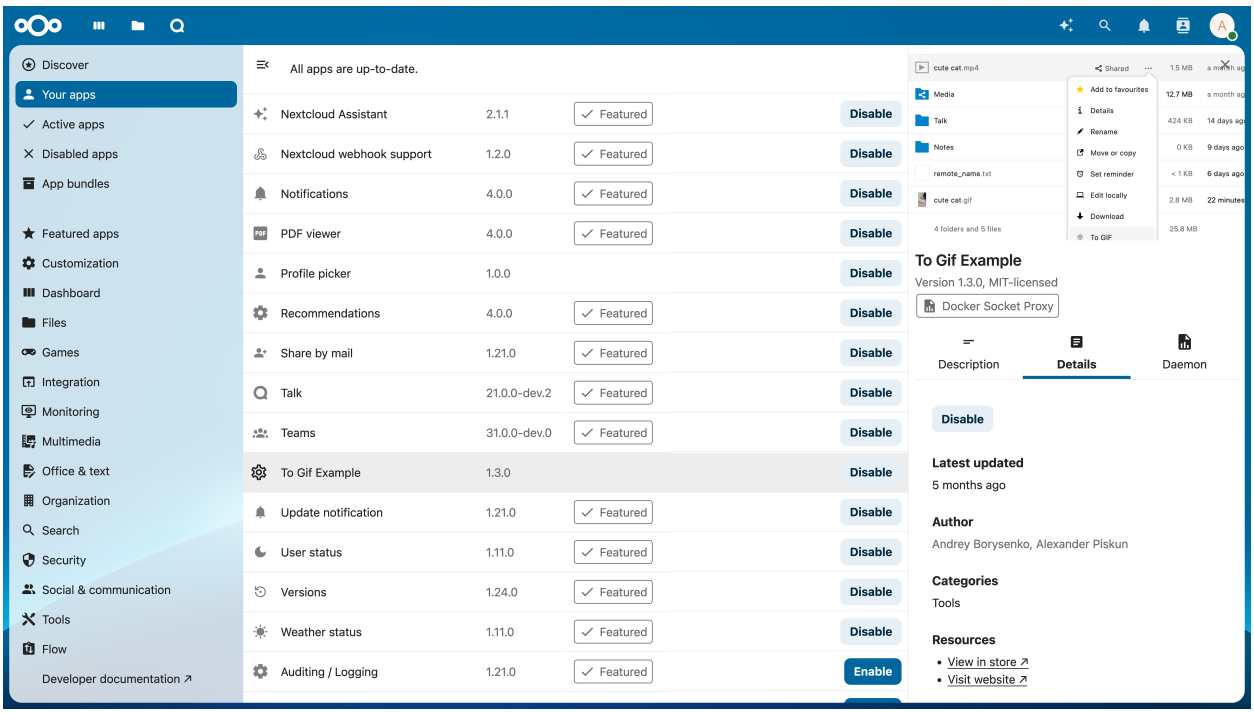
For remote DSP setup, it should expose the ports on the host.



Deployment configuration examples can be found *here*.

18.1.3 Installing ExApps

You can now install ExApps from the Nextcloud App Store by clicking “Install” on the respective app in the Apps page. If successful, the ExApp will be displayed under the “Your apps” list.



18.1.4 FAQ

- **I have two graphics cards (e.g. NVIDIA RTX 3060) with 8 GB of VRAM each. How can I run something which does not fit into one graphics card?**
  - Distributing models across multiple GPUs is currently not supported. You will need a GPU that fits all of the model you are trying to use.
- **I have a graphics card that does not support CUDA - can I use it and how?**
  - No, our AI apps require GPUs with CUDA support to function at this time.
- **What is the minimum VRAM size requirement for the GPU if I want to install multiple apps?**
  - When running multiple ExApps on the same GPU, the GPU must hold the largest model amongst the apps you install.
- **Is it possible to add more graphics cards for my instance to enable parallel requests or to speed up one request?**
  - Parallel processing of AI workloads for the same app with multiple GPUs is currently not supported.
- **Can I use the CPU and GPU in parallel for AI processing?**
  - No, you can only process AI workloads on either the CPU or GPU for one app. For different apps, you can decide whether to run them on CPU or GPU.
- **Can I run the same ex app container once for two different nextcloud instances at the same time?**
  - No, each ExApp container can only be used for one Nextcloud instance at a time. You would need to run separate containers and even separate docker hosts for each Nextcloud instance.

18.1.5 Docker Socket Proxy vs HaRP

HaRP can be seen as Docker Socket Proxy version 2.0. It does all what Docker Socket Proxy does, but also addresses the main pain point of ExApps not being reachable by the Nextcloud server (or AppAPI).

FRP is used to create a tunnel between the ExApp and the HaRP container so there is no need for the ExApp containers to expose any ports to the host or to be reachable from the Nextcloud server.

The Nextcloud server can reach the ExApp containers through the HaRP container.

HaRP has an additional benefit of being able to proxy requests coming from the Web interface or an API to the ExApp container without being proxied through the Nextcloud server, saving resources, improving performance and supporting additional protocols like WebSockets.

HaRP is the recommended way to run ExApps, but if you are not able to use it, Docker Socket Proxy is still supported.

Frontend requests in case of Docker Socket Proxy:

Frontend requests in case of HaRP:

18.2 Deployment configurations

Currently, two kinds of application deployments are supported:

- *Docker Deploy Daemon (Docker Socket Proxy)*
- *Docker Deploy Daemon (HaRP)*

18.2.1 Docker Deploy Daemon

Orchestrates the deployment of applications as Docker containers.

Warning

The administrator is responsible for the security actions taken to configure the Docker daemon connected to the Nextcloud instance.

These schemes are only examples of possible configurations.

For Docker Deploy Daemon (HaRP), AppAPI HaRP is required.

For Docker Deploy Daemon (Docker Socket Proxy), we recommend that you use the AppAPI Docker Socket Proxy or *AIO Docker Socket Proxy* container for Nextcloud AIO.

There are several Docker Daemon Deploy configurations (example schemes):

- Nextcloud and Docker on the **same host** (via socket, DockerSocketProxy, or HaRP)
- Nextcloud on the host and Docker on a **remote** host (via DockerSocketProxy with HTTPS, or HaRP)
- Nextcloud and **ExApps** in the **same Docker network** (via DockerSocketProxy, or HaRP)
- Nextcloud in AIO Docker and **ExApps** in the **same Docker network** (via AIO DockerSocketProxy)

18.2.2 Docker Deploy Daemon (HaRP)

With HaRP, the ExApps initiate the connection for tunneling to the Nextcloud instance and the HaRP container so there is no need to expose any ports or open any firewall rules.
See the diagrams of the respective configurations in the *Docker Deploy Daemon (Docker Socket Proxy)* section below.

A little introduction to the default ports of the HaRP container is given below. More about it can be found in the HaRP's readme.

- Port `8780` is the HTTP communication port used where Nextcloud connects to the HaRP container.
- Port `8781` is the HTTPS communication port when setup.
- Port `8782` is the FRP tunnel port used by ExApps to connect to the HaRP container.

In any of the cases, the following connections should succeed:

- Nextcloud -> HaRP container (on port 8780/8781)
- HaRP container -> Nextcloud (through proxy or directly as the NC_INSTANCE_URL env var dictates)
- ExApp -> HaRP container (on port 8782)
- ExApp -> Nextcloud (through proxy or directly as the `Nextcloud URL` in the daemon config dictates)

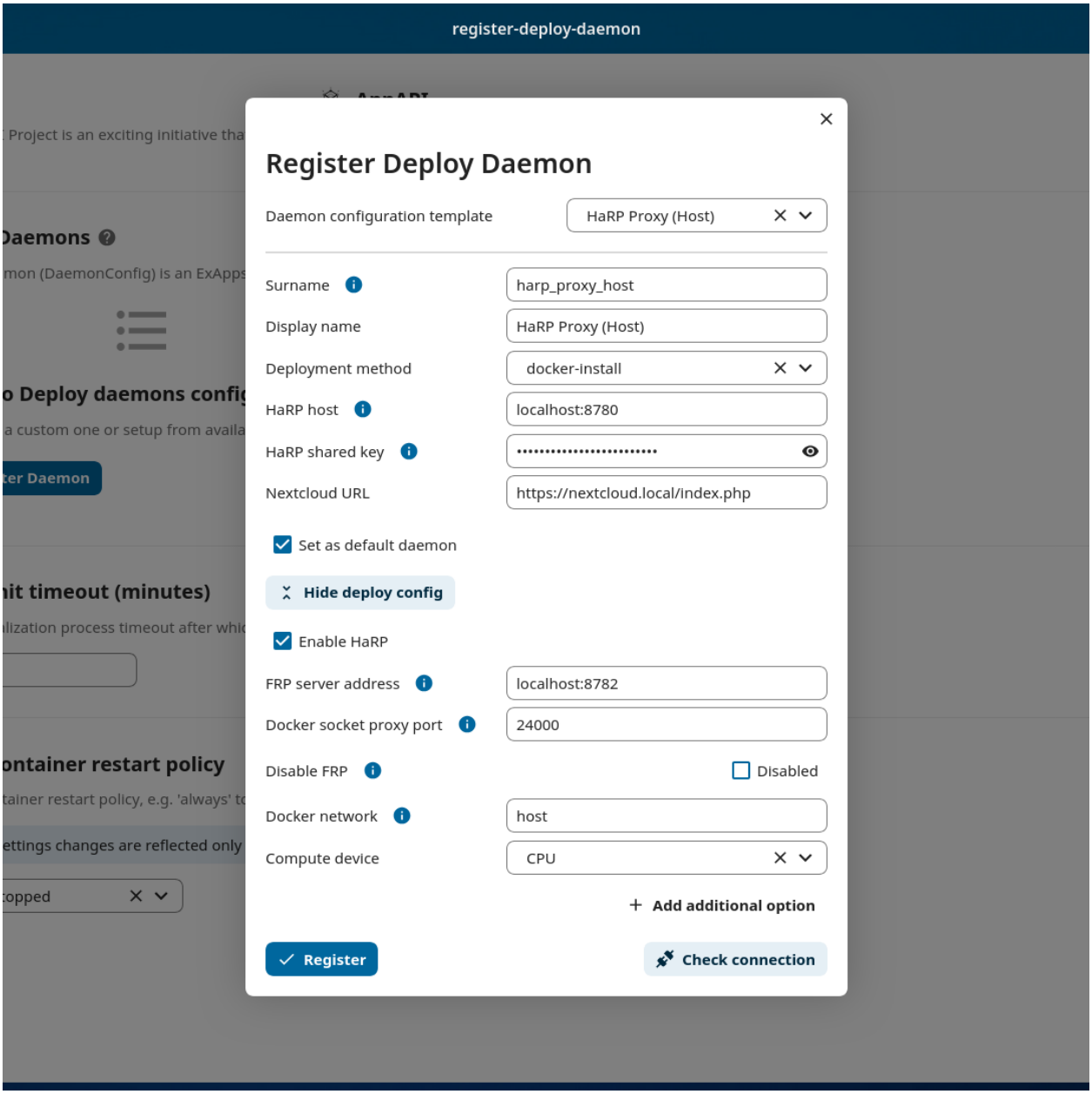
Nextcloud and Docker on the same host - with Nextcloud bare metal

The simplest configuration is when Nextcloud is installed on the host and docker is on the same host and applications are deployed to it.

Create a HaRP container with either `--network host` option or expose the ports `8780` and `8782` to the host.

```
docker run \
  -e HP_SHARED_KEY="some_very_secure_password" \
  -e NC_INSTANCE_URL="https://127.0.0.1:8080" \
  -v /var/run/docker.sock:/var/run/docker.sock \
  -v `pwd`/certs:/certs \
  --name appapi-harp -h appapi-harp \
  --restart unless-stopped \
  -p 8780:8780 \
  -p 8782:8782 \
  -d ghcr.io/nextcloud/nextcloud-appapi-harp:release
```

Go to AppAPI admin settings and register a `HaRP Proxy (Host)` daemon.



Finally, test the whole setup with "Test deploy" in the 3-dots menu of the deploy daemon.

Nextcloud and Docker on the same host - with Nextcloud in Docker

When Nextcloud is installed in Docker, the HaRP container can be created in the same docker network as the Nextcloud instance.

Create a HaRP container with `--network <nextcloud_docker_network_name>` option, where `<nextcloud_docker_network_name>` is the name of the Docker network in which Nextcloud is accessible.

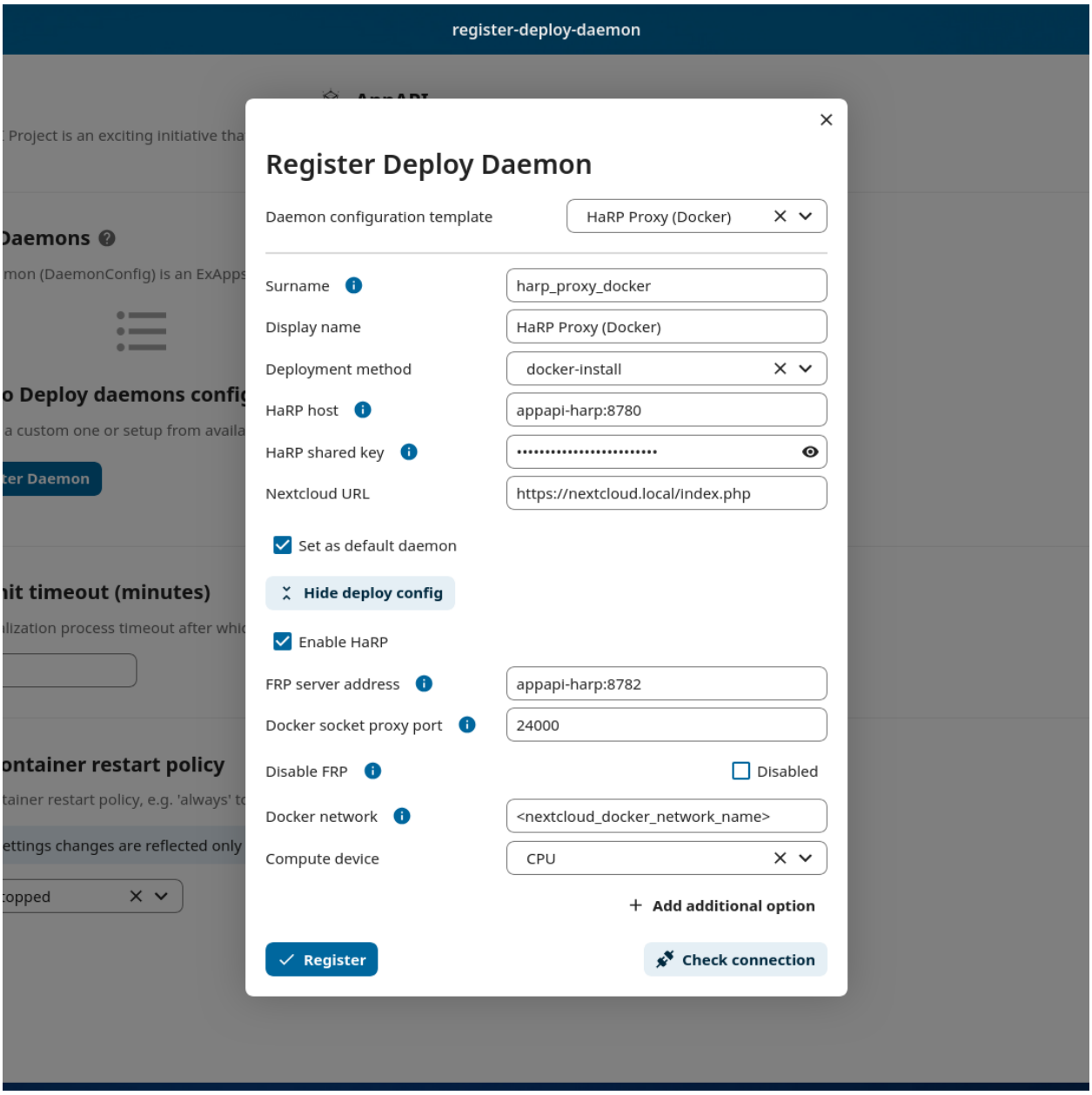
```
docker run \
  -e HP_SHARED_KEY="some_very_secure_password" \
  -e NC_INSTANCE_URL="https://nextcloud.tld" \
  -v /var/run/docker.sock:/var/run/docker.sock \
```

(continues on next page)

(continued from previous page)

```
  -v `pwd`/certs:/certs \
  --name appapi-harp -h appapi-harp \
  --restart unless-stopped \
  --net <nextcloud_docker_network_name> \
  -d ghcr.io/nextcloud/nextcloud-appapi-harp:release
```

Go to AppAPI admin settings and register a `HaRP Proxy (Docker)` daemon. Take note of the `<nextcloud_docker_network_name>` value in the `Docker network` field.



Finally, test the whole setup with "Test deploy" in the 3-dots menu of the deploy daemon.

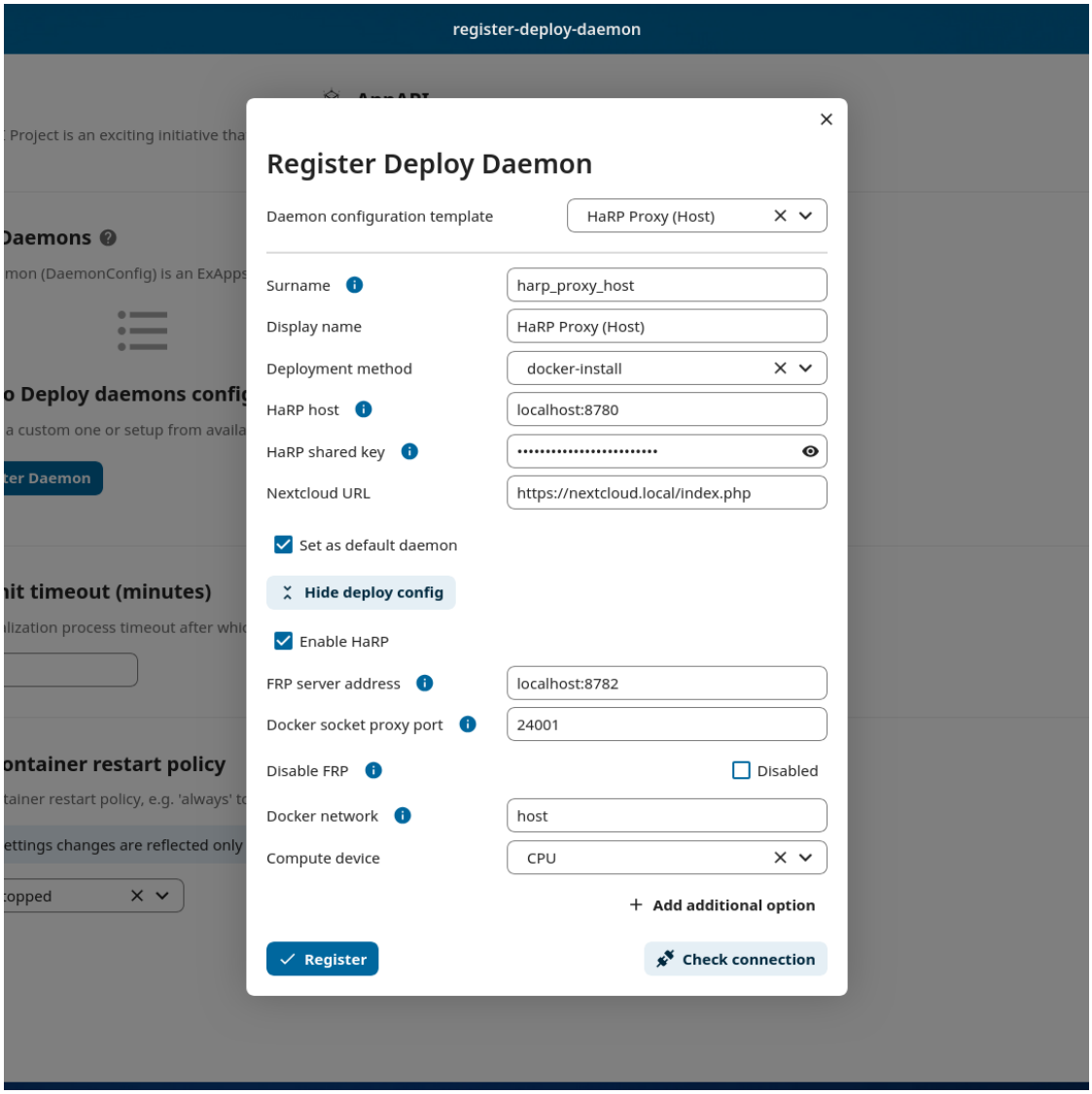
Docker on a remote host - with HaRP container on the local host

This configuration is suited for deployments that want to offload the heavy lifting of the ExApps to a remote host, especially when using GPUs as compute devices. There can be multiple deploy daemons that can be used to deploy ExApps on different remote hosts for different compute capabilities. Here the HaRP container is deployed on the local host and the remote host tunnels the remote host's docker socket to the local host over the FRP secure tunnel. The ExApps are deployed on the remote host. A setup with the HaRP container itself on the remote is not supported.

1. Create a HaRP container in the local host following *the above examples* but without the docker socket mount.

```
docker run \
  -e HP_SHARED_KEY="some_very_secure_password" \
  -e NC_INSTANCE_URL="https://127.0.0.1:8080" \
  -v `pwd`/certs:/certs \
  --name appapi-harp -h appapi-harp \
  --restart unless-stopped \
  -p 8780:8780 \
  -p 8782:8782 \
  -d ghcr.io/nextcloud/nextcloud-appapi-harp:release
```

2. Create a matching deploy daemon with `Docker socket proxy port` set to `24001`.



3. The FRP generated client certificates should be present in the `certs` folder locally. Copy the files `client.crt`, `client.key` and `ca.crt` inside the `certs` folder to the remote host.
4. Create a folder structure on the remote host: `mkdir -p certs/frp` and copy the files `client.crt`, `client.key` and `ca.crt` to the `certs/frp` folder.
5. Create a new file `frpc.toml` with the following contents.

```
# frpc.toml
serverAddr = "your.harp.server.address"          # Replace with your HP_
↪FRP_ADDRESS host
serverPort = 8782                                # Default port for FRP␣
↪or the port your reverse proxy listens on
loginFailExit = false                            # If the FRP (HaRP)␣
↪server is unavailable, continue trying to log in.

transport.tls.certFile = "certs/frp/client.crt"
transport.tls.keyFile = "certs/frp/client.key"
transport.tls.trustedCaFile = "certs/frp/ca.crt"
```

(continues on next page)

(continued from previous page)

```
transport.tls.serverName = "harp.nc"                # DO NOT CHANGE THIS␣
↪VALUE

metadatas.token = "some_very_secure_password"       # HP_SHARED_KEY in quotes

[[proxies]]
remotePort = 24001                                  # Unique remotePort for␣
↪each Docker Engine (range: 24001-24099)
name = "deploy-daemon-1"                            # Unique name for each␣
↪Docker Engine
type = "tcp"
[proxies.plugin]
type = "unix_domain_socket"
unixPath = "/var/run/docker.sock"
```

Make sure to replace the `your.harp.server.address` with the actual address of the local host where the HaRP container is running.

You might want to open the port `8782` on the local host firewall to allow the remote host to connect to it,

or use a reverse proxy to forward the requests to the HaRP container. An example with nginx is given below. Feel free to adjust the port you want to listen on. The FRP client will connect to this port exposed port.

With the reverse proxy config below, the whole setup would only need the main Nextcloud proxy to be exposed and reachable from the outside world, simplifying the network setup.

```
stream {
    server {
        listen 8782;  # Replace with the port you want to listen on
        proxy_pass 127.0.0.1:8782;
        proxy_protocol off;
        proxy_connect_timeout 10s;
        proxy_timeout 300s;
    }
}
```

6. Download a release of the FRP client from the official releases or our snapshot from here.
7. Extract and copy the `frpc` binary to an appropriate location on the remote host, e.g. `/usr/local/bin`.
8. Make it executable: `chmod +x /usr/local/bin/frpc`.
9. Start the FRP client with the command: `frpc -c /path/to/frpc.toml`.
10. Finally, test the whole setup with “Test deploy” in the 3-dots menu of the deploy daemon.

Docker / Reverse Proxy / Nextcloud on 3 independent hosts - with HaRP container

This is the related infrastructure

Please see below the steps I follow All of the following steps are based on a Almalinux Distro. Please customize for your distribution.

1. On the Host2 Docker

1.1. Creation of Cert folder (if necessary)

```
mkdir -p /some/path/certs
```

1.2. Open ports

```
firewall-cmd --permanent --zone=public --add-port=8780/tcp
firewall-cmd --permanent --zone=public --add-port=8782/tcp
firewall-cmd --reload
```

1.3. Deploy of the HaRP Container

```
docker run \
  -e HP_SHARED_KEY="some_very_secure_password" \
  -e NC_INSTANCE_URL="https://cloud.acme.com" \
  -e HP_TRUSTED_PROXY_IPS="192.168.0.0/24" \  # Replace with your actual␣
↪trusted proxy subnet (Host3's IP or subnet)
  -v /var/run/docker.sock:/var/run/docker.sock \
  -v /some/path/certs:/certs \
  -p 8780:8780 \
  -p 8782:8782 \
  --name appapi-harp -h appapi-harp \
  --restart unless-stopped \
  -d ghcr.io/nextcloud/nextcloud-appapi-harp:release
```

2. On the Host3 Apache Reverse Proxy - Reverse proxy redirections

On the virtual Host “cloud.acme.com” of the apache conf file Add the following lines (before the existing configuration)

```
#  AppAPI Configuration
ProxyPass /exapps/ http://<IP_host2_docker>:8780/exapps/
ProxyPassReverse /exapps/ http://<IP_host2_docker>:8780/exapps/
```

3. On the Nextcloud Web Interface - Daemon Register

Add the following configuration :

- Daemon configuration template : `HaRP Proxy (HOST)`
- Surname : `appapi-harp`
- Display name : `appapi-harp`
- Deployment method : `docker-install`
- HaRP host : `<IP_host2_docker>:8780`
- HaRP shared key : `some_very_secure_password`
- Nextcloud URL : `https://cloud.acme.com`
- FRP server address : `<IP_host2_docker>:8782`
- Docker network : `bridge`

Finally, test the whole setup with “Test deploy” in the 3-dots menu of the deploy daemon.

4. Additional tests from the network of your hosts

Note

The `docker-engine-port` header tells HaRP which internal virtual port to route to the Docker engine. When HaRP has the Docker socket mounted directly (as in this setup), it uses `24000` as the default virtual port for that local socket. This value does not require any additional firewall or port configuration.

```
curl -fsS \
  -H "harp-shared-key: some_very_secure_password" \
  -H "docker-engine-port: 24000" \
  http://<IP_host2_docker>:8780/exapps/app_api/v1.41/_ping
curl -fsS \
  -H "harp-shared-key: some_very_secure_password" \
  -H "docker-engine-port: 24000" \
  https://cloud.acme.com/exapps/app_api/v1.41/_ping
```

18.2.3 Docker Deploy Daemon (Docker Socket Proxy)

NC & Docker on the Same-Host

The simplest configuration is when Nextcloud is installed on the host and Docker is on the same host and applications are deployed to it.

Suggested config values(template *Custom default*):

1. Daemon host: `/var/run/docker.sock`
2. HTTPS checkbox: *not supported using docker socket*
3. Network: `host`
4. HaProxy password: **not supported using raw docker socket, should be empty**

—

Suggested way to communicate with Docker via Docker Socket Proxy container.

Suggested config values(template *Docker Socket Proxy*):

1. **Daemon host: `localhost:2375`**

   Choose A or B option:

   A. Docker Socket Proxy should be deployed with `network=host` and `BIND_ADDRESS=127.0.0.1`

   B. Docker Socket Proxy should be deployed with `network=bridge` and it's port should be published to host's 127.0.0.1(e.g. **-p 127.0.0.1:2375:2375**)

2. HTTPS checkbox: **disabled**
3. Network: `host`
4. HaProxy password: **should not be empty**

Warning

Be careful with option `A`, by default **Docker Socket Proxy** binds to `*` if `BIND_ADDRESS` is not specified during container creation. Check opened ports after finishing configuration.

Docker on a remote host

Distributed configuration occurs when Nextcloud is installed on one host and Docker is located on a remote host, resulting in the deployment of applications on the remote host.

Benefit: no performance impact on Nextcloud host.

In this case, the AppAPI uses a Docker Socket Proxy deployed on remote host to access docker socket and ExApps.

Suggested config values(template *Docker Socket Proxy*):

1. Daemon host: ADDRESS_OF_REMOTE_MACHINE (e.g. **server_name.com:2375**)
2. HTTPS checkbox: `enabled`
3. Network: `host`
4. HaProxy password: **should not be empty**

NC & ExApps in the same Docker

Applications are deployed in the same Docker where Nextcloud resides.

Suggested way to communicate with Docker: via `docker-socket-proxy`.

Suggested config values(template *Docker Socket Proxy*):

1. Daemon host: nextcloud-appapi-dsp:2375
2. HTTPS checkbox: `disabled`
3. Network: user defined network
4. HaProxy password: **should not be empty**

> **Note**
>
> Network **should not be the default docker's bridge** as it does not support DNS resolving by container names.
>
> This means that **Docker Socket Proxy**, **Nextcloud** and **ExApps** containers should all be in the same docker network, different from the default **bridge**.

Nextcloud in Docker AIO (all-in-one)

In the case of AppAPI in Docker AIO setup (installed in Nextcloud container).

> **Note**
>
> AIO Docker Socket Proxy container must be enabled.

AppAPI will automatically create the default DaemonConfig for AIO Docker Socket Proxy in order to use it as an orchestrator to create ExApp containers.

> **Note**
>
> Default DaemonConfig will be created only if the default DaemonConfig is not already registered.

Default AIO Deploy Daemon (Docker Socket Proxy)

Nextcloud AIO has a specifically created Docker Socket Proxy container to be used as the Deploy Daemon in AppAPI. It has fixed parameters:

- Name: `docker_aio`
- Display name: `AIO Docker Socket Proxy`
- Accepts Deploy ID: `docker-install`
- Protocol: `http`
- Host: `nextcloud-aio-docker-socket-proxy:2375`
- Compute device: `CPU`
- Network: `nextcloud-aio`
- Nextcloud URL (passed to ExApps): `https://$NC_DOMAIN`

Docker Socket Proxy security

AIO Docker Socket Proxy has strictly limited access to the Docker APIs described in HAProxy configuration.

18.2.4 Container log rotation

AppAPI deploys each ExApp as a container but does not set a logging configuration on it, so each ExApp container uses the logging settings of the deploy daemon that runs it. If those logs have no size limit they are never rotated and can grow until they fill the host disk. Configure rotation on the daemon, where it applies to every container the daemon runs.

Docker

Docker's default `json-file` driver sets no size limit. Add the following to the Docker daemon configuration file (`/etc/docker/daemon.json` on most Linux hosts) and restart the daemon:

```
{
    "log-driver": "json-file",
    "log-opts": {
        "max-size": "10m",
        "max-file": "3"
    }
}
```

```
sudo systemctl restart docker
```

Each container then keeps at most `max-file` log files of `max-size` each (here three files of 10 MB, so up to 30 MB per container). Adjust the values to suit your hosts.

Note

A container keeps the logging configuration it was created with, and disabling and enabling an ExApp only stops and starts the existing container. Remove and install the affected ExApps again (or update them) so their containers are recreated with the new settings.

Important

Keep the `json-file` or `journald` driver: AppAPI can only download container logs for these two drivers, so a different driver disables log downloads for ExApp containers.

If you deliberately use a different driver (`syslog`, `fluentd`, and so on) to forward logs elsewhere, keep your existing setup: rotation is then handled by that driver and the steps above do not apply.

Podman

When the deploy daemon is Podman, set a global default in `/etc/containers/containers.conf` (or the per-user file) and re-deploy the affected ExApps:

```
[containers]
log_driver = "json-file"
log_size_max = 10485760
```

Kubernetes

For the Kubernetes deploy daemon, container log rotation is handled by the kubelet, not by AppAPI. Set `containerLogMaxSize` and `containerLogMaxFiles` in the kubelet configuration on each node and restart the kubelet for the change to take effect (it loads its configuration only at startup); see the Kubernetes logging architecture documentation.

18.2.5 NC to ExApp Communication

Communications between Nextcloud and ExApps are done via the AppAPI. With Docker Socket Proxy, the requests are sent to the ExApp container directly. For HaRP, the communication goes through the main Nextcloud proxy and the HaRP container.

Each type of DeployDaemon necessarily implements the `resolveExAppUrl` function.

It has the prototype:

```
public function resolveExAppUrl(
    string $appId, string $protocol, string $host, array $deployConfig, int $port,␣
↪array &$auth
) {}
```

where:

- **protocol** is daemon protocol value
- **host** is daemon host value, *can be DNS:port or IP:PORT or even path to docker socket*.
- **port** is an integer with ExApp port
- **deployConfig** can be custom for each Daemon type
- **auth** is an optional array, with *Basic Authentication* data if needed to access ExApp

Note

Applies only to Docker Socket Proxy.

The optional additional parameter *OVERRIDE_APP_HOST* can be used to override the host that will be used for ExApp binding.

It can be `0.0.0.0` in some specific configurations, when VPN is used or both Nextcloud instance and ExApps are one the same physical machine but different virtual environments.

Also you can specify something like `10.10.2.5` and in this case `ExApp` will try to bind to that address and AppAPI will try to send request s directly to this address assuming that ExApp itself bound on it.

The simplest implementation is in the **Manual-Install** deploy type:

```
public function resolveExAppUrl(
    string $appId, string $protocol, string $host, array $deployConfig, int $port,␣
↪array &$auth
): string {
    if (boolval($deployConfig['harp'] ?? false)) {
        $url = rtrim($deployConfig['nextcloud_url'], '/');
        if (str_ends_with($url, '/index.php')) {
            $url = substr($url, 0, -10);
        }
        return sprintf('%s/exapps/%s', $url, $appId);
    }

    $auth = [];
    if (isset($deployConfig['additional_options']['OVERRIDE_APP_HOST']) &&
        $deployConfig['additional_options']['OVERRIDE_APP_HOST'] !== ''
    ) {
        $wideNetworkAddresses = ['0.0.0.0', '127.0.0.1', '::', '::1'];
        if (!in_array($deployConfig['additional_options']['OVERRIDE_APP_HOST'],
↪$wideNetworkAddresses)) {
            $host = $deployConfig['additional_options']['OVERRIDE_APP_HOST'];
        }
    }
    return sprintf('%s://%s:%s', $protocol, $host, $port);
}
```

Here we see that AppAPI sends requests to the **host**:**port** specified during daemon creation for manual-install without HaRP.
But it exclusively uses the `http(s)://nextcloud.example.tld/exapps/` route for manual deployments using the HaRP proxy. `http(s)://nextcloud.example.tld` is the Nextcloud URL specified in the daemon config. Take care to configure the `/exapps/` route in your reverse proxy accordingly if your Nextcloud instance is on a subpath `https://nextcloud.example.tld/nextcloud`. See Configuring Your Reverse Proxy in the HaRP readme for examples.

Now, let's take a look at the Docker Daemon implementation of `resolveExAppUrl`:

```
public function resolveExAppUrl(
    string $appId, string $protocol, string $host, array $deployConfig, int $port,␣
↪array &$auth
): string {
    if (boolval($deployConfig['harp'] ?? false)) {
        $url = rtrim($deployConfig['nextcloud_url'], '/');
        if (str_ends_with($url, '/index.php')) {
```

(continues on next page)

(continued from previous page)

```
            $url = substr($url, 0, -10);
        }
        return sprintf('%s/exapps/%s', $url, $appId);
    }

    $auth = [];
    if (isset($deployConfig['additional_options']['OVERRIDE_APP_HOST']) &&
        $deployConfig['additional_options']['OVERRIDE_APP_HOST'] !== ''
    ) {
        $wideNetworkAddresses = ['0.0.0.0', '127.0.0.1', '::', '::1'];
        if (!in_array($deployConfig['additional_options']['OVERRIDE_APP_HOST'],
↪$wideNetworkAddresses)) {
            return sprintf(
                '%s://%s:%s', $protocol, $deployConfig['additional_options'][
↪'OVERRIDE_APP_HOST'], $port
            );
        }
    }
    $host = explode(':', $host)[0];
    if ($protocol == 'https') {
        $exAppHost = $host;
    } elseif (isset($deployConfig['net']) && $deployConfig['net'] === 'host') {
        $exAppHost = 'localhost';
    } else {
        $exAppHost = $appId;
    }
    if ($protocol == 'https' && isset($deployConfig['haproxy_password']) &&
↪$deployConfig['haproxy_password'] !== '') {
        // we only set haproxy auth for remote installations, when all requests come␣
↪through HaProxy.
        $haproxyPass = $this->crypto->decrypt($deployConfig['haproxy_password']);
        $auth = [self::APP_API_HAPROXY_USER, $haproxyPass];
    }
    return sprintf('%s://%s:%s', $protocol, $exAppHost, $port);
}
```

The route for HaRP setups remain the same here as in the previous example. All the requests are sent to the Nextcloud URL with the `/exapps/` route.

For Docker Socket Proxy, however, we have much more complex algorithm of detecting to where requests should be send.

First of all, if the protocol is set to `https`, AppAPI always sends requests to the daemon host, and in this case, it is a HaProxy that will forward requests to ExApps that will be listening on `localhost`.

Briefly, it will look like this (*haproxy_host==daemon host value*):

NC -> *https* -> `haproxy_host:ex_app_port` -> *http* -> `localhost:ex_app_port`

When the protocol is not `https` but `http`, then what will be the endpoint where to send requests is determined by `$deployConfig['net']` value.

If `net` is defined and equal to `host`, then AppAPI assumes that ExApp is installed somewhere in the current host network and will be available on `localhost` loop-back adapter.

NC -> *http* -> `localhost:ex_app_port`

In all other cases, the ExApp should be available by it's name: e.g. when using docker **custom bridge** network all containers available by DNS.

NC –> *http* –> `app_container_name:ex_app_port`

These three different types of communication cover most popular configurations.

18.3 Managing Deploy Daemons

18.3.1 OCC CLI

There are a few OCC CLI commands to manage Deploy Daemons:

1. Register `occ app_api:daemon:register`
2. Unregister `occ app_api:daemon:unregister`
3. List registered daemons `occ app_api:daemon:list`

Register

Register Deploy Daemon (DaemonConfig).

Command: `app_api:daemon:register [--net NET] [--haproxy_password HAPROXY_PASSWORD] [--compute_device COMPUTE_DEVICE] [--set-default] [--harp] [--harp_frp_address HARP_FRP_ADDRESS] [--harp_shared_key HARP_SHARED_KEY] [--harp_docker_socket_port HARP_DOCKER_SOCKET_PORT] [--harp_exapp_direct] [--] <name> <display-name> <accepts-deploy-id> <protocol> <host> <nextcloud_url>`

Arguments

- `name` - unique name of the daemon (e.g. `docker_local_sock`)
- `display-name` - name of the daemon (e.g. `My Local Docker`, will be displayed in the UI)
- `accepts-deploy-id` - type of deployment (`docker-install` or `manual-install`)
- `host` - **path to docker-socket** or the Docker Socket Proxy: `address:port`
- `protocol` - protocol used to communicate with the Daemon/ExApps (`http` or `https`)
- `nextcloud_url` - Nextcloud URL, Daemon config required option (e.g. `https://nextcloud.local`)

Options

- `--net [network-name]` - `[required]` network name to bind docker container to (default: `host`)
- `--haproxy_password HAPROXY_PASSWORD` - `[optional]` password for AppAPI Docker Socket Proxy
- `--compute_device GPU` - `[optional]` GPU device to expose to the daemon (e.g. `cpu|cuda|rocm`, default: `cpu`)
- `--set-default` - `[optional]` set created daemon as default for ExApps installation
- `--harp` - `[optional]` Flag to set daemon to use HaRP for all docker and exapp communication
- `--harp_frp_address` - `[optional]` [host]:[port] of the HaRP FRP server, default host is same as HaRP host and port is 8782
- `--harp_shared_key` - `[optional]` HaRP shared key for secure communication between HaRP and AppAPI

- `--harp_docker_socket_port - [optional]` 'remotePort' of the FRP client of the remote docker socket proxy. There is one included in the harp container so this can be skipped for default setups. (default: "24000")
- `--harp_exapp_direct - [optional]` Flag for the advanced setups only. Disables the FRP tunnel between ExApps and HaRP.

Usage Examples

- Register a HaRP deploy daemon within the `nextcloud` docker network, with the `appapi-harp` container as the host and the `appapi-harp:8782` as the FRP server address. This can be paired with a HaRP container running in the same network.

```
occ app_api:daemon:register harp_proxy_docker "Harp Proxy (Docker)"
↪"docker-install" "http" "appapi-harp:8780" "http://nextcloud.local" --
↪net nextcloud --harp --harp_frp_address "appapi-harp:8782" --harp_
↪shared_key "some_very_secure_password" --set-default --compute_
↪device=cuda
```

- Register a HaRP deploy daemon with the `localhost` as the host and the `localhost:8782` as the FRP server address. This can be paired with a HaRP container running in the host network mode or that has exposed the ports `8780` and `8782` to the host.

```
app_api:daemon:register harp_proxy_host "Harp Proxy (Host)" "docker-
↪install" "http" "localhost:8780" "http://nextcloud.local" --harp --harp_
↪frp_address "localhost:8782" --harp_shared_key "some_very_secure_
↪password" --set-default --compute_device=cuda
```

- Register a manual install deploy daemon with HaRP support. This can be paired with a HaRP container running in the same network. The HaRP container need not have access to a docker socket or any other ports exposed to the host. It will not create docker containers of the ExApps but will only proxy the requests to the ExApp process manually launched by the user.

> **Note**
>
> The ExApp process should have a FRP Client (frpc) running in the same network as the HaRP container or should be able to connect to the ports exposed by the HaRP container.
> If the communication has to go without the FRP client, the `--harp_exapp_direct` flag should be provided. The localhost IP address is always used as the host in this case for manual deployments and `OVERRIDE_APP_HOST` or the `<app_id>` is used for ExApp deployments. Take care not to use the host network mode or the default bridge network for this.

```
app_api:daemon:register manual_install_harp "Harp Manual Install" "manual-
↪install" "http" "appapi-harp:8780" "http://nextcloud.local" --net␣
↪nextcloud --harp --harp_frp_address "appapi-harp:8782" --harp_shared_
↪key "some_very_secure_password"
```

- Register a Docker Socket Proxy deploy daemon with the `nextcloud-appapi-dsp:2375` as the host and the `nextcloud` docker network. This can be paired with a Docker Socket Proxy container running in the same network with the default port `2375`.

```
app_api:daemon:register docker_install "Docker Socket Proxy" "docker-
↪install" "http" "nextcloud-appapi-dsp:2375" "http://nextcloud.local" --
↪net=nextcloud --set-default --compute_device=cuda
```

- Register a manual deploy daemon with `host.docker.internal` as the host used to connect to the ExApps.

```
app_api:daemon:register manual_install "Manual Install" "manual-install"
↪"http" null "http://nextcloud.local"
```

- Register a local docker deploy daemon with the `/var/run/docker.sock` as the socket and the host, and the `nextcloud` docker network. This does not need a Docker Socket Proxy container. The compute device used by this daemon is `CPU`.

```
app_api:daemon:register local_docker "Docker Local" "docker-install" "http
↪" "/var/run/docker.sock" "http://nextcloud.local" --net=nextcloud
```

- Register a local docker deploy daemon with the `/var/run/docker.sock` as the socket and the host, and the `nextcloud` docker network. This does not need a Docker Socket Proxy container. The compute device used by this daemon is `CUDA` (NVIDIA).

```
app_api:daemon:register local_docker "Docker Local" "docker-install" "http
↪" "/var/run/docker.sock" "http://nextcloud.local" --net=nextcloud --set-
↪default --compute_device=cuda
```

DeployConfig

DeployConfig is a set of additional options in Daemon config, which are used in deployment algorithms to configure ExApp container.

```
{
    "net": "host",
    "nextcloud_url": "https://nextcloud.local",
    "haproxy_password": "some_secure_password",
    "computeDevice": {
        "id": "cuda",
        "name": "CUDA (NVIDIA)",
    },
    "harp": {
        "frp_address": "localhost:8782",
        "docker_socket_port": "24000",
        "exapp_direct": false
    }
}
```

DeployConfig options

- `net` **[required]** - network name to bind docker container to (default: `host`)
- `nextcloud_url` **[required]** - Nextcloud URL (e.g. `https://nextcloud.local`)
- `haproxy_password` *[optional]* - password for AppAPI Docker Socket Proxy
- `computeDevice` *[optional]* - Compute device to attach to the daemon (e.g. `{ "id": "cuda", "label": "CUDA (NVIDIA)" }`)

- **`harp` *[optional]* - HaRP options, can be `null` in case of non-HaRP setups**
  - `frp_address` *[optional]* - [host]:[port] of the HaRP FRP server, default host is same as HaRP host and port is 8782
  - `docker_socket_port` *[optional]* - 'remotePort' of the FRP client of the remote docker socket proxy. There is one included in the harp container so this can be skipped for default setups. [default: "24000"]
  - `exapp_direct` *[optional]* - Flag for the advanced setups only. Disables the FRP tunnel between ExApps and HaRP.

Unregister

Unregister Deploy Daemon (DaemonConfig).

Command: `app_api:daemon:unregister <daemon-config-name>`

List registered daemons

List registered Deploy Daemons (DaemonConfigs).

Command: `app_api:daemon:list`

18.3.2 Nextcloud AIO

In the case of AppAPI installed in AIO, a default Deploy Daemon is registered automatically. It is possible to register additional Deploy Daemons using the same methods as described above.

18.3.3 Additional options

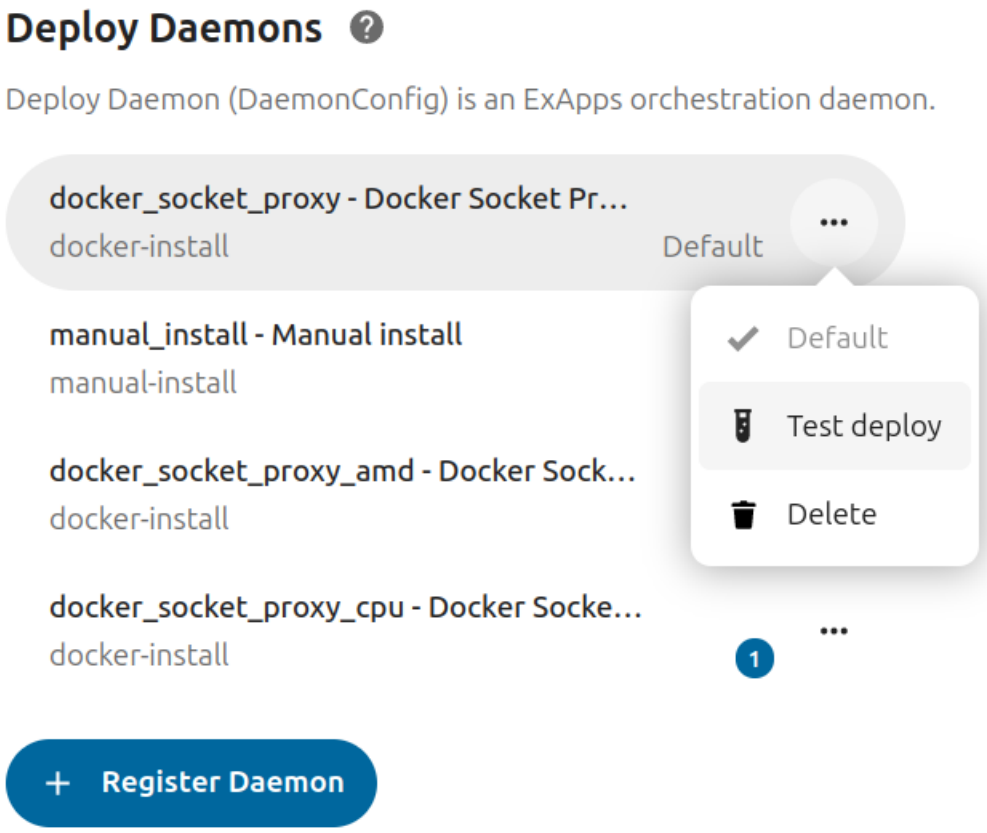
There is a possibility to add additional options to the Deploy Daemon configuration, which are key-value pairs.
This should not be used for HaRP.

Currently, the following options are available:

- `OVERRIDE_APP_HOST` - can be used to override the host that will be used for ExApp binding (not passed to ExApp container envs)

18.4 Test Deploy Daemon

You can test each Daemon configuration deployment from the AppAPI Admin settings.

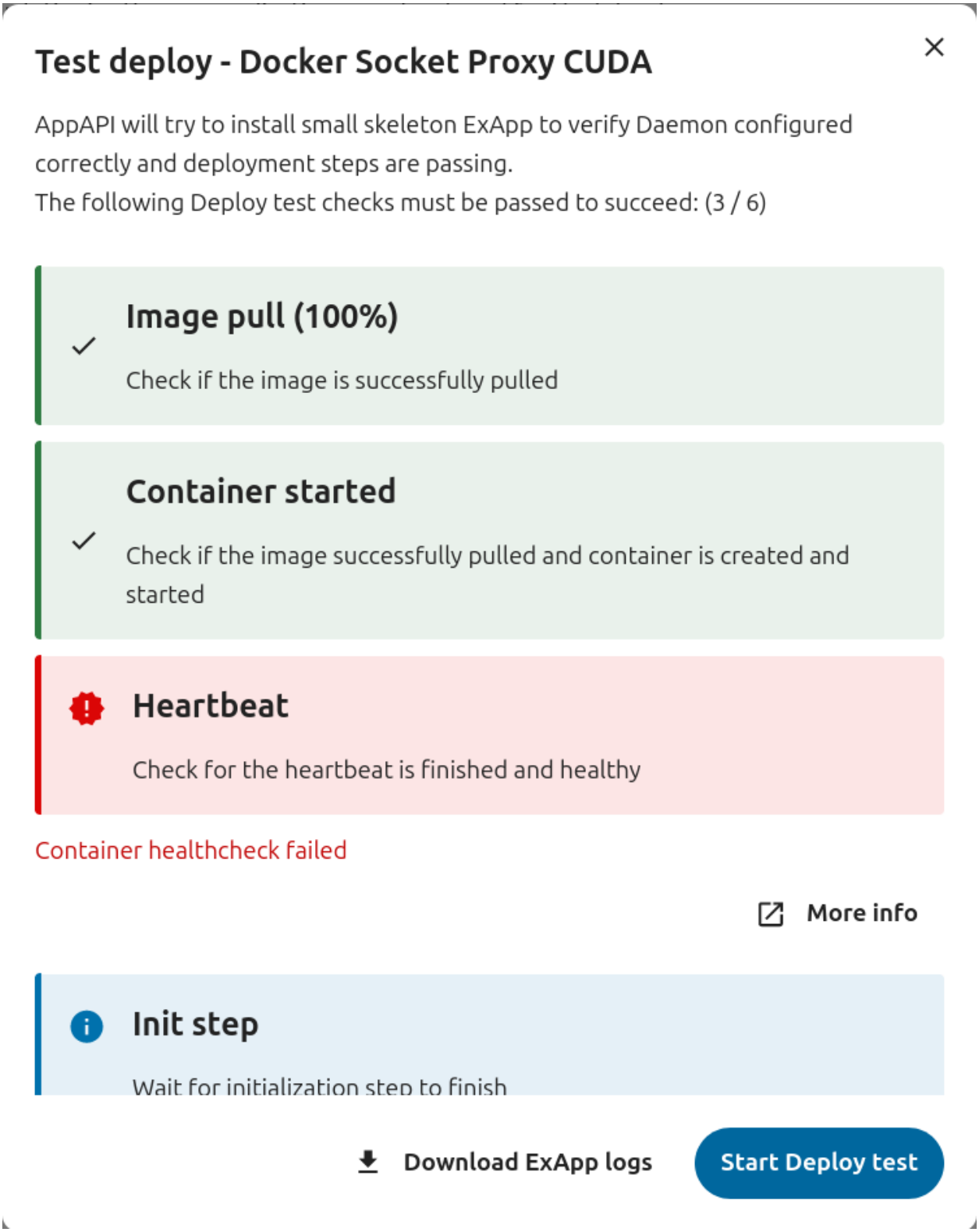


18.4.1 Status Checks

The Deploy test installs a test-deploy ExApp to verify each step of the deployment process, including a hardware support check - for each compute device, there is a separate Docker image.

Note

The Test Deploy ExApp container is not removed after the test as it's needed for logs and status checks. You can remove it after testing from the Apps page. The Docker images are also not removed from the Daemon; you can clean up unused images with the `docker image prune` command.


Test deploy - Docker Socket Proxy CUDA
AppAPI will try to install small skeleton ExApp to verify Daemon configured correctly and deployment steps are passing.
The following Deploy test checks must be passed to succeed: (3 / 6)
Image pull (100%)
Check if the image is successfully pulled
Container started
Check if the image successfully pulled and container is created and started
Heartbeat
Check for the heartbeat is finished and healthy
Container healthcheck failed
More info
Init step
Wait for initialization step to finish
Download ExApp logs
Start Deploy test

Register

The Register step is the first step; it checks if the ExApp is registered in Nextcloud.

Image Pull

The Image Pull step downloads the ExApp Docker image.

Possible errors:

- Image not found (e.g. not public, no image found for your hardware architecture)
- Image pull failed (e.g., due to network issues)
- Image pull timeout
- Your Docker Socket Proxy/HaRP is not configured correctly and blocks access to this Docker Engine API

See `journalctl -f -u docker.service` for more details in systemd based systems.

Container Started

The Container Started step verifies that the ExApp container is created and started successfully.

Possible errors:

- **Container failed to start with GPU support (may be missing or misconfigured)**
  - For NVIDIA, refer to the NVIDIA Docker configuration docs.
  - For AMD, refer to the ROCm Docker configuration docs.
- The ExApp issue during startup (e.g. not enough memory). The app would show repeated starting attempts in the logs.

See `docker logs nc_app_<app_id>` for more details.

Heartbeat

The Heartbeat step checks if the container's health check is finished and the container is healthy. The ExApp might have additional pre-configuration logic during this step.

Possible errors:

- ExApp failed to start a web server, e.g., if the port is already in use (this should be visible in the container logs)
- ExApp heartbeat_count keeps increasing, this may indicate that the ExApp couldn't start properly
- **Nextcloud can not reach the ExApp container, e.g.,**
  - due to a network issue or a firewall (this should be visible in the server logs or the firewall logs)
  - due to a "http" protocol deploy daemon. In this case, the ExApp's container listens on localhost (127.0.0.1 or ::1) which might not be reachable from the Nextcloud server and you might want to listen on a different IP address. See `OVERRIDE_APP_HOST` in *Additional options* in the Deploy Daemon form. This issue can be identified using this command: `lsof -i -P -n | grep LISTEN`
- For HaRP, the main Nextcloud proxy might not be configured to redirect requests to the HaRP container correctly. See the Configuring Your Reverse Proxy section in the HaRP readme.

Init

The Init step checks if the ExApp is initialized and ready to use. During the init step, the ExApp may perform downloads of extra stuff required for it.

Possible errors:

- Initialization failed (e.g., due to network issues or timeout)
- ExApp not being able to reach the Nextcloud server (e.g., due to a network issue or a firewall)

Enabled

The Enabled step checks if the ExApp is enabled and ready to use. During this step, the ExApp registers all the required and available APIs of the Nextcloud AppFramework.

Possible errors:

- ExApp did not respond to the enable request
- ExApp failed to enable due to a failure in registering AppAPI Nextcloud AppFramework APIs (this should be visible both in the container logs and in the Nextcloud logs if there are any errors)

18.4.2 Download Logs

You can download the logs of the last test deploy attempt container.

> **Note**
>
> Downloading Docker container logs is only possible for containers using the json-file or journald logging drivers.

18.5 Managing ExApps

Managing ExApps can be done from App Management UI as with other Nextcloud Apps, but you can also use the AppAPI commands in the OCC CLI tool.

There are several commands to work with ExApps:

1. Register
2. Unregister
3. Update
4. Enable
5. Disable
6. List ExApps

18.5.1 Register

Command: `app_api:app:register [--force-scopes] [--info-xml INFO-XML] [--json-info JSON-INFO] [--wait-finish] [--silent] [--test-deploy-mode] [--env [ENV]] [--mount [MOUNT]] [--] <appid> [<daemon-config-name>]`

The register command is the first ExApp installation step.

Arguments

- `appid` - unique name of the ExApp (e.g. `app_python_skeleton`, must be the same as in deployed container)
- `daemon-config-name` - unique name of the daemon (e.g. `docker_local_sock`)

Options

- `--json-info JSON-INFO` *[optional]* - ExApp deploy JSON info (json string)
- `--info-xml INFO-XML` *[optional]* - path to info.xml file (url or local absolute path)
- `--wait-finish` *[optional]* - wait until initialization finished
- `--silent` *[optional]* - do not print to console
- `--test-deploy-mode` *[optional]* - test deploy mode with additional status checks and slightly different logic

Advanced Deploy Options

- `--env` *[optional]* - environment (ENV_NAME=ENV_VALUE), passed to ExApp container as environment variables (multiple values allowed)
- `--mount` *[optional]* - mount options (SRC_PATH=DST_PATH), passed to ExApp container as volume mounts (multiple values allowed)

18.5.2 Unregister

Command: `app_api:app:unregister [--rm-data] [--force] [--silent] [--] <appid>`

To remove an ExApp, you can use the unregister command. By default, this command does *not* delete the ExApp's persistent storage (data volume) to avoid accidental removal of any user data.

Arguments

- `appid` - unique name of the ExApp (e.g. `app_python_skeleton`, must be the same as in deployed container)

Options

- `--rm-data` *[optional]* - remove ExApp persistent storage (data volume)
- `--force` *[optional]* - continue removal even if some error occurs
- `--silent` *[optional]* - print a minimum of information, display only some errors, if any

18.5.3 Update

Command: `app_api:app:update [--info-xml INFO-XML] [--force-update] [-e|--enabled] [--] <appid>`

ExApp will be updated if there is a new version available.

Arguments

- `appid` - unique name of the ExApp (e.g. `app_python_skeleton`, must be the same as in deployed container)

Options

- `--info-xml INFO-XML` *[optional]* - path to info.xml file (url or local absolute path)
- `-e|--enabled` *[optional]* - enable ExApp after update

18.5.4 Enable

Command: `app_api:app:enable <appid>`

18.5.5 Disable

Command: `app_api:app:disable <appid>`

18.5.6 List ExApps

Command: `app_api:app:list`

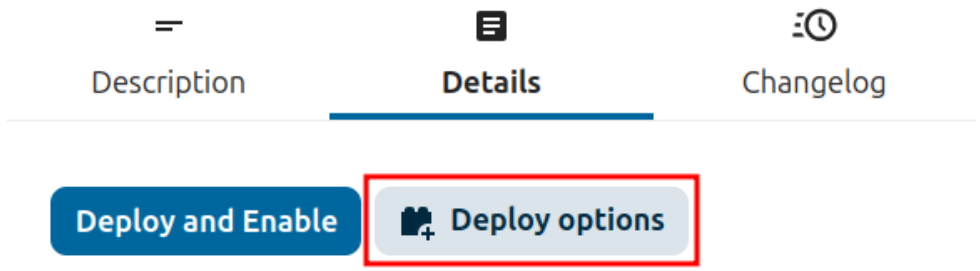
ListExApps command will show all ExApps:

```
ExApps:
appid (Display Name): version [enabled/disabled]
to_gif_example (To Gif Example): 1.0.0 [enabled]
upscaler_example (Upscaler Example): 1.0.0 [enabled]
```

18.6 Advanced Deploy Options

AppAPI allows optionally to configure environment variables and mounts for the ExApp container.

It is available via "Deploy options" modal next to "Deploy and Enable" button in the sidebar of the ExApp page on the Apps management page:

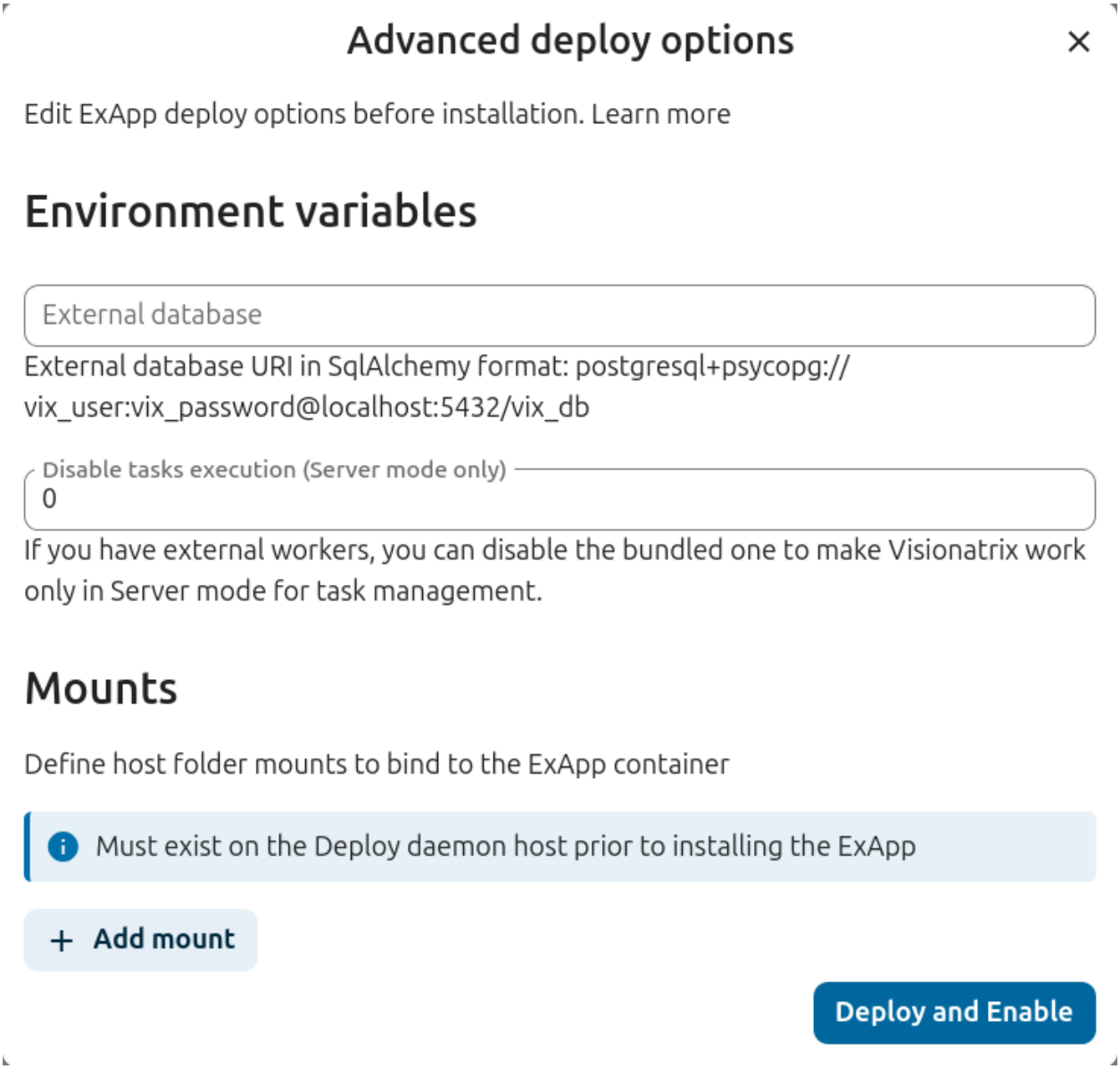


Or via CLI (*Advanced Deploy Options*).

18.6.1 Environment Variables

Environment variables enable more precise configuration of the ExApp. ExApp developers can define the list of supported environment variables with descriptions, only these variables will be available for configuration.

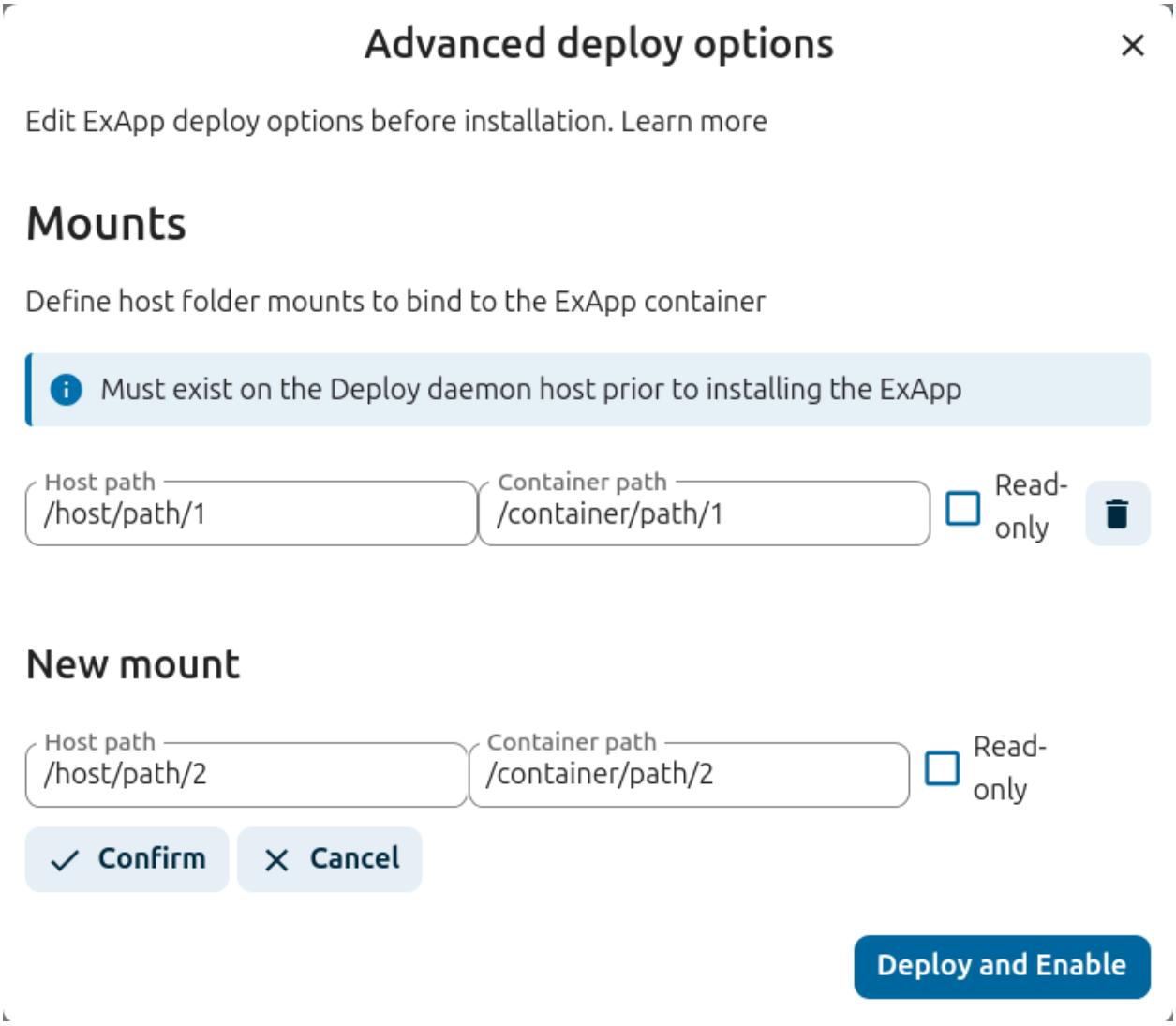
By default there are only mounts available for configuration.



When ExApp installed the list of set environment variables will be displayed.

18.6.2 Mounts

Mounts can be used to provide additional data to the ExApp container from the host. For example, it will be useful for some apps to provide a folder with SSL certs of your cloud, so the app can handle HTTPS correctly without any additional re-installation of the ExApp.


Advanced deploy options
Edit ExApp deploy options before installation. Learn more
Mounts
Define host folder mounts to bind to the ExApp container
Must exist on the Deploy daemon host prior to installing the ExApp
Host path
/host/path/1
Container path
/container/path/1
Read-only
New mount
Host path
/host/path/2
Container path
/container/path/2
Read-only
Confirm
Cancel
Deploy and Enable

CHAPTER

NINETEEN

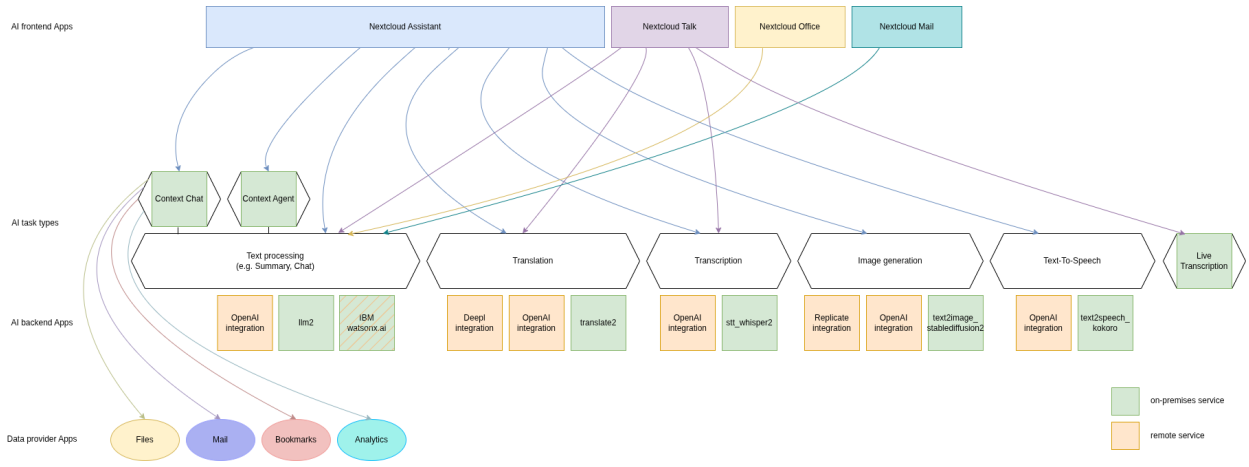
ARTIFICIAL INTELLIGENCE

19.1 Overview

We strive to bring Artificial Intelligence features to Nextcloud. This section highlights these features, how they work and where to find them. All of these features are completely optional. If you want to have them on your server, you need to install them via separate Nextcloud Apps.

19.1.1 Overview of AI features

Nextcloud uses modularity to separate raw AI functionality from the Graphical User interfaces and apps that make use of said functionality. Each instance can thus make use of various backends that provide the functionality for the same frontends and the same functionality can be implemented by multiple apps using on-premises processing or third-party AI service providers.



Feature	App	Rating	Open source
Smart inbox	Mail	Green	Yes
Image object recognition	Recognize	Green	Yes
Image face recognition	Recognize	Green	Yes
Video action recognition	Recognize	Green	Yes
Audio music genre recognition	Recognize	Green	Yes
Suspicious login detection	Suspicious Login	Green	Yes
Related resources	Related Resources	Green	Yes
Recommended files	recommended_files	Green	Yes

Table 1 – continued fro

Feature	App	Rating	Open source
Text processing using LLMs	Local large language model 2 (ExApp) (ExApp)	Green	Yes
	OpenAI and LocalAI integration (via OpenAI API)	Red	No
	OpenAI and LocalAI integration (via LocalAI)	Yellow	Yes
	OpenAI and LocalAI integration (via Ollama)	Yellow	Yes
	OpenAI and LocalAI integration (via IONOS AI Model Hub)	Orange	No
	OpenAI and LocalAI integration (via Plusserver)	Orange	No
	OpenAI and LocalAI integration (via Groqcloud)	Orange	No
	OpenAI and LocalAI integration (via MistralAI)	Orange	No
	IBM watsonx.ai integration (via IBM watsonx.ai as a Service)	Orange	No
	IBM watsonx.ai integration (via IBM watsonx.ai software)	Orange	No
Machine translation	Local Machine Translation 2 (ExApp)	Green	Yes
	DeepL integration	Red	No
	OpenAI and LocalAI integration (via OpenAI API)	Red	No
	OpenAI and LocalAI integration (via LocalAI)	Green	Yes
	OpenAI and LocalAI integration (via Ollama)	Yellow	Yes
	OpenAI and LocalAI integration (via IONOS AI Model Hub)	Orange	No
	OpenAI and LocalAI integration (via Plusserver)	Orange	No
	OpenAI and LocalAI integration (via Groqcloud)	Orange	No
	OpenAI and LocalAI integration (via MistralAI)	Orange	No
Speech to Text	Local Whisper Speech-To-Text 2 (ExApp)	Yellow	Yes
	OpenAI and LocalAI integration	Yellow	Yes
	OpenAI and LocalAI integration (via LocalAI)	Green	Yes
	OpenAI and LocalAI integration (via Ollama)	Yellow	Yes
	OpenAI and LocalAI integration (via IONOS AI Model Hub)	Orange	No
	OpenAI and LocalAI integration (via Plusserver)	Orange	No
	OpenAI and LocalAI integration (via Groqcloud)	Orange	No
	OpenAI and LocalAI integration (via MistralAI)	Orange	No
	Replicate integration	Yellow	Yes
Image generation	Local Stable Diffusion 2 (ExApp)	Yellow	Yes
	Local Image Generation: Flux (ExApp)	Yellow	Yes
	Replicate integration	Yellow	Yes
	OpenAI and LocalAI integration (via OpenAI API)	Red	No
	OpenAI and LocalAI integration (via LocalAI)	Green	Yes
	OpenAI and LocalAI integration (via Ollama)	Yellow	Yes
	OpenAI and LocalAI integration (via IONOS AI Model Hub)	Orange	No
	OpenAI and LocalAI integration (via Plusserver)	Orange	No
	OpenAI and LocalAI integration (via Groqcloud)	Orange	No
	OpenAI and LocalAI integration (via MistralAI)	Orange	No
Context Chat	Nextcloud Assistant Context Chat	Yellow	Yes
	Nextcloud Assistant Context Chat (Backend)	Yellow	Yes
Context Chat Search	Nextcloud Assistant Context Chat	Yellow	Yes
Context Agent	Nextcloud Context Agent (ExApp)	Green	Yes
Text To Speech	Open AI Text To Speech	Red	No
	Local Text To Speech (ExApp)	Yellow	Yes
Document generation	Nextcloud Office	Green	Yes
Live Transcription	Local Live Transcription	Yellow	Yes
Optical Character Recognition	Local OCR: PaddleOCR	Yellow	Yes
	OpenAI and LocalAI integration >= v5.0.0 (via OpenAI API)	Red	No
	OpenAI and LocalAI integration >= v5.0.0 (via LocalAI)	Green	Yes
	OpenAI and LocalAI integration >= v5.0.0 (via Ollama)	Yellow	Yes

Table 1 – continued fro

Feature	App	Rating	Open source
	OpenAI and LocalAI integration >= v5.0.0 (via IONOS AI Model Hub)	Orange	No
	OpenAI and LocalAI integration >= v5.0.0 (via Plusserver)	Orange	No
	OpenAI and LocalAI integration >= v5.0.0 (via Groqcloud)	Orange	No
	OpenAI and LocalAI integration >= v5.0.0 (via MistralAI)	Orange	No

19.1.2 Ethical AI Rating

Until Hub 3, we succeeded in offering features without relying on proprietary blobs or third party services. Yet, while there is a large community developing ethical, safe and privacy-respecting technologies, there are many other relevant technologies users might want to use. We want to provide users with these cutting-edge technologies – but also be transparent. For some use cases, ChatGPT might be a reasonable solution, while for more private, professional or sensitive data, it is paramount to have a local, on-prem, open solution. To differentiate these, we developed an Ethical AI Rating.

The rating has four levels:

- Red
- Orange
- Yellow
- Green

It is based on points from these factors:

- Is the software (both for inferencing and training) under a free and open source license?
- Is the trained model freely available for self-hosting?
- Is the training data available and free to use?

If all of these points are met, we give a Green label. If none are met, it is Red. If 1 condition is met, it is Orange and if 2 conditions are met, Yellow.

19.1.3 Features used by other apps

Some of our AI features are realized as generic APIs that any app can use and any app can provide an implementation for by registering a provider. So far, these are Machine translation, Speech-To-Text, Text-To-Speech, Image generation, Text processing and Context Chat.

Text processing

As you can see in the table above we have multiple apps offering text processing using Large language models. In downstream apps like Context Chat and assistant, users can use the text processing functionality regardless of which app implements it behind the scenes.

Frontend apps

- Assistant for offering a graphical UI for the various tasks, a smart picker and "Chat with AI" functionality
- Mail for summarizing mail threads (see *the Nextcloud Mail docs* for how to enable this). For smart replies. To active it - Administration settings > Groupware > Mail app > Enable text processing through LLMs
- Summary Bot for summarizing chat histories in Talk
- Talk for summarizing chat history (see Nextcloud Talk docs for how to enable this)
- Text for offering an inline graphical UI for the various tasks

- Collectives integrating Assistant through the smart picker to offer a graphical UI for the various tasks
- Notes integrating Assistant through the smart picker to offer a graphical UI for the various tasks
- Whiteboard integrating Assistant through the smart picker to offer a graphical UI for the various tasks
- Deck integrating Assistant through the smart picker to offer a graphical UI for the various tasks
- Nextcloud Office integrating Assistant through the smart picker to offer a graphical UI for the various tasks in documents
- Desktop Clients for simple "Chat with AI"

Backend apps

- *llm2* - Runs open source AI LLM models on your own server hardware (Customer support available upon request)
- OpenAI and LocalAI integration (via OpenAI API) - Integrates with the OpenAI API to provide AI functionality from OpenAI servers (Customer support available upon request; see *AI as a Service*)
- IBM watsonx.ai integration (via IBM watsonx.ai as a Service) - Integrates with the IBM watsonx.ai API to provide AI functionality from IBM Cloud servers (Customer support available upon request; see *AI as a Service*)

Machine translation

As you can see in the table above we have multiple apps offering machine translation capabilities. Each app brings its own set of supported languages. In downstream apps like the Text app, users can use the translation functionality regardless of which app implements it behind the scenes.

Frontend apps

- Assistant offering a graphical translation UI
- Analytics for translating graph labels
- Talk for translating messages and live translations in calls in conjunction with the *Live Transcription app*
- Deck for offering a translation UI in the card description
- *Text* for offering the translation menu
- Notes for offering a translation UI in the note content
- Collectives for offering a translation UI in the page content
- Whiteboard for offering a translation UI through the assistant
- Nextcloud Office for offering translation UI in the document content
- Mail for offering translations for messages

Backend apps

- *translate2 (ExApp)* - Runs open source AI translation models locally on your own server hardware (Customer support available upon request)
- OpenAI and LocalAI integration (via OpenAI API) - Integrates with the OpenAI API to provide AI functionality from OpenAI servers (Customer support available upon request; see *AI as a Service*)
- DeepL integration - Integrates with the deepl API to provide translation functionality from Deepl.com servers (Only community supported)

Speech-To-Text

As you can see in the table above we have multiple apps offering Speech-To-Text capabilities. In downstream apps like the Talk app, users can use the transcription functionality regardless of which app implements it behind the scenes.

Frontend apps

- Assistant offering a graphical translation UI, a smart picker and Audio Chat
- Talk for transcribing calls (see Nextcloud Talk docs for how to enable this)

Backend apps

- *stt_whisper2* - Runs open weights AI Speech-To-Text models on your own server hardware (Customer support available upon request)
- OpenAI and LocalAI integration (via OpenAI API) - Integrates with the OpenAI API to provide AI functionality from OpenAI servers (Customer support available upon request; see *AI as a Service*)

Image generation

As you can see in the table above we have multiple apps offering Image generation capabilities. In downstream apps like the Text-to-Image helper app, users can use the image generation functionality regardless of which app implements it behind the scenes.

Frontend apps

- Assistant for offering a graphical UI and a smart picker
- Deck for inserting images with the smart picker
- *Text* for inserting images with the assistant and smart picker
- Notes for inserting images with the assistant and smart picker
- Collectives for inserting images with the assistant and smart picker
- Whiteboard for inserting images with the assistant and smart picker, generating diagrams and flowcharts with Mermaid
- Nextcloud Office for inserting images with the assistant and smart picker

Backend apps

- Local Stable Diffusion 2 (ExApp) (Customer support available upon request)
- *Local Image Generation: Flux (ExApp)* (Customer support available upon request)
- OpenAI and LocalAI integration (via OpenAI API) - Integrates with the OpenAI API to provide AI functionality from OpenAI servers (Customer support available upon request; see *AI as a Service*)
- *integration_replicate* - Integrates with the replicate API to provide AI functionality from replicate servers (see *AI as a Service*)

Text-To-Speech

As you can see in the table above we have multiple apps offering speech generation capabilities. In downstream apps like the assistant app, users can use the speech generation functionality regardless of which app implements it behind the scenes.

Frontend apps

- Assistant for offering a audio chat

Backend apps

- OpenAI and LocalAI integration (via OpenAI API) - Integrates with the OpenAI API to provide AI functionality from OpenAI servers (Customer support available upon request; see *AI as a Service*)
- Local Text To Speech (ExApp) (Customer support available upon request)

Context Chat

Our Context Chat feature was introduced in Nextcloud Hub 7 (v28). It allows asking questions to the assistant related to your documents in Nextcloud. You will need to install both the context_chat app as well as the context_chat_backend External App. Be prepared that things might break or be a little rough around the edges. We look forward to your feedback!

Frontend apps

- Assistant for offering a graphical UI for the context chat tasks
- Nextcloud Context Agent for offering Context chat as a tool that the agent can execute in the "Chat with AI" feature

Backend apps

- *context_chat + context_chat_backend* - (Customer support available upon request)

Provider apps

Apps can integrate their content with Context Chat to make it available for querying using Context Chat. The following apps have implemented this integration so far:

- *files*
- Analytics
- Mail
- Bookmarks

Context Chat Search

Our Context Chat Search feature allows searching through your documents using natural language. You will need to install both the context_chat app as well as the context_chat_backend External App. We look forward to your feedback!

Frontend apps

- Assistant for offering a graphical UI for the context chat search tasks

Backend apps

- *context_chat + context_chat_backend* - (Customer support available upon request)

Provider apps

See *Context Chat* section above.

Context Agent

Our Context Agent feature was introduced in Nextcloud Hub 9 (v30). It allows asking the assistant to execute tasks related to Nextcloud. You will need to install both the context_agent app as well as a text processing provider.

Frontend apps

- Assistant for offering a graphical UI for the "Chat with AI" feature

Backend apps

- Nextcloud Context Agent for agentic AI capabilities in the "Chat with AI" feature (Customer support available upon request)

Provider apps

- *llm2* - Runs open source AI LLM models on your own server hardware (Customer support available upon request)
- OpenAI and LocalAI integration (via OpenAI API) - Integrates with the OpenAI API to provide AI functionality from OpenAI servers (Customer support available upon request; see *AI as a Service*)

Document generation

Since Hub 11 you can let Nextcloud automatically generate Office documents with content. This functionality is available in the assistant app and made possible by the Nextcloud Office app.

Frontend apps

- Assistant for offering a graphical UI for the context chat search tasks

Backend apps

- Nextcloud Office

Provider apps

- *llm2* - Runs open source AI LLM models on your own server hardware (Customer support available upon request)
- OpenAI and LocalAI integration (via OpenAI API) - Integrates with the OpenAI API to provide AI functionality from OpenAI servers (Customer support available upon request; see *AI as a Service*)
- IBM watsonx.ai integration (via IBM watsonx.ai as a Service) - Integrates with the IBM watsonx.ai API to provide AI functionality from IBM Cloud servers (Customer support available upon request; see *AI as a Service*)

Live transcription

Since Hub 25 Autumn you can let Nextcloud automatically generate subtitles for video and audio calls in Nextcloud Talk.

Frontend apps

- Talk for displaying the subtitles in calls

Backend apps

- *live_transcription* - Runs open weights AI Speech-To-Text models on your own server hardware (Customer support available upon request)

Optical Character Recognition

As you can see in the table above we have multiple apps offering OCR capabilities. In downstream apps like the assistant app, users can use OCR functionality regardless of which app implements it behind the scenes.

Frontend apps

- Assistant

Backend apps

- OpenAI and LocalAI integration (via OpenAI API) >= v5.0.0 - Integrates with the OpenAI API to provide AI functionality from OpenAI servers (Customer support available upon request; see *AI as a Service*)
- *Local OCR: PaddleOCR (ExApp)* - (Customer support available upon request)

Windmill workflows

You can use the AI endpoints in your *Windmill workflows*. Some example workflows showcasing the possibilities of combining Windmill workflows with Nextcloud AI are available on the Windmill Hub

19.1.4 Improve AI task pickup speed

Most AI tasks will be run as part of the background job system in Nextcloud which only runs jobs every 5 minutes by default. To pick up scheduled jobs faster you can set up background job workers inside your Nextcloud main server/container that process AI tasks as soon as they are scheduled. If the PHP code or the Nextcloud settings values are changed while a worker is running, those changes won't be effective inside the runner. For that reason, the worker needs to be restarted regularly. It is done with a timeout of N seconds which means any changes to the settings or the code will be picked up after N seconds (worst case scenario). This timeout does not, in any way, affect the processing or the timeout of the AI tasks.

Changed in version 32.0.7: The command to run the worker has changed from `background-job:worker` to `taskprocessing:worker`. If you are running an older version of Nextcloud, please use the old command.

Screen or tmux session

Run the following occ command inside a screen or a tmux session, preferably 4 or more times for parallel processing of multiple requests by different or the same user (and as a requirement for some apps like context_chat). It would be best to run one command per screen session or per tmux window/pane to keep the logs visible and the worker easily restartable.

```
set -e; while true; do sudo -E -u www-data php occ taskprocessing:worker -v -t 60;␣
↪done
```

For Nextcloud-AIO you should use this command on the host server.

```
set -e; while true; do docker exec -it nextcloud-aio-nextcloud sudo -E -u www-data␣
↪php occ taskprocessing:worker -v -t 60; done
```

You may want to adjust the number of workers and the timeout (in seconds) to your needs. The logs of the worker can be checked by attaching to the screen or tmux session.

Systemd service

1. Create a systemd service file in `/etc/systemd/system/nextcloud-ai-worker@.service` with the following content:

```
[Unit]
Description=Nextcloud AI worker %i
After=network.target

[Service]
ExecStart=/opt/nextcloud-ai-worker/taskprocessing.sh %i
Restart=always
StartLimitInterval=60
StartLimitBurst=10

[Install]
WantedBy=multi-user.target
```

2. Create a shell script in `/opt/nextcloud-ai-worker/taskprocessing.sh` with the following content and make sure to make it executable:

```
#!/bin/sh
echo "Starting Nextcloud AI Worker $1"
cd /path/to/nextcloud
sudo -E -u www-data php occ taskprocessing:worker -v -t 60
```

You may want to adjust the timeout to your needs (in seconds).

3. Enable and start the service 4 or more times:

```
for i in {1..4}; do systemctl enable --now nextcloud-ai-worker@$i.service; done
```

The status of the workers can be checked with (replace 1 with the worker number):

```
systemctl status nextcloud-ai-worker@1.service
```

The list of workers can be checked with:

```
systemctl list-units --type=service | grep nextcloud-ai-worker
```

The complete logs of the workers can be checked with (replace 1 with the worker number):

```
journalctl -xeu nextcloud-ai-worker@1.service -f
```

19.1.5 Frequently Asked Questions

Why is my prompt slow?

Reasons for slow performance from a user perspective can be

- Using CPU processing instead of GPU (sometimes this limit is imposed by the used app)
- High user demand for the feature: User prompts and AI tasks are usually processed in the order they are received, which can cause delays when a lot of users access these features at the same time.

19.2 Nextcloud Assistant

Nextcloud assistant is the primary graphical user interface for interacting with artificial intelligence features in Nextcloud.

It offers the graphical user interface for the unified AI Task processing API offering features like summarizing text, generating headlines, asking arbitrary questions, transcription of media files, image generation and it integrates with the context_chat app to offer in-context answers about your own data stored in Nextcloud. The assistant app also offers a chat interface to interact with the chosen language model, including multimodal chat with file attachments when a compatible backend is configured. Nextcloud can provide customer support upon request, please talk to your account manager for the possibilities.

Find the user documentation here: AI assistant user documentation

19.2.1 Installation

You can install the *assistant* app via the "Apps" page in Nextcloud, or by executing

```
sudo -E -u www-data php occ app:enable assistant
```

19.2.2 App store

You can also find the app in our app store, where you can write a review: https://apps.nextcloud.com/apps/assistant

19.2.3 Repository

You can find the app's code repository on GitHub where you can report bugs and contribute fixes and features: https://github.com/nextcloud/assistant

Nextcloud customers should file bugs directly with our Customer Support.

19.2.4 Related apps

Artificial intelligence at Nextcloud is built in a modular way, allowing you to choose from a variety of solutions for your needs. In order to make use of the various features of the Assistant you will need additional apps that act as backends to provide the actual implementation of the AI functionality. In the Nextcloud administration settings under "Artificial intelligence" you can select which AI backend app to use for which tasks. Note that some of the backend apps are only community maintained, while others are available for Customer support upon request.

The AI admin settings will show all types of Assistant Tasks that are implemented by all your installed apps. Task types can be disabled in the AI admin settings so they are not available for the Assistant or other apps even if they are implemented. All implemented Task types are enabled by default.

Note: At Nextcloud we focus on creating on-premise AI apps that run fully self-hosted on your own servers in order to preserve your privacy and data sovereignty. However, you can also offload these resource-heavy tasks to an *"AI as a Service" provider*.

Note: When using our on-premise AI apps, make sure you have a GPU with enough VRAM that fits all the features you need. For each app documented here you will find its hardware requirements.

Machine translation

In order to make use of machine translation features in the assistant, you will need an app that provides a translation backend: * *translate2 (ExApp)* - Runs open source AI translation models locally on your own server hardware (Customer support available upon request) * *integration_deepl* - Integrates with the deepl API to provide translation functionality from Deepl.com servers (Only community supported)

Speech-To-Text

In order to make use of Speech-to-Text, you will need an app that provides a Speech-To-Text backend:

- *stt_whisper2* - Runs open source AI Speech-To-Text models on your own server hardware (Customer support available upon request)
- OpenAI and LocalAI integration (via OpenAI API) - Integrates with the OpenAI API to provide AI functionality from OpenAI servers (Customer support available upon request; see *AI as a Service*)

Text processing

In order to make use of text processing features in the assistant, you will need an app that provides a Text processing backend:

- *llm2* - Runs open source AI language models locally on your own server hardware (Customer support available upon request)
- OpenAI and LocalAI integration (via OpenAI API) - Integrates with the OpenAI API to provide AI functionality from OpenAI servers (Customer support available upon request; see *AI as a Service*)
- *integration_watsonx* - Integrates with the IBM watsonx.ai API to provide AI functionality from IBM Cloud servers (Customer support available upon request; see *AI as a Service*)

These apps currently implement the following Assistant Tasks:

- *Generate text* (Tested with OpenAI GPT-3.5 and Llama 3.1 8B)
- *Summarize* (Tested with OpenAI GPT-3.5 and Llama 3.1 8B)
- *Generate headline* (Tested with OpenAI GPT-3.5 and Llama 3.1 8B)
- *Extract topics* (Tested with OpenAI GPT-3.5 and Llama 3.1 8B)
- *Context write* (Tested with OpenAI GPT-3.5 and Llama 3.1 8B)
- *Reformulate text* (Tested with OpenAI GPT-3.5 and Llama 3.1 8B)

These tasks may work with other models, but we can give no guarantees.

Text-To-Image

In order to make use of Text-To-Image features, you will need an app that provides an image generation backend:

- *text2image_stablediffusion2* (Customer support available upon request)
- *text2image_flux* - Local text-to-image and image editing with FLUX.2 [klein] 4B (Customer support available upon request)
- OpenAI and LocalAI integration (via OpenAI API) - Integrates with the OpenAI API to provide AI functionality from OpenAI servers (Customer support available upon request; see *AI as a Service*)
- *integration_replicate* - Integrates with the replicate API to provide AI functionality from replicate servers (see *AI as a Service*)

Context Chat

In order to make use of our special Context Chat feature, offering insights about your own documents and data stored in Nextcloud, you will need the following apps:

- *context_chat + context_chat_backend* - (Customer support available upon request)

You will also need a text processing provider as specified above (ie. llm2, integration_openai or integration_watsonx).

Chat

In order to make use of our "Chat with AI" feature you will need any one of the following apps:

- *llm2* - Runs open source AI language models locally on your own server hardware (Customer support available upon request)
- OpenAI and LocalAI integration (via OpenAI API) - Integrates with the OpenAI API to provide AI functionality from OpenAI servers (Customer support available upon request; see *AI as a Service*)

Multimodal chat

Multimodal chat extends "Chat with AI" so users can attach files for the language model to analyze in the conversation, such as images and, with some models, audio and video. Text files can also be attached as context.

In order to make use of multimodal chat you will need any one of the following backends with a multimodal-capable model:

- *llm2* - Use a multimodal model such as the shipped Qwen 3.5 9B (vision) or Gemma 4 E4B (vision and audio) (Customer support available upon request)
- OpenAI and LocalAI integration (via OpenAI API) - Integrates with multimodal models via the OpenAI API (Customer support available upon request; see *AI as a Service*)

Voice Chat

In order to make use of our "Audio chat" feature which allows you to interact with the assistant Chat via your Voice and Ears as if in a real conversation, you will need any of the following set of apps:

- **Either**
  - OpenAI and LocalAI integration (via OpenAI API) - Integrates with the OpenAI API to provide AI functionality from OpenAI servers (Customer support available upon request; see *AI as a Service*)
- **Or**
  - *llm2* - Runs open source AI language models locally on your own server hardware (Customer support available upon request)
  - *stt_whisper2* - Runs open source AI Speech-To-Text models on your own server hardware (Customer support available upon request)
  - *text2speech_kokoro* - Runs open source AI Text-To-Speech models on your own server hardware (Customer support available upon request)

Context Agent

In order to make use of our AI agent feature, offering the execution of actions on behalf of the user based on the AI chat, you will need the following apps:

- *context_agent* - (Customer support available upon request)

You will also need a text processing provider as specified above (ie. *llm2* or *integration_openai*).

Text-To-Speech

In order to make use of Text-To-Speech, you will need an app that provides a Text-To-Speech backend, which is one of the following:

- OpenAI and LocalAI integration (via OpenAI API) - Integrates with the OpenAI API to provide AI functionality from OpenAI servers (Customer support available upon request; see *AI as a Service*)

- *text2speech_kokoro* - Runs open source AI Text-To-Speech models on your own server hardware (Customer support available upon request)

19.2.5 Configuration

The Assistant admin settings can be found under the "Artificial intelligence" section. You can disable the assistant top menu entry there. You can also disable the AI-related smart pickers. The occ commands to change the options are listed below.

Assistant configuration

1. Top-right Assistant

```
occ config:app:set assistant assistant_enabled --value=1 --type=string
```

To enable/disable the assistant button in the navigation bar for all users.

2. AI text generation smart picker

```
occ config:app:set assistant free_prompt_picker_enabled --value=1 --type=string
```

To enable/disable the AI text generation smart picker for all the users.

3. Text-to-image smart picker

```
occ config:app:set assistant text_to_image_picker_enabled --value=1 --type=string
```

To enable/disable the text-to-image smart picker for all the users.

4. Speech-to-text smart picker

```
occ config:app:set assistant speech_to_text_picker_enabled --value=1 --type=string
```

To enable/disable the speech-to-text smart picker for all the users.

5. Data folder

```
occ config:app:set assistant default_data_folder --value="Assistant" --type=string
```

The assistant stores the content it generates, such as generated images and speech-to-text output, in a folder in the files of each user. This sets the name that folder is given when it is created. It defaults to `Assistant`, and is also editable in the Assistant admin settings.

Users can choose their own name in their personal Assistant settings, which is stored per user and takes precedence over the server-wide default:

```
occ user:setting $USER_ID assistant data_folder "Assistant"
```

Two things to keep in mind:

- Changing either value does not rename or move a folder that already exists. It only affects the name used the next time a folder has to be created, so existing generated content stays where it is.
- A user who already has a folder under the built-in name `Assistant`, from before a default was set, keeps using it. This avoids starting a second folder for them and leaving their earlier output behind.

Task processing

1. List Tasks

```
occ taskprocessing:task:list
```

lists all task processing tasks.

2. Get Task

```
occ taskprocessing:task:get $TASK_ID
```

shows all information for a specific task.

3. Enable or disable a Task type

```
occ taskprocessing:task-type:set-enabled $TASK_TYPE_ID 1
```

Set 1 to enable and 0 to disable an implemented task type.

4. Get Task statistics

```
occ taskprocessing:task:stats
```

shows statistics for all task processing Tasks.

5. Clean-up old tasks

```
occ taskprocessing:task:cleanup
```

delete tasks that are older than this number of seconds, defaults to 4 months.

Image storage

Days until generated images are deleted if they are not viewed.

```
occ config:app:set assistant max_image_generation_idle_time --value=90 --type=integer
```

Chat with AI

1. Chat User Instructions for Chat Completions

```
occ config:app:set assistant chat_user_instructions --value="hello world"
```

The user instructions that are prepended before the chat messages for the AI model to understand the context of the block of text. This is a good place not only to instruct the AI model to be polite and kind but also to for example answer all the queries in a particular language or better yet, follow the user's language. The sky is the limit.

Note: The default instructions are optimized to work well across a variety of language models, but may not be optimal for the specific model you choose. Specifically, the model may be tempted to mention the user's name a bit too often and may mention the user's language in an unusual manner.

2. Chat User Instructions for Title Generation

```
occ config:app:set assistant chat_user_instructions_title --value="hello title"
```

This field is appended to the block of chat messages, i.e. attached after the messages. It is done this way to allow it to be used even with text completion models which could have the instructions as "The title for the above conversation could be "".

3. Last N messages to consider for chat completions

```
occ config:app:set assistant chat_last_n_messages --value=10
```

The number of latest messages to consider for generating the next message. This does not include the user instructions, which is always considered in addition to this. This value should be adjusted in case you are hitting the token limit in your conversations too often. The AI text generation provider should ideally handle the max token limit case.

Improve AI task pickup speed

See *the relevant section in AI Overview* for more information.

19.3 App: Local Machine translation 2 (translate2)

The *translate2* app is one of the apps that provide machine translation functionality in Nextcloud and act as a translation backend for the *Nextcloud Assistant app*. The *translate2* app specifically runs only open source models and does so entirely on-premises. Nextcloud can provide customer support upon request, please talk to your account manager for the possibilities.

The app currently supports 400+ languages. See the complete list here: https://huggingface.co/datasets/allenai/MADLAD-400

19.3.1 Requirements

- Minimal Nextcloud version: 30
- This app is built as an External App and thus depends on AppAPI v3.1.0 or higher
- Nextcloud AIO is supported
- We currently support NVIDIA GPUs and x86_64 CPUs
- CUDA >= v12.2.2 on your host system
- GPU Sizing
  - A NVIDIA GPU with at least 4 GB VRAM
  - At least 6 GB of system RAM
- CPU Sizing
  - x86 CPU with 4-8 cores for the app to use (The more cores the faster it will be)
  - At least 6 GB of RAM for the app should be enough (includes software+libraries and the model)

Space usage

- ~ 2.95 GB for the docker container
- ~ 2.77 GB for the default model

19.3.2 Installation

0. Make sure the *Nextcloud Assistant app* is installed
1. *Install AppAPI and setup a Deploy Demon*
2. Install the "Local Machine Translation" (translate2) ExApp via the "Apps" page in the Nextcloud web admin user interface

19.3.3 Model Switch

1. Remove `hf_model_path` key from `loader` object in the `config.json` file in the docker container named `nc_app_translate2`.
2. Change `model_name` to the new model name to `Nextcloud-AI/madlad400-7b-mt-bt-ct2-int8_float32`.
3. Restart the docker container `docker restart nc_app_translate2`

19.3.4 App store

You can also find the app in our app store, where you can write a review: https://apps.nextcloud.com/apps/translate2

19.3.5 Repository

You can find the app's code repository on GitHub where you can report bugs and contribute fixes and features: https://github.com/nextcloud/translate2

Nextcloud customers should file bugs directly with our Customer Support.

19.3.6 Ethical AI Rating

Rating:

Positive: * the software for training and inference of this model is open source * the trained model is freely available, and thus can be run on-premises * the training data is freely available, making it possible to check or correct for bias or optimise the performance and CO2 usage.

Learn more about the Nextcloud Ethical AI Rating in our blog.

19.3.7 Known Limitations

- AI translations are not a replacement for human professional translations and in many cases post-editing is required. AI translations can be used for understanding the main content of a text but not for translations that require special knowledge (such as technical content or legal content), or translations that require specific writing style to convey style, deeper meaning, or emotions (such as marketing content or translating books).
- While the quality of the output will be fine for the most common languages (English, French, Spanish) the quality will suffer for languages that have less coverage in the original training set.
- Make sure to test the translation model you are using it for whether it meets the use-case's quality requirements. The default model is the smallest of the batch and might produce duplicate translation outputs. Switch to a larger model if you need better quality and less artifacts, see *Model Switch*.
- Language models notoriously have a high energy consumption.
- Customer support is available upon request, however we can't solve false or problematic output, most performance issues, or other problems caused by the underlying models. Support is thus limited only to bugs directly caused by the implementation of the app (connectors, API, front-end, AppAPI).

19.4 App: Local large language model (llm2)

The *llm2* app is one of the apps that provide text processing functionality using Large language models in Nextcloud and act as a text processing backend for the *Nextcloud Assistant app*, the *mail* app and *other apps making use of the core Text Processing API*. The *llm2* app specifically runs only open source models and does so entirely on-premises. Nextcloud can provide customer support upon request, please talk to your account manager for the possibilities.

This app uses llama.cpp under the hood and is thus compatible with any model in *gguf* format.

However, we primarily test with the models listed below. Output quality will differ depending on which model you use and downstream tasks like summarization or Context Chat may not work on other models. We thus recommend the following models:

- Qwen 3.5 9B (good quality; multimodal vision; comes shipped with the app)
- Gemma 4 E4B (fast; multimodal vision and audio; comes shipped with the app)
- OLMo 3 7B Instruct (fully open; green Ethical AI rating; text only; comes shipped with the app; see addendum below)

19.4.1 Multimodal chat

With a multimodal-capable model, *llm2* supports multimodal chat in the *Nextcloud Assistant app*: users can attach images (and with Gemma 4, also audio) to chat messages so the model can reason about their content. Text files can also be attached as context.

The following shipped models support multimodality:

- **Qwen 3.5 9B** - vision (images)
- **Gemma 4 E4B** - vision (images) and audio

Vision-capable models also register the *Analyze images* and *OCR* task types in addition to the usual text-processing tasks.

Multimodal models need a matching multimodal projector (`mmproj`) *gguf* file next to the model file. The projector files for the shipped Qwen 3.5 and Gemma 4 models are downloaded automatically with the models. If you supply your own multimodal model, place its `mmproj` file in `/nc_app_llm2_data` as well and set `mmproj_path` in the model's JSON configuration (see *Configuring alternate models* below).

19.4.2 Multilinguality

This app supports input and output in languages other than English if the underlying model supports the language.

Qwen 3.5 supports the following 201 languages: <https://qwen.ai/blog?id=qwen3.5>

Gemma 4 provides multilingual support, offering high-quality performance in 35 languages and basic support for 140 languages.

Llama 3.1 supports the following languages:

- English
- Portuguese
- Spanish
- Italian
- German
- French
- Hindi
- Thai

Note that other languages may work as well, but only the above languages are guaranteed to work with Llama 3.1.

19.4.3 Requirements

- This app is built as an External App and thus depends on AppAPI v3.1.0 or higher
- Nextcloud AIO is supported
- We currently support NVIDIA GPUs and x86_64 CPUs
- CPU that supports AVX and AVX2 instruction
- CUDA >= v12.4 on your host system
- GPU Sizing
  - A NVIDIA GPU with at least 8GB VRAM
  - At least 12GB of system RAM
- CPU Sizing
  - At least 12GB of system RAM
  - The more cores you have and the more powerful the CPU the better, we recommend 10-20 cores
  - The app will hog all cores by default, so it is usually better to run it on a separate machine

19.4.4 Installation

0. Make sure the *Nextcloud Assistant app* is installed
1. *Install AppAPI and setup a Deploy Demon*
2. Install the "Local large language model" ExApp via the "Apps" page in the Nextcloud web admin user interface

Supplying alternate models

This app allows supplying alternate LLM models as *gguf* files in the `/nc_app_llm2_data` directory of the docker container.

1. Download a **gguf** model e.g. from huggingface
2. Copy the **gguf** file to `/nc_app_llm2_data` inside the docker container
3. Restart the llm2 ExApp
4. Select the new model in the Nextcloud AI admin settings

Configuring alternate models

Since every model requires slightly different inference parameters, you can pass along a configuration file for the alternate model files you supply.

The configuration file for a model file must have the same name as the model file but must end in `.json` instead of `.gguf`.

The strings `{system_prompt}` and `{user_prompt}` are variables that will be filled in by the app, so they must be part of your prompt template.

Here is an example config file for Llama 2:

```
{
  "prompt": "<|im_start|> system\n{system_prompt}\n<|im_end|>\n<|im_start|> user\n
↪{user_prompt}\n<|im_end|>\n<|im_start|> assistant\n",
  "loader_config": {
     "n_ctx": 4096,
```

(continues on next page)

(continued from previous page)

```
    "max_tokens": 2048,
    "stop": ["<|im_end|>"]
  }
}
```

Here is an example configuration for Llama 3:

```
{
  "prompt": "<|begin_of_text|><|start_header_id|>system<|end_header_id|>\n{system_
↪prompt}<|eot_id|><|start_header_id|>user<|end_header_id|>\n{user_prompt}<|eot_id|>\n
↪<|start_header_id|>assistant<|end_header_id|>\n",
  "loader_config": {
      "n_ctx": 8000,
      "max_tokens": 4000,
      "stop": ["<|eot_id|>"],
      "temperature": 0.3
  }
}
```

19.4.5 Scaling

It is currently not possible to scale this app, we are working on this. Based on our calculations an instance has a rough capacity of 1000 user requests per hour. However, this number is based on theory and we do appreciate real-world feedback on this. If you would like to scale up your language model usage, we recommend using an *AI as a Service provider* or hosting a service compatible with the OpenAI API yourself that can be scaled up and connecting nextcloud to it via the integration_openai app.

19.4.6 App store

You can also find the app in our app store, where you can write a review: https://apps.nextcloud.com/apps/llm2

19.4.7 Repository

You can find the app's code repository on GitHub where you can report bugs and contribute fixes and features: https://github.com/nextcloud/llm2

Nextcloud customers should file bugs directly with our Support system.

19.4.8 Known Limitations

- We currently only support languages that the underlying model supports; correctness of language use in languages other than English may be poor depending on the language's coverage in the model's training data (We recommend Qwen 3.5 for Asian languages, while Gemma 4 is better for European languages)
- Language models can be bad at reasoning tasks
- Language models can be bad at math
- Language models are likely to generate false information and should thus only be used in situations that are not critical. It's recommended to only use AI at the beginning of a creation process and not at the end, so that outputs of AI serve as a draft for example and not as final product. Always check the output of language models before using it.
- Make sure to test the language model you are using it for whether it meets the use-case's quality requirements

- Language models notoriously have a high energy consumption, if you want to reduce load on your server you can choose smaller models or quantized models in exchange for lower accuracy
- Multimodal features (chat with image or audio attachments, Analyze images, OCR) require a multimodal-capable model and its matching `mmproj` file; text-only models such as Llama 3.1 will not process image or audio attachments
- Customer support is available upon request, however we can't solve false or problematic output, most performance issues, or other problems caused by the underlying model. Support is thus limited only to bugs directly caused by the implementation of the app (connectors, API, front-end, AppAPI)

19.4.9 Addendum: Running with a fully open model

If you would like to use a fully open model that scores a green score on our Ethical AI rating, we recommend the following model:

- Olmo 3 (either in 7B or 32B): https://huggingface.co/allenai/Olmo-3-7B-Instruct

What makes OLMo a fully open model?

- The code for training, fine-tuning and inference of the model is publicly available and fully open source
- The training data with which the model is pretrained is publicly available
- The model itself is publicly available and fully open source
- The instruction tuning data is publicly available
- The Reinforcement learning model is publicly available and fully open source

Limitations

- OLMo currently only works well with English language input
- In our tests it sometimes produced hallucinated or garbled output; make sure to thoroughly test the model for your use case
- It cannot use tools, so cannot be used in conjunction with *Context Agent*

19.5 App: Local Whisper Speech-To-Text (stt_whisper2)

The *stt_whisper2* app is one of the apps that provide Speech-To-Text functionality in Nextcloud and act as a media transcription backend for the *Nextcloud Assistant app*, the *talk* app and *other apps making use of the core Speech-To-Text API*. The *stt_whisper2* app specifically runs only open source models and does so entirely on-premises. Nextcloud can provide customer support upon request, please talk to your account manager for the possibilities.

This app supports input and output in languages other than English if the underlying model supports the language.

This app uses faster-whisper under the hood. Output quality will differ depending on which model you use, we recommend the following models:

- OpenAI Whisper large v3 turbo (multilingual)
- OpenAI Whisper medium.en (English only)

Whisper large v3 supports about ~100 languages and shows outstanding performance in ~10 of them. For more details see the OpenAI Whisper paper

19.5.1 Requirements

- Minimal Nextcloud version: 28
- This app is built as an External App and thus depends on AppAPI v2.3.0
- Nextcloud AIO is supported
- We currently support NVIDIA GPUs and x86_64 CPUs
- CUDA >= v12.2 on your host system
- GPU Sizing
  - A NVIDIA GPU with at least 4GB VRAM
- CPU Sizing
  - The more cores you have and the more powerful the CPU the better, we recommend 10-20 cores
  - The app will hog all cores by default, so it is usually better to run it on a separate machine
  - 4GB for the app

19.5.2 Installation

0. Make sure the *Nextcloud Assistant app* is installed
1. *Install AppAPI and setup a Deploy Demon*
2. Install the *stt_whisper2* "Local Speech-To-Text" ExApp via the "Apps" page in the Nextcloud web admin user interface

Supplying alternate models

This app allows supplying alternate models in the `/nc_app_stt_whisper2_data` directory of the docker container. You can use any *faster-whisper* model by Systran on hugging face in the following way:

1. git cloning the respective repository
2. Copying the folder with the git repository to `/nc_app_stt_whisper2_data` inside the docker container.
3. Restarting the Whisper ExApp
4. Selecting the respective model in the Nextcloud AI admin settings

19.5.3 Scaling

It is currently not possible to scale this app, we are working on this. Based on our calculations an instance has a rough capacity of 4h of transcription throughput per minute (measured with 8 CPU threads on an Intel(R) Xeon(R) Gold 6226R). It is unclear how close to real-world usage this number is, so we do appreciate real-world feedback on this.

19.5.4 App store

You can also find this app in our app store, where you can write a review: https://apps.nextcloud.com/apps/stt_whisper2

19.5.5 Repository

You can find the app's code repository on GitHub where you can report bugs and contribute fixes and features: https://github.com/nextcloud/stt_whisper2

Nextcloud customers should file bugs directly with our customer support.

19.5.6 Known Limitations

- We currently do not support live transcription
- We currently only support languages supported by the underlying Whisper models
- The whisper models perform unevenly across languages, and may show lower accuracy on low-resource and/or low-discoverability languages or languages where there was less training data available. The models also exhibit disparate performance on different accents and dialects of particular languages, which may include higher word error rate across speakers of different genders, races, ages, or other demographic criteria.
- Language models are likely to generate false information and should thus only be used in situations that are not critical. It's recommended to only use AI at the beginning of a creation process and not at the end, so that outputs of AI serve as a draft for example and not as final product. Always check the output of language models before using it.
- Make sure to test the language model you are using it for whether it meets the use-case's quality requirements
- Language models notoriously have a high energy consumption, if you want to reduce load on your server you can choose smaller models or quantized models in exchange for lower accuracy
- Customer support is available upon request, however we can't solve false or problematic output, most performance issues, or other problems caused by the underlying model. Support is thus limited only to bugs directly caused by the implementation of the app (connectors, API, front-end, AppAPI)

19.6 App: Local Image Generation (text2image_stablediffusion2)

The *text2image_stablediffusion2* app is one of the apps that provide image generation functionality in Nextcloud and act as an image generation backend for the *Nextcloud Assistant app* and other apps making use of the image generation functionality. The *text2image_stablediffusion2* app specifically runs only open source models and does so entirely on-premises. Nextcloud can provide customer support upon request, please talk to your account manager for the possibilities.

19.6.1 Requirements

- This app is built as an External App and thus depends on AppAPI v3.1.0 or higher
- Nextcloud AIO is supported
- We currently support NVIDIA GPUs and x86_64 CPUs
- CUDA >= v12.2 on your host system
- GPU Sizing
  - A NVIDIA GPU with at least 8GB VRAM
- CPU Sizing
  - At least 8GB of system RAM
  - The more cores you have and the more powerful the CPU the better, we recommend 10-20 cores
  - The app will hog all cores by default, so it is usually better to run it on a separate machine

19.6.2 Installation

- Make sure the *Nextcloud Assistant app* is installed
- *Install AppAPI and setup a Deploy Demon*
- Install the "Local large language model" ExApp via the "Apps" page in the Nextcloud web admin user interface

19.6.3 Scaling

It is currently not possible to scale this app, we are working on this. Based on our calculations an instance has a rough capacity of 120 image requests per hour (each user request can be for multiple images). However, this number is based on theory and we do appreciate real-world feedback on this.

19.6.4 App store

You can also find the app in our app store, where you can write a review: https://apps.nextcloud.com/apps/text2image_stablediffusion2

19.6.5 Repository

You can find the app's code repository on GitHub where you can report bugs and contribute fixes and features: https://github.com/nextcloud/text2image_stablediffusion2

Nextcloud customers should file bugs directly with our Support system.

19.6.6 Known Limitations

- The generated images are of a fixed resolution (512x512 pix), and the model does not achieve perfect photorealism
- The model cannot render legible text
- Faces and people in general may not be generated properly
- The results for certain image generation requests can be biased and may enforce stereotypes
- We currently only support languages that the underlying model supports; correctness of language use in languages other than English may be poor depending on the language's coverage in the model's training data
- Make sure to test the app for whether it meets the use-case's quality requirements
- Customer support is available upon request, however we can't solve false or problematic output, most performance issues, or other problems caused by the underlying model. Support is thus limited only to bugs directly caused by the implementation of the app (connectors, API, front-end, AppAPI)

19.7 App: Local Image Generation: Flux (text2image_flux)

The *text2image_flux* app is one of the apps that provide image generation functionality in Nextcloud and act as an image generation backend for the *Nextcloud Assistant app* and *other apps making use of the image generation functionality*. It also provides image editing (image-to-image) from a text prompt. The *text2image_flux* app specifically runs only open source models and does so entirely on-premises. Nextcloud can provide customer support upon request, please talk to your account manager for the possibilities.

19.7.1 Providers

The app registers the following Task Processing providers:

- **Nextcloud Local Image Generation: Flux 2 Klein 4B** - text-to-image (`core:text2image`)
- **Nextcloud Local Image Generation: Flux 2 Klein 4B (Enhanced)** - text-to-image with automatic prompt refinement via a text-to-text provider (`core:text2text`) before generation
- **Nextcloud Local Image Editing: Flux 2 Klein 4B** - image editing from a prompt (`core:image2image` on Nextcloud 36+; a custom `text2image_flux:image2image` task type on Nextcloud 35 and older)

Select the providers in the Nextcloud AI admin settings. For the Enhanced provider, also install and enable a text-to-text backend such as *llm2*.

Default output size is `1024x1024`. You can override the size per task (`WxH` format) or change the default via the `DEFAULT_SIZE` deploy option (see *Configuration* below).

19.7.2 Requirements

- Minimal Nextcloud version: 30
- This app is built as an External App and thus depends on AppAPI v3.1.0 or higher
- Nextcloud AIO is supported
- We currently support NVIDIA GPUs and x86_64 CPUs
- CUDA >= v12.2 on your host system
- GPU Sizing
  - A NVIDIA GPU with at least 8GB VRAM (roughly 7GB used)
  - At least 8GB of system RAM
- CPU Sizing
  - CPU inference is supported but very slow; for faster generation prefer a GPU or use *text2image_stablediffusion2*
  - At least 12GB of system RAM
  - The more cores you have and the more powerful the CPU the better, we recommend 10-20 cores
  - The app will hog all cores by default, so it is usually better to run it on a separate machine

19.7.3 Installation

0. Make sure the *Nextcloud Assistant app* is installed
1. *Install AppAPI and setup a Deploy Demon*
2. Install the **Local Image Generation: Flux** ExApp via the "Apps" page in the Nextcloud web admin user interface
3. Optionally install a text-to-text provider such as *llm2* if you want to use the Enhanced text-to-image provider
4. Select the Flux providers in the Nextcloud AI admin settings

19.7.4 Configuration

You can set the following environment variables in the *Deploy Options* when installing or redeploying the ExApp:

- `NUM_INFERENCE_STEPS` - Number of denoising steps used when generating or editing images. Default is `4`.
- `DEFAULT_SIZE` - Default size of generated or edited images in `WxH` format (for example `1024x1024`). Used when no size is provided with the task. Default is `1024x1024`.

19.7.5 Scaling

It is currently not possible to scale this app, we are working on this. Throughput is lower than with *text2image_stablediffusion2* because Flux generation is slower. We do appreciate real-world feedback on capacity.

19.7.6 App store

You can also find the app in our app store, where you can write a review: https://apps.nextcloud.com/apps/text2image_flux

19.7.7 Repository

You can find the app's code repository on GitHub where you can report bugs and contribute fixes and features: https://github.com/nextcloud/text2image_flux

Nextcloud customers should file bugs directly with our Support system.

19.7.8 Ethical AI Rating

Rating: Yellow

Positive:

- The software for training and inferencing of this model is open source
- The trained model is freely available under Apache 2.0, and thus can be run on-premises

Negative:

- The training data is not freely available, making it not possible to check or correct for bias or optimise the performance and CO2 usage

Learn more about the Nextcloud Ethical AI Rating in our blog.

19.7.9 Known Limitations

- Image generation and editing are slower than with *text2image_stablediffusion2*; prefer that app if speed matters more than quality
- The Enhanced provider depends on a working text-to-text Task Processing provider; if prompt improvement fails, the original prompt is used
- Faces and people in general may not be generated properly
- The results for certain image generation requests can be biased and may enforce stereotypes
- We currently only support languages that the underlying model supports; correctness of language use in languages other than English may be poor depending on the language's coverage in the model's training data
- Make sure to test the app for whether it meets the use-case's quality requirements
- Customer support is available upon request, however we can't solve false or problematic output, most performance issues, or other problems caused by the underlying model. Support is thus limited only to bugs directly caused by the implementation of the app (connectors, API, front-end, AppAPI)

19.8 App: Recognize

The *recognize* app provides media tagging and face recognition functionality for the memories app. *Recognize* can group similar faces on user's photos ("face recognition"); it can add fitting tags to photos detecting landscapes, food, vehicles, buildings animals and other objects, as well as known landmarks and monuments; it can recognize music genres in user's audio files and adds tags for those; it can recognize human actions on user's video files and add tags for them. It specifically runs only open source models and does so entirely on-premises. Nextcloud can provide customer support upon request, please talk to your account manager for the possibilities.

19.8.1 Front-end

Tagged files will appear in the Memories app under the "Tags" section as well as in the normal Files app. Face recognition results will appear under the "People" section in the Memories app.

19.8.2 Requirements

- Nextcloud AIO is not supported but will likely work at sub optimal speed
- Minimum supported Nextcloud version: 26
- x86 CPU
- GNU lib C
- Background Jobs must be executed via cron
- Using GPU processing is supported, but not required; slow performance is expected if you are not using a GPU
- We currently only support NVIDIA GPUs
- For GPU support you need to install:
  - NVIDIA® GPU drivers version 450.80.02 or higher.
  - CUDA® Toolkit 11.x
  - cuDNN SDK 8.x
- GPU Sizing
  - The models used by recognize require about 1GB of VRAM or less
- CPU Sizing
  - If you don't have a GPU, this app will utilize your CPU cores
  - The more cores you have and the more powerful the CPU the better, we recommend 10-20 cores
  - In the app settings you can set the number of cores to use
  - At least ~4GB of RAM dedicated for recognize

Disk space usage

- ~1.5GB for all models in total

19.8.3 Installation

1. Install the *recognize* app via the "Apps" page in Nextcloud, or by executing

   occ app:enable recognize

2. Execute the following command on your server terminal of each node that runs background jobs:

   occ recognize:download-models

3. Go to your Nextcloud Administration settings and open the *recognize* admin settings page
4. Enable all modes of operation that you want the app to undertake
5. Enable GPU mode if you have a GPU that you want to use; if you want to use CPU only, you can set the number of cores to use here
6. Execute the following command on your server terminal to stop background processing of existing files:

   occ recognize:clear-background-jobs

7. Execute the following command on your server terminal to process all existing files in bulk (This may take a long time, depending on how many files you have on your instance):

   occ recognize:classify

8. Execute the following command on your server terminal to calculate face clusters from faces found in all existing files (Run this repeatedly until no more clusters are found):

   occ recognize:cluster-faces

9. All new files from this point on will be automatically processed in background tasks without manual intervention

19.8.4 Scaling

It is possible to scale this app by adding multiple "background" nodes to your cluster that will only process background jobs by executing cron.php.

19.8.5 App store

You can also find the app in our app store, where you can write a review: https://apps.nextcloud.com/apps/recognize

19.8.6 Repository

You can find the app's source repository on GitHub where you can report bugs and contribute fixes and features: https://github.com/nextcloud/recognize

Nextcloud customers should file bugs directly with our Support system.

19.8.7 Known Limitations

- Make sure to test whether the functionality meets the use-case's quality requirements
- Machine learning models notoriously have a high energy consumption
- Customer support is available upon request, however we can't solve false or problematic output, most performance issues, or other problems caused by the underlying model. Support is thus limited only to bugs directly caused by the implementation of the app (connectors, API, front-end, AppAPI)

19.8.8 Ethical AI Rating

Rating for Photo object detection: Green

Positive:

- The software for training and inference of this model is open source
- The trained model is freely available, and thus can be run on-premises
- The training data is freely available, making it possible to check or correct for bias or optimize the performance and CO2 usage.

Rating for Photo face recognition: Green

Positive:

- The software for training and inference of this model is open source
- The trained model is freely available, and thus can be run on-premises
- The training data is freely available, making it possible to check or correct for bias or optimize the performance and CO2 usage.

Rating for Video action recognition: Green

Positive:

- The software for training and inferencing of this model is open source
- The trained model is freely available, and thus can be ran on-premises
- The training data is freely available, making it possible to check or correct for bias or optimize the performance and CO2 usage.

Rating Music genre recognition: Yellow

Positive:

- The software for training and inference of this model is open source
- The trained model is freely available, and thus can be run on-premises

Negative:

- The training data is not freely available, limiting the ability of external parties to check and correct for bias or optimise the model's performance and CO2 usage.

Learn more about the Nextcloud Ethical AI Rating in our blog.

19.9 App: Context Chat

Context Chat is an *assistant* feature that is implemented via an ensemble of two apps:

- the `context_chat` app, written purely in PHP
- the `context_chat_backend` ExternalApp written in Python

Together they provide the ContextChat *text processing* and *search* tasks accessible via the *Nextcloud Assistant app*.

The `context_chat` and `context_chat_backend` apps will use the configured text-to-text task processing provider, which is required on a fresh install. It can be configured to run open source models entirely on-premises, see the list of providers *here* in the "Backend apps" section.

This app supports input and output in the same languages that the currently configured text-to-text task processing provider supports.

19.9.1 Requirements

- Minimal Nextcloud version: 32
- Nextcloud AIO is supported
- We currently support NVIDIA GPUs, AMD GPUs (via Vulkan) and x86_64 CPUs
- CPU that supports AVX and AVX2 instruction
- CUDA >= v12.8 on your host system if NVIDIA GPUs are used
- Both podman and docker are supported
- GPU Setup Sizing
  - **A GPU with at least 2GB VRAM**
    - The requirements for the text-to-text providers should be checked separately for each app *here* in the "Backend apps" section, as they can vary greatly based on the model used and whether the provider is hosted locally or remotely.

  - **At least 8GB of system RAM**
    - 2 GB + additional 500MB for each concurrent request made to the backend if configuration parameters are changed
- CPU Setup Sizing
  - **At least 12GB of system RAM**
    - 2 GB + additional 500MB for each additional concurrent query request
    - 8 GB is recommended in the above case for the default settings
  - This app makes use of the configured text-to-text task processing provider instead of running its own language model by default, thus 4+ cores for the embedding model is needed
- A dedicated machine is recommended

Space usage

This app employs a bundled DB with Vector support called PostgreSQL. All the users' textual data is duplicated, chunked and stored on disk in this vector DB along with semantic embedding vectors for the content.

19.9.2 Installation

1. Make sure the *Nextcloud Assistant app* is installed
2. Setup a *Deploy Daemon* in AppAPI Admin settings
3. Install the `context_chat_backend` ExApp via the "Apps" page in Nextcloud, or by executing (checkout the readme at https://github.com/nextcloud/context_chat_backend for manual install steps)

```
occ app_api:app:register context_chat_backend
```

4. Install the `context_chat` app via the "Apps" page in Nextcloud, or by executing

```
occ app:enable context_chat
```

5. Install a text-to-text provider (text generation provider) via the "Apps" page in Nextcloud. A list of providers can be found *here* in the "Backend apps" section.
6. Optionally but recommended, setup background workers for faster pickup of tasks. See *the relevant section in AI Overview* for more information.

> **Note**
>
> Both apps need to be installed and both major version and minor version of the two apps must match for the functionality to work (ie. "v1.3.4" and "v1.3.1"; but not "v1.3.4" and "v2.1.6"; and not "v1.3.4" and "v1.4.5"). Keep this in mind when updating.

19.9.3 Initial loading of data

Auto-indexing

Context chat will automatically load user data into the Vector DB. Context chat backend pulls the queued files and content providers' items from the context chat PHP app and indexes them.
The initial loading of data can take a long time depending on the number of files and their size.

19.9.4 Scaling

Listed below are the major parts of the system that can be scaled independently to improve performance:

1. The text-to-text task processing provider (from among the list of providers *here* in the "Backend apps" section)

   The text-to-text task processing provider can be scaled by using a hosted service using the OpenAI and LocalAI integration (via OpenAI API) like OpenAI or by hosting your own model on powerful hardware.

2. The vector DB performance

   The vector DB performance can be scaled by using a dedicated or cluster setup for PostgreSQL with the pgvector extension.
   The connection string of the external vector DB can be set using the environment variable `EXTERNAL_DB` during deployment in the "Deploy Options".

3. The embedding model performance

   The embedding model performance can be scaled by using a hosted embedding service, locally or remotely hosted. It should be able to serve an OpenAI-compatible API.
   The embedding service URL can be set using the environment variable `CC_EM_BASE_URL` during deployment in the "Deploy Options". Other options like the model name, api key, or username and password can be set using the environment variables `CC_EM_MODEL_NAME`, `CC_EM_APIKEY`, `CC_EM_USERNAME`, and `CC_EM_PASSWORD` respectively.

   Warning

   The embedding model cannot be changed after installing the app. To use a different embedding model or service, you will need to do a full uninstall (removing all data of the ExApp) and reinstall the `context_chat_backend` ExApp with the new environment variables and an empty vector DB. If the vector DB is external, the connected database (database may be named `ccb`) should be dropped before installing the ExApp again.

   For the `context_chat` app, obtain a clean slate by dropping all the `<PREFIX>_context_chat_*` tables in the database, and removing all the config values: And re-installing it.

   ```
   drop table if exists oc_context_chat_action_queue;
   drop table if exists oc_context_chat_content_queue;
   drop table if exists oc_context_chat_fs_events;
   drop table if exists oc_context_chat_queue;
   delete from oc_appconfig where appid = 'context_chat';
   ```

4. The parsing of the documents to extract text

   The parsing of the documents to extract text is done in a single instance of the `context_chat_backend` ExApp in a docker-based environment. It is a CPU-bound task, so having a powerful CPU will help speed up the parsing process.
   This can be scaled by using Kubernetes for deployment, allowing multiple instances of the `context_chat_backend` ExApp to handle the parsing concurrently, see *the Kubernetes section*.

If `context_chat_backend` is already deployed, you can change these environment variables by redeploying it with the new values.

1. Go to Apps page -> search for "Context Chat Backend"
2. Disable and remove the app taking care the data is not removed (except when the embedding model is changed, in which case the data should be removed)
3. Set the "Deploy Options" with the new environment variables
4. Reinstall the app

Kubernetes

Starting with version 5.4.0 of the app and Nextcloud 34, Kubernetes is supported for deployment of the backend to scale ContextChat to large instances. Nextcloud customers can find details about Kubernetes deployment in the customer documentation or by contacting support.

19.9.5 App store

You can also find the `context_chat` app in our app store, where you can write a review: https://apps.nextcloud.com/apps/context_chat

19.9.6 Repository

You can find the app's code repository on GitHub where you can report bugs and contribute fixes and features: https://github.com/nextcloud/context_chat and https://github.com/nextcloud/context_chat_backend

Nextcloud customers should file bugs directly with our Customer Support.

19.9.7 Commands (OCC)

The options for each command can be found like this, using scan as example: `context_chat:scan --help`

- **`context_chat:prompt`**
  Ask a question about your data, with options for selective context.
- **`context_chat:search`**
  Perform a semantic (vector DB based) search on your indexed documents, with options for selective context.
- **`context_chat:stats`**

  Shows the time taken to complete the initial indexing of the documents if it has finished,
  and the current no. of items in the indexer and actions queue.
  "Actions" refers to tasks like file deletions, ownership changes through share changes, etc.
  These file and ownership changes are synced with the backed through this actions queue.
- **`context_chat:reindex`**

  Schedule a full re-crawl of all the files in all the mounts. Indexed files are not re-indexed when compared against context_chat_backend's vector DB.
  Content providers are not re-indexed.

19.9.8 Configuration Options

- **`auto_indexing` string (default: 'true')**
  To allow/disallow the IndexerJob from running in the background. Not required to be configured normally.

```
occ config:app:set context_chat auto_indexing --value='true' --type=string
```

Context chat backend's configuration options are available through the environment variable in *Deploy Options*.

19.9.9 Logs

Logs for both the `context_chat` PHP app and the `context_chat_backend` ExApp can be found in the admin settings of your Nextcloud GUI as well as in the Context Chat log file, which is usually located in the Nextcloud data directory. The log file is named `context_chat.log`.

The internal PostgreSQL Vector DB logs (in case of non-Kubernetes setups) can be found inside the docker container at `/nc_app_context_chat_backend_data/vector_db_data/pgsql/logfile`. It can be copied to host using `docker cp nc_app_context_chat_backend:/nc_app_context_chat_backend_data/vector_db_data/pgsql/logfile /tmp/vectordb-logfile`.
This might be needed when the automated setting up of the vector db fails with something like: "pg_ctl: could not start server".

When running in Kubernetes, the logs for the backend will be in the respective pod's logs, which can be accessed using `kubectl logs <pod_name>` command.

19.9.10 Troubleshooting

1. If the docker container seems to suddenly restart during indexing or querying, it could be related to RAM/storage filling up, or AVX being unavailable on the system. AVX can be checked using `grep -i avx /proc/cpuinfo` command on the host system. If AVX is not available, the app will not work.
2. Look for issues in the diagnostic logs, the server logs and the docker container `nc_app_context_chat_container` logs. If unsure, open an issue in either of the repositories.
3. Check "Admin settings -> Context Chat" for statistics and information about the indexing process.

19.9.11 File access control rules not supported

In Nextcloud you can set up file access control rules using the files_accesscontrol app to restrict access to certain files.

Context Chat does **not** follow these rules.

It is thus possible for users who have been denied access to a document via the files_accesscontrol app to still gain access via Context Chat if the document is visible in the files app for the user in question.

19.9.12 Known Limitations

- Language models are likely to generate false information and should thus only be used in situations that are not critical. It's recommended to only use AI at the beginning of a creation process and not at the end, so that outputs of AI serve as a draft for example and not as final product. Always check the output of language models before using it and make sure whether it meets your use-case's quality requirements.
- Customer support is available upon request, however we can't solve false or problematic output, most performance issues, or other problems caused by the underlying model. Support is thus limited only to bugs directly caused by the implementation of the app (connectors, API, front-end, AppAPI).

- Files larger than 100MB are not supported
- Password protected PDFs or any other files are not supported. There will be error logs mentioning cryptography and AES in the docker container when such files are encountered but it is nothing to worry about, they will be simply ignored and the system will continue to function normally.
- External storages (through `files_external`) may not work as well as the local storage.
- Multi-tenant installation is not supported. You can only run one context chat instance connected to one Nextcloud instance (see *the AppAPI FAQ*).

19.10 App: Context Agent (context_agent)

The *context_agent* app is the app that provides AI agent functionality in Nextcloud's "Chat with AI" feature and acts as a backend for the *Nextcloud Assistant app*. Nextcloud can provide customer support upon request, please talk to your account manager for the possibilities.

When the Context Agent app is installed the AI Chat in Nextcloud Assistant will be able to interact with your Nextcloud apps via virtual integrations that are called "tools". They allow the Assistant to perform actions in your Nextcloud upon sending instructions in a chat message. Tool groups are only available if their requirements are met. They can be en- and disabled in the AI admin settings.

Additionally, the app makes all tools available via an MCP server that you can connect third-party agents to (see below).

19.10.1 Currently implemented tools

Artificial intelligence tools

- Ask a question to context chat (requires *Context Chat*)
  - Example prompt: *"What is the company's sick leave process?"*
- Transcribe a media file (requires Transcribe audio task type enabled)
  - Example prompt: *"Can you transcribe the following file? https://mycloud.com/f/9825679"* (Can be selected via smart picker.)
- Generate documents (requires Nextcloud Office)
  - Example prompt: *"Can you generate me a slide deck for my presentation about cats?"*
  - Example prompt: *"Can you generate me a spreadsheet with some plausible numbers for countries and their population count?"*
  - Example prompt: *"Can you generate me a pdf with an outline about what to see in Berlin?"*
- Generate images (requires Image generation task type enabled)
  - Example prompt: *"Can you generate me an image of a cartoon drawing of a roman soldier typing something on a laptop?"*

Calendar tools

- List the user's calendars
  - Example prompt: *"List my calendars"*
- Schedule an event in the user's calendar
  - Example prompt: *"Schedule an event with Andrew tomorrow at noon."*
- Find free times in users' calendar

  - Example prompt: *"Find a free 1-hour slot for a meeting with me and Marco next week."*

Tasks tools

- Create a task
  - Example prompt: *"Create a task for grocery shopping with due date tomorrow."*
- List tasks
  - Example prompt: *"List my outstanding tasks"*
- Complete a task
  - Example prompt: *"Mark the grocery shopping task as completed."*
- Update a task's details
  - Example prompt: *"Change the priority of the grocery shopping task to the highest possible priority."*
  - Example prompt: *"Change the due date of my work report task to the beginning of next week."*
- Delete a task
  - Example prompt: *"Delete the grocery shopping task in my tasks."*

Circles/teams tools

- List circles
  - Example prompt: *"List all my teams."*
- List circle members
  - Example prompt: *"List all members of my Content Marketing team."*
- Create new circle
  - Example prompt: *"Create a new team called 'Hiking group'."*
- Add members to circles
  - Example prompt: *"Add Ralph to the Hiking group team."*
- Remove members from circles
  - Example prompt: *"Remove ralph from the Hiking group team."*
- Change circle details
  - Example prompt: *"Change the name of the Hiking group team to 'Outdoor group'."*
  - Example prompt: *"Add the following description to the Hiking group team: We go hiking together once a month. Come join us."*
- Delete a circle
  - Example prompt: *"Delete the Hiking group team."*
- Share a file with a circle
  - Example prompt: *"Share my Hiking plans.md file with the Hiking group team."*

Collectives tools (require Collectives)

- List the user's collectives
  - Example prompt: *"List my collectives."*
- List the pages in a collective
  - Example prompt: *"List the pages in my Engineering collective."*
- Get metadata about a page in a collective
  - Example prompt: *"Show me the details of the 'Onboarding' page in my Engineering collective."*
- Get the markdown content of a page
  - Example prompt: *"Fetch the content of the 'Onboarding' page in my Engineering collective."*
- List trashed pages in a collective
  - Example prompt: *"List the trashed pages in my Engineering collective."*
- Create a new page in a collective under a given parent
  - Example prompt: *"Create a new 'Q3 retro' page in my Engineering collective under the 'Retros' page."*
- Replace the content of a page (a note saying the page was edited with the help of the AI Assistant is appended automatically)
  - Example prompt: *"Update the 'Q3 retro' page in my Engineering collective with the meeting notes."*
- Rename a page
  - Example prompt: *"Rename the 'Q3 retro' page in my Engineering collective to 'Q3 2026 retro'."*
- Move a page to a different parent
  - Example prompt: *"Move the 'Q3 2026 retro' page under the 'Archive' page in my Engineering collective."*
- Set the emoji icon of a page
  - Example prompt: *"Set the emoji of the 'Onboarding' page in my Engineering collective to .."*
- Move a page to the collective's trash
  - Example prompt: *"Trash the 'Old draft' page in my Engineering collective."*
- Restore a page from the collective's trash
  - Example prompt: *"Restore the 'Old draft' page from the trash of my Engineering collective."*
- Permanently delete a page
  - Example prompt: *"Permanently delete the 'Old draft' page in my Engineering collective."*

Contacts tools

- Find a contact
  - Example prompt: *"What is Anna's email address?"*
- Find a user's ID
  - Example prompt: *"What is Ralph's userID?"*
- Find the current user's details
  - Example prompt: *"Where do I live?"*

Cookbook tools

- List recipes
  - Example prompt: *"List my recipes."*
- Search for recipes
  - Example prompt: *"Do I have any Spaghetti recipes?"*
- Get recipe details
  - Example prompt: *"Can you give me the details of my Spaghetti Carbonara recipe?"*
- Create a new recipe
  - Example prompt: *"Create a recipe for Guacamole in my cookbook."*
- Delete a recipe
  - Example prompt: *"Remove the Guacamole recipe from my cookbook."*
- List recipe categories
  - Example prompt: *"Which recipe categories do I have in my cookbook?"*

Deck tools (require Deck)

- List deck boards
  - Example prompt: *"List the deck boards I have access to."*
- List the cards on a deck board
  - Example prompt: *"List the cards on my Personal deck board."*
- Add a new card
  - Example prompt: *"Can you add a card with title 'Repair kitchen sink' to my Personal deck board?"*
- Add a label to a card
  - Example prompt: *"Can you add the label 'Urgent' to the 'repair kitchen sink' card in my personal deck board?"*
- Assign a card to a user
  - Example prompt: *"Can you assign the 'Repair kitchen sink' card in my Personal deck board to Andrew?"*
- Delete a card
  - Example prompt: *"Delete the 'Repair kitchen sink' card in my Personal deck board."*
- List the comments on a deck card
  - Example prompt: *"Show the comments on the 'Repair kitchen sink' card in my Personal deck board."*
- Add a comment to a deck card
  - Example prompt: *"Add a comment 'I'll handle this Friday' to 'Repair kitchen sink' in my Personal deck board."*
- Edit a comment on a deck card
  - Example prompt: *"Update my last comment on the 'Repair kitchen sink' card to say 'Moved to Saturday'."*
- Delete a comment on a deck card
  - Example prompt: *"Delete my last comment on the 'Repair kitchen sink' card in my Personal deck board."*

Files tools

- Get contents of a file
  - Example prompt: *"Can you fetch the following file in my documents? Design/Planning.md"*
  - Example prompt: *"Can you fetch the following file in my documents? https://mycloud.com/f/98543234"*
- Retrieve folder tree
  - Example prompt: *"List my files."*
- Create a public link for a file or folder
  - Example prompt: *"Create a public link for the following file: Design/Planning.md"*
- Create a new file
  - Example prompt: *"Create a new file Ideas.md in my files and fill it with ideas for hiking destinations in the black forest."*
- Create a new folder
  - Example prompt: *"Create a new folder 'Hiking plans' in my files."*
- Move a file
  - Example prompt: *"Move the Ideas.md file into the Hiking plans folder."*
- Copy a file
  - Example prompt: *"Copy the Ideas.md file into my Notes folder."*
- Delete a file
  - Example prompt: *"Delete the Ideas.md file."*

Forms tools

- List all forms
  - Example prompt: *"List all the forms I have access to."*
- Get details of a form
  - Example prompt: *"Can you give me all details about the Retreat signup form?"*
- Add a question to a form
  - Example prompt: *"Add the following question to the retreat signup form: 'Number of days attending'."*
- Retrieve all responses of a form
  - Example prompt: *"List all responses to the Retreat signup form."*
- Update form settings
  - Example prompt: *"Make the Retreat signup form expire end of next week."*
- Delete a form
  - Example prompt: *"Delete the Retreat signup form."*

Bookmarks tools

- List all bookmarks
  - Example prompt: *"List all my bookmarks."*
- Add a bookmark
  - Example prompt: *"Add a bookmark for https://nextcloud.com with title 'Nextcloud homepage'."*
- Delete a bookmark
  - Example prompt: *"Delete the bookmark for https://nextcloud.com."*
- Update a bookmark
  - Example prompt: *"Change the title of the bookmark for https://nextcloud.com to 'Nextcloud official homepage'."*
  - Example prompt: *"Add the tag 'cloud' to the bookmark for https://nextcloud.com."*
  - Example prompt: *"Remove the tag 'cloud' from the bookmark for https://nextcloud.com."*
  - Example prompt: *"Put the bookmark for https://nextcloud.com into the 'work' folder."*
- List bookmark folders
  - Example prompt: *"Which bookmark folders do I have?"*
- Create a bookmark folder
  - Example prompt: *"Create a bookmark folder called 'work'."*
- List bookmark tags
  - Example prompt: *"Which bookmark tags do I have?"*

Search tools

All search providers in Nextcloud are also automatically available as tools.

- Search for files
  - Example prompt: *"List all the powerpoint presentations in my files with file ending pptx."*

Share tools

- List shares
  - Example prompt: *"List all files that were shared with me."*
  - Example prompt: *"List the shares of the Design/Ideas.md file."*
- Share a file or folder with a user
  - Example prompt: *"Share the Design/Ideas.md file with the user martin."*
- Share a file or folder with a group
  - Example prompt: *"Share the Design/Ideas.md file with the group Designers."*
- Update a share's permissions
  - Example prompt: *"Only allow martin read only access on the share of the Design/Ideas.md file."*
- Delete a share
  - Example prompt: *"Remove the share of the Design/Ideas.md file with martin."*
- List user groups

  - Example prompt: *"Which user groups are there?"*
- Retrieve share details
- Example prompt: *"Does martin have write access to the Design/Ideas.md file I shared with him?"*

Talk tools (require Talk)

- List the user's talk conversations
  - Example prompt: *"List my talk conversations"*
- List messages in a talk conversation
  - Example prompt: *"List the latest messages in my conversation with Andrew"*
- Send a message to a talk conversation
  - Example prompt: *"Can you send a joke to Andrew in talk?"*
- Create a public talk conversation
  - Example prompt: *"Can you create a new public talk conversation titled 'Press conference'?"*
- Reply to a specific message in a talk conversation
  - Example prompt: *"Reply to Andrew's last message in our talk conversation with 'Got it, thanks!'"*
- Add an emoji reaction to a message in a talk conversation
  - Example prompt: *"React with 👍 to Andrew's last message in our talk conversation"*
- Remove an emoji reaction from a message in a talk conversation
  - Example prompt: *"Remove my 👍 reaction from Andrew's last message"*
- List the reactions on a message in a talk conversation
  - Example prompt: *"Who reacted to Andrew's last message in our talk conversation?"*
- Create a poll in a talk conversation
  - Example prompt: *"Create a poll in the 'Team standup' conversation asking 'Which day works for the offsite?' with options Monday, Tuesday, Wednesday"*
- Get the question, options, and current results of a poll in a talk conversation
  - Example prompt: *"Show the current results of the offsite poll in the 'Team standup' conversation"*
- Cast a vote on a poll in a talk conversation
  - Example prompt: *"Vote for Tuesday on the offsite poll in the 'Team standup' conversation"*
- Close a poll in a talk conversation
  - Example prompt: *"Close the offsite poll in the 'Team standup' conversation"*
- Share a Nextcloud Files item to a talk conversation
  - Example prompt: *"Share the file 'Q3 plan.pdf' to my conversation with Andrew"*
- List items of a given type (e.g. file, location, poll) that were shared in a talk conversation
  - Example prompt: *"List the files shared in my conversation with Andrew"*
- Get an overview of items shared in a talk conversation across all types
  - Example prompt: *"Give me an overview of what's been shared in my conversation with Andrew"*

Mail tools (require Mail)

- Send an email via Nextcloud Mail
  - Example prompt: *"Send a test email from my carry@company.com account to Andrew@company.com"*
- List all connected mail accounts
  - Example prompt: *"List my mail accounts"*
- List all mail folders of an email account
  - Example prompt: *"List the folders of my carry@company.com account"*
- List the mails in a mail folder
  - Example prompt: *"List the last 5 mails in the inbox of my carry@company.com account"*

Miscellaneous tools

- Get coordinates for an Address from Open Street Maps Nomatim
  - Example prompt: *"What are the coordinates for Berlin, Germany?"*
- Get the URL for a map of a location using Open Street Maps
  - Example prompt: *"Can you show me a map of New York, please"*
- Get the current weather at a location
  - Example prompt: *"How is the weather in Berlin?"*
- Search for youtube videos
  - Example prompt: *"Show me the youtube video of the Nextcloud hub 10 launch."*
- Search Duckduckgo
  - Example prompt: *"Show me search results for quick pasta recipes, please."*
- Determine public transport routes (requires a HERE API key configured in the admin settings)
  - Example prompt: *"How can I get from Würzburg Hauptbahnhof to Berlin Hauptbahnhof?"*
- List all projects in OpenProject (requires the OpenProject integration)
  - Example prompt: *"List all my projects in OpenProject, please"*
- List all available assignees of a project in OpenProject (requires the OpenProject integration)
  - Example prompt: *"List all available assignees for the 'Product launch' project in OpenProject"*
- Create a new work package in a given project in OpenProject (requires the OpenProject integration) * Example prompt: *"Create a work package called 'Publish release video' in the 'Product launch' project in OpenProject"*

19.10.2 Combining tools

These tools can also be combined by the agent to fulfil tasks like the following:

- *"How is the weather where Andrew lives?"*
  - Uses contacts to look up Andrew's address and then checks the weather
- *"How is the weather where I live?"*
  - Look up the current user's address and then checks the weather
- *"Send an email from carry@company.com to Andrew"*

  - Uses contacts to look up Andrew's email and then sends an email
- *"Which of my files are from Anna?"*
  - Looks up Anna's userID and searches for files that belong to her
- *"Send the content of my draft.md file to Andrew in Talk"*
  - Gets the content of the file and sends it in a 1-1 Talk conversation with Andrew

19.10.3 Custom Tools using MCP

Model Context Protocol (MCP) is a protocol that enables Large Language Models (LLMs) to interact with external data sources and tools. The Context Agent app allows administrators to extend its capabilities by adding custom services via MCP. This can be configured in the admin settings under "MCP Config," where you can provide a JSON configuration in the following format:

```
{
  "service-name": {
    "url": "https://service-url.com/endpoint",
    "transport": "streamable_http"
  }
}
```

19.10.4 Requirements

- This app is built as an External App and thus depends on AppAPI v3.1.0 or higher
- Nextcloud AIO is supported
- No GPU is necessary for Context Agent but one might be useful if you use it with a self-hosted provider like llm2
- CPU Sizing
  - At least 1GB of system RAM

19.10.5 Installation

0. Make sure the *Nextcloud Assistant app* is installed
1. *Install AppAPI and setup a Deploy Demon*
2. Install the "Context Agent" ExApp via the "Apps" page in the Nextcloud web admin user interface
3. If you want to use the agent in the Nextcloud UI, install a text generation backend like *llm2* or *integration_openai* via the "Apps" page in Nextcloud; if you only want to use the MCP server you don't need these.

Model requirements

This app requires underlying Large language models to support tool calling. The default model in *llm2* does support tool calling since version 2.4.0. Other models that may give good results are:

- Google Gemma 3 12B or higher
- Mistral 3 small 24B
- Qwen 2.5 8B or higher (May not work well with languages other than English)

See *llm2 documentation* on how to configure alternate models.

19.10.6 Using Nextcloud MCP Server

Context Agent exposes an MCP server that can be used by other large language models or applications to access the tools provided by Context Agent. The server will be available at *https://your-nextcloud-domain.com/index.php/apps/app_api/proxy/context_agent/mcp/*, and it requires authentication via an app password passed in the *Authorization* header. Ex: *Authorization: Bearer <app-password>*.

19.10.7 Scaling

It is currently not possible to scale this app, we are working on this.

19.10.8 App store

You can also find the app in our app store, where you can write a review: https://apps.nextcloud.com/apps/context_agent

19.10.9 Repository

You can find the app's code repository on GitHub where you can report bugs and contribute fixes and features: https://github.com/nextcloud/context_agent

Nextcloud customers should file bugs directly with our Support system.

19.10.10 Known Limitations

- Make sure to test the language model you are using in concert with this app for whether they meet the use-case's quality requirements
- Most models have difficulties with languages other than English. Some models sometimes answer in a different language than the one used by the user.
- Customer support is available upon request, however we can't solve false or problematic output, most performance issues, or other problems caused by the underlying model. Support is thus limited only to bugs directly caused by the implementation of the app (connectors, API, front-end, AppAPI). We still try to optimize this as far as possible, so if you encounter any false or problematic output, you can report it in a dedicated Github issue to help us improve this app.
- When multiple MCP services are configured that have tools with the same name undefined behavior will occur.
- Only remote MCP services are supported (streamable_http transport).
- MCP services that require different access tokens for each user are not currently supported.

19.11 App: Summary Bot (Talk chat summarize bot)

The *Summary Bot* app utilizes Large Language Model (LLM) providers in Nextcloud and can be added to a conversation in *Nextcloud Talk* to generate summaries from the chat messages of that room either on-demand or following a schedule. It can run on only open source or proprietary models either on-premises or in the cloud leveraging apps like Local large language model app or OpenAI and LocalAI integration app.

Nextcloud can provide customer support upon request, please talk to your account manager for the possibilities.

The app currently supports the following languages:

- English (en)

The quality of summaries depends directly on the quality of the underlying model. It is recommended to test the model for the desired use-case before applying it.

19.11.1 Requirements

- Minimal Nextcloud version: 30
- Docker
- AppAPI >= 3.0.0
- Talk
- Task Processing Provider like Local large language model app (llm2) or OpenAI and LocalAI integration app (integration_openai)

Space usage

- ~100MB

19.11.2 Installation

0. Make sure the following apps are installed:
    - Nextcloud AppAPI app
    - Nextcloud Talk app (Spreed)
    - One of the following AI model providers:
        - Nextcloud Local large language model app
        - Nextcloud OpenAI and LocalAI integration app
        - Nextcloud IBM watsonx.ai integration app

Setup (via App Store)

1. Install the *Summary Bot* app via the "Apps" page in Nextcloud
2. Enable the *Summary Bot* Bot for the selected Chatroom via the three dots menu of the Chatroom (The Bots settings are located inside the *Bots* section)

Setup (Manual)

After cloning this app *manually* (cloned via git to your apps directory) you will need to execute the following steps:

1. Change to the folder you have cloned the source to: .. code-block:

```
cd  /path/to/your/nextcloud/webroot/apps/summary_bot/
```

2. Build the docker image: .. code-block:

```
docker build --no-cache -f Dockerfile -t local_summary_bot .
```

3. Run the docker image:

Info:

- APP_VERSION environment variable should be equal to the version of the *Summary Bot* you are using
- NEXTCLOUD_URL environment variable must be set to your Nextcloud instance's URL, ensuring it's reachable by the docker image.

```
sudo docker run -ti -v /etc/localtime:/etc/localtime:ro -v /etc/timezone:/etc/
↪timezone:ro -e APP_ID=summary_bot -e APP_DISPLAY_NAME="Summary Bot" -e APP_HOST=0.0.
↪0.0 -e APP_PORT=9031 -e APP_SECRET=12345 -e APP_VERSION=1.0.0 -e NEXTCLOUD_URL='
↪<YOUR_NEXTCLOUD_URL_REACHABLE_FROM_INSIDE_DOCKER>' -p 9031:9031 local_summary_bot
```

4. Un-register the Summary Bot if its already installed

```
sudo -E -u www-data php occ app_api:app:unregister summary_bot
```

5. Register the Summary Bot so that your Nextcloud instance is aware of it

Info: Adjust the host value in the following example to the IP address of the docker container (for added security)

```
sudo -E -u www-data php occ app_api:app:register summary_bot manual_install --json-
↪info '{ "id": "summary_bot", "name": "Summary Bot", "daemon_config_name": "manual_
↪install", "version": "1.0.0", "secret": "12345", "host": "0.0.0.0", "port": 9031,
↪"scopes": ["AI_PROVIDERS", "TALK", "TALK_BOT"], "protocol": "http"}' --force-scopes␣
↪--wait-finish
```

6. Enable the *Summary Bot* for the selected Chatroom via the three dots menu of the Chatroom (The Bots settings are located inside the *Bots* section)

19.11.3 Usage

After enabling the *Summary Bot* in a Chatroom, you can test its functionality by simply sending the message below:

"@summary" or "@summary help"

19.11.4 App store

You can also find the app in our app store, where you can write a review: https://apps.nextcloud.com/apps/summary_bot

19.11.5 Repository

You can find the app's code repository on GitHub where you can report bugs and contribute fixes and features: https://github.com/nextcloud/summary_bot

Nextcloud customers should file bugs directly with our Customer Support.

19.11.6 Ethical AI Rating

The ethical rating of the *Summary Bot*, which utilizes a model for text processing through the Nextcloud Assistant app, is significantly influenced by the choice and implementation of the underlying model.

Learn more about the Nextcloud Ethical AI Rating in our blog.

19.11.7 Known Limitations

- The Summary Bot cannot access previous conversations, it only recognizes messages from the moment it was enabled in the chatroom.
- Summary of maximum 40000 characters is supported. This assumes the underlying model can handle this amount of text (which should be close to 16000 context length).
- Languages other than English are not supported. The underlying model may still be able to understand other languages.

- AI models may occasionally produce inaccurate information. Therefore, they should be employed with caution in non-critical scenarios. It's essential to verify the accuracy of the bot's output before application.
- Be aware that AI models can consume a significant amount of energy. It's advisable to consider this factor in the planning and operation of AI systems if hosted on-premises or sustainability is a concern.
- AI models can exhibit extended processing times when run on CPUs. For enhanced efficiency, utilizing GPU support is recommended to expedite request handling.
- Customer support is available upon request, however we can't solve false or problematic output (hallucinations), most performance issues, or other problems caused by the underlying models. Support is thus limited only to bugs directly caused by the implementation of the app (connectors, API, front-end, AppAPI)

19.12 App: Local Text-To-Speech (text2speech_kokoro)

The *text2speech_kokoro* app is one of the apps that provide Text-To-Speech functionality in Nextcloud and act as a speech generation backend for the *Nextcloud Assistant app* and *other apps making use of the core `Text-To-Speech Task type*. The *text2speech_kokoro* app specifically runs only open source models and does so entirely on-premises. Nextcloud can provide customer support upon request, please talk to your account manager for the possibilities.

This app uses Kokoro under the hood.

The used model supports the following languages:

- American English
- British English
- Spanish
- French
- Italian
- Hindi
- Portuguese
- Japanese
- Mandarin

19.12.1 Requirements

- Minimal Nextcloud version: 31
- This app is built as an External App and thus depends on AppAPI v2.3.0
- Nextcloud AIO is supported
- We currently support x86_64 CPUs
- We do not support GPUs
- CPU Sizing
  - The more cores you have and the more powerful the CPU the better, we recommend around 10 cores
  - The app will hog all cores by default, so it is usually better to run it on a separate machine
  - 800MB RAM

19.12.2 Installation

0. Make sure the *Nextcloud Assistant app* is installed
1. *Install AppAPI and setup a Deploy Demon*
2. Install the *text2speech_kokoro* "Local Text-To-Speech" ExApp via the "Apps" page in the Nextcloud web admin user interface

19.12.3 Scaling

It is currently not possible to scale this app, we are working on this. Based on our calculations an instance has a rough capacity of 4h of transcription throughput per minute (measured with 8 CPU threads on an Intel(R) Xeon(R) Gold 6226R). It is unclear how close to real-world usage this number is, so we do appreciate real-world feedback on this.

19.12.4 App store

You can also find this app in our app store, where you can write a review: https://apps.nextcloud.com/apps/text2speech_kokoro

19.12.5 Repository

You can find the app's code repository on GitHub where you can report bugs and contribute fixes and features: https://github.com/nextcloud/text2speech_kokoro

Nextcloud customers should file bugs directly with our customer support.

19.12.6 Known Limitations

- We currently only support languages supported by the underlying Kokoro model
- The Kokoro models perform unevenly across languages, and may show lower accuracy on low-resource and/or low-discoverability languages or languages where there was less training data available.
- Make sure to test the language model you are using it for whether it meets the use-case's quality requirements
- Customer support is available upon request, however we can't solve false or problematic output, most performance issues, or other problems caused by the underlying model. Support is thus limited only to bugs directly caused by the implementation of the app (connectors, API, front-end, AppAPI)

19.13 App: Live Transcription and Translation in Nextcloud Talk (live_transcription)

This app provides live transcription and translation of speech in Nextcloud Talk calls using open source AI models provided by Vosk.
The transcription is done on your own server, preserving your privacy and data sovereignty, while the translation is done using a translation task processing provider like the *translate2 app*. OpenAI and LocalAI integration and DeepL integration apps will soon also be supported for translation.

A good set of language models for transcription are auto-downloaded. They include Arabic, Arabic (Tunisian), Breton, Catalan, Czech, German, English, Esperanto, Spanish, Persian (Farsi), French, Hindi, Italian, Japanese, Kazakh, Korean, Dutch, Polish, Portuguese (Brazilian), Russian, Telegu, Tajik, Turkish, Ukrainian, Uzbek, Vietnamese and Chinese.
The translation capabilities depend on the installed translation task processing provider app. A list of translation-capable apps can be found *here* in the "Backend apps" section.

19.13.1 Installation

1. Make sure the Nextcloud Talk app is installed.
2. Make sure the High-Performance Backend (latest or released after September 2025) is installed and configured in Nextcloud Talk settings. See the Nextcloud Talk install manual for more information.
3. Setup a *Deploy Daemon* in AppAPI Admin settings.
4. Install the **live_transcription** app via the "Apps" page in Nextcloud, or by executing

```
occ app_api:app:register live_transcription \
  --env LT_HPB_URL=wss://cloud.example.com/standalone-signaling/spreed \
  --env LT_INTERNAL_SECRET=1234 \
  --wait-finish
```

Important

The environment variables `LT_HPB_URL` and `LT_INTERNAL_SECRET` must be set in the *Deploy Options* during installation, and the High-Performance Backend must be functionally configured in Nextcloud Talk settings for the app to work.

Changing these environment variables after installation is possible through a re-installation of the app after uninstalling it first.

5. Install a Text-to-text task processing provider app for translation capabilities from the "Backend apps" section *here*.

19.13.2 Requirements

- Minimal Nextcloud version: 33
- Nextcloud AIO is supported
- We currently support NVIDIA GPUs and x86_64 CPUs. Only CPU-based transcription is also supported and works well on modern x86 CPUs.
- CUDA >= v12.4.1 on your host system for GPU-based transcription
- GPU Sizing
  - A NVIDIA GPU with at least 10 GB VRAM
  - 16 GB of system RAM should be enough for one or two concurrent calls
- CPU Sizing
  - x86 CPU with 4 threads. Additional 2 threads per concurrent call.
  - 16 GB of RAM should be enough for one or two concurrent calls
- **Space usage**
  - ~ 2.8 GB for the docker container
  - ~ 6.0 GB for the default models

Note

We currently have very little real-world experience running this software on production instances. The above sizing recommendations come from our estimates and are not real-world benchmarks. Actual requirements will vary based

on factors such as the number of concurrent calls, audio quality, and selected languages. Please do thorough testing to confirm your hardware meets your needs.

19.13.3 Logs

Warnings and errors are logged to the main Nextcloud server logs.
To inspect the detailed JSON logs for lower levels, the following commands can help.

To view docker stdout/err logs:

```
docker logs -f -n400 nc_app_live_transcription
```

To view the JSON logs:
The logs are rotated when file size exceeds 20 MiB. The older log files are named `lt.log.1`, `lt.log.2`, and so on.

```
docker exec -it nc_app_live_transcription tail -f -n400 /nc_app_live_transcription_
↪data/logs/lt.log
```

To download the JSON logs:

```
docker cp nc_app_live_transcription:/nc_app_live_transcription_data/logs/ /tmp/nc_app_
↪live_transcription_logs
```

19.13.4 App store

You can also find the app in our app store, where you can write a review: https://apps.nextcloud.com/apps/live_transcription

19.13.5 Repository

You can find the app's code repository on GitHub where you can report bugs and contribute fixes and features: https://github.com/nextcloud/live_transcription

Nextcloud customers should file bugs directly with our Customer Support.

19.13.6 Limitations

- The generated transcripts may not be perfect and may contain errors. It can also depend on the audio quality and the speaker's accent.
- The app currently supports only a limited number of languages. More languages may be added in the future.
- The languages other than English may have lower accuracy mainly due to the shipped models being smaller.
- The app currently does not support punctuation in the transcription.
- OpenAI and LocalAI integration and DeepL integration apps are not yet supported for translation.

19.14 App: Local Optical Character Recognition (ocr_paddle)

The *ocr_paddle* app is one of the apps that provide Optical Character Recognition functionality in Nextcloud and act as an OCR backend for the *Nextcloud Assistant app* and *other apps making use of the core OpticalCharacterRecognition Task type*. The *ocr_paddle* app specifically runs only open source models and does so entirely on-premises. Nextcloud can provide customer support upon request, please talk to your account manager for the possibilities.

This app uses PaddleOCR_VL under the hood.

The used model supports 109 languages, covering major global languages, including but not limited to Chinese, English, Japanese, Latin, and Korean, as well as languages with different scripts and structures, such as Russian (Cyrillic script), Arabic, Hindi (Devanagari script), and Thai.

PDFs are supported; multi-page PDFs come back as one document.

19.14.1 Requirements

- Minimal Nextcloud version: 35
- This app is built as an External App and thus depends on at least AppAPI v3.1.0
- Nextcloud AIO is supported
- We currently support x86_64 & arm64 CPUs as well as NVIDIA GPUs
- CPU Sizing
  - If you don't have a GPU, this app will utilize your CPU cores
  - The more cores you have and the more powerful the CPU the better, we recommend 10-20 cores, but the runtime will still be very slow on CPU only
  - The app will hog all cores by default, so it is usually better to run it on a separate machine
  - 7.5GB of free system RAM
- GPU Sizing
  - NVIDIA GPU
  - Minimum of 6GB free VRAM

19.14.2 Installation

0. Make sure the *Nextcloud Assistant app* is installed
1. *Install AppAPI and setup a Deploy Daemon*
2. Install the *ocr_paddle* "Local OCR: PaddleOCR" ExApp via the "Apps" page in the Nextcloud web admin user interface

19.14.3 Scaling

It is currently not possible to scale this app, we are working on this. If you have real world data on how many files this app can process per minute, we will gladly list it here.

19.14.4 App store

You can also find this app in our app store, where you can write a review: https://apps.nextcloud.com/apps/ocr_paddle

19.14.5 Repository

You can find the app's code repository on GitHub where you can report bugs and contribute fixes and features: https://github.com/nextcloud/ocr_paddle

Nextcloud customers should file bugs directly with our customer support.

19.14.6 Known Limitations

- We currently only support languages supported by the underlying PaddleOCR-VL model
- The PaddleOCR-VL model performs unevenly across languages, and may show lower accuracy on low-resource and/or low-discoverability languages or languages where there was less training data available.
- Only the first 50 pages of PDF files are read (configurable via the `OCR_MAX_PDF_PAGES` deploy option), longer PDFs are silently truncated.
- PDF pages are rasterized at 170 DPI (configurable via the `OCR_PDF_DPI` deploy option) before OCR
- Inputs are downscaled to ≤2.8 MP (max_pixels: 2822400), so high-resolution scans lose detail before OCR runs.
- Make sure to test the language model you are using it for whether it meets the use-case's quality requirements
- Customer support is available upon request, however we can't solve false or problematic output, most performance issues, or other problems caused by the underlying model. Support is thus limited only to bugs directly caused by the implementation of the app (connectors, API, front-end, AppAPI)

19.15 AI as a Service

At Nextcloud, we focus on creating on-premise AI apps that run fully self-hosted on your own servers in order to preserve your privacy and data sovereignty. However, you can also offload these resource-heavy tasks to an "AI as a Service" provider offering API access in exchange for payment. Examples of such providers are OpenAI, with its ChatGPT APIs providing language model access among other APIs, as well as Replicate and IBM watsonx.

19.15.1 Installation

In order to use these providers you will need to install the respective app from the app store:

- `integration_openai`
- `integration_replicate`
- `integration_watsonx`

You can then add your account information, set rate limits, and set the providers live in the "Artificial intelligence" section of the administration settings.

Optionally (but recommended), setup background workers for faster pickup of tasks. See *the relevant section in AI Overview* for more information.

19.15.2 OpenAI integration

With this application, you can also connect to a self-hosted LocalAI or Ollama instance or to any service that implements an API similar enough to the OpenAI API, for example IONOS AI Model Hub, Plusserver, Groqcloud, MistralAI or Together AI.

Do note, however, that we test the Assistant tasks that this app implements only with OpenAI models and only against the OpenAI API, we thus cannot guarantee other models and APIs will work. Some APIs claiming to be compatible with OpenAI might not be fully compatible so we cannot guarantee that they will work with this app.

19.15.3 IBM watsonx.ai integration

With this application, you can also connect to a self-hosted cluster running the IBM watsonx.ai software.

Do note, however, that we test the Assistant tasks that this app implements only with the provided foundation models and only against IBM Cloud servers. We thus cannot guarantee that other models or server instances will work.

19.15.4 Improve performance

Prompts from these apps can have a delay of up to 5 minutes. This can be optimized and more information can be found in *the relevant section in AI Overview*.

19.16 Insight and debugging

In order to gain insights and debug AI tasks, there are a number of occ commands available in the *taskprocessing* namespace.

All commands listed here accept an *–output* parameter that can be set to *plain*, *json* or *json_pretty*.

Tasks are retained in the database for 6 months. Note, however, that there is no strict write-only guarantee for these logs. This means Nextcloud will not change the task logs in the database once the task has ended, but any administrator with database access has the power to change them.

19.16.1 Get a task by id

```
$ occ taskprocessing:task:get <task-id>
```

For example

```
$ occ taskprocessing:task:get --output=json_pretty 42
{
  "id": 1,
  "type": "core:text2text:chat",
  "lastUpdated": 1759739466,
  "status": "STATUS_SUCCESSFUL",
  "userId": "admin",
  "appId": "assistant:chatty-llm",
  "input": {
    "system_prompt": "This is a conversation in a specific language between the user␣
↪and you, Nextcloud Assistant. You are a kind, polite and helpful AI that helps the␣
↪user to the best of its abilities. If you do not understand something, you will ask␣
↪for clarification. Detect the language that the user is using. Make sure to use the␣
↪same language in your response. Do not mention the language explicitly.",
    "input": "What's the weather in Berlin today?",
    "history": []
  },
  "output": {
    "output": "I'm happy to help, but I'm a large language model, I don't have real-
↪time access to current weather conditions. However, I can suggest checking a␣
↪reliable weather website or app, such as AccuWeather or OpenWeatherMap, for the␣
↪most up-to-date information on the weather in Karlsruhe today. Would you like me to␣
↪help with anything else?"
  },
  "customId": "chatty-llm:1",
```

(continues on next page)

(continued from previous page)

```
  "completionExpectedAt": 1759739513,
  "progress": 1,
  "scheduledAt": 1759739453,
  "startedAt": 1759739455,
  "endedAt": 1759739466,
  "allowCleanup": true,
  "error_message": null
}
```

Each task has the following fields:

- *id* The internal ID of the task, also referenced in user-facing error messages in case something goes wrong
- *type* The type of the task
- *status* The current status of the task (can be either *"STATUS_CANCELLED"*, *"STATUS_FAILED"*, *"STATUS_SUCCESSFUL"*, *"STATUS_SCHEDULED"*, *"STATUS_RUNNING"*, or *"STATUS_UNKNOWN"*)
- *userId* The Id of the user who requested the task
- *lastUpdated* When the task was last updated
- *scheduledAt* When the task was scheduled/created
- *startedAt* When the task was started to be processed by the set task processing provider
- *completionExpectedAt* When the system expects/expected the task to be finished
- *endedAt* When the task finished be it successfully or unsuccessfully
- *appid* The ID of the app that scheduled the task
- *input* The values that were part of the task input
- *output* The values that were part of the task output
- *error_message* The error message in case the task failed

19.16.2 List and filter tasks

```
$ occ taskprocessing:task:list [options]

-u, --userIdFilter[=USERIDFILTER]        only get the tasks for one user ID
-t, --type[=TYPE]                        only get the tasks for one task type
    --appId[=APPID]                      only get the tasks for one app ID
    --customID[=CUSTOMID]                only get the tasks for one custom ID
-s, --status[=STATUS]                    only get the tests that have a specific␣
↪status
    --scheduledAfter[=SCHEDULEDAFTER]    only get the tasks that were scheduled after␣
↪a specific date (Unix timestamp)
    --endedBefore[=ENDEDBEFORE]          only get the tasks that ended before a␣
↪specific date (Unix timestamp)
```

For example

```
$ occ taskprocessing:task:list --output=json_pretty --status=3 --
↪scheduledAfter=1759740266 --endedBefore=1759743900
[
```

(continues on next page)

(continued from previous page)

```
  {
      ...
  }
]
```

19.17 Legal: Compliance with EU AI Act

19.17.1 Implementation of the transparency requirements

This section describes of how Nextcloud and its AI products implemented the transparency requirements.

- All functionality that outputs AI generated content which significantly altered the user's input has in the software UI a visual warning that the content is generated using AI and urge users to double check the correctness of any claims therein.
- Additionally, AI generated files like documents, images, and audio contain a note that it was generated by AI. We also add a machine-readable tag ("Generated using AI"). In file formats that support metadata, we add metadata with the same information ("Generated using Artificial Intelligence.").
- Agentic interactions with third parties always include a note that the interaction was generated by AI (e.g. E-Mails sent on behalf of the user, calendar events created on behalf of the user, etc.).
- Employees of Nextcloud GmbH have been instructed through the internal AI policy to inform audience when they send or publish content that were generated by AI or significantly altered by AI.
- All interactions of users with AI in Nextcloud are retained in the database for observability and transparency. Refer to *Insight and Debugging* for details on how to explore and examine these records.
- All changes to AI configuration will be logged in the *audit log*.

19.17.2 Reliability and robustness

This section describes the measures that ensure the software is reliable and robust.

- All merged code is reviewed by at least one extra employee who checks for potential issues.
- The design and implementation of larger and more risky new features are always discussed with multiple experts to ensure a robust implementation.
- Our software is entirely open-source and anyone can raise bugs they face and propose fixes to the bugs they face. The incoming bugs are regularly reviewed by our employees and prioritized accordingly. Bugs that affect the functionality of the software for many users and instances are prioritized high and the work to develop a fix is added to our roadmap. For the other bugs, anyone is welcome to propose a fix and our employees will review the code change.
- We offer an enterprise subscription for downstream providers with critical infrastructure. This will guarantee support and fixes for any issue they face within a SLA agreement.
- We regularly test our software using standard testing procedures, like static code analysis and integration tests where appropriate.
- We maintain several test instances to ensure the reliability and stability of our AI features. One instance is updated daily with the latest development versions and a selection of self-hosted AI features are tested here. Another instance is used to validate upcoming core releases prior to announcement, where we test a selection of AI features of which most depend on OpenAI as a backend. In addition, we operate an instance that resembles the production environment of a small to mid-sized company, where we perform end-to-end testing of selected AI features. This environment combines AI-as-a-service providers for text generation capabilities with self-hosted models for other capabilities, allowing us to verify real-world performance and usability.

- When a feature relies on large language models as its core component, we cannot guarantee complete reliability due to the unpredictable nature of an LLM, and have thus documented the limitations of said features and provide AI literacy training to our employees and customers. We select the LLM models we recommend to use based their results on industry standard benchmarks as well as on a custom suite of tests for multilingual usage and tool calling. In the user-facing UI we prompt users to always double check the AI generated content.

19.17.3 Interoperability

This section describes the measures that ensure the software is interoperable.

- We provide our AI features via an open API with publicly accessible OpenAPI specs which allows developers to build on top of our features.
- As our software is fully open-source, anyone can adjust the software to meet their needs. For example, anyone can adjust the core code, adjust the code of existing applications, or develop a custom application for Nextcloud.
- We implement integrations for the major model hosting providers and their protocols upon request of customers. We are interoperable with OpenAI and IBM watsonx. As Nextcloud is an open app ecosystem, anyone can develop an integration with a model hosting provider on their own.
- We implement the agent interoperability protocol MCP both as a client and as a server to allow users to connect the AI Agent software to existing services and connect existing AI Agents to our software.
- We implement a local model hosting mechanism that can be used to host GGUF models (most open weight models can be converted using an Open Source tool called llama.cpp).

19.17.4 Cybersecurity and physical security of the hardware

This section describes the measures how we guarantee cybersecurity and the physical security of the hardware.

- All merged code is reviewed by at least one employee who also checks on security.
- We have a security bounty program at HackerOne.
- Our software is self-hosted by our customers and downstream providers.
  - We provide our customers extra support to ensure the software is secure, such as long-term support and early security notifications.
  - We cannot guarantee the security of the hardware of our customers and downstream providers. We cannot guarantee the security of the hardware as we only deliver software. Hardware security is the responsibility of those who host the software.
  - For internal use of Nextcloud GmbH, we use a well-respected EU-based hosting provider for our hardware (Hetzner) and a well-respected EU-based AI service provider (Ionos) with whom we have a data processing agreement.

19.17.5 Additional requirements when using large AI models

Nextcloud's AI products are designed to be used with smaller AI models that can also run on-premise. Nextcloud's AI Act compliance efforts thus assume you are using models that were trained using less than 10^25 floating point operations. However, Nextcloud's AI products are designed (in accordance with the AI act) to be interoperable and therefore it is technically possible to use larger models. If you decide to use larger models, this qualifies as a significant modification of the system and additional legal requirements for systemic-risk General purpose AI systems apply. Please refer to a lawyer and to the EU AI Act.

CHAPTER

TWENTY

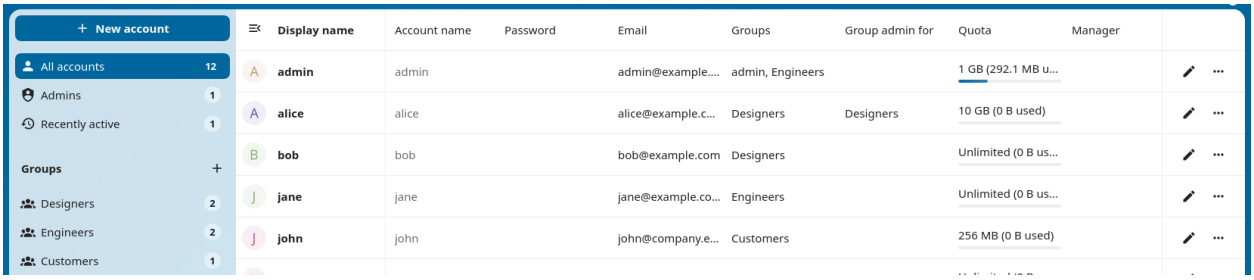
USER MANAGEMENT

20.1 User management

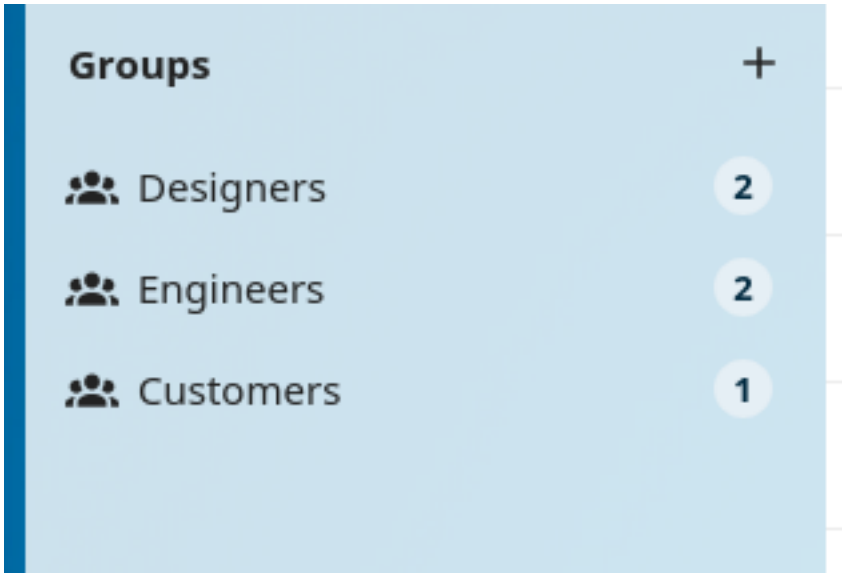
On the User management page of your Nextcloud Web UI you can:

- Create new users
- View all of your users in a single scrolling window
- Filter users by group
- See what groups they belong to
- Edit their full names and passwords
- See their data storage locations
- View and set quotas
- Create and edit their email addresses
- Send an automatic email notification to new users
- Disable and enable users
- Delete them with a single click

The default view displays basic information about your users.



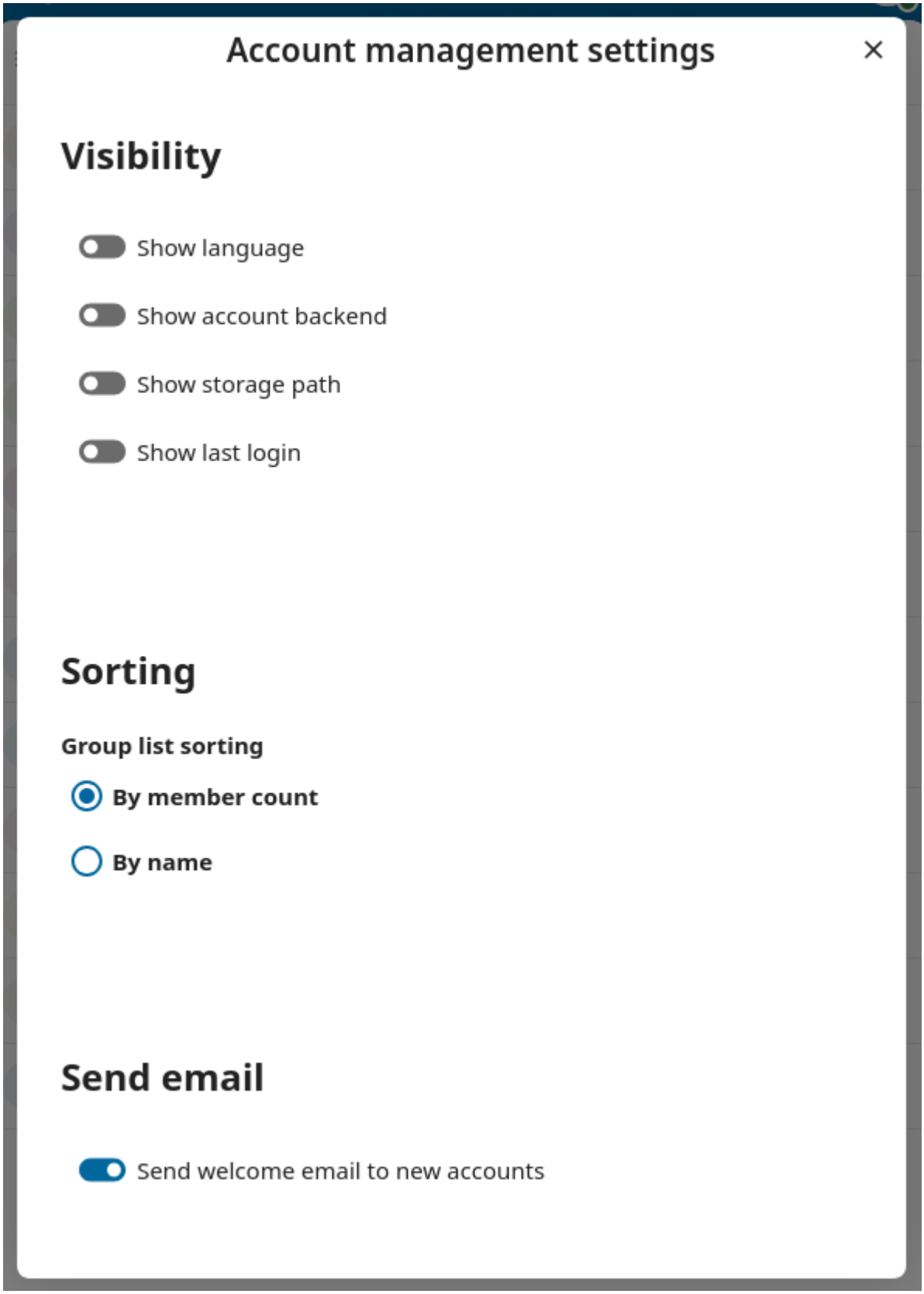
The group filters on the left sidebar let you quickly filter users by their group memberships and create new groups.



Note

User counts for certain groups, such as “All accounts,” may not be visible when using certain backends such as LDAP/AD/SAML.

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**.


Account management settings
Visibility
Show language
Show account backend
Show storage path
Show last login
Sorting
Group list sorting
By member count
By name
Send email
Send welcome email to new accounts

User accounts have the following properties:

Login Name (Username)
The unique ID of a Nextcloud user, which 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 at any time. If the Full Name is not set, it defaults to the login name.

Password
The admin sets the new user's initial password. Both the user and the admin can change the user's password at any time.

Email address
You can set an email address for a user. This address can be used when you first set up an account so the user receives an email asking them to create a password if none is provided. This address can also be used for password reset requests.

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 for specific groups and can create and remove users from their groups. This means they can modify the username, password, email, quota, etc., of members of the group. Group admins are not allowed to add existing users to their groups.

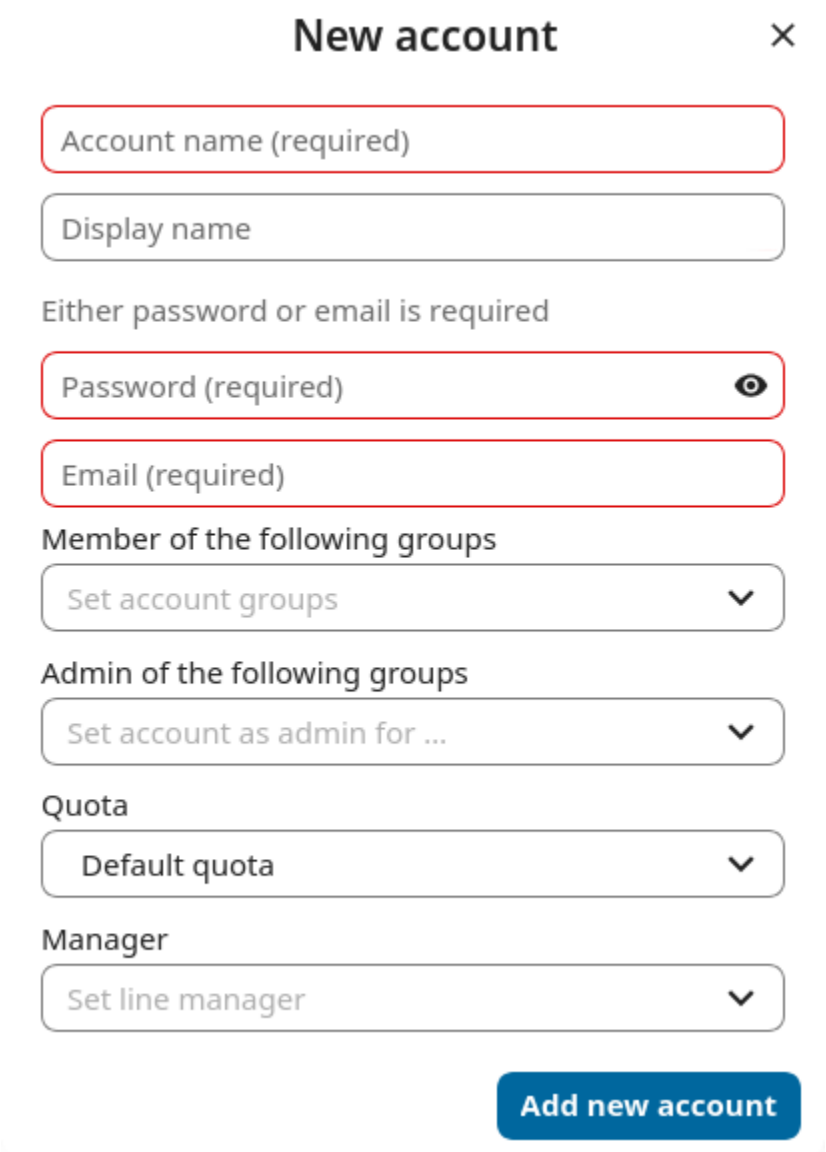
Quota
The maximum disk space assigned to each user. Any user who exceeds the quota cannot upload or sync data. You have the option to include external storage in user quotas.

Manager
Every user can have one organizational manager. The manager property goes into the system address book card of the user and is used for the Contacts app's organization chart, for example. Setting a manager does **not** change any authorization level of the user or their manager.

20.1.1 Creating a new user

To create a user account:

- Enter the new user's **Login Name** and their initial **Password**
- Optionally, assign **Groups** memberships
- Click the **Create** button



Login names may contain letters (a-z, A-Z), numbers (0-9), dashes (-), underscores (_), periods (.), spaces ( ), and at signs (@). After creating the user, you may fill in their **Full Name** if it is different from 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*).

If you check the **Send email to new user** checkbox, you can leave the **Password** field empty. The user will receive an activation email to set their own password.

20.1.2 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 *Server-side Encryption*.

20.1.3 Renaming a user

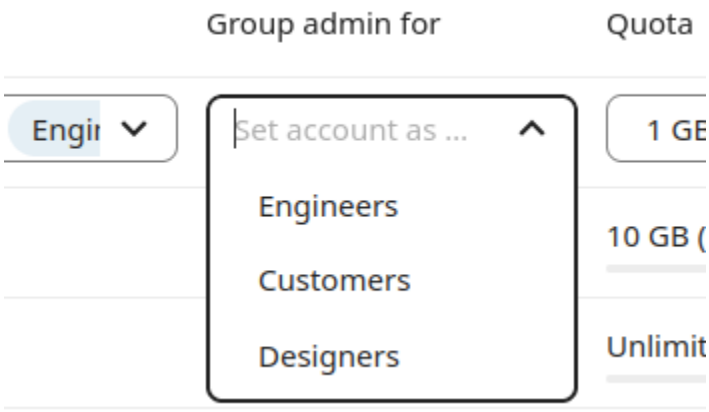
Each Nextcloud user has two names: a unique **Login Name** used for authentication, and a **Full Name**, which is their display name. You can edit the display name of a user, but you cannot change the login name of any user.

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

20.1.4 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 groups for which they are not **Group Administrators**. 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.

20.1.5 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.

20.1.6 Setting storage quotas

Click the gear icon 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 standard 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 users' quotas. This is still experimental and may not work as expected. The default is to not count external storage as part of user storage quotas. If you prefer to include it, then change the default `false` to `true`.

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

Note

If an external storage is defined as root, the quota will not be calculable and will be **ignored**.

Metadata (such as thumbnails, temporary files, and encryption keys) takes up about 10% of disk space but is not counted against user quotas. Users can check their used and available space on their Personal pages. Only files that originate with users count against their quotas, and not files shared with them that originate from other users. For example, if you upload files to a different user's share, those files count against your quota. If you re-share a file that another user shared with you, that file does not count against your quota, but the originating user's.

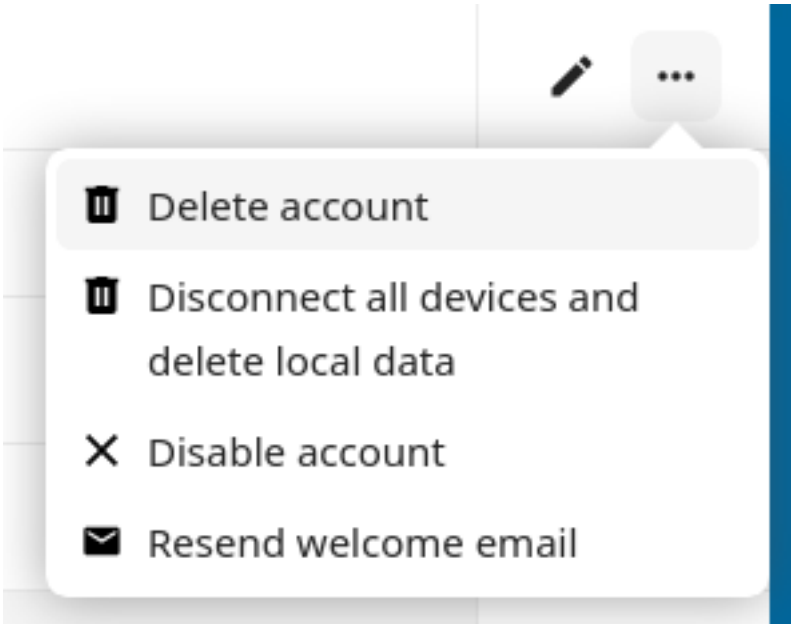
Encrypted files are a little larger than unencrypted files; the unencrypted size is calculated against the user's quota.

Deleted files that are still in the trash bin do not count against quotas. The trash bin is set at 50% of quota. Deleted file aging is set at 30 days. When deleted files exceed 50% of quota, the oldest files are removed until the total is below 50%.

When version control is enabled, 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.

20.1.7 Disable and enable users



Sometimes you may want to disable a user without permanently deleting their settings and files. The user can be activated again at any time, without data loss.

Hover your cursor over their name on the **Users** page until the "…" menu icon appears at the far right. After clicking on it, you will see the **Disable** option.

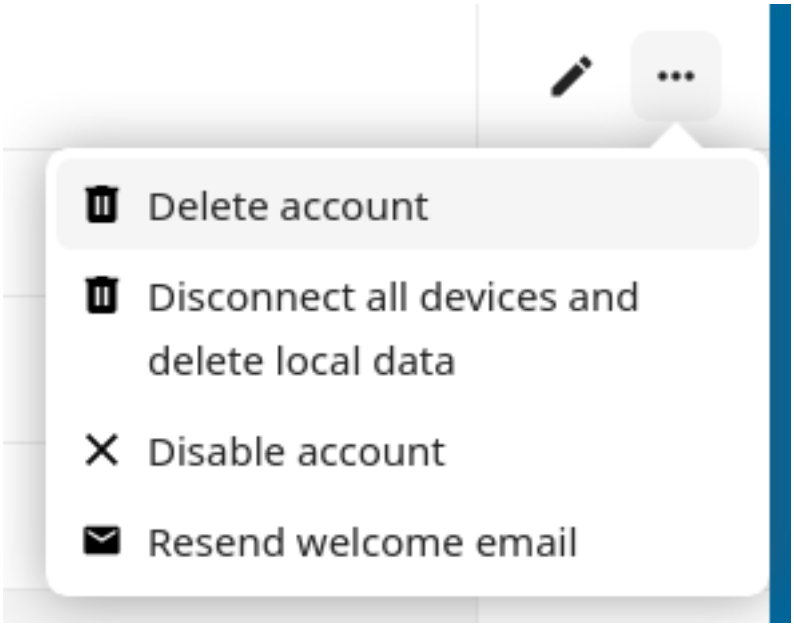
The user will no longer be able to access their Nextcloud until you enable them again. Also, all external shares, via public link or email, will not be accessible. Internal shares will still be working, so other users on Nextcloud can continue working.

If you wish for internal shares to be disabled as well when a user is disabled, activate the configuration option files_sharing:hide_disabled_user_shares:

```
occ config:app:set files_sharing hide_disabled_user_shares --value yes
```

You will find all disabled users in the **disabled** section on the left pane. Enabling users is as easy as disabling them. Just click on the "…" menu, and select **Enable**.

20.1.8 Deleting users



Deleting a user is easy: hover your cursor over their name on the **Users** page until the "…" menu icon appears at the far right. After clicking on it, you will see the **Delete** option. Clicking on it deletes a user with all their data immediately.

You'll see an undo button at the top of the page, which remains for a few seconds. When the undo button is gone, you cannot recover the deleted user.

All of the files owned by the user are deleted as well, including all files they have shared. If you need to preserve the user's files and shares, you must first download them from your Nextcloud Files page, which compresses them into a zip file, or use a sync client to copy them to your local computer. See *File Sharing* to learn how to create persistent file shares that survive user deletions.

20.1.9 Disabling the "Your email address [...] was changed" email

If an email address of a user is changed by an admin, then it triggers an email to the user that states "Your email address on [URL] was changed by an administrator.". In some cases this should not be triggered, because it was a normal maintenance change. To disable this specific email the appconfig option `disable_email.email_address_changed_by_admin` can be set to `yes`:

```
occ config:app:set settings disable_activity.email_address_changed_by_admin --value␣
↪yes
```

To disable this behaviour change it to any other value or delete the app config:

```
occ config:app:delete settings disable_activity.email_address_changed_by_admin
```

20.2 Resetting a lost admin password

The normal ways to recover a lost password are:

1. Click the password reset link on the login screen; this appears after a failed login attempt. This works only if you have entered your email address on your Personal page in the Nextcloud Web interface, so that the Nextcloud server can email a reset link to you.
2. Ask another Nextcloud server admin to reset it for you.

If neither of these is an option, then you have a third option, and that is using the `occ` command. See *Using the occ command* to learn more about using the `occ` command.

```
$ sudo -E -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.

20.3 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 that can handle an automatic reset:

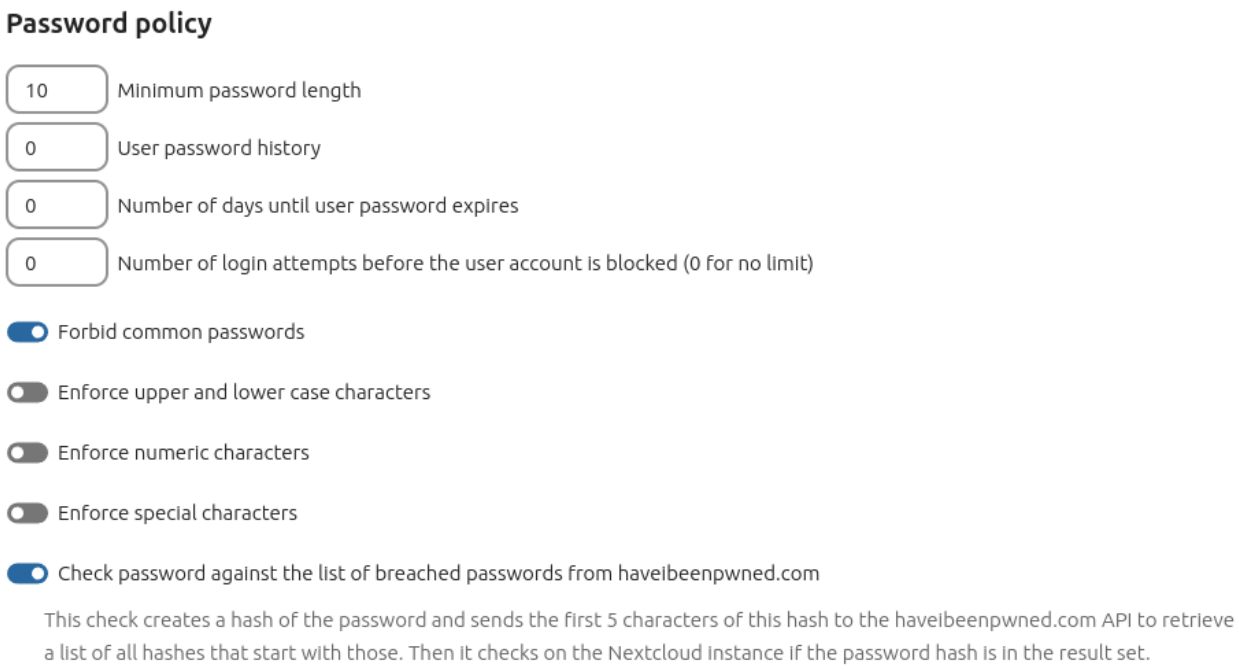
```
'lost_password_link' => 'https://example.org/link/to/password/reset',
```

20.4 User password policy

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

In the security-section of your administrator-settings you can configure

- a minimal length of a password. Default is 10 characters.
- a password history
- a password expiration period
- a lockout policy
- to forbid common passwords like 'password' or 'login'.
- to enforce upper and lower case characters
- to enforce numeric characters
- to enforce special characters like ! or :
- to check the password against the list of breached passwords from haveibeenpwnd.com (hashed check via haveibeenpwnd.com-API)



20.5 Authentication

20.5.1 App passwords

App passwords allow users to authenticate multiple client applications against their Nextcloud account without giving the application the login password. App passwords are mandatory for accounts with *two-factor authentication* enabled.

Some clients support *remote wipe*, which makes the connected application delete its local data.

Automated clean-up

Added in version 30.

Nextcloud will delete unused passwords. Passwords set for *remote wipe* are deleted after 60 days of no usage. App passwords of client applications are deleted after 365 days of no usage.

The time spans can be overwritten with configuration:

```
sudo -E -u www-data php occ config:system:set token_auth_wipe_token_retention --
↪type=int --value 2592000 # 60*60*24*30 - 30 days
sudo -E -u www-data php occ config:system:set token_auth_token_retention --type=int --
↪value 63072000     # 60*60*24*365*2 - 2 years
```

Values are set in **seconds**.

20.6 Two-factor authentication

Two-factor authentication adds an additional layer of security to user accounts. In order to log in on an account when two-factor authentication (2FA) is enabled, you must provide both the login password and another factor.

To use 2FA two things must happen:

- At least one 2FA provider must be enabled by the administrator.

- A user must activate 2FA on their account (or) the administrator must enforce the use of 2FA.

Both steps are described below.

20.6.1 Enabling two-factor authentication

2FA in Nextcloud is pluggable, meaning that various 2FA providers can be used to support different types of factors. Three providers are automatically installed (but may need to be enabled):

Two-Factor TOTP Provider

- A 2FA factor provider that enables the use of a TOTP (RFC 6238) app installed on a phone (or other device) to be used as the secondary factor
- Compatible with any RFC 6238 compliant TOTP client app (such as Aegis or Google Authenticator).
- Disabled by default. Go to *Apps->Disabled apps* and find *Two-Factor TOTP Provider* to enable this factor.

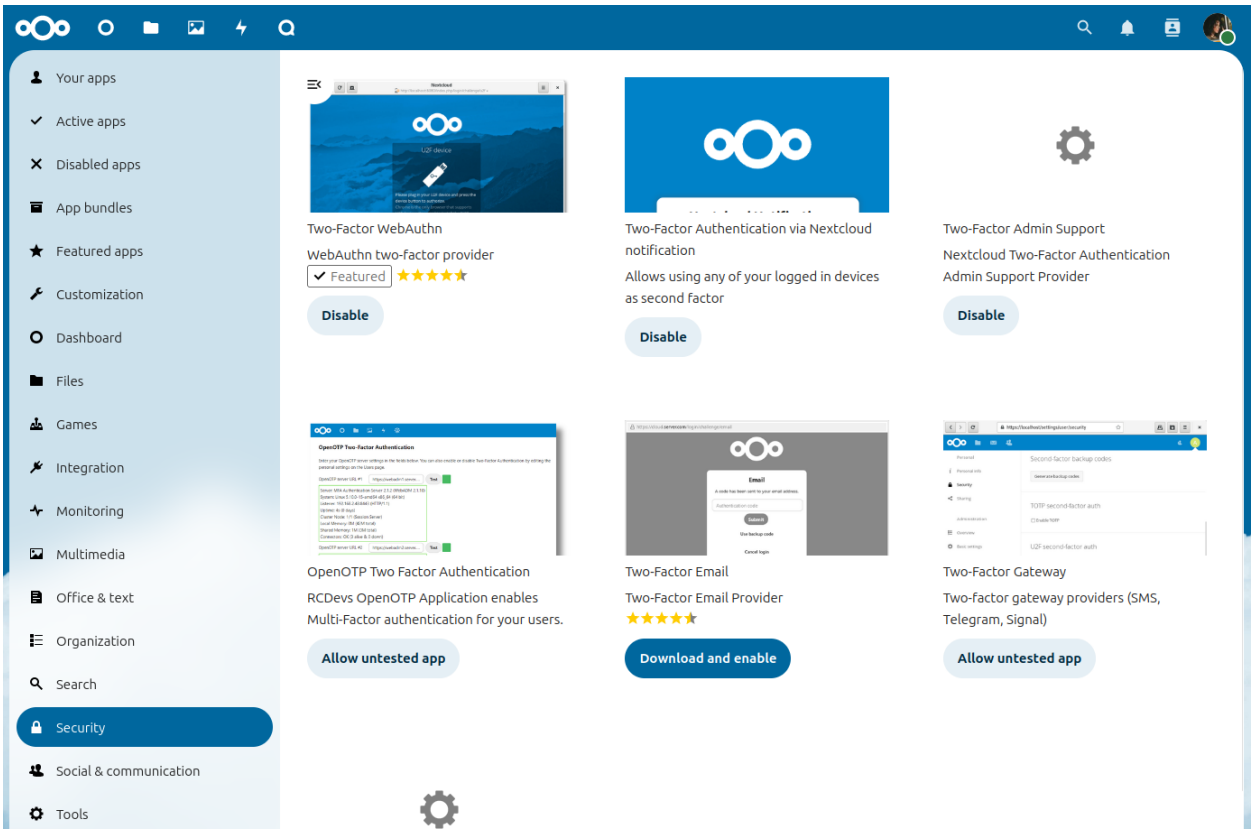
Two-Factor Authentication via Nextcloud notifications

- A 2FA factor provider that enables the use of a logged in device as the secondary factor.
- Disabled by default. Go to *Apps->Disabled apps* and find *Two-Factor Authentication via Nextcloud notification* to enable this factor.

Two-Factor Backup Codes

- A special 2FA factor provider enables users to generate backup codes.
- Facilitates recovery of access if a 2FA device is unavailable (i.e. gets stolen or is not working).
- Generates ten backup codes (which can, of course, only be used once).
- Always enabled.

Other 2FA providers may be found in the App Store.



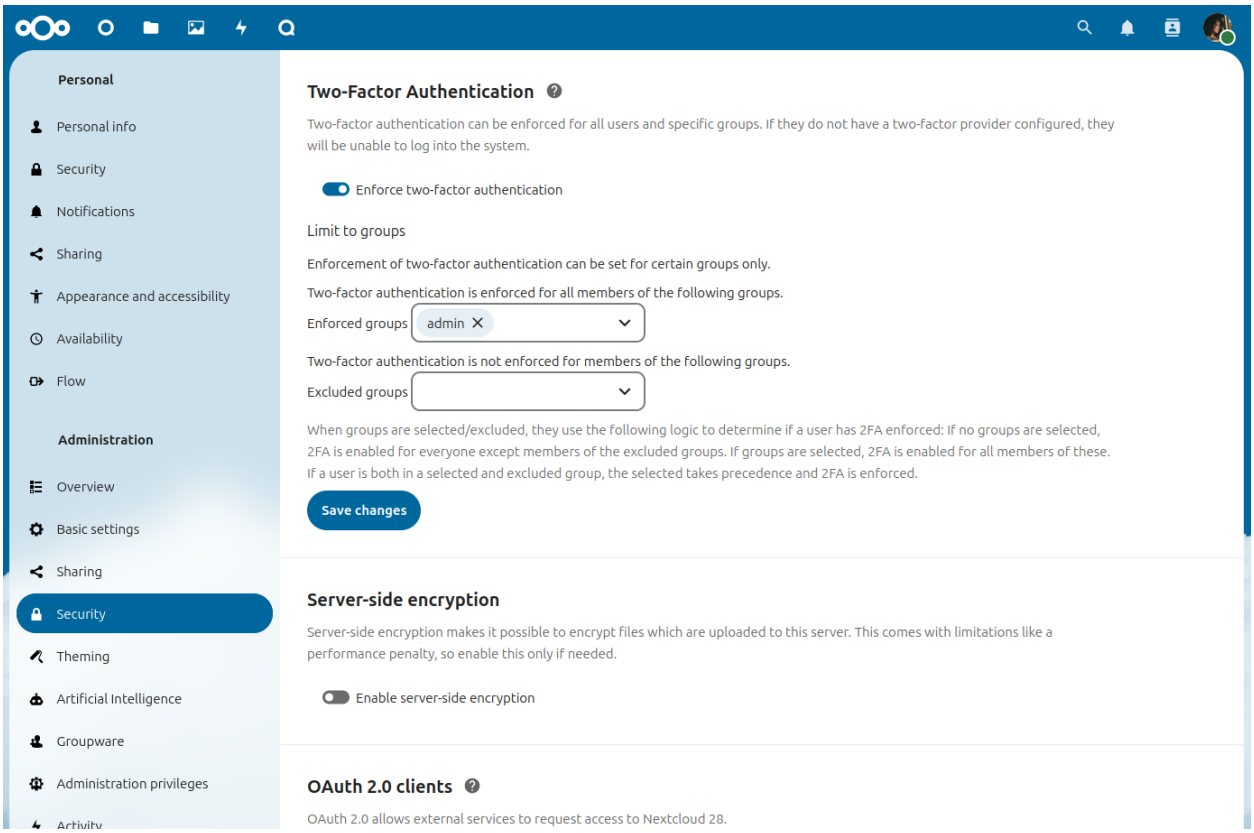
Developers can also implement new two-factor provider apps.

20.6.2 Enforcing two-factor authentication

By default 2FA is *optional*, hence users are given the choice whether to enable it for their account under their personal settings. Admins may, however, enforce the use of 2FA.

Enforcement is possible system-wide (all users) or for selected groups only. Select groups can also be excluded from 2FA requirements.

These settings can be found under *Administration Settings->Security*.



When groups are selected/excluded, they use the following logic to determine if a user has 2FA enforced:

- If no groups are selected, 2FA is enabled for everyone except members of the excluded groups
- If groups are selected, 2FA is enabled for all members of these. If a user is both in a selected *and* excluded group, the selected takes precedence and 2FA is enforced.

20.6.3 Provider removal

Nextcloud keeps records about the enabled two-factor authentication providers of every user. If a provider is simply removed/*disabled*, Nextcloud will still consider the provider active for the user at login and show a warning like *Could not load at least one of your enabled two-factor auth methods*.

The associations of removed providers can be cleaned up via *occ*:

```
sudo -E -u www-data php occ twofactorauth:cleanup <provider_id>
```

Warning

This operation is irreversible. Only run it for providers you do not intend to enable again as then you have to setup the configuration for all users from scratch.

20.6.4 Disabling two-factor authentication

Two-factor providers can be disabled via *occ*:

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

This can be useful if the user forgot or lost their second factor. Afterwards users are free to enable this provider again via their personal settings.

Note

This operation has to be supported by the provider. If this support is missing, Nextcloud will abort and show an error.

It is also possible to check the current two-factor user status via *occ*:

```
sudo -E -u www-data php occ twofactorauth:state <uid>
```

20.7 User authentication with LDAP

Nextcloud ships with an LDAP application to allow LDAP users (including Active Directory) to appear in your Nextcloud user listings. These users will authenticate to Nextcloud with their LDAP credentials, so you don't have to create separate Nextcloud user accounts for them. You will manage their Nextcloud group memberships, quotas, and sharing permissions just like any other Nextcloud user.

Note

The PHP LDAP module is required; this is supplied by `php-ldap` on most distributions.

The LDAP application supports:

- LDAP group support
- File sharing with Nextcloud users and groups
- Access via WebDAV and Nextcloud Desktop Client
- Versioning, external Storage and all other Nextcloud features
- Seamless connectivity to Active Directory, with no extra configuration required
- Support for primary groups in Active Directory
- Auto-detection of LDAP attributes such as base DN, email, and the LDAP server port number
- Only read access to your LDAP (edit or delete of users on your LDAP is not supported)
- Optional: Allow users to change their LDAP password from Nextcloud

Note

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

20.7.1 Configuration

First enable the `LDAP user and group backend` app on the Apps page in Nextcloud. Then go to your Admin page to configure it.

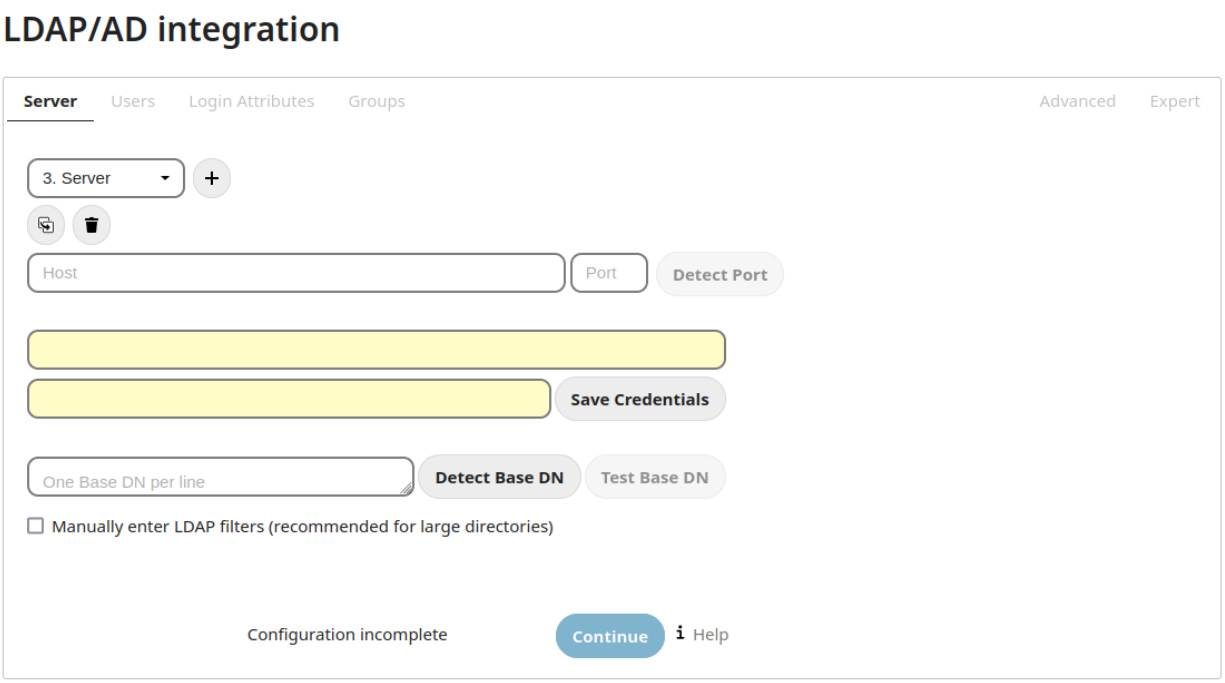
The LDAP configuration panel has four tabs. A correctly completed first tab ("Server") is mandatory to access the other tabs. A green indicator lights when the configuration is correct. Hover your cursor over the fields to see some pop-up tooltips.

Server tab

Start with the Server tab. You may configure multiple servers if you have them.

> **Note**
>
> Do not configure any failover LDAP hosts here. See *Advanced settings* for instructions instead.

At a minimum you must supply the LDAP server's hostname. If your server requires authentication, enter your credentials on this tab. Nextcloud will then attempt to auto-detect the server's port and base DN. The base DN and port are mandatory, so if Nextcloud cannot detect them you must enter them manually.



Server configuration:
: Configure one or more LDAP servers. Click the **Delete Configuration** button to remove the active configuration.

Host:
: The host name or IP address of the LDAP server. It can also be a **ldaps://** URI. If you enter the port number, it speeds up server detection.

  Examples:

  - *directory.my-company.com*
  - *ldaps://directory.my-company.com*

- *directory.my-company.com:9876*

Port:
The port on which to connect to the LDAP server. The field is disabled in the beginning of a new configuration. If the LDAP server is running on a standard port, the port will be detected automatically. If you are using a non-standard port, Nextcloud will attempt to detect it. If this fails you must enter the port number manually.

Example:

- *389*

User DN:
The name as DN of a user who has permissions to do searches in the LDAP directory. Leave it empty for anonymous access. We recommend that you have a special LDAP system user for this.

Example:

- *uid=nextcloudsystemuser,cn=sysusers,dc=my-company,dc=com*

Password:
The password for the user given above. Empty for anonymous access.

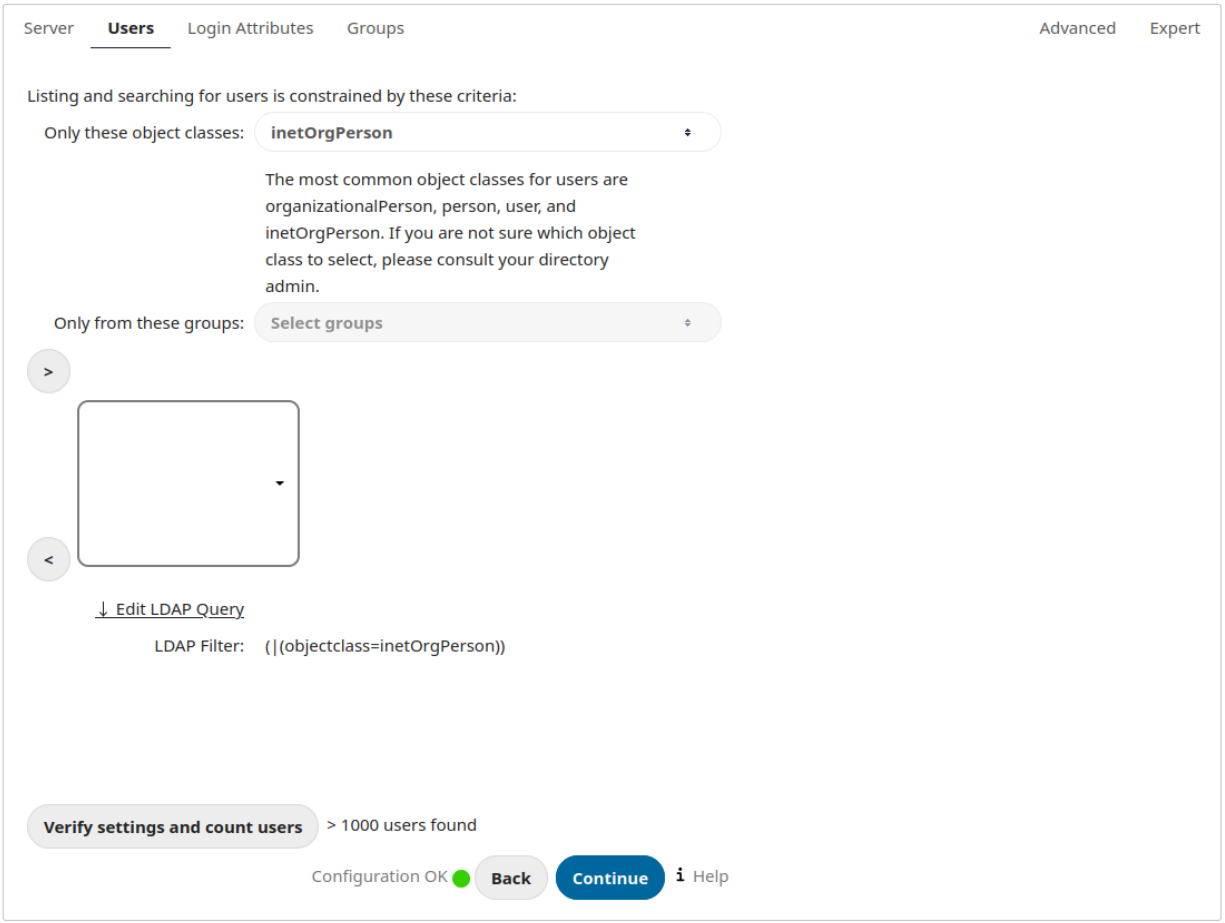
Base DN:
The base DN of LDAP, from where all users and groups can be reached. You may enter multiple base DNs, one per line. (Base DNs for users and groups can be set in the Advanced tab.) This field is mandatory. Nextcloud attempts to determine the Base DN according to the provided User DN or the provided Host, and you must enter it manually if Nextcloud does not detect it.

Example:

- *dc=my-company,dc=com*

Users tab

Use this to control which LDAP users are listed as Nextcloud users on your Nextcloud server. In order to control which LDAP users can login to your Nextcloud server use the **Login Attributes** tab. Those LDAP users who have access but are not listed as users (if there are any) will be hidden users. You may bypass the form fields and enter a raw LDAP filter if you prefer.



Only those object classes:
Nextcloud will determine the object classes that are typically available for user objects in your LDAP. Nextcloud will automatically select the object class that returns the highest amount of users. You may select multiple object classes.

Only from those groups:
If your LDAP server supports the `member-of-overlay` in LDAP filters, you can define that only users from one or more certain groups are allowed to appear in user listings in Nextcloud. By default, no value will be selected. You may select multiple groups.

If your LDAP server does not support the `member-of-overlay` in LDAP filters, the input field is disabled. Please contact your LDAP administrator.

Edit LDAP Query:
Clicking on this text toggles the filter mode and you can enter the raw LDAP filter directly. Example:

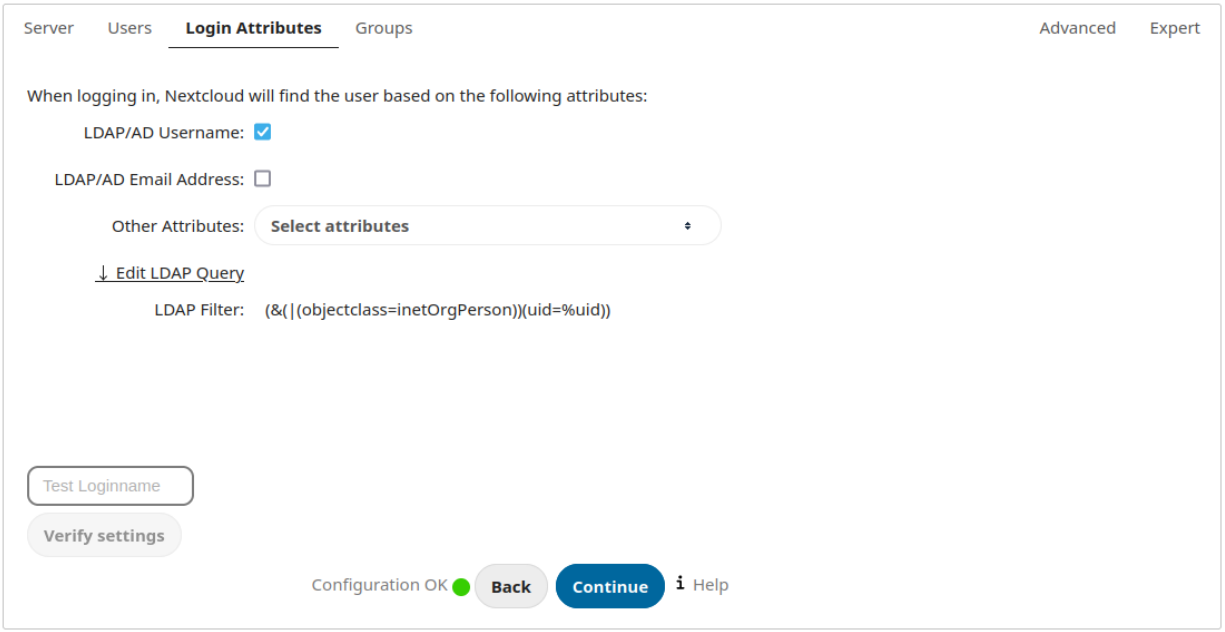
```
(&(objectClass=inetOrgPerson)(memberOf=cn=nextcloudusers,ou=groups,
dc=example,dc=com))
```

x users found:
This is an indicator that tells you approximately how many users will be listed in Nextcloud. The number updates automatically after any changes.

Login attributes tab

The settings in the Login Attributes tab determine which LDAP users can log in to your Nextcloud system and which attribute or attributes the provided login name is matched against (e.g. LDAP/AD username, email address). You may select multiple user details. (You may bypass the form fields and enter a raw LDAP filter if you prefer.)

You may override your User Filter settings on the Users tab by using a raw LDAP filter.



LDAP Username:
If this value is checked, the login value will be compared to the username in the LDAP directory. The corresponding attribute, usually *uid* or *samaccountname* will be detected automatically by Nextcloud.

LDAP Email Address:
If this value is checked, the login value will be compared to an email address in the LDAP directory; specifically, the *mailPrimaryAddress* and *mail* attributes.

Other Attributes:
This multi-select box allows you to select other attributes for the comparison. The list is generated automatically from the user object attributes in your LDAP server.

Edit LDAP Query:
Clicking on this text toggles the filter mode and you can enter the raw LDAP filter directly.

The **%uid** placeholder is replaced with the login name entered by the user upon login.

Examples:

- only username:

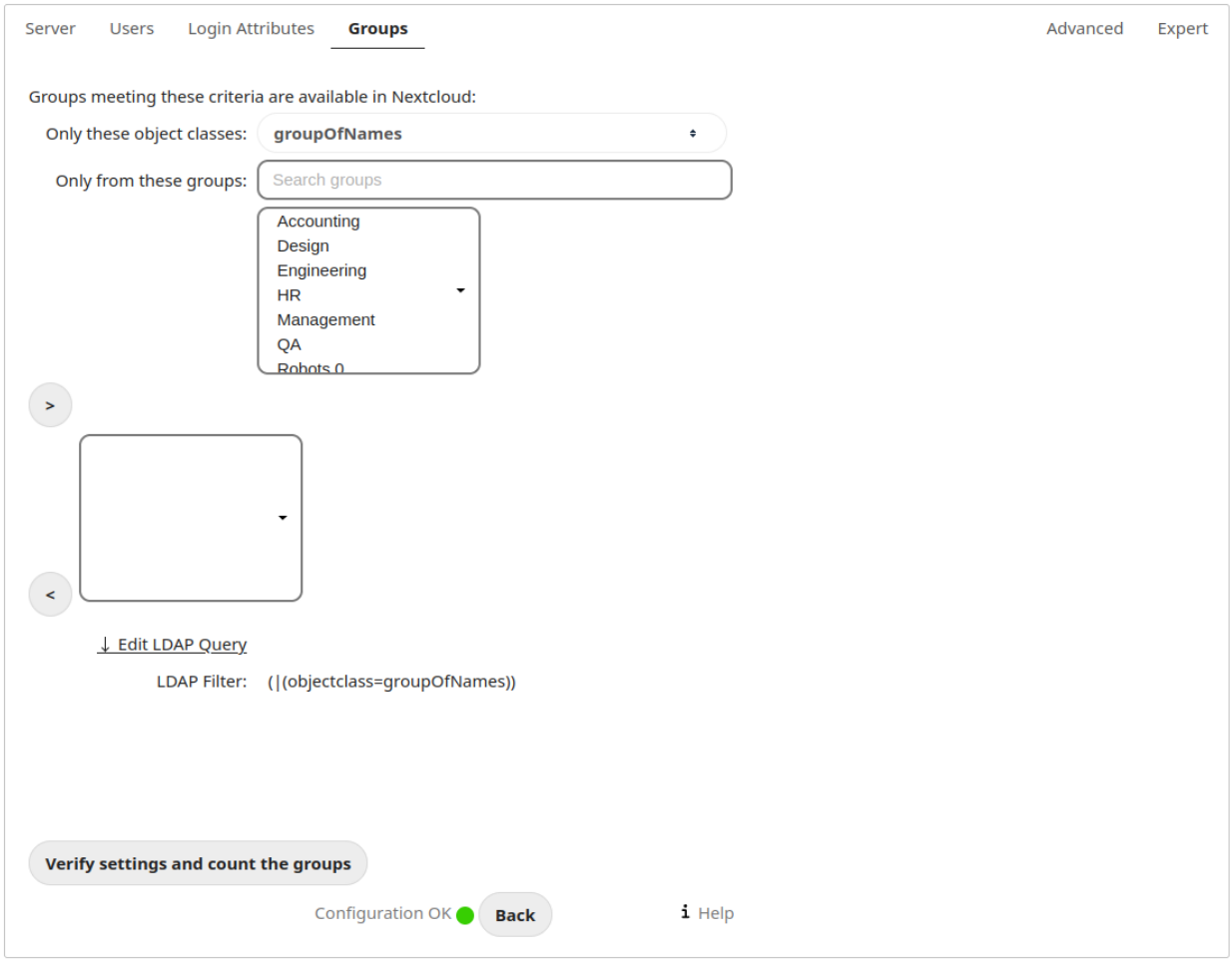
```
(&(objectClass=inetOrgPerson)(memberOf=cn=nextcloudusers,ou=groups,
dc=example,dc=com)(uid=%uid)
```

- username or email address:

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

Groups tab

By default, no LDAP groups will be available in Nextcloud. The settings in the Groups tab determine which groups will be available in Nextcloud. You may also elect to enter a raw LDAP filter instead.



Only these object classes:
: Nextcloud will determine the object classes that are typically available for group objects in your LDAP server. Nextcloud will only list object classes that return at least one group object. You can select multiple object classes. A typical object class is “group”, or “posixGroup”.

Only from these groups:
: Nextcloud will generate a list of available groups found in your LDAP server. Then you select the group or groups that get access to your Nextcloud server.

Edit LDAP Query:
: Clicking on this text toggles the filter mode and you can enter the raw LDAP filter directly.

  Example:

  - *objectClass=group*
  - *objectClass=posixGroup*

y groups found:
: This tells you approximately how many groups will be available in Nextcloud. The number updates automatically after any change.

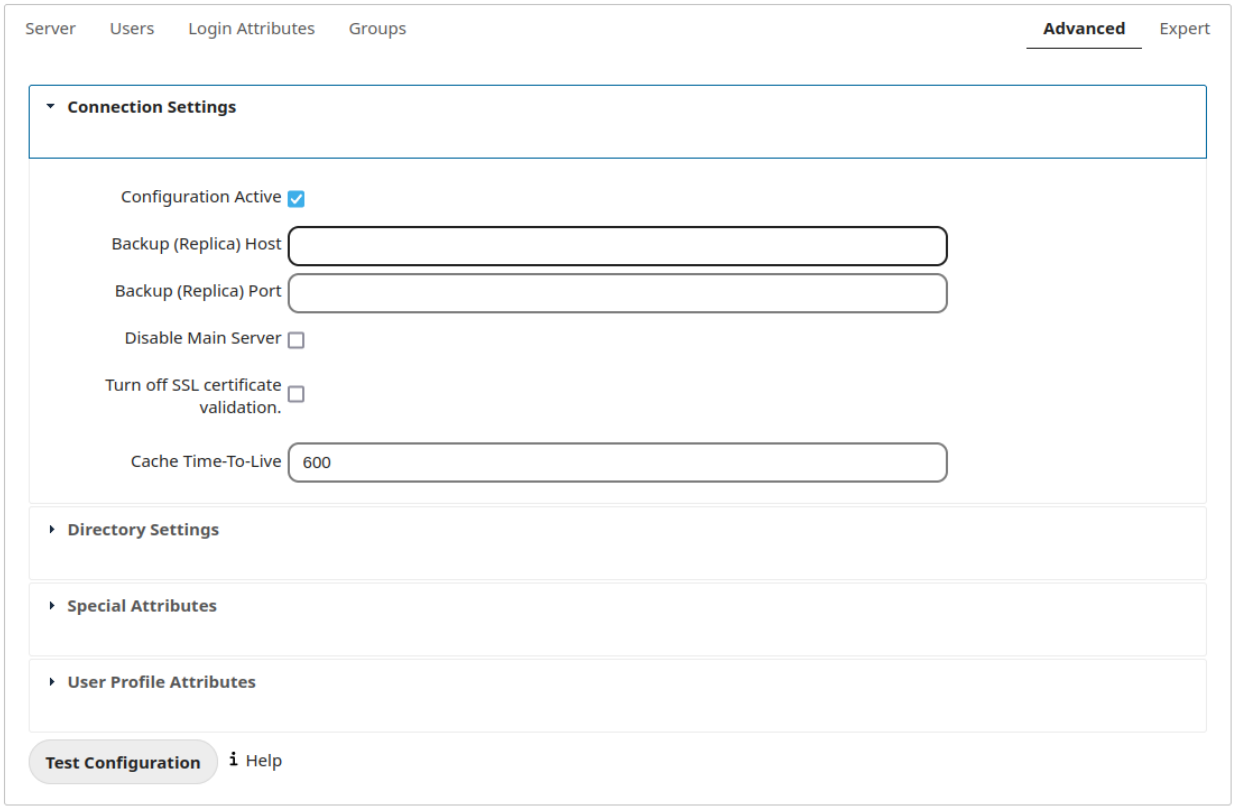
20.7.2 Advanced settings

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 four parts:

- Connection Settings
- Directory Settings
- Special Attributes
- User Profile Attributes

Connection settings



Configuration Active:
Enables or Disables the current configuration. By default, it is turned off. When Nextcloud makes a successful test connection it is automatically turned on.

Backup (Replica) Host:
If you have a backup LDAP server, enter the connection settings here. Nextcloud will then automatically connect to the backup when the main server cannot be reached. The backup server must be a replica of the main server so that the object UUIDs match.

Example:

- *directory2.my-company.com*

Backup (Replica) Port:
The connection port of the backup LDAP server. If no port is given, but only a host, then the main port (as specified above) will be used.

Example:

- *389*

Disable Main Server:
You can manually override the main server and make Nextcloud only connect to the backup server. This is useful for planned downtimes.

Turn off SSL certificate validation:
Turns off SSL certificate checking. Use it for testing only! *Note*: The effect of this setting depends on the PHP system configuration. It does for example not work with the [official Nextcloud container image](https://github.com/nextcloud/docker). To disable certificate verification for a particular use, append the following configuration line to your */etc/ldap/ldap.conf*:

```
` TLS_REQCERT ALLOW `
```

Cache Time-To-Live:
A cache is introduced to avoid unnecessary LDAP traffic, for example caching usernames so they don't have to be looked up for every page, and speeding up loading of the Users page. Saving the configuration empties the cache. The time is given in seconds.

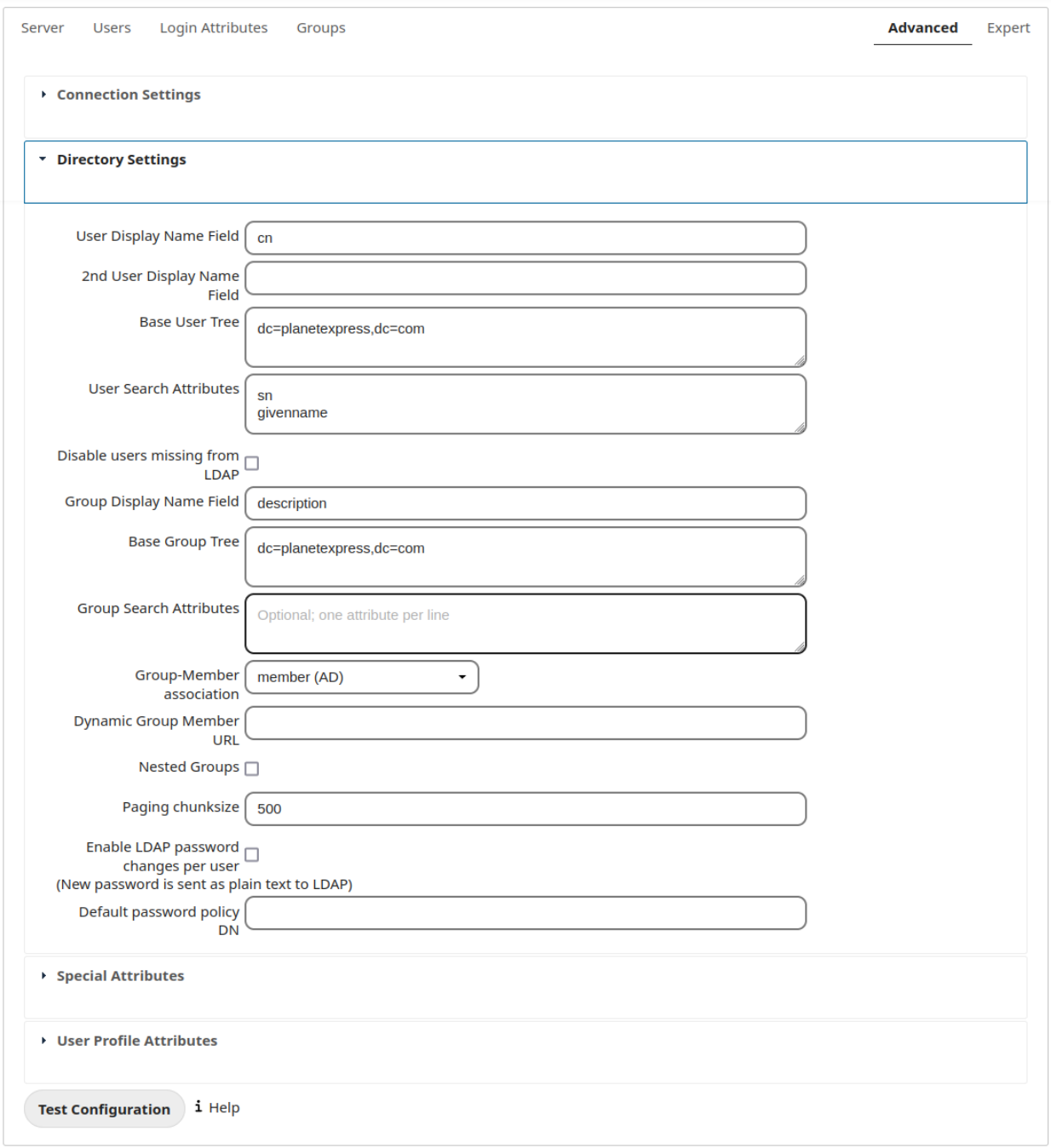
Note that almost every PHP request requires a new connection to the LDAP server. If you require fresh PHP requests we recommend defining a minimum lifetime of 15s or so, rather than completely eliminating the cache.

Examples:

- ten minutes: *600*
- one hour: *3600*

See the Caching section below for detailed information on how the cache operates.

Directory settings



User Display Name Field:
The attribute that should be used as display name in Nextcloud.

- Example: *displayName*

2nd User Display Name Field:
An optional second attribute displayed in brackets after the display name, for example using the `mail` attribute displays as `Molly Foo (molly@example.com)`.

Base User Tree:
The base DN of LDAP, from where all users can be reached. This must be a complete DN, regardless of what you have entered for your Base DN in the Basic setting. You can specify multiple base trees, one on each line.

- Example:

cn=programmers,dc=my-company,dc=com
cn=designers,dc=my-company,dc=com

User Search Attributes:
These attributes are used when searches for users are performed, for example in the share dialogue. The user display name attribute is the default. You may list multiple attributes, one per line.

If an attribute is not available on a user object, the user will not be listed, and will be unable to login. This also affects the display name attribute. If you override the default you must specify the display name attribute here.

- Example:

displayName
mail

Disable users missing from LDAP
If this is enabled, users which are missing from LDAP, also known as remnants, will behave as if disabled in Nextcloud. This means for instance that public shares by these users will not work anymore. see also *LDAP user cleanup*.

Group Display Name Field:
The attribute that should be used as Nextcloud group name. Nextcloud allows a limited set of characters (a-zA-Z0-9.-_@). Once a group name is assigned it cannot be changed.

- Example: *cn*

Base Group Tree:
The base DN of LDAP, from where all groups can be reached. This must be a complete DN, regardless of what you have entered for your Base DN in the Basic setting. You can specify multiple base trees, one in each line.

- Example:

cn=barcelona,dc=my-company,dc=com
cn=madrid,dc=my-company,dc=com

Group Search Attributes:
These attributes are used when a search for groups is done, for example in the share dialogue. By default the group display name attribute as specified above is used. Multiple attributes can be given, one in each line.

If you override the default, the group display name attribute will not be taken into account, unless you specify it as well.

- Example:

cn

description

Group Member association:
The attribute that is used to indicate group memberships, i.e. the attribute used by LDAP groups to refer to their users.

Nextcloud detects the value automatically. You should only change it if you have a very valid reason and know what you are doing.

- Example: *uniquemember*

Nested groups:
Enable group member retrieval from sub groups.

To allow user listing and login from nested groups, please see **User listing and login per nested groups** in the section **Troubleshooting, Tips and Tricks**.

Enable LDAP password changes per user:
Allow LDAP users to change their password and allow Super Administrators and Group Administrators to change the password of their LDAP users.

To enable this feature, the following requirements have to be met:

- General requirements:
- Access control policies must be configured on the LDAP server to grant permissions for password changes. The User DN as configured in *Server Settings* needs to have write permissions in order to update the userPassword attribute.
- Passwords are sent in plaintext to the LDAP server. Therefore, transport encryption must be used for the communication between Nextcloud and the LDAP server, e.g. employ LDAPS.
- Enabling password hashing on the LDAP server is highly recommended. While Active Directory stores passwords in a one-way format by default, OpenLDAP users could configure the `ppolicy_hash_cleartext` directive of the ppolicy overlay that ships with OpenLDAP.
- Additional requirements for Active Directory:
- At least a 128-bit transport encryption must be used for the communication between Nextcloud and the LDAP server.
- Make sure that the `fUserPwdSupport` char of the dSHeuristics is configured to employ the `userPassword` attribute as `unicodePwd` alias. While this is set accordingly on AD LDS by default, this is not the case on AD DS.

Default password policy DN:
This feature requires OpenLDAP with ppolicy. The DN of a default password policy will be used for password expiry handling in the absence of any user specific password policy. Password expiry handling features the following:

- When a LDAP password is about to expire, display a warning message to the user showing the number of days left before it expires. Password expiry warnings are displayed through the notifications app for Nextcloud.
- Prompt LDAP users with expired passwords to reset their password during login, provided that an adequate number of grace logins is still available.

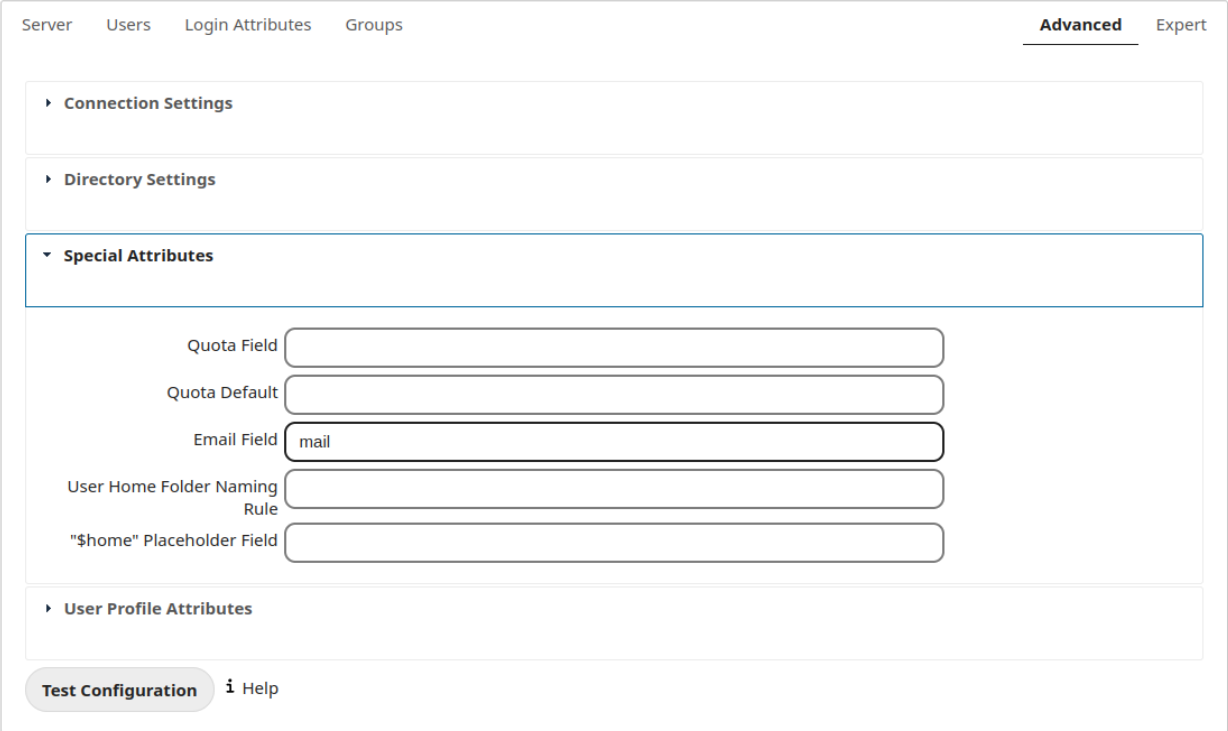
Leave the setting empty to keep password expiry handling disabled.

For the password expiry handling feature to work, LDAP password changes per user must be enabled and the LDAP server must be running OpenLDAP with its ppolicy module configured accordingly.

- Example:

cn=default,ou=policies,dc=my-company,dc=com

Special attributes



Quota Field:
Nextcloud can read an LDAP attribute and set the user quota according to its value. Specify the attribute here, and it will return human-readable values, e.g. “2 GB”.

- Example: *NextcloudQuota*

Warning

LDAP quota parameters override quota parameters set in the Nextcloud user management page.

Quota Default:
Specifies a default quota for LDAP users who do not have a quota set in the above Quota Field.

- Example: *15 GB*

Warning

LDAP quota parameters override quota parameters set in the Nextcloud user management page.

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.

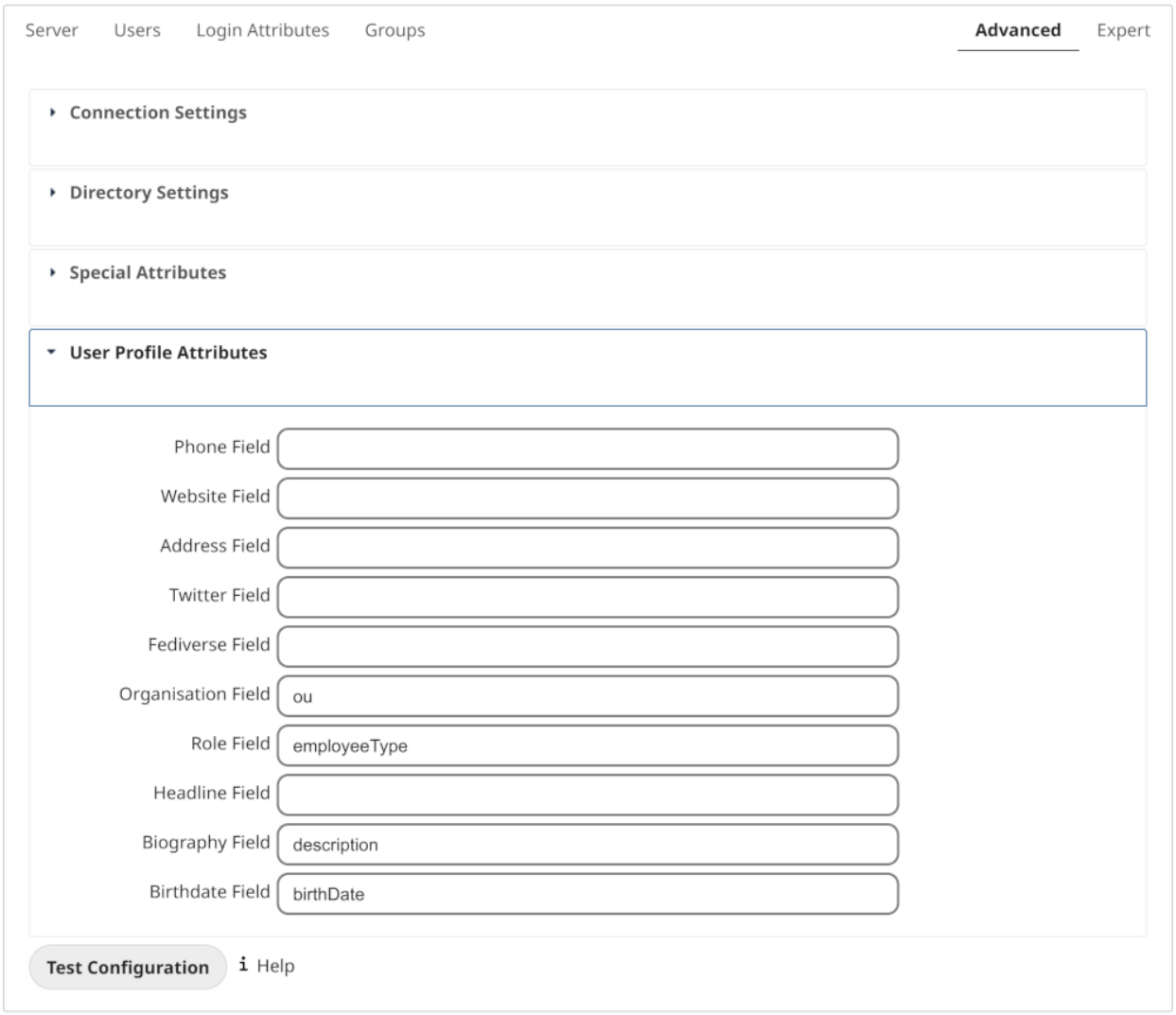
- Example: *cn*

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 -E -u www-data php occ config:app:set user_ldap enforce_home_folder_naming_rule -
↪-value=1
```

User Profile attributes



After configuring those attributes, the User Profile data will be overwritten with the corresponding data from LDAP. The checksum of data from LDAP will be stored in user settings `user_ldap`, `lastProfileChecksum` and profile update is skipped as long as data from LDAP doesn't change. If `memcache.distributed` is enabled in `config.php` the checksum will be cached and the checking will be skipped, as long as the cached value exists (expires after `ldapCacheTTL` seconds).

Please be aware:

- The user can change the data in profile, but it will get overwritten if changed in LDAP
- The user can change the visibility scope in profile
- The default visibility can be adjusted with setting the `account_manager.default_property_scope` array in `config.php`
- If multiple attribute values are present, only the first distributed value is used
- All user profile properties are limited to 2048 character
- Having misformatted data in LDAP will most probably leave you with empty user profile fields

- Setting the global `profile.enabled => false` on `config.php` skips the code

By calling `sudo -E -u www-data php occ ldap:check-user --update <uid>` the users data from LDAP will be displayed and the profile gets updated. To get the correct `<uid>` value for any user you can use `php occ user:list`.

Note

After unsetting an attribute name here, the data won't be deleted from user profile. Setting an nonexisting attribute will empty the corresponding profile field.

Phone Field:
The LDAP Attribute holding the phone number, to copy to the Profile Phone field. The phone number has to be formatted in international syntax without delimiters (E.164). Be sure to format phone numbers like `+4966612345678`.

- Example: *telephoneNumber*
- Example: *mobile*

Note

You should set your `default_phone_region` in `config.php`.

Website Field:
The LDAP attribute holding the website URI. The URI must start with `https://` or `http://` others are currently not allowed in Nextcloud user profile. If using `labeledURI` attributes the label (everything after first SPACE) gets removed.

- Example: *wWWHomePage*
- Example: *labeledURI*

Address Field:
The LDAP attribute holding the users address. Named Location on user profile page. Nextcloud wants a single line value like `city, country` or `somewhere under the loving sun`. Multi line postalAddress format will get reformatted, DOLLAR sign delimiter gets replaced with COMMA+SPACE.

- Example: *postalAddress*
- Example: *localityName*

Twitter Field:
The LDAP attribute holding the Twitter account name.

Fediverse Field:
The LDAP attribute holding the users Fediverse address.

Organisation Field:
The LDAP attribute holding the Organisation name.

- Example: *company*
- Example: *o* or *organizationName*

Role Field:
The LDAP attribute holding the organizational role, within the organisation or job title.

- Example: *title*

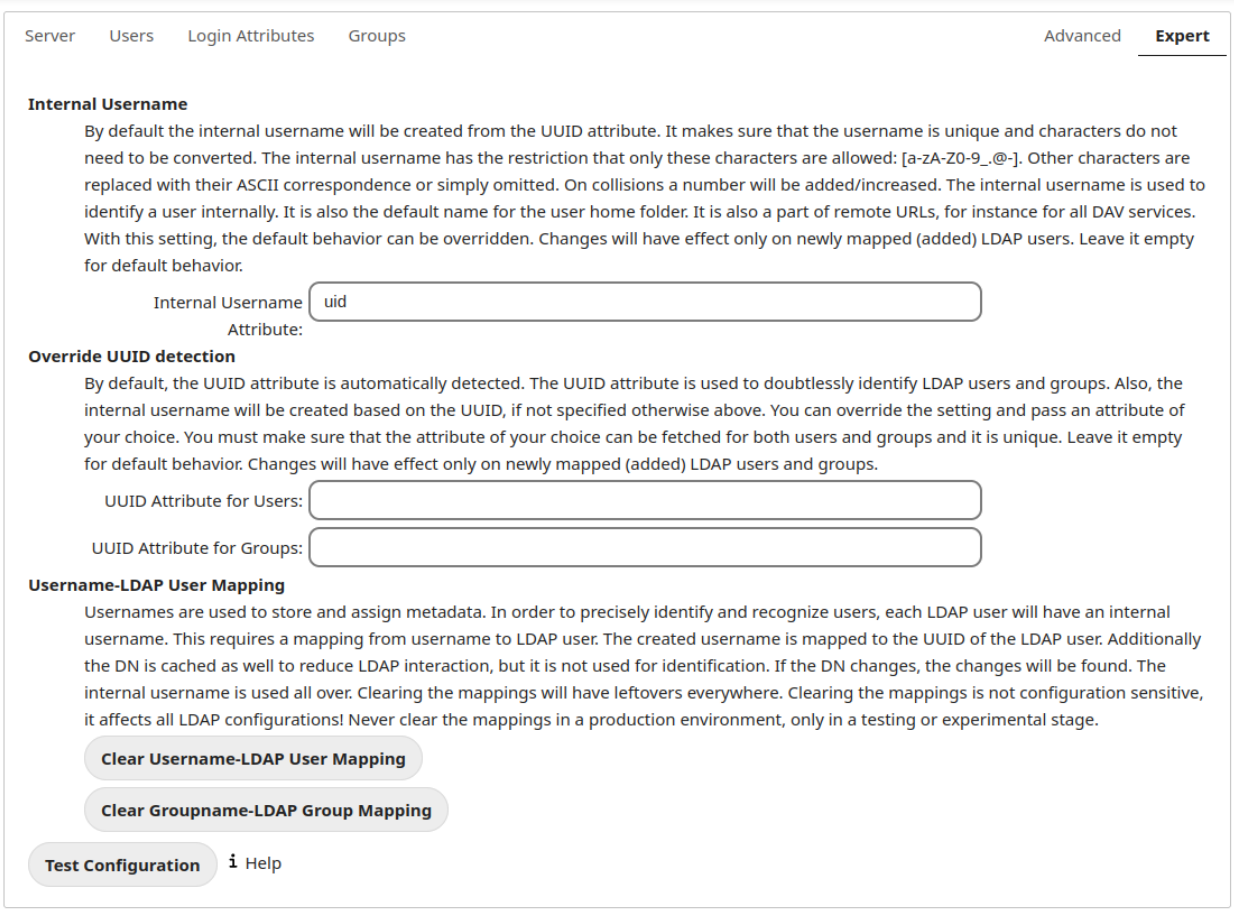
Headline Field:
The LDAP attribute holding the users headline.

Biography Field:
The LDAP attribute holding the user's biography (short description). Multi line value with unix LF line ending. Windows CRLF and Macintosh CR line endings will be replaced with unix LF line ending.

Birthdate Field:
The LDAP attribute holding the user's date of birth. Allowed formats:

- LDAP GeneralizedTime
- `YYYY-MM-DD`
- `YYYYMMDD`

20.7.3 Expert settings



In the Expert Settings fundamental behavior can be adjusted to your needs. The configuration should be well-tested before starting production use.

Internal Username:
The internal username is the identifier in Nextcloud for LDAP users. By default it will be created from the UUID attribute. The UUID attribute ensures that the username is unique, and that characters do not need to be converted. Only these characters are allowed: `[a-zA-Z0-9_.@-]`. Other characters are replaced with their ASCII equivalents, or are simply omitted.

The LDAP backend ensures that there are no duplicate internal usernames in Nextcloud, i.e. that it is checking all

other activated user backends (including local Nextcloud users). On collisions a random number (between 1000 and 9999) will be attached to the retrieved value. For example, if "alice" exists, the next username may be "alice_1337".

The internal username is the default name for the user home folder in Nextcloud. It is also a part of remote URLs, for instance for all *DAV services.

You can override all of this with the Internal Username setting. Leave it empty for default behavior. Changes will affect only newly mapped LDAP users.

When configuring this, be aware that the username in Nextcloud is considered immutable and cannot be changed afterwards. This can cause issues when using an attribute that might change, e.g. the email address of a user that will get changed during name change.

- Example: *uid*

Override UUID detection
By default, Nextcloud auto-detects the UUID attribute. The UUID attribute is used to uniquely identify LDAP users and groups. The internal username will be created based on the UUID, if not specified otherwise.

You can override the setting and pass an attribute of your choice. You must make sure that the attribute of your choice can be fetched for both users and groups and it is unique. Leave it empty for default behavior. Changes will have effect only on newly mapped LDAP users and groups. It also will have effect when a user's or group's DN changes and an old UUID was cached, which will result in a new user. Because of this, the setting should be applied before putting Nextcloud in production use and clearing the bindings (see the `User and Group Mapping` section below).

- Example: *cn*

Username-LDAP User Mapping
Nextcloud uses usernames as keys to store and assign data. In order to precisely identify and recognize users, each LDAP user will have a internal username in Nextcloud. This requires a mapping from Nextcloud username to LDAP user. The created username is mapped to the UUID of the LDAP user. Additionally the DN is cached as well to reduce LDAP interaction, but it is not used for identification. If the DN changes, the change will be detected by Nextcloud by checking the UUID value.

The same is valid for groups.

The internal Nextcloud name is used all over in Nextcloud. Clearing the Mappings will have leftovers everywhere. Never clear the mappings in a production environment, but only in a testing or experimental server.

> **Warning**
>
> Clearing the Mappings is not configuration sensitive, it affects all LDAP configurations!

20.7.4 Testing the configuration

> **Warning**
>
> Before testing, make sure the **Configuration Active** checkbox is enabled. It is disabled by default and will cause authentication to fail with a "wrong password" error even if all other settings are correct. Nextcloud only enables it automatically after a successful test connection.

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.

20.7.5 Additional configuration options via occ

Few configuration settings can only be set on command line via `occ`.

Background sync interval

The LDAP backend updates user attributes (email, quota, avatar, and others) on every login, on first detection of a new user, and periodically via a background job.

The background job recalculates its run interval after each cycle. The goal is to process every known LDAP user approximately once per day. The interval is derived from the total number of mapped users and the smallest configured LDAP paging size, then clamped to a minimum of 30 minutes and a maximum of 12 hours.

Administrative Group mapping

It is possible to promote **one** LDAP per connection as an admin group, so that all its members also have administrative privileges in Nextcloud.

A group can either be promoted via a dedicated `occ` call providing a group parameter that can be either a nextcloud group ID or a group name that will be search against. When a search is executed an exact match is required.

Example usage:

```
$ sudo -E -u www-data php occ ldap:promote-group --help
Description:
  declares the specified group as admin group (only one is possible per LDAP␣
↪configuration)

Usage:
  ldap:promote-group [options] [--] <group>

Arguments:
  group                 the group ID in Nextcloud or a group name

Options:
  -y, --yes             do not ask for confirmation
…

# Example
$ sudo -E -u www-data php occ ldap:promote-group  "Nextcloud Admins"
Promote Nextcloud Admins to the admin group (y|N)? y
Group Nextcloud Admins was promoted

$ sudo -E -u www-data php occ ldap:promote-group  "Paramount Court"
Promote Nextcloud Admins to the admin group and demote Nextcloud Admins (Group ID:␣
↪nextcloud_admins) (y|N)? y
Group Paramount Court was promoted

$ sudo -E -u www-data php occ ldap:promote-group  "Paramount Court"
The specified group is already promoted
```

Note

Note the group ID will only be displayed when it differs from the group's display name.

It is also possible to set the admin group mapping using `occ ldap:set-config $configId ldapAdminGroup $groupId`, but as the Nextcloud group ID might not be known (yet) it is recommended (especially for automated setups) to use the *promote-group* command, that would also pull in the group and determine the group ID.

In order to demote or reset a promotion, an empty string should be set against to the targeted config's ldapAdminGroup:

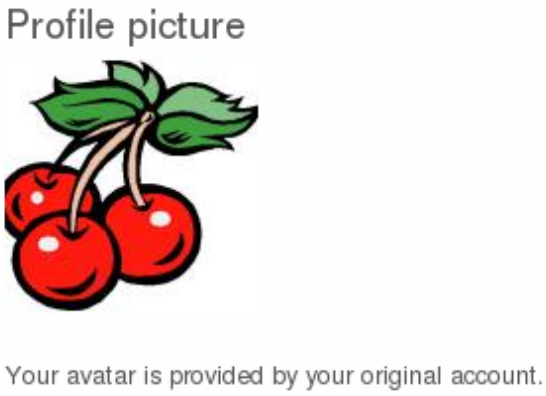
```
# Reset an admin group mapping via set-config
occ ldap:set-config $configId ldapAdminGroup ""
# Example
occ ldap:set-config s01 ldapAdminGroup ""
```

Tip

To have more than one administrative groups in a connection, create a holding group in your LDAP directory that contains the single groups as nested members, and promote this one.

20.7.6 Nextcloud avatar integration

Nextcloud supports user profile pictures, which are also called avatars. If a user has a photo stored in the *jpegPhoto* or *thumbnailPhoto* attribute on your LDAP server, it will be used as their avatar. In this case the user cannot alter their avatar (on their Personal page) as it must be changed in LDAP. *jpegPhoto* is preferred over *thumbnailPhoto*.



If the *jpegPhoto* or *thumbnailPhoto* attribute is not set or empty, then users can upload and manage their avatars on their Nextcloud Personal pages. Avatars managed in Nextcloud are not stored in LDAP.

The *jpegPhoto* or *thumbnailPhoto* attribute is fetched once a day to make sure the current photo from LDAP is used in Nextcloud. LDAP avatars override Nextcloud avatars, and when an LDAP avatar is deleted then the most recent Nextcloud avatar replaces it.

Photos served from LDAP are automatically cropped and resized in Nextcloud. This affects only the presentation, and the original image is not changed.

Use a specific attribute or turn off loading of images

It is possible to turn off the avatar integration or specify a single, different attribute to read the image from. It is expected to contain image data just like *jpegPhoto* or *thumbnailPhoto* do.

The behaviour can be changed using the occ command line tool only. Essentially those options are available:

- The default behaviour as described above should be used

  ```
  occ ldap:set-config "s01" "ldapUserAvatarRule" "default"
  ```

- User images shall not be fetched from LDAP

  ```
  occ ldap:set-config "s01" "ldapUserAvatarRule" "none"
  ```

- The image should be read from the attribute "selfiePhoto"

  ```
  occ ldap:set-config "s01" "ldapUserAvatarRule" "data:selfiePhoto"
  ```

The "s01" refers to the configuration ID as can be retrieved per `occ ldap:show-config`.

20.7.7 Troubleshooting, tips and tricks

Logging

Nextcloud's LDAP implementation is capable of logging lots of additional details about its activities. When diagnosing problems, it can be useful to temporarily adjust your `loglevel` to INFO (`1`) or DEBUG (`0`).

SSL certificate verification (LDAPS, TLS)

A common mistake with SSL certificates is that they may not be known to PHP. If you have trouble with certificate validation make sure that

- You have the certificate of the server installed on the Nextcloud server
- The certificate is announced in the system's LDAP configuration file (usually */etc/ldap/ldap.conf*)
- Using LDAPS, also make sure that the port is correctly configured (by default 636)

Microsoft Active Directory

Compared to earlier Nextcloud versions, no further tweaks need to be done to make Nextcloud work with Active Directory. Nextcloud will automatically find the correct configuration in the set-up process.

memberOf / read memberof permissions

If you want to use `memberOf` within your filter you might need to give your querying user the permissions to use it. For Microsoft Active Directory this is described here.

User listing and login per nested groups

When it is intended to allow user listing and login based on a specific group having subgroups ("nested groups"), checking **Nested groups** on **Directory Settings** is not enough. Also the User (and Login) filter need to be changed, by specifying the `LDAP_MATCHING_RULE_IN_CHAIN` matching rule. Change the filter parts containing the *memberof* condition according to this example:

- (memberof=cn=Nextcloud Users Group,ou=Groups,…)

to

- (memberof:1.2.840.113556.1.4.1941:=cn=Nextcloud Users Group,ou=Groups,…)

Duplicating server configurations

In case you have a working configuration and want to create a similar one or "snapshot" configurations before modifying them you can do the following:

1. Go to the **Server** tab
2. On **Server Configuration** choose *Add Server Configuration*
3. Answer the question *Take over settings from recent server configuration?* with *yes*.
4. (optional) Switch to **Advanced** tab and uncheck **Configuration Active** in the *Connection Settings*, so the new configuration is not used on Save
5. Click on **Save**

Now you can modify and enable the configuration.

20.7.8 Nextcloud LDAP internals

Some parts of how the LDAP backend works are described here.

User and group mapping

In Nextcloud the user or group name is used to have all relevant information in the database assigned. To work reliably a permanent internal user name and group name is created and mapped to the LDAP DN and UUID. If the DN changes in LDAP it will be detected, and there will be no conflicts.

Those mappings are done in the database table `ldap_user_mapping` and `ldap_group_mapping`. The user name is also used for the user's folder (except if something else is specified in *User Home Folder Naming Rule*), which contains files and meta data.

The internal user name and a visible display name are separated. This is not the case for group names, yet, i.e. a group name cannot be altered.

That means that your LDAP configuration should be good and ready before putting it into production. The mapping tables are filled early, but as long as you are testing, you can empty the tables any time. Do not do this in production.

The attributes of users are fetched on demand (i.e. for sharing autocompletion or in the user management) and then stored inside the Nextcloud database to allow a better performance on our side. They are typically checked twice a day in batches from all users again. Beside that they are also refreshed during a login for this user or can be fetched manually via the occ command `occ ldap:check-user --update USERID` where `USERID` is Nextcloud's user id.

For groups, a cache of memberships is stored in the database to be able to trigger events when a membership is added or removed. This cache is updated by a background job, and can be force updated using `occ ldap:check-group --update GROUPID`.

Caching

The LDAP information is cached in Nextcloud memory cache, and you must install and configure the memory cache (see *Memory caching*). The Nextcloud **Cache** helps to speed up user interactions and sharing. It is populated on demand, and remains populated until the **Cache Time-To-Live** for each unique request expires. User logins are not cached, so if you need to improve login times set up a slave LDAP server to share the load.

You can adjust the **Cache Time-To-Live** value to balance performance and freshness of LDAP data. All LDAP requests will be cached for 10 minutes by default, and you can alter this with the **Cache Time-To-Live** setting. The cache answers each request that is identical to a previous request, within the time-to-live of the original request, rather than hitting the LDAP server.

The **Cache Time-To-Live** is related to each single request. After a cache entry expires there is no automatic trigger for re-populating the information, as the cache is populated only by new requests, for example by opening the User administration page, or searching in a sharing dialog.

There is one trigger which is automatically triggered by a certain background job which keeps the `user-group-mappings` up-to-date, and always in cache.

Under normal circumstances, all users are never loaded at the same time. Typically the loading of users happens while page results are generated, in steps of 30 until the limit is reached or no results are left. For this to work on a Nextcloud-Server and LDAP-Server, **Paged Results** must be supported.

Nextcloud remembers which user belongs to which LDAP-configuration. That means each request will always be directed to the right server unless a user is defunct, for example due to a server migration or unreachable server. In this case the other servers will also receive the request.

Handling with backup server

When Nextcloud is not able to contact the main LDAP server, Nextcloud assumes it is offline and will not try to connect again for the time specified in **Cache Time-To-Live**. If you have a backup server configured Nextcloud will connect to it instead. When you have scheduled downtime, check **Disable Main Server** to avoid unnecessary connection attempts.

20.7.9 Note

When a LDAP object's name or surname, that is display name attribute, by default "displayname", is left empty, Nextcloud will treat it as an empty object, therefore no results from this user or AD-Object will be shown to avoid gathering of technical accounts.

20.8 LDAP user cleanup

LDAP User Cleanup is a new feature in the `LDAP user and group backend` application. LDAP User Cleanup is a background process that automatically searches the Nextcloud LDAP mappings table, and verifies if the LDAP users are still available. Any users that are not available are marked as `deleted` in the `oc_preferences` database table. Then you can run a command to display this table, displaying only the users marked as `deleted`, and then you have the option of removing their data from your Nextcloud data directory.

These items are removed upon cleanup:

- Local Nextcloud group assignments
- User preferences (DB table `oc_preferences`)
- User's Nextcloud home folder
- User's corresponding entry in `oc_storages`

There are two prerequisites for LDAP User Cleanup to operate:

1. Set `ldapUserCleanupInterval` in `config.php` to your desired check interval in minutes. The default is 51 minutes.
2. All configured LDAP connections are enabled and operating correctly. As users can exist on multiple LDAP servers, you want to be sure that all of your LDAP servers are available so that a user on a temporarily disconnected LDAP server is not marked as `deleted`.

The background process examines 50 users at a time, and runs at the interval you configured with `ldapUserCleanupInterval`. For example, if you have 200 LDAP users and your `ldapUserCleanupInterval` is 20 minutes, the process will examine the first 50 users, then 20 minutes later the next 50 users, and 20 minutes later the next 50, and so on.

The amount of users to check can be set to a custom value via occ command. The following example sets it to 300:

```
sudo -E -u www-data php occ config:app:set --value=300 user_ldap cleanUpJobChunkSize
```

There are two `occ` commands to use for examining a table of users marked as deleted, and then manually deleting them. The `occ` command is in your Nextcloud directory, for example `/var/www/nextcloud/occ`, and it must be run as your HTTP user. To learn more about `occ`, see *Using the occ command*.

These examples are for Ubuntu Linux:

1. `sudo -E -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 -E -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 -E -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   |
+-----------------+-----------------+------------------+------------------------------
↪--------+
```

Following flags can be specified additionally:

- `--short-date`: formats the dates for `Last login` and `Detected on` in a short Y-m-d format (e.g. 2019-01-14)
- `--json`: instead of a table, the output is json-encoded. This makes it easy to process the data programmatically.

Then you can run `sudo -E -u www-data php occ user:delete aaliyah_brown` to delete user aaliyah_brown. You must use the user's Nextcloud name.

20.8.1 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.

20.9 The LDAP configuration API

All methods require that the "OCS-APIREQUEST" header be set to "true". Methods take an optional "format" parameter, which may be "xml" (the default) or "json".

20.9.1 Creating a configuration

Creates a new and empty LDAP configuration. It returns its ID. Authentication is done by sending a basic HTTP authentication header.

Syntax: ocs/v2.php/apps/user_ldap/api/v1/config

- HTTP method: POST

Example

```
$ curl -X POST https://admin:secret@example.com/ocs/v2.php/apps/user_ldap/api/v1/
↪config -H "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>
```

20.9.2 Deleting a configuration

Deletes a given LDAP configuration. Authentication is done by sending a basic HTTP authentication header.

Syntax: ocs/v2.php/apps/user_ldap/api/v1/config/{configID}

- HTTP method: DELETE

Example

```
$ curl -X DELETE ``https://admin:secret@example.com/ocs/v2.php/apps/user_ldap/api/v1/
↪config/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>
```

20.9.3 Reading a configuration

Returns all keys and values of the specified LDAP configuration. Authentication is done by sending a basic HTTP authentication header.

Syntax: ocs/v2.php/apps/user_ldap/api/v1/config/{configID}

- HTTP method: GET
- url argument: showPassword - int, optional, default 0, whether to return the password in clear text

Example

```
$ curl -X GET https://admin:secret@example.com/ocs/v2.php/apps/user_ldap/api/v1/
↪config/s02?showPassword=1 -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>
```

(continues on next page)

(continued from previous page)

```
  <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>
```

20.9.4 Modifying a configuration

Updates a configuration with the provided values. Authentication is done by sending a basic HTTP authentication header.

Syntax: ocs/v2.php/apps/user_ldap/api/v1/config/{configID}

- HTTP method: PUT
- url argument: configData - array, see table below for the fields. All fields are optional. The values must be url-encoded.

Example

```
$ curl -X PUT https://admin:secret@example.com/ocs/v2.php/apps/user_ldap/api/v1/
↪config/s01 -H "OCS-APIREQUEST: true" -d "configData[ldapHost]=ldap%3A%2F%2Fldap.
↪server.tld &configData[ldapPort]=389"
```

- updates the LDAP configuration

XML output

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>200</statuscode>
  <message>OK</message>
 </meta>
```

(continues on next page)

(continued from previous page)

```
 <data/>
</ocs>
```

20.9.5 Configuration keys

Key	Mode	Required	Description
ldapHost	rw	yes	LDAP server host, supports protocol
ldapPort	rw	yes	LDAP server port
ldapBackupHost	rw	no	LDAP replica host
ldapBackupPort	rw	no	LDAP replica port
ldapOverrideMainServer	rw	no	Whether replica should be used instead
ldapBase	rw	yes	Base
ldapBaseUsers	rw	no	Base for users, defaults to general base if not specified
ldapBaseGroups	rw	no	Base for groups, defaults to general base if not specified
ldapAgentName	rw	no	DN for the (service) user to connect to LDAP
ldapAgentPassword	rw	no	Password for the service user
ldapTLS	rw	no	Whether to use StartTLS
turnOffCertCheck	rw	no	Turns off certificate validation for TLS connections
ldapIgnoreNamingRules	rw	no	Backwards compatibility, do not set it.
ldapUserDisplayName	rw	yes	Attribute used as display name for users
ldapUserDisplayName2	rw	no	Additional attribute, if set show on brackets next to the main attribute
ldapUserAvatarRule	rw	no	Specify the avatar integration behavior, possible values: “default”, “none”,
ldapGidNumber	rw	no	group ID attribute, needed for primary groups on OpenLDAP (and compa
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, unic
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 sto
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

Table 1 – continued from previous page

Key	Mode	Required	Description
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 av
ldapExpertUUIDUserAttr	rw	no	override the LDAP servers UUID attribute to identify LDAP user records
ldapExpertUUIDGroupAttr	rw	no	override the LDAP servers UUID attribute to identify LDAP group record
lastJpegPhotoLookup	r	no	set in runtime
ldapNestedGroups	rw	no	Whether LDAP supports nested groups
ldapPagingSize	rw	no	Number of results to return per page
turnOnPasswordChange	rw	no	Whether users are allowed to change passwords (hashing must happen on L
ldapDynamicGroupMemberURL	rw	no	URL for dynamic groups
ldapDefaultPPolicyDN	rw	no	PPolicy DN for password rules
ldapConnectionTimeout	rw	no	Set the `LDAP_OPT_NETWORK_TIMEOUT` connection options. Default to 15

20.10 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. 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 Provisioning API is `https://cloud.example.com/ocs/v1.php/cloud`.

All calls to OCS endpoints require the `OCS-APIRequest` header to be set to `true`.

All POST requests require the `Content-Type: application/x-www-form-urlencoded` header. (Note: Some libraries like cURL set this header automatically, others require setting the header explicitly.)

20.10.1 Instruction set for users

Add a new user

Create a new user on the Nextcloud server. Authentication is done by sending a basic HTTP authentication header.

Syntax: ocs/v1.php/cloud/users

- HTTP method: POST
- POST argument: userid - string, the required username for the new user
- POST argument: password - string, the password for the new user, leave empty to send welcome mail
- POST argument: displayName - string, the display name for the new user
- POST argument: email - string, the email for the new user, required if password empty
- POST argument: groups - array, the groups for the new user
- POST argument: subadmin - array, the groups in which the new user is subadmin
- POST argument: quota - string, quota for the new user
- POST argument: language - string, language for the new user

Status codes:

- 101 - invalid argument
- 102 - user already exists

- 103 - cannot create sub-admins for admin group
- 104 - group does not exist
- 105 - insufficient privileges for group
- 106 - no group specified (required for sub-admins)
- 107 - hint exceptions
- 108 - an email address is required, to send a password link to the user.
- 109 - sub-admin group does not exist
- 110 - required email address was not provided
- 111 - could not create non-existing user ID

Example

```
$ curl -X POST http://admin:secret@example.com/ocs/v1.php/cloud/users -d userid="Frank
↪" -d password="frankspassword" -H "OCS-APIRequest: true"
```

- Creates the user `Frank` with password `frankspassword`
- optionally groups can be specified by one or more `groups[]` query parameters: `URL -d groups[]="admin" -D groups[]="Team1"`

XML output

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

Search/get users

Retrieves a list of users from the Nextcloud server. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users

- HTTP method: GET
- url arguments: search - string, optional search string
- url arguments: limit - int, optional limit value
- url arguments: offset - int, optional offset value

Status codes:

- 100 - successful

Example

```
$ curl -X GET http://admin:secret@example.com/ocs/v1.php/cloud/users?search=Frank -H
↪"OCS-APIRequest: true"
```

- Returns list of users matching the search string.

XML output

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

Get data of a single user

Retrieves information about a single user. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}

- HTTP method: GET

Status codes:

- 100 - successful

Example

```
$ curl -X GET http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank -H "OCS-
↪APIRequest: true"
```

- Returns information on the user `Frank`

XML output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>100</statuscode>
    <status>ok</status>
  </meta>
  <data>
    <enabled>true</enabled>
    <id>Frank</id>
    <quota>0</quota>
    <email>frank@example.org</email>
```

(continues on next page)

(continued from previous page)

```
    <displayname>Frank K.</displayname>
    <display-name>Frank K.</display-name>
    <phone>0123 / 456 789</phone>
    <address>Foobar 12, 12345 Town</address>
    <website>https://nextcloud.com</website>
    <twitter>Nextcloud</twitter>
    <groups>
     <element>group1</element>
     <element>group2</element>
    </groups>
  </data>
</ocs>
```

Edit data of a single user

Edits attributes related to a user. Users are able to edit email, displayname and password; admins can also edit the quota value. Further restrictions may apply, check the *List of editable data fields* endpoint. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}

- HTTP method: PUT
- PUT argument: key, the field to edit:
  - email
  - quota
  - displayname
  - display (**deprecated** use *displayname* instead)
  - phone
  - address
  - website
  - twitter
  - password
- PUT argument: value, the new value for the field

Status codes:

- 101 - invalid argument
- 107 - password policy (hint exception)
- 112 - Setting the password is not supported by the users backend
- 113 - editing field not allowed / field doesn't exist

Examples

```
$ curl -X PUT http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank -d key=
↪"email" -d value="franksnewemail@example.org" -H "OCS-APIRequest: true"
```

- Updates the email address for the user `Frank`

```
$ curl -X PUT http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank -d key=
↪"quota" -d value="100MB" -H "OCS-APIRequest: true"
```

- Updates the quota for the user `Frank`

XML output

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

List of editable data fields

Edits attributes related to a user. Users are able to edit email, displayname and password; admins can also edit the quota value. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/user/fields

- HTTP method: GET

Status codes:

- 100 - successful

Examples

```
$ curl -X GET http://admin:secret@example.com/ocs/v1.php/cloud/user/fields -H "OCS-
↪APIRequest: true"
```

- Gets the list of fields

XML output

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <status>ok</status>
  <statuscode>100</statuscode>
  <message>OK</message>
 </meta>
 <data>
  <element>displayname</element>
  <element>email</element>
  <element>phone</element>
  <element>address</element>
  <element>website</element>
  <element>twitter</element>
 </data>
</ocs>
```

Disable a user

Disables a user on the Nextcloud server so that the user cannot login anymore. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}/disable

- HTTP method: PUT

Statuscodes:

- 100 - successful
- 101 - failure

Example

```
$ curl -X PUT http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/disable -H
↪"OCS-APIRequest: true"
```

- Disables the user `Frank`

XML output

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

Enable a user

Enables a user on the Nextcloud server so that the user can login again. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}/enable

- HTTP method: PUT

Statuscodes:

- 100 - successful
- 101 - failure

Example

```
$ curl -X PUT http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/enable -H
↪"OCS-APIRequest: true"
```

- Enables the user `Frank`

XML output

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

Delete a user

Deletes a user from the Nextcloud server. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}

- HTTP method: DELETE

Statuscodes:

- 100 - successful
- 101 - failure

Example

```
$ curl -X DELETE http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank -H "OCS-
↪APIRequest: true"
```

- Deletes the user `Frank`

XML output

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

Get user's groups

Retrieves a list of groups the specified user is a member of. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}/groups

- HTTP method: GET

Status codes:

- 100 - successful

Example

```
$ curl -X GET http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/groups -H
↪"OCS-APIRequest: true"
```

- Retrieves a list of groups of which `Frank` is a member

XML output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>100</statuscode>
    <status>ok</status>
  </meta>
  <data>
    <groups>
      <element>admin</element>
      <element>group1</element>
    </groups>
  </data>
</ocs>
```

Add user to group

Adds the specified user to the specified group. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}/groups

- HTTP method: POST
- POST argument: groupid, string - the group to add the user to

Status codes:

- 100 - successful
- 101 - no group specified
- 102 - group does not exist
- 103 - user does not exist
- 104 - insufficient privileges
- 105 - failed to add user to group

Example

```
$ curl -X POST http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/groups -d␣
↪groupid="newgroup" -H "OCS-APIRequest: true"
```

- Adds the user `Frank` to the group `newgroup`

XML output

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

Remove user from group

Removes the specified user from the specified group. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}/groups

- HTTP method: DELETE
- DELETE argument: groupid, string - the group to remove the user from

Status codes:

- 100 - successful
- 101 - no group specified
- 102 - group does not exist
- 103 - user does not exist
- 104 - insufficient privileges
- 105 - failed to remove user from group

Example

```
$ curl -X DELETE http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/groups -
↪d groupid="newgroup" -H "OCS-APIRequest: true"
```

- Removes the user `Frank` from the group `newgroup`

XML output

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

Promote user to subadmin

Makes a user the subadmin of a group. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}/subadmins

- HTTP method: POST
- POST argument: groupid, string - the group of which to make the user a subadmin

Status codes:

- 100 - successful
- 101 - user does not exist
- 102 - group does not exist
- 103 - unknown failure

Example

```
$ curl -X POST https://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/
↪subadmins -d groupid="group" -H "OCS-APIRequest: true"
```

- Makes the user `Frank` a subadmin of the `group` group

XML output

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

Demote user from subadmin

Removes the subadmin rights for the user specified from the group specified. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}/subadmins

- HTTP method: DELETE
- DELETE argument: groupid, string - the group from which to remove the user's subadmin rights

Status codes:

- 100 - successful
- 101 - user does not exist
- 102 - user is not a subadmin of the group / group does not exist
- 103 - unknown failure

Example

```
$ curl -X DELETE https://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/
↪subadmins -d groupid="oldgroup" -H "OCS-APIRequest: true"
```

- Removes `Frank's` subadmin rights from the `oldgroup` group

XML output

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

Get user's subadmin groups

Returns the groups in which the user is a subadmin. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}/subadmins

- HTTP method: GET

Status codes:

- 100 - successful
- 101 - user does not exist
- 102 - unknown failure

Example

```
$ curl -X GET https://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/subadmins␣
↪-H "OCS-APIRequest: true"
```

- Returns the groups of which `Frank` is a subadmin

XML output

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

Resend the welcome email

The request to this endpoint triggers the welcome email for this user again.

Syntax: ocs/v1.php/cloud/users/{userid}/welcome

- HTTP method: POST

Status codes:

- 100 - successful
- 101 - email address not available
- 102 - sending email failed

Example

```
$ curl -X POST https://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/welcome -
↪H "OCS-APIRequest: true"
```

- Sends the welcome email to `Frank`

XML output

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

20.10.2 Instruction set for groups

Search/get groups

Retrieves a list of groups from the Nextcloud server. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/groups

- HTTP method: GET
- url arguments: search - string, optional search string
- url arguments: limit - int, optional limit value
- url arguments: offset - int, optional offset value

Status codes:

- 100 - successful

Example

```
$ curl -X GET http://admin:secret@example.com/ocs/v1.php/cloud/groups?search=adm -H
↪"OCS-APIRequest: true"
```

- Returns list of groups matching the search string.

XML output

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

Create a group

Adds a new group. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/groups

- HTTP method: POST
- POST argument: groupid, string - the new groups name

Status codes:

- 100 - successful
- 101 - invalid input data
- 102 - group already exists
- 103 - failed to add the group

Example

```
$ curl -X POST http://admin:secret@example.com/ocs/v1.php/cloud/groups -d groupid=
↪"newgroup" -H "OCS-APIRequest: true"
```

- Adds a new group called `newgroup`

XML output

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

(continues on next page)

(continued from previous page)

```
  </meta>
  <data/>
</ocs>
```

Get members of a group

Retrieves a list of group members. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/groups/{groupid}

- HTTP method: GET

Status codes:

- 100 - successful

Example

```
$ curl -X GET http://admin:secret@example.com/ocs/v1.php/cloud/groups/admin -H "OCS-
↪APIRequest: true"
```

- Returns a list of users in the `admin` group

XML output

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

Get subadmins of a group

Returns subadmins of the group. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/groups/{groupid}/subadmins

- HTTP method: GET

Status codes:

- 100 - successful
- 101 - group does not exist
- 102 - unknown failure

Example

```
$ curl -X GET https://admin:secret@example.com/ocs/v1.php/cloud/groups/mygroup/
↪subadmins -H "OCS-APIRequest: true"
```

- Return the subadmins of the group: `mygroup`

XML output

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

Edit data of a single group

Edits attributes related to a group. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/groups/{groupid}

- HTTP method: PUT
- PUT argument: key, string - the field to edit:
  - displayname
- PUT argument: value, string - the new value for the field

Status codes:

- 100 - successful
- 101 - not supported by backend

Examples

```
$ curl -X PUT http://admin:secret@example.com/ocs/v1.php/cloud/groups/mygroup -d key=
↪"displayname" -d value="My Group Name" -H "OCS-APIRequest: true"
```

- Updates the display name for the group `mygroup`

XML output

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

(continues on next page)

(continued from previous page)

```
  <data/>
</ocs>
```

Delete a group

Removes a group. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/groups/{groupid}

- HTTP method: DELETE

Status codes:

- 100 - successful
- 101 - group does not exist
- 102 - failed to delete group

Example

```
$ curl -X DELETE http://admin:secret@example.com/ocs/v1.php/cloud/groups/mygroup -H
↪"OCS-APIRequest: true"
```

- Delete the group `mygroup`

XML output

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

20.11 Profiles

The user profile displays information about an account. Profiles are enabled by default.

Users can enable or disable their own profile in **Personal settings** under **Personal info**.

As an administrator, you can:

- set the default profile state for new users, and
- disable profiles globally.

Profile data can also be used by other features (for example the *system address book*), but what is exposed depends on privacy controls.

> **Note**
>
> Profile visibility is layered.

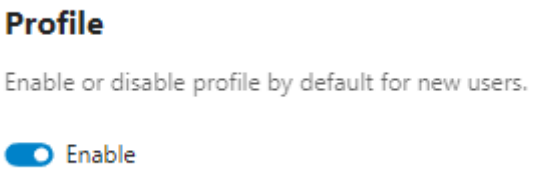
- **Profile enablement** determines if the profile feature is active at all.
- **Profile field visibility settings** control whether a field is shown.
- **Account property scopes** (for example `private`, `local`, `federated`, `published`) define the intended audience for each property.
- **Discovery restrictions** (for example sharing/autocomplete enumeration rules) can further reduce what other accounts can find or see.

In short: effective visibility is the most restrictive result of all applicable controls.

20.11.1 Configuration

Set the profile default for new users

In **Administration settings -> Basic settings**, use the profile default toggle.



You can also set this via `occ`:

```
occ config:app:set settings profile_enabled_by_default --value="0"
```

Use `--value="1"` to enable it by default again.

See *Using the occ command* for more `occ` usage details.

Disable profiles globally

To disable profile functionality for all users, add this to `config.php`:

```
'profile.enabled' => false,
```

20.11.2 Profile field visibility settings

Each profile field has its own **profile visibility** setting (stored per user in the profile configuration):

- **Show to everyone** (`show`): visible to anyone, including unauthenticated visitors, *subject to the field's property scope*.
- **Show to logged in accounts only** (`show_users_only`): visible only to authenticated users, *subject to the field's property scope*.
- **Hide** (`hide`): never shown on profile surfaces regardless of property scope.

These correspond to the visibility options in **Personal settings -> Personal info -> Edit your Profile visibility**.

Important

Effective visibility is the most restrictive result of **both** controls: the profile visibility setting and the property scope.

Defaults

Most profile fields default to **Show to everyone**, while some contact-related fields (e.g. address, email, and phone) default to **Show to logged in accounts only**.

Administrators should note that these defaults are independent from the default *property scopes* described below.

Server configuration preset categories can override these initial defaults:

- **Shared**, **School**, and **University**: address, email, and phone default to **Hide**.
- **Private**, **Family**, and **Club**: email defaults to **Show to everyone**.
- **Small**, **Medium**, and **Large**: email and phone default to **Show to everyone**.

These are initial per-user profile-visibility values. Existing stored profile visibility settings are not changed merely by selecting or changing a preset.

20.11.3 Property visibility scopes

User properties (Display name, Address, Website, Role, etc.) have visibility scopes: Private, Local, Federated, Published.

These scopes are evaluated per attribute. A profile being reachable does not imply that all its attributes are visible.

The visibility scopes are:

Private
: The most restrictive level. Data is hidden from public profiles, federation, and public lookup. On the local server, it is only shown in specific features and typically only to authenticated users who have a recognized relationship with the account owner (for example, as a known contact).

Local
: Contact details visible on the local instance and in some public contexts where profile/account attributes are required (for example owner/uploader metadata). Not shared to federated servers and not published to the public lookup server.

Federated
: Contact details visible on the local instance, in relevant public contexts, and on trusted federated servers.

Published
: Contact details visible on the local instance, in relevant public contexts, on trusted federated servers, and published to the public lookup server.

Note

Public lookup server: a public directory used to find users across Nextcloud instances. Only profile fields marked Published may be exposed there.

Note

Not all fields are eligible for lookup publication even if their scope is set to `Published`. Some fields are intentionally never published (for example Biography, Headline, Organisation, Role, Birthdate).

In other words: `Published` is necessary but not always sufficient for lookup publication.

Important

A reachable profile does not mean all attributes are public. Each attribute is filtered by its own scope, and effective visibility can also depend on the consuming feature.

Important

On profile surfaces, the effective visibility is the more restrictive of profile-visibility settings and property scope.

Scope visibility matrix

Scope	User themself [1]	Other users on same local instance	Public contexts (feature-dependent)	Trusted federation	Public lookup server
Private	Yes	Limited on profile surfaces: authenticated + known-user relation required [3]	No	No	No
Local	Yes	Yes	Yes (where applicable) [2]	No	No
Federated	Yes	Yes	Yes (where applicable) [2]	Yes	No
Published	Yes	Yes	Yes (where applicable) [2]	Yes	Yes

Notes:

1. Scope primarily governs exposure to others; owner access follows account/endpoint behavior.
2. Public-context visibility depends on feature path; scope alone does not guarantee display.
3. Some non-profile surfaces may exclude Private-scoped properties entirely (for example generated system address book cards), even for authenticated users.

Note

The matrix describes **profile visibility behavior**. Other consuming features may apply additional filtering and may not expose Private-scoped properties at all.

Known-user rule for `Private` scope

For `Private` properties, Nextcloud may allow visibility on specific local feature paths only when the requester is considered a *known user* of the target user.

In practical terms, this relation is derived by server-side *known-contact matching*. In current Nextcloud versions this matching is primarily established via **phone-number matching** (for example through the Talk mobile contact integration), and it is directional (e.g., Alice might be known to Bob, but Bob isn't necessarily known to Alice). Users are always known to themselves.

What local users can see

A common question is what one user can see about another user on the same instance.

In general, profile visibility is controlled by each property's scope, but the exact UI/API surface depends on the consuming feature (for example profile page, share dialogs, search, mentions, Contacts, and other integrations).

For local users on the same instance:

- `Private`: not generally visible to all local users; visibility is restricted on applicable paths to authenticated users that satisfy the known-user relation and other feature constraints.
- `Local`: visible on the local instance.
- `Federated`: visible on the local instance (and also shared with trusted federated servers).
- `Published`: visible on the local instance (and also federated + public lookup).

Note

System address book exposure is scope-aware and context-aware: private/empty-scope properties are excluded from generated cards, and federated reads strip local-scoped properties.

How to verify scope behavior

Because effective visibility can vary by feature path, administrators should verify scope behavior in their own deployment.

Recommended test procedure:

1. Create test users:
   - `alice` (target profile owner)
   - `bob` (authenticated local user)
   - `charlie` (second local user for control)
2. As `alice`, set distinct test values for profile fields and assign different scopes where possible (for example Private vs Local via the UI, and Federated/Published via API/administrative tooling if supported in your deployment).
3. Verify as `alice`:
   - Confirm owner-visible values as expected.
4. Verify as `bob` (authenticated local user):
   - Check the feature paths used in your instance (for example profile page, user card, share dialog, search, mentions, Contacts integrations).
   - Confirm `Local/Federated/Published` fields are visible where expected.
   - Confirm `Private` fields are visible only on paths that satisfy the known-user relation and other feature constraints.
5. Verify as unauthenticated user (private browser session):
   - Confirm only public-appropriate fields are visible.
6. Verify federation/publication behavior (if enabled):
   - From a trusted federated server, confirm Federated/Published behavior.
   - Confirm only Published fields are exposed to the public lookup server.
7. Re-test with a newly created user after changing `account_manager.default_property_scope`:

- Confirm new defaults apply only to newly initialized accounts.
- Confirm existing users retain stored scopes unless explicitly changed.

Scope defaults and precedence

Visibility is determined per property using this order:

1. **Server defaults** from `OC\Accounts\AccountManager::DEFAULT_SCOPES`.
2. **Administrator default override** via `account_manager.default_property_scope`.
3. **User-set value** in personal/profile settings (subject to server-side constraints).

Practical implications:

- Admin overrides in `account_manager.default_property_scope` are applied at account initialization and therefore affect **new users**.
- Existing users keep already stored scopes unless changed explicitly.
- `PROPERTY_DISPLAYNAME` and `PROPERTY_EMAIL` cannot be `Private`; server-side validation/enforcement requires at least `Local`.

Default scope values (reference)

Default values are defined in server code and may change over time. The authoritative source is the `DEFAULT_SCOPES` constant in `OC\Accounts\AccountManager`: latest source.

Example defaults (verify against your deployed version):

Property	Default visibility scope
Display name	Federated
Address	Local
Website	Local
Email	Federated
Avatar	Federated
Phone	Local
Twitter	Local
Bluesky	Local
Fediverse	Local
Organisation	Local
Role	Local
Headline	Local
Biography	Local
Birthdate	Local
Pronouns	Federated

Override default scopes in `config.php`

To override one or several default visibility scopes for *new users*, use `account_manager.default_property_scope` (default: empty array):

```
'account_manager.default_property_scope' => [
  \OCP\Accounts\IAccountManager::PROPERTY_PHONE => \OCP\Accounts\
↪IAccountManager::SCOPE_PRIVATE,
  \OCP\Accounts\IAccountManager::PROPERTY_ROLE => \OCP\Accounts\
```

(continues on next page)

(continued from previous page)

```
↪IAccountManager::SCOPE_FEDERATED,
]
```

In the above example, phone and role are overwritten to `Private` and `Federated` respectively.

> **Note**
>
> Use `\OCP\Accounts\IAccountManager` constants for both property keys and scope values.

FAQ: How to lock profile visibility down

If your goal is maximum privacy:

1. Disable profiles globally (strictest option):

   ```
   'profile.enabled' => false,
   ```

   Effect:

   - Profile functionality is removed.
   - Profile-based discoverability/usability features are reduced accordingly.

2. If profiles must remain enabled, set restrictive defaults for new users:

   ```
   'account_manager.default_property_scope' => [
     \OCP\Accounts\IAccountManager::PROPERTY_ADDRESS => \OCP\Accounts\
   ↪IAccountManager::SCOPE_PRIVATE,
     \OCP\Accounts\IAccountManager::PROPERTY_PHONE => \OCP\Accounts\
   ↪IAccountManager::SCOPE_PRIVATE,
     \OCP\Accounts\IAccountManager::PROPERTY_WEBSITE => \OCP\Accounts\
   ↪IAccountManager::SCOPE_PRIVATE,
     \OCP\Accounts\IAccountManager::PROPERTY_TWITTER => \OCP\Accounts\
   ↪IAccountManager::SCOPE_PRIVATE,
     \OCP\Accounts\IAccountManager::PROPERTY_BLUESKY => \OCP\Accounts\
   ↪IAccountManager::SCOPE_PRIVATE,
     \OCP\Accounts\IAccountManager::PROPERTY_FEDIVERSE => \OCP\Accounts\
   ↪IAccountManager::SCOPE_PRIVATE,
     \OCP\Accounts\IAccountManager::PROPERTY_ORGANISATION => \OCP\Accounts\
   ↪IAccountManager::SCOPE_PRIVATE,
     \OCP\Accounts\IAccountManager::PROPERTY_ROLE => \OCP\Accounts\
   ↪IAccountManager::SCOPE_PRIVATE,
     \OCP\Accounts\IAccountManager::PROPERTY_HEADLINE => \OCP\Accounts\
   ↪IAccountManager::SCOPE_PRIVATE,
     \OCP\Accounts\IAccountManager::PROPERTY_BIOGRAPHY => \OCP\Accounts\
   ↪IAccountManager::SCOPE_PRIVATE,
     \OCP\Accounts\IAccountManager::PROPERTY_BIRTHDATE => \OCP\Accounts\
   ↪IAccountManager::SCOPE_PRIVATE,
     \OCP\Accounts\IAccountManager::PROPERTY_PRONOUNS => \OCP\Accounts\
   ↪IAccountManager::SCOPE_PRIVATE,
     \OCP\Accounts\IAccountManager::PROPERTY_AVATAR => \OCP\Accounts\
   ↪IAccountManager::SCOPE_PRIVATE,
   ]
   ```

Notes:

- `PROPERTY_DISPLAYNAME` and `PROPERTY_EMAIL` cannot be set to `Private`; server-side enforcement requires at least `Local`.
- Defaults apply to **new users**. Existing users keep stored scopes unless changed.

What becomes limited when you lock it down?

With more restrictive scopes (especially `Private`), expect reduced visibility in:

- User discovery/search/user cards
- Share dialogs and mention/autocomplete context
- Public/share-related contexts where account metadata may be shown
- Federated visibility of profile attributes
- Public lookup publication (only `Published` appears there)

In short: tighter privacy reduces profile-based convenience and discoverability.

20.11.4 Scopes and existing users

The `account_manager.default_property_scope` config only applies to **new** users. Existing users keep their stored scopes.

There is currently no admin-level mechanism to bulk-change scopes for existing users. The OCS provisioning API only allows users to change their **own** scopes — admins cannot set scopes on behalf of other users.

Users can update their own scopes in **Personal settings → Personal info → Edit your Profile visibility**, or via the API:

```
curl -s -u alice:password -X PUT \
  "https://cloud.example.com/ocs/v2.php/cloud/users/alice" \
  -H "OCS-APIRequest: true" \
  -d "key=phoneScope&value=v2-private"
```

The scope key is the property name with `Scope` appended. Available keys:

`displaynameScope`, `emailScope`, `phoneScope`, `addressScope`, `websiteScope`, `twitterScope`, `blueskyScope`, `fediverseScope`, `organisationScope`, `roleScope`, `headlineScope`, `biographyScope`, `birthdateScope`, `avatarScope`, `pronounsScope`

Allowed scope values:

- `v2-private` — Private
- `v2-local` — Local
- `v2-federated` — Federated
- `v2-published` — Published

> **Note**
>
> `displaynameScope` and `emailScope` cannot be set to `v2-private`. The server enforces a minimum of `v2-local` for these properties.

20.11.5 See also

- *File Sharing* — sharing and autocomplete settings that interact with profile visibility
- User manual: Personal settings — user-facing profile and personal info settings

20.12 User authentication with OpenID Connect

Nextcloud users can authenticate via an external identity provider. Nextcloud can also be an identity provider itself.

20.12.1 Authentication in Nextcloud

The OpenID Connect user backend app makes it possible for users to authenticate using external Oidc identity providers.

This app can optionally be in charge of user provisioning (by creating users when they first connect) or rely on other user backends and only take care of authentication.

More details in the project's README

20.12.2 Using Nextcloud as an identity provider

The OIDC Identity Provider community app can be installed to make Nextcloud an identity provider for other services.

This app will allow any Nextcloud user (managed by any user backend) to authenticate during an Oidc login flow. This is useful if you want your Nextcloud instance to be the authority regarding authentication and user profile data among multiple services.

20.12.3 Bearer token validation

Nextcloud can accept Oidc ID tokens and access tokens as valid bearer token for API requests. If using an external identity provider, only the `user_oidc` app is necessary.

If Nextcloud is the identity provider, you will naturally need the `oidc` app to make Nextcloud an Oidc provider, and also the `user_oidc` app because it will take care of validating API requests authentication. In user_oidc, the `oidc_provider_bearer_validation` config flag needs to be set to true so `user_oidc` knows it needs to ask the `oidc` app to validate the received bearer tokens.

More details on bearer token validation

CHAPTER

TWENTYONE

DESKTOP CLIENTS

Available for Windows, macOS, and various Linux distributions, the Nextcloud Desktop Sync client enables you to:

- Specify one or more directories on your computer that you want to synchronize to the Nextcloud server.
- Always have the latest files synchronized, wherever they are located.

Your files are always automatically synchronized between your Nextcloud server, computer and mobile device.

21.1 Desktop Client Deployment and Setup

This chapter describes administrator-facing deployment and setup options for the Nextcloud Desktop client.

These options are intended for managed rollouts and other administrative scenarios, such as deployment scripts, software-management platforms, login scripts, MDM or RMM workflows, and other automated installation or configuration processes.

Depending on your deployment goals, you can use the desktop client in one of several ways:

- customize Windows installation behavior,
- create an account automatically during deployment without user interaction, or
- prefill or constrain the interactive setup wizard.

21.1.1 Choosing the right approach

Use the option that best matches your deployment goal:

I want to customize how the client is installed on managed Windows systems.
See *Advanced Windows deployment options*.

I want to silently create an account during deployment.
See *Non-interactive account provisioning*.

I want users to complete setup interactively, but with predefined values or restrictions.
See *Interactive wizard preconfiguration*.

21.1.2 Advanced Windows deployment options

If you just want to install the desktop client on your local system, simply launch the `.msi` file and follow the installation wizard.

The following options are intended for advanced Windows installations, for example when automating deployment or customizing installed features.

> **Note**
>
> Windows installation customization controls how the client is installed. It does not by itself create or configure a desktop-client account. If you want to configure an account after installation, see *Non-interactive account provisioning* or *Interactive wizard preconfiguration*.

Features

The MSI installer provides several features that can be installed or removed individually, which you can also control via the command line. If you are automating the installation, run the following command:

```
msiexec /passive /i Nextcloud-x.y.z-x64.msi
```

The command installs the client into the default location with the default features enabled.

If you want to disable, for example, desktop shortcut icons, you can change the command to the following:

```
msiexec /passive /i Nextcloud-x.y.z-x64.msi REMOVE=DesktopShortcut
```

See the following table for a list of available features:

Feature	Enabled by default	Description	Property to disable
Client	Yes, required	The actual client	
DesktopShortcut	Yes	Adds a shortcut to the desktop	`NO_DESKTOP_SHORTCUT`
StartMenuShortcuts	Yes	Adds a shortcut to the start menu	`NO_START_MENU_SHORTCUTS`
ShellExtensions	Yes	Adds Explorer integration	`NO_SHELL_EXTENSIONS`

Installation

You can also choose to only install the client itself by using the following command:

```
msiexec /passive /i Nextcloud-x.y.z-x64.msi ADDDEFAULT=Client
```

For example, if you want to install everything except the `DesktopShortcut` and the `ShellExtensions` feature, you have two possibilities:

1. Explicitly name all the features you actually want to install (whitelist), where `Client` is always installed anyway:

   ```
   msiexec /passive /i Nextcloud-x.y.z-x64.msi ADDDEFAULT=StartMenuShortcuts
   ```

2. Pass the `NO_DESKTOP_SHORTCUT` and `NO_SHELL_EXTENSIONS` properties:

   ```
   msiexec /passive /i Nextcloud-x.y.z-x64.msi NO_DESKTOP_SHORTCUT="1" NO_SHELL_
   ↪EXTENSIONS="1"
   ```

> **Note**
>
> The Nextcloud `.msi` remembers these properties, so you do not need to specify them again on upgrades.

> **Note**
>
> You cannot use these to change the installed features after installation. If you want to do that, see the next section.

Changing installed features

You can change the installed features later by using the `REMOVE` and `ADDDEFAULT` properties.

To add the desktop shortcut later, run the following command:

```
msiexec /passive /i Nextcloud-x.y.z-x64.msi ADDDEFAULT="DesktopShortcut"
```

To remove it, run the following command:

```
msiexec /passive /i Nextcloud-x.y.z-x64.msi REMOVE="DesktopShortcut"
```

Windows keeps track of the installed features, and `REMOVE` or `ADDDEFAULT` affects only the specified features.

> **Note**
>
> You cannot specify `REMOVE` on initial installation, as it will disable all features.

Installation folder

You can adjust the installation folder by specifying the `INSTALLDIR` property:

```
msiexec /passive /i Nextcloud-x.y.z-x64.msi INSTALLDIR="C:\Program Files\Non Standard␣
↪Nextcloud Client Folder"
```

Be careful when using PowerShell instead of `cmd.exe`; getting the whitespace escaping right can be tricky.

Specifying `INSTALLDIR` like this only works on first installation; you cannot simply re-run the `.msi` with a different path. If you still need to change it, uninstall the client first and then reinstall it to the new location.

Disabling automatic updates

To disable automatic updates, pass the `SKIPAUTOUPDATE` property:

```
msiexec /passive /i Nextcloud-x.y.z-x64.msi SKIPAUTOUPDATE="1"
```

Launch after installation

To launch the client automatically after installation, pass the `LAUNCH` property:

```
msiexec /i Nextcloud-x.y.z-x64.msi LAUNCH="1"
```

This option also removes the checkbox that lets users decide whether to launch the client during non-passive or non-quiet installations.

> **Note**
>
> This option does not have any effect without a GUI.

No reboot after installation

The client schedules a reboot after installation to make sure the Explorer extension is correctly loaded or unloaded. If you are handling reboots yourself, you can set the `REBOOT` property:

```
msiexec /i Nextcloud-x.y.z-x64.msi REBOOT=ReallySuppress
```

This makes `msiexec` exit with error `ERROR_SUCCESS_REBOOT_REQUIRED` (3010). If your deployment tooling interprets this as an actual error and you want to avoid that, you may want to set `DO_NOT_SCHEDULE_REBOOT` instead:

```
msiexec /i Nextcloud-x.y.z-x64.msi DO_NOT_SCHEDULE_REBOOT="1"
```

21.1.3 Non-interactive account provisioning

The Nextcloud Desktop client supports non-interactive account provisioning from the command line.

This is intended for deployment automation and other managed-installation scenarios where an administrator wants to create a desktop-client account without requiring the user to go through the graphical setup wizard.

In a typical deployment workflow, you deploy the desktop client package first, then invoke the desktop client executable with provisioning parameters as part of your automation.

When the required parameters are provided, the client attempts to create an account and store it in the client's normal configuration so that subsequent launches behave as if the account had been added through the graphical user interface.

If account creation succeeds, the client exits with code `0`. If account creation fails, the client exits with code `1`.

Detailed status and failure information is written to the desktop client logs.

> **Note**
>
> This workflow is distinct from interactive wizard preconfiguration. If you want to prefill or constrain the setup wizard instead of directly provisioning an account, see *Interactive wizard preconfiguration*.

Required parameters

The following parameters are required for non-interactive account provisioning:

`--userid`
: The user ID to configure in the desktop client.

`--apppassword`
: The app password to use for authentication.

  App passwords should be used instead of a user's regular login password. See the server documentation for the login flow and app password generation process.

`--serverurl`
: The base URL of the Nextcloud server to use for the account.

Optional parameters

`--localdirpath`
: The local path to use for the sync folder.

  If omitted, the desktop client chooses its normal default sync-folder location for the platform.

  If this option is provided and the target directory already exists and is not empty, account provisioning fails.

  If this option is provided and the directory does not yet exist, the client attempts to create it.

--remotedirpath
: The remote path to sync.

  If omitted, the default is `/` (the account root on the server).

  Example: if the server contains folders such as `/Photos`, `/Documents`, and `/Music`, specifying `/Music` creates a sync connection for that remote subfolder instead of the entire account root.

--isvfsenabled
: Controls whether the created sync connection should use virtual files.

  Use `1` to enable virtual files and `0` to disable them.

  Virtual file support depends on the client build, operating system, and runtime availability of the required virtual-file functionality. Only enable this option on platforms and builds where virtual files are supported.

--confdir
: Overrides the configuration directory used by the client process.

  This is a general client option rather than a provisioning-specific option. It is only needed if you intentionally want to use a non-default configuration directory.

  If you use this option during provisioning, subsequent desktop-client launches must use the same configuration directory, or the provisioned account may not appear in later GUI sessions.

Behavior

When started with the required provisioning parameters, the desktop client attempts to:

1. validate the provided command-line parameters,
2. create the local sync folder if needed,
3. authenticate to the server using the provided app password,
4. verify server access, and
5. store the resulting account in the client's normal configuration.

On later launches, the configured account is available as a normal desktop-client account, provided the client is started with the same configuration context.

Important

The desktop client does not mount the server into the local path like a network filesystem. It creates a sync connection. An empty local folder does not by itself mean that provisioning failed.

Common failure cases

Account provisioning can fail for reasons including:

- one or more required parameters are missing or invalid,
- an account for the same user and server already exists,
- the specified local sync folder already exists and is not empty,
- the specified local sync folder could not be created,
- the server URL is incorrect,
- the app password is invalid,
- the authenticated request is redirected unexpectedly,

- the server denies access to the configured account, or
- the server returns an invalid response to the authenticated WebDAV request.

In these cases, the client exits with code `1` and writes more detailed information to the logs.

Examples

Windows:

```
"C:\Program Files\Nextcloud\nextcloud.exe" ^
  --userid admin ^
  --apppassword␣
↪Jliy12356785jxnHa2ZCiZ9MX48ncECwDso95Pq3a5HABjY34ZvhZiXrPfpKWUg7aOHAX5 ^
  --serverurl https://cloud.example.com ^
  --localdirpath "D:\Nextcloud-sync-folder" ^
  --remotedirpath /Music ^
  --isvfsenabled 1
```

Linux:

```
nextcloud \
  --userid admin \
  --apppassword␣
↪Jliy12356785jxnHa2ZCiZ9MX48ncECwDso95Pq3a5HABjY34ZvhZiXrPfpKWUg7aOHAX5 \
  --serverurl https://cloud.example.com \
  --localdirpath "/home/admin/Nextcloud-sync-folder" \
  --remotedirpath /Music \
  --isvfsenabled 0
```

macOS:

```
nextcloud \
  --userid admin \
  --apppassword␣
↪Jliy12356785jxnHa2ZCiZ9MX48ncECwDso95Pq3a5HABjY34ZvhZiXrPfpKWUg7aOHAX5 \
  --serverurl https://cloud.example.com \
  --localdirpath "/Users/admin/Nextcloud-sync-folder" \
  --remotedirpath /Music \
  --isvfsenabled 1
```

Recommendations

For reliable deployment automation:

- use app passwords rather than regular user passwords,
- ensure that the server URL is the correct base URL for the Nextcloud instance,
- use a new or empty local sync directory,
- validate that the target remote path exists and is accessible to the user,
- enable `--isvfsenabled` only on supported platforms and builds, and
- avoid `--confdir` unless you intentionally need a non-default configuration location.

21.1.4 Interactive wizard preconfiguration

If you want to automate the Account Setup Wizard so that users can skip entering the server URL and local sync folder path in the UI, you can use command-line parameters.

When you specify both, the desktop client's Account Setup Wizard will jump straight to opening a browser for account authentication or connection without the need to enter any of the connection details.

The local sync folder will also be selected to the one you specify instead of using the default path.

The following parameters are supported:

`--overridelocaldir`
: Specify a local directory to be used in the account setup wizard.

  Example: `/home/nextcloud-sync-folder`

`--overrideserverurl`
: Specify a server URL to use for the force override in the account setup wizard.

  Example: `https://cloud.example.com`

Behavior

These options affect the behavior of the interactive setup wizard. They do not directly create an account in the same way as non-interactive account provisioning.

Use this approach when you want the user to complete setup interactively, but want to prefill, constrain, or guide the process.

Examples

Windows:

```
"C:\Program Files\Nextcloud\nextcloud.exe" --overridelocaldir "D:/work/nextcloud-sync-
↪folder" --overrideserverurl https://cloud.example.com
```

Linux and macOS:

```
nextcloud --overridelocaldir "/home/<user>/nextcloud-sync-folder" --overrideserverurl␣
↪https://cloud.example.com
```

Important distinction from non-interactive provisioning

Use wizard preconfiguration when you still want the user to complete setup in the graphical interface.

Use non-interactive account provisioning when you want deployment automation to create the account directly.

In other words:

- non-interactive provisioning creates the account automatically;
- wizard preconfiguration influences the setup wizard, but still relies on interactive completion.

21.1.5 Summary

The Nextcloud Desktop client supports multiple administrator-facing deployment and setup workflows.

Choose the workflow that best matches your environment:

- use advanced Windows deployment options to control installation behavior on managed Windows systems,
- use non-interactive account provisioning to create desktop-client accounts automatically during deployment,

- use interactive wizard preconfiguration to guide users through a constrained or prefilled first-run setup process.

21.2 Block desktop client access at the HTTP layer

Server administrators can reject Nextcloud Desktop requests at a web server, reverse proxy, or web application firewall (WAF) by matching the HTTP `User-Agent` header.

This can be useful as a compatibility guard or an operational policy, for example to temporarily block all desktop clients or require a particular client release.

> **Warning**
>
> This is not a security boundary. The client controls the `User-Agent` header and can change or omit it. Do not use this rule as the only control when access must be prevented against a deliberately modified client.

21.2.1 Identify desktop client requests

The standard desktop client sends a header with this shape:

```
Mozilla/5.0 (Macintosh) mirall/34.0.2 (Nextcloud, macos-25.5.0 ClientArchitecture:␣
↪arm64 OsArchitecture: arm64)
```

The stable marker is `mirall/`; the value immediately after it is the desktop client version. The current client constructs this value in Utility::userAgentString().

Use a case-insensitive match for:

```
mirall/
```

Apply the rule to every request reaching the Nextcloud virtual host. Blocking only one OCS or WebDAV endpoint does not block desktop access because the client uses multiple OCS, WebDAV, login-flow, status, and provisioning endpoints.

Nextcloud Server's `USER_AGENT_CLIENT_DESKTOP` matcher also recognizes the legacy marker `csyncoC/`. To include those clients, replace `mirall/` with `(mirall|csyncoC)/` in regular-expression rules, or add a second substring rule where the configuration format does not support regular expressions.

21.2.2 Choose the policy

Block every standard desktop client

Match the case-insensitive substring:

```
mirall/
```

Block releases older than 34.0.2

Web-server regular expressions do not compare arbitrary version strings as semantic versions. The following case-insensitive expression is deliberately written for one minimum release, `34.0.2`:

```
mirall/(([0-9]|[12][0-9]|3[0-3])[.]|34[.]0[.](0|1)([^0-9]|$))
```

It has these results:

Desktop version	Result
3.14.3	blocked
33.0.0	blocked
34.0.0	blocked
34.0.1	blocked
34.0.2	allowed by this rule
34.1.0	allowed by this rule
35.0.0	allowed by this rule

The expression assumes normal release versions with three numeric components. Do not change only the version text in the expression. Build and test a new set of numeric ranges whenever the minimum release changes. A WAF that can extract the version and compare three integer fields is preferable when the minimum changes frequently.

Return `403 Forbidden`. Do not return `401 Unauthorized`: by HTTP semantics, that status asks the client to authenticate and obscures that this is an administrative policy. The desktop client is not guaranteed to display a custom response body, so the administrator should communicate the policy separately to users.

The examples below assume that Nextcloud has its own virtual host at `cloud.example.com`. If the host serves other applications, additionally scope the rule to the Nextcloud URL prefix.

21.2.3 nginx

Define one `map` in the `http` context. Enable exactly one of the two match entries:

```
http {
    map $http_user_agent $block_nextcloud_desktop {
        default 0;

        # Policy A: block every standard desktop client.
        ~*mirall/ 1;

        # Policy B: instead block releases older than 34.0.2.
        # ~*mirall/(([0-9]|[12][0-9]|3[0-3])[.]|34[.]0[.](0|1)([^0-9]|$)) 1;
    }

    server {
        server_name cloud.example.com;

        if ($block_nextcloud_desktop) {
            return 403 "Nextcloud desktop client access is disabled.\n";
        }

        # The existing Nextcloud configuration follows here.
    }
}
```

The `map` directive is valid only in the `http` context. The resulting variable is then checked in the Nextcloud `server` block. A conditional containing only `return` is sufficient here; no URI rewrite is involved.

Validate and reload the configuration with the commands appropriate to the installation, commonly:

```
nginx -t
systemctl reload nginx
```

See the nginx documentation for map and return.

21.2.4 Apache HTTP Server 2.4

Place `SetEnvIfNoCase` in the Nextcloud virtual host and deny requests carrying the resulting environment variable. Enable exactly one `SetEnvIfNoCase` line:

```
<VirtualHost *:443>
    ServerName cloud.example.com

    # Policy A: block every standard desktop client.
    SetEnvIfNoCase User-Agent "mirall/" block_nextcloud_desktop

    # Policy B: instead block releases older than 34.0.2.
    # SetEnvIfNoCase User-Agent "mirall/(([0-9]|[12][0-9]|3[0-3])[.]|34[.]0[.](0|1)([^
↪0-9]|$))" block_nextcloud_desktop

    <Location "/">
        <RequireAll>
            Require all granted
            Require not env block_nextcloud_desktop
        </RequireAll>
    </Location>

    # The existing Nextcloud configuration follows here.
</VirtualHost>
```

For a subdirectory installation, such as `https://cloud.example.com/nextcloud`, use `<Location "/nextcloud">` instead. The modules `mod_setenvif` and `mod_authz_core` must be enabled.

Validate and reload the configuration with the commands appropriate to the installation, commonly:

```
apachectl configtest
systemctl reload apache2
```

The Apache documentation describes this SetEnvIfNoCase and Require blocking pattern and also notes that a client can circumvent a `User-Agent` rule.

21.2.5 Caddy 2

Place the matcher and response before the existing Nextcloud handler. Enable exactly one matcher line:

```
cloud.example.com {
    # Policy A: block every standard desktop client.
    @blockedNextcloudDesktop header_regexp User-Agent (?i)mirall/

    # Policy B: instead block releases older than 34.0.2.
    # @blockedNextcloudDesktop header_regexp User-Agent (?i)mirall/(([0-9]|[12][0-
↪9]|3[0-3])[.]|34[.]0[.](0|1)([^0-9]|$))

    respond @blockedNextcloudDesktop "Nextcloud desktop client access is disabled."␣
↪403

    # The existing Nextcloud file_server, php_fastcgi, or reverse_proxy
```

(continues on next page)

(continued from previous page)

```
    # directives follow here.
}
```

Caddy uses RE2 regular expressions. The example therefore uses capturing groups rather than PCRE-only constructs.

Validate and reload the configuration with the commands appropriate to the installation, commonly:

```
caddy validate --config /etc/caddy/Caddyfile
systemctl reload caddy
```

See the Caddy documentation for header_regexp and respond.

21.2.6 HAProxy

Add the ACL and deny action to the HTTP `frontend` that receives Nextcloud traffic. Enable exactly one `blocked_nextcloud_desktop` ACL:

```
frontend https_frontend
    bind :443 ssl crt /etc/haproxy/certs/cloud.example.com.pem

    acl nextcloud_host hdr(host) -i cloud.example.com cloud.example.com:443

    # Policy A: block every standard desktop client.
    acl blocked_nextcloud_desktop req.hdr(User-Agent) -m sub -i mirall/

    # Policy B: instead block releases older than 34.0.2.
    # acl blocked_nextcloud_desktop req.hdr(User-Agent) -m reg -i mirall/(([0-
↪9]|[12][0-9]|3[0-3])[.]|34[.]0[.](0|1)([^0-9]|$))

    http-request deny deny_status 403 if nextcloud_host blocked_nextcloud_desktop

    # The existing backend selection follows here.
```

Keep the host ACL when this frontend serves more than Nextcloud. Validate and reload the configuration with the commands appropriate to the installation, commonly:

```
haproxy -c -f /etc/haproxy/haproxy.cfg
systemctl reload haproxy
```

See the HAProxy documentation for ACLs and http-request deny.

21.2.7 Verify the rule

Test from a system that reaches the same proxy or WAF path as real users. The following probes do not require Nextcloud credentials:

```
# Must be 403 for both policies.
curl -sS -o /dev/null -w '%{http_code}\n' \
    -A 'Mozilla/5.0 (Linux) mirall/34.0.1 (Nextcloud, linux)' \
    https://cloud.example.com/status.php

# Must not be blocked by the minimum-version policy. It is still 403 when
# the block-all policy is active.
```

(continues on next page)

(continued from previous page)

```
curl -sS -o /dev/null -w '%{http_code}\n' \
    -A 'Mozilla/5.0 (Linux) mirall/34.0.2 (Nextcloud, linux)' \
    https://cloud.example.com/status.php

# Must not be blocked by either desktop rule.
curl -sS -o /dev/null -w '%{http_code}\n' \
    -A 'Mozilla/5.0 (compatible; policy-check)' \
    https://cloud.example.com/status.php
```

Also verify these operational details:

1. The rule covers all Nextcloud paths and HTTP methods, including WebDAV methods such as `PROPFIND`, `PUT`, and `DELETE`.
2. The origin server cannot be reached through another hostname or IP address that bypasses the proxy or WAF.
3. Proxy layers preserve the original `User-Agent` header until the component enforcing the rule has evaluated it.
4. Access and WAF logs record blocked requests, the matched policy, and the returned status.
5. A rollback consists of disabling the match rule and reloading the configuration; it does not require changing Nextcloud itself.

21.3 Troubleshooting

The following two general issues can result in failed synchronization:

- The server setup is incorrect.
- The client contains a bug.

When reporting bugs, it is helpful if you first determine what part of the system is causing the issue.

21.3.1 Identifying Basic Functionality Problems

Performing a general Nextcloud Server test

The first step in troubleshooting synchronization issues is to verify that you can log on to the Nextcloud web application. To verify connectivity to the Nextcloud server try logging in via your Web browser.

If you are not prompted for your username and password, or if a red warning box appears on the page, your server setup requires modification. Please verify that your server installation is working correctly.

Ensure the WebDAV API is working

If all desktop clients fail to connect to the Nextcloud Server, but access using the Web interface functions properly, the problem is often a misconfiguration of the WebDAV API.

The Nextcloud Client uses the built-in WebDAV access of the server content. Verify that you can log on to Nextcloud's WebDAV server. To verify connectivity with the Nextcloud WebDAV server:

- Open a browser window and enter the address to the Nextcloud WebDAV server.

For example, if your Nextcloud instance is installed at `http://yourserver.com/nextcloud`, your WebDAV server address is `http://yourserver.com/nextcloud/remote.php/dav`.

If you are prompted for your username and password but, after providing the correct credentials, authentication fails, please ensure that your authentication backend is configured properly.

Use a WebDAV command line tool to test

A more sophisticated test method for troubleshooting synchronization issues is to use a WebDAV command line client and log into the Nextcloud WebDAV server. One such command line client – called `cadaver` – is available for Linux distributions. You can use this application to further verify that the WebDAV server is running properly using PROPFIND calls.

As an example, after installing the `cadaver` app, you can issue the `propget` command to obtain various properties pertaining to the current directory and also verify WebDAV server connection.

21.3.2 "CSync unknown error"

If you see this error message stop your client, delete the `.sync_xxxxxxx.db` file, and then restart your client. There is a hidden `.sync_xxxxxxx.db` file inside the folder of every account configured on your client.

> **Note**
>
> Please note that this will also erase some of your settings about which files to download.

See https://github.com/owncloud/client/issues/5226 for more discussion of this issue.

21.3.3 "Connection closed" message when syncing files

This message can be caused by using chunks that are too big or time-outs that are set too liberally. You can configure the chunking behavior of the client in the config file. For example, change these settings:

`chunkSize`	`10000000` (10 MB)	Specifies the chunk size of uploaded files in bytes. The client will dynamically adjust this size within the maximum and minimum bounds (see below).
`minChunkSize`	`1000000` (1 MB)	Specifies the minimum chunk size of uploaded files in bytes.
`maxChunkSize`	`50000000` (1000 MB)	Specifies the maximum chunk size of uploaded files in bytes.
`targetChunkUploadDuration`	`6000` (1 minute)	Target duration in milliseconds for chunk uploads. The client adjusts the chunk size until each chunk upload takes approximately this long. Set to 0 to disable dynamic chunk sizing.

Setting `maxChunkSize` to 50000000, for example, will decrease the individual chunk to about 50 mb. This causes additional overhead but might be required in some situations, for example behind CloudFlare which has been seen limiting upload chunks to 100mb. In other situations, limiting `targetChunkUploadDuration` can help to avoid time-outs.

21.3.4 Connection issues with the macOS client on "insecure" connections

When using macOS devices to connect to a Nextcloud server that uses a what maybe be classified as an insecure connection (i.e. connecting to a server with a self-signed certificate, or a certificate with what Apple may consider an insufficiently secure cipher), the macOS client may not connect to the server. This is because macOS requires a valid certificate to establish a connection.

To resolve this issue, you must ensure the server is signed with a certificate that is accepted by Apple's App Transport Security requirements. More information on the requirements can be found in Apple's documentation pages.

https://developer.apple.com/documentation/security/preventing-insecure-network-connections

21.3.5 Isolating other issues

Other issues can affect synchronization of your Nextcloud files:

- If you find that the results of the synchronizations are unreliable, please ensure that the folder to which you are synchronizing is not shared with other synchronization applications.
- Synchronizing the same directory with Nextcloud and other synchronization software such as Unison, rsync, Microsoft Windows Offline Folders, or other cloud services such as Dropbox or Microsoft SkyDrive is not supported and should not be attempted. In the worst case, it is possible that synchronizing folders or files using Nextcloud and other synchronization software or services can result in data loss.
- If you find that only specific files are not synchronized, the synchronization protocol might be having an effect. Some files are automatically ignored because they are system files, other files might be ignored because their filename contains characters that are not supported on certain file systems. For more information about ignored files, see *ignored-files*.
- If you are operating your own server, and use the local storage backend (the default), make sure that Nextcloud has exclusive access to the directory.

> **Warning**
>
> The data directory on the server is exclusive to Nextcloud and must not be modified manually.

- If you are using a different file backend on the server, you can try to exclude a bug in the backend by reverting to the built-in backend.
- If you are experiencing slow upload/download speed or similar performance issues be aware that those could be caused by on-access virus scanning solutions, either on the server (like the files_antivirus app) or the client.

21.3.6 Log Files

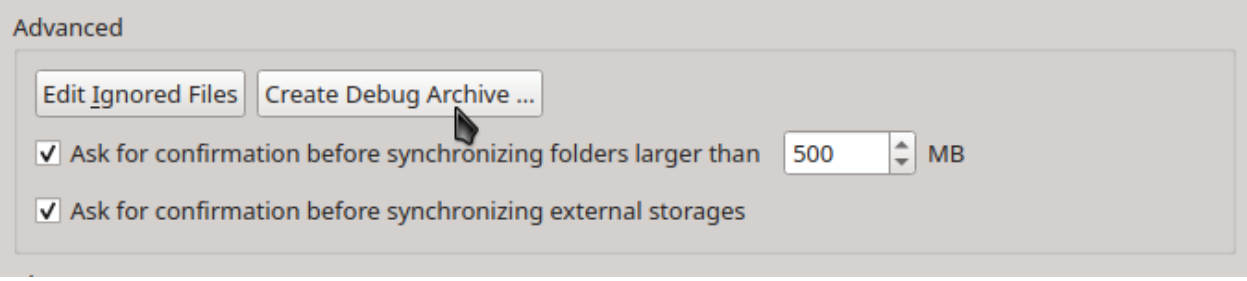
Effectively debugging software requires as much relevant information as can be obtained. To assist the Nextcloud support personnel, please try to provide as many relevant logs as possible. Log output can help with tracking down problems and, if you report a bug, log output can help to resolve an issue more quickly.

> **Warning**
>
> Log files contain sensitive information. You may wish to redact sensitive details or to only share limited excerpts.

Obtaining the Client Log File

Create Debug Archive

Since the 3.1.0 release we made it easier for users to provide debug information: debug logging is enabled by default with expiration time set to 24 hours and under the “General” settings, you can click on “Create Debug Archive …” to pick the location of where the desktop client will export the logs and the database to a zip file.

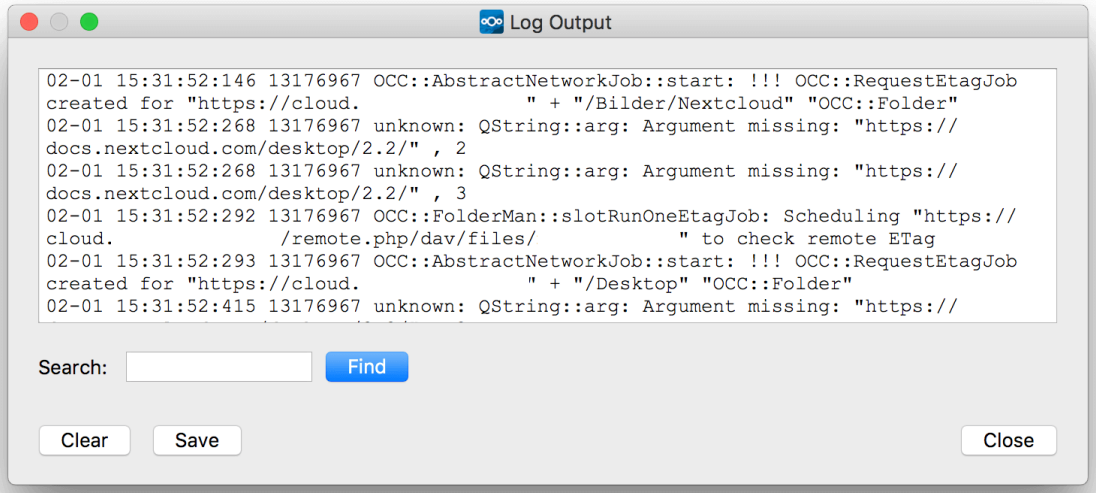


Keyboard shortcut

Another way to obtain the client log file:

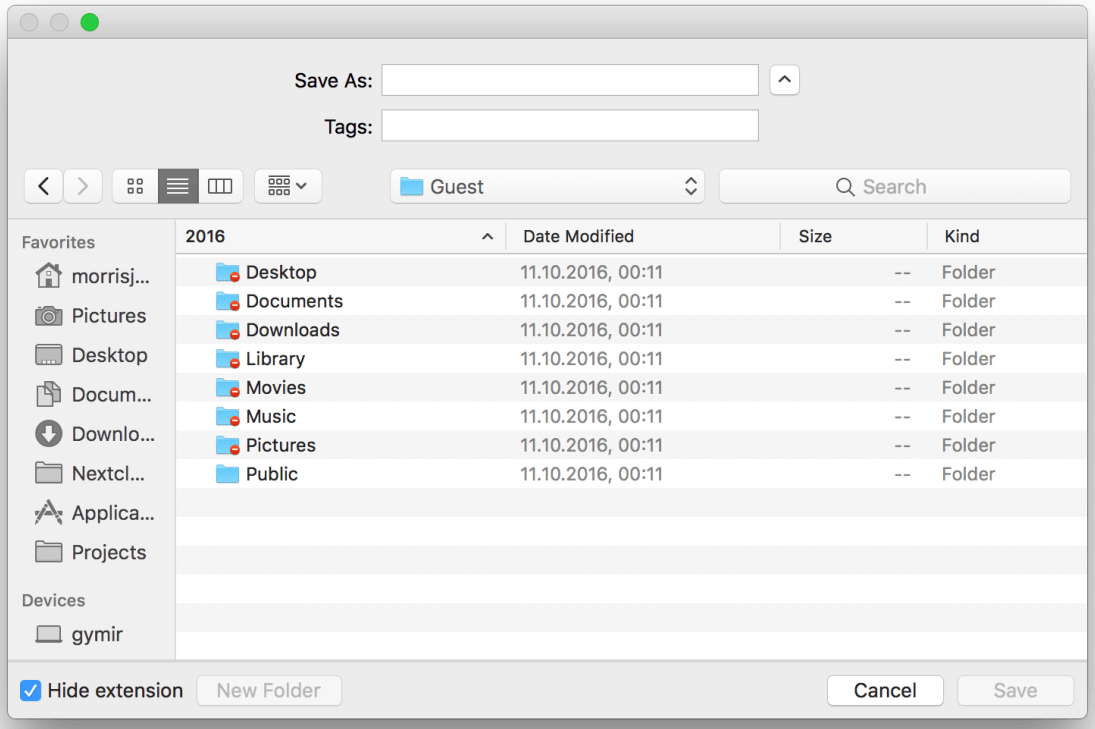
1. Open the Nextcloud Desktop Client.
2. Press `F12` or `Ctrl+L` on your keyboard.

The Log Output window opens.



3. Click the ‘Save’ button.

The Save Log File window opens.



4. Navigate to a location on your system where you want to save your log file.
5. Name the log file and click the 'Save' button.

   The log file is saved in the location specified.

Command line

Alternatively, you can launch the Nextcloud Log Output window using the `--logwindow` command. After issuing this command, the Log Output window opens to show the current log. You can then follow the same procedures mentioned above to save the log to a file.

> **Note**
>
> You can also open a log window for an already running session, by restarting the client using the following command:
>
> - Windows: `C:\Program Files\Nextcloud\nextcloud.exe --logwindow`
> - macOS: `/Applications/nextcloud.app/Contents/MacOS/nextcloud --logwindow`
> - Linux: `nextcloud --logwindow`

Config file

The Nextcloud client enables you to save log files directly to a predefined file or directory. This is a useful option for troubleshooting sporadic issues as it enables you to log large amounts of data and bypass the limited buffer settings associated with the log window.

To enable logging to a directory, stop the client and add the following to the General section in the configuration file:

```
[General]
logDebug=true
logExpire=<hours>
logDir=<dir>
```

Independent of platform you must use slash (/) as a path separator:

> **Note**
>
> - Correct: C:/Temp
> - Not correct: C:Temp

As an example, to keep log data for two days in a directory called temp:

```
[General]
logDebug=true
logExpire=48
logDir=C:/Temp
```

Once you restart the client, you will find the log file in the `<dir>` defined in `logDir`.

> **Note**
>
> You will find the configuration file in the following locations:
>
> - Microsoft Windows systems: `%APPDATA%\Nextcloud\nextcloud.cfg`
> - macOS systems: `$HOME/Library/Preferences/Nextcloud/nextcloud.cfg`
> - Linux distributions: `$HOME/.config/Nextcloud/nextcloud.cfg`

Alternatively, you can start the client in the command line with parameters:

1. To save to a file, start the client using the `--logfile <file>` command, where `<file>` is the filename to which you want to save the file.
2. To save to a directory, start the client using the `--logdir <dir>` command, where `<dir>` is an existing directory.

When using the `--logdir` command, each sync run creates a new file. To limit the amount of data that accumulates over time, you can specify the `--logexpire <hours>` command. When combined with the `--logdir` command, the client automatically erases saved log data in the directory that is older than the specified number of hours.

As an example, to define a test where you keep log data for two days, you can issue the following command:

` nextcloud --logdir /tmp/nextcloud_logs --logexpire 48 `

Nextcloud server Log File

The Nextcloud server also maintains an Nextcloud specific log file. This log file must be enabled through the Nextcloud Administration page. On that page, you can adjust the log level. We recommend that when setting the log file level that you set it to a verbose level like `Debug` or `Info`.

You can view the server log file using the web interface or you can open it directly from the file system in the Nextcloud server data directory.

See *Logging* in the admin manual for details on configuring log levels and log file locations.

Webserver Log Files

It can be helpful to view your webserver's error log file to isolate any Nextcloud-related problems. For Apache on Linux, the error logs are typically located in the `/var/log/apache2` directory. Some helpful files include the following:

- `error_log` – Maintains errors associated with PHP code.
- `access_log` – Typically records all requests handled by the server; very useful as a debugging tool because the log line contains information specific to each request and its result.

You can find more information about Apache logging at `http://httpd.apache.org/docs/current/logs.html`.

21.3.7 Core Dumps

On macOS and Linux systems, and in the unlikely event the client software crashes, the client is able to write a core dump file. Obtaining a core dump file can assist Nextcloud Customer Support tremendously in the debugging process.

To enable the writing of core dump files, you must define the `OWNCLOUD_CORE_DUMP` environment variable on the system.

For example:

```
` OWNCLOUD_CORE_DUMP=1 nextcloud `
```

This command starts the client with core dumping enabled and saves the files in the current working directory.

Note

Core dump files can be fairly large. Before enabling core dumps on your system, ensure that you have enough disk space to accommodate these files. Also, due to their size, we strongly recommend that you properly compress any core dump files prior to sending them to Nextcloud Customer Support.

21.3.8 Citrix Workspace known issues

These are known issues when running the desktop client in a Citrix workspace:

- The Windows user's Roaming profile needs to be persisted between sessions. The failure on doing so will result in result in users having to set up their account again in every new session.
- The user's synchronization folder also needs to be persisted between sessions. The client will throw errors because it can not find the sync folder once the users logs in a new session.
- Every time the user logs in a Citrix environment, it creates a session with the desktop client and in that session the client will sync the user's files, that can quickly lead to the storage running out of space.

You can find additional information here:

- User manual
- Developer manual

CHAPTER

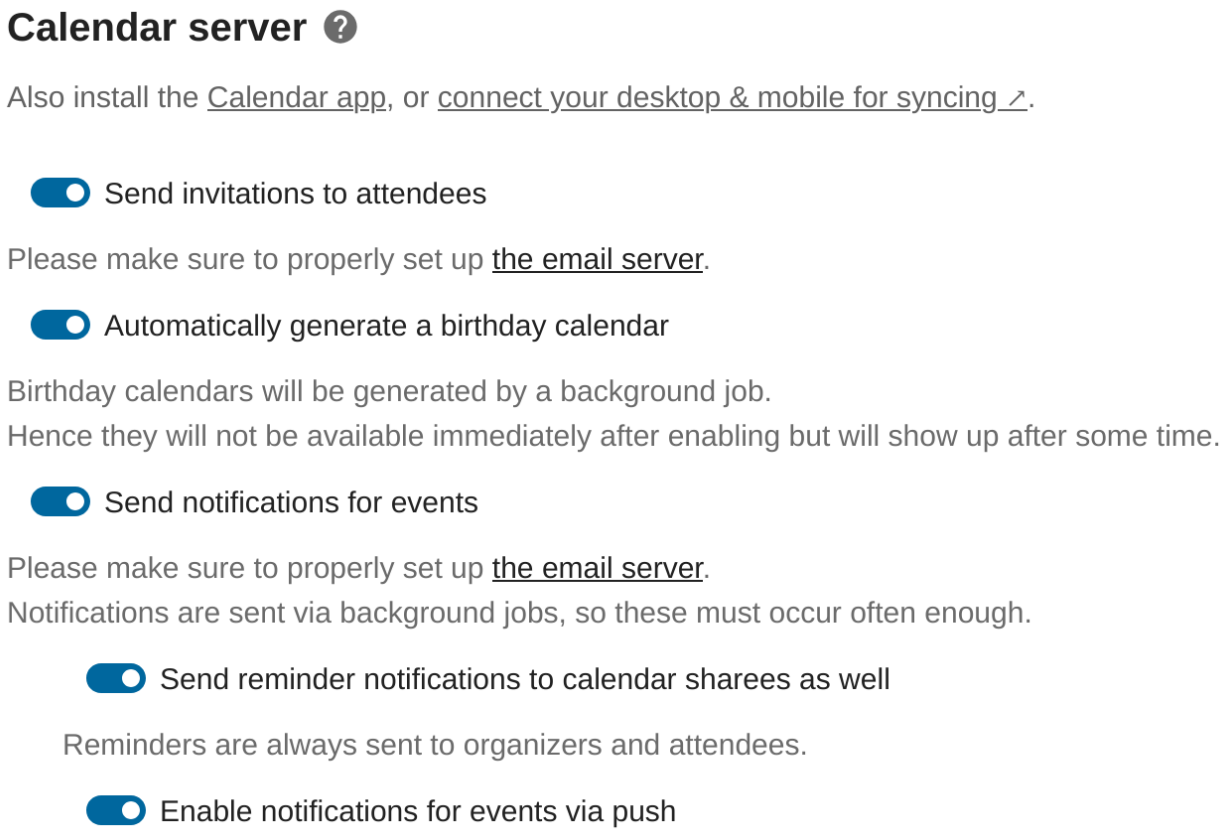
TWENTYTWO

GROUPWARE

22.1 Calendar / CalDAV

22.1.1 Calendar server settings

The calendar server can be configured on the Groupware admin settings page. You can globally disable sending invitation emails for events, generating the built-in birthday calendar, and sending email notifications about upcoming events.



Added in version 30.0.0: The section will be hidden if no app makes use of the CalDAV backend.

Starting from Nextcloud 30, the calendar server settings section will be hidden if no app uses the CalDAV backend. Install and enable an appropriate app to show the section again, e.g. Calendar or Tasks.

22.1.2 Events

You can customize the events user interface.

Hide export buttons

By default users can export their calendar data from the event editor. Admins can disable this feature:

```
sudo -E -u www-data php occ config:app:set calendar hideEventExport --value=yes
```

22.1.3 Invitations

Nextcloud can send invitations for event attendees if this option is activated. Be sure to have configured the email server first so that the invitations go through. See *Email*.

You must also make sure the "Send invitations to attendees" setting is activated in the admin setting groupware section for the emails to be sent.

Administrators can disable the sending of invitations to external participants with the following command:

```
sudo -E -u www-data php occ config:app:set dav caldav_external_attendees_disabled --
↪value yes
```

This prevents invitations from being sent to attendees outside the instance and hides external contacts from the invitee search.

22.1.4 Birthday calendar

Contacts that have a birthday date filled are automatically added as events to a special Birthday calendar. If you deactivate this option, all users will no longer have this calendar.

When activating this option, users birthday calendars won't be available right away because they need to be generated by a background task. See *Using the occ command* section DAV commands.

22.1.5 Reminder notifications

Nextcloud handles sending notifications for events.

Nextcloud currently handles two types of reminder notifications: Built-in Nextcloud notifications and email notifications. For the emails to be send, you'll need a configured email server. See *Email*.

Make sure the "Send notifications for events" and the "Enable notifications for events via push" are activated in the admin setting groupware section for this feature to work.

Background jobs

Running background jobs can be an expensive task when there are a large number of events, reminders, event sharees and attendees. However, this needs to happen often enough so that the notifications are sent on time. To accomplish this you should use a dedicated `occ` command that runs more often than the standard `cron` system:

```
# crontab -u www-data -e
*/5 * * * * php -f /var/www/nextcloud/occ dav:send-event-reminders
```

See *Using the occ command* section Dav commands.

You'll also need to change the sending mode from `background-job` to `occ`:

```
sudo -E -u www-data php occ config:app:set dav sendEventRemindersMode --value occ
```

If you don't use this dedicated command, the reminders will just be sent as soon as possible when the background jobs run.

22.1.6 Event alarm types

Nextcloud allows users to set notification and email reminders for events. Admins can enforce one of the two options:

```
occ config:app:set calendar forceEventAlarmType --value=EMAIL
```

Allowed values are `EMAIL` (email) and `DISPLAY` (notification).

> **Note**
>
> This only enforces alarm types for events created with the Nextcloud Calendar. This setting has no influence for other connected applications.

22.1.7 FreeBusy

When logged-in, Nextcloud can return FreeBusy information for all users of the instance, to know when they are available so that you can schedule an event at the right time. If you don't wish for users to have this capability, you can disable FreeBusy for the whole instance with the following setting:

```
sudo -E -u www-data php occ config:app:set dav disableFreeBusy --value yes
```

22.1.8 Subscriptions

Custom public calendars

In addition to the public holiday calendars, it is possible to define your own calendar. They act in the same way as the holiday calendars and can be configured with the following command:

```
sudo -E -u www-data php occ config:app:set calendar publicCalendars --value '[{"name":
↪"My custom calendar","source":"http://example.com/example.ics"}]'
```

The setting is specified as a JSON array of objects with the following options:

- `name` - name of the calendar in the listing
- `source` - URL of the calendar's ICS file
- `displayName` - optional, to overwrite the name of the subscribed calendar
- `description` - optional, description in the listing
- `authors` - optional, copyrights and so on

Refresh rate

Calendar subscriptions are cached on server and refreshed periodically. If the calendar server provides a refresh interval, it is respected. Otherwise the default refresh rate is one day.

To set up a different default refresh rate for calendars without server side refresh rates, change the `calendarSubscriptionRefreshRate` option:

```
sudo -E -u www-data php occ config:app:set dav calendarSubscriptionRefreshRate --
↪value "PT6H"
```

Where the value is a DateInterval, for instance with the above command all of the Nextcloud instance's calendars would be refreshed every 6 hours.

Allow subscriptions on local network

Because of security issues, Nextcloud forbids subscriptions from local network hosts. If you need to allow this, change the following parameter to:

```
sudo -E -u www-data php occ config:app:set dav webcalAllowLocalAccess --value yes
```

22.1.9 Federated calendar shares

Added in version 32.0.0.

Changed in version 33.0.0: Federated calendar shares are now read/write.

Nextcloud supports creating federated calendar shares. A user is able to share a calendar with a remote user on a federated instance. Starting with Nextcloud 33, remote users are able to create, edit, and delete events inside the shared calendar. In Nextcloud 32, shares were read-only.

The feature can be optionally disabled through an app config. Run the following command to disable creating new federated calendar shares for all users:

```
sudo -E -u www-data php occ config:app:set dav enableCalendarFederation --
↪type=boolean --value=false
```

Note that existing shares will be deleted when the feature is disabled as they will fail to sync.

22.1.10 Trash bin

Nextcloud supports a calendar, events and tasks trash bin.

The default delay before objects are purged from the trash bin is 30 days. A background job runs every 6 hours to clean up expired objects.

To set up a different retention period, change the `calendarRetentionObligation` option:

```
sudo -E -u www-data php occ config:app:set dav calendarRetentionObligation --
↪value=2592000
```

Where the value is the number of seconds for the period. Setting the value to `0` disables the trash bin.

22.1.11 Resources and rooms

The Nextcloud CalDAV backend supports resources and rooms. Resources and rooms can be booked for appointments, and the system will schedule them so they can only be used once at a time. Those resources and rooms have to be provided by an app that provides a backend for this.

Once a backend app is installed, the app typically allows admins, or even users, to define the resources, but this is subject of the specific implementation.

Nextcloud periodically queries all registered backends, therefore new/updated resources and rooms will show with a delay.

Known backends

- Calendar Resource Management: database backend with CLI configuration for admins

22.1.12 Rate limits

Nextcloud rate limits the creation of calendars and subscriptions if too many items are created within a short time frame. The default is 10 calendars or subscriptions per hour. This can be customized as follows:

```
# Set limit to 15 items per 30 minutes
sudo -E -u www-data php occ config:app:set dav rateLimitCalendarCreation --
↪type=integer --value=15
sudo -E -u www-data php occ config:app:set dav rateLimitPeriodCalendarCreation --
↪type=integer --value=1800
```

Additionally, the maximum number of calendars and subscriptions a user may create is limited to 30 items. This can be customized too:

```
# Allow users to create 50 calendars/subscriptions
sudo -E -u www-data php occ config:app:set dav maximumCalendarsSubscriptions --
↪type=integer --value=50
```

or:

```
# Allow users to create calendars/subscriptions without restriction
sudo -E -u www-data php occ config:app:set dav maximumCalendarsSubscriptions --
↪type=integer --value=-1
```

22.1.13 Example event

Added in version 32.0.0.

When a user logs in for the first time an example event is created in their personal calendar.

This feature is enabled by default and is controlled by the `create_example_event` app config.

As an admin, you can disable the creation of the example event. It is also possible to replace the default event with a custom one.

To disable the creation of the example event for new users:

1. Navigate to the Groupware settings in the admin settings.
2. Scroll down to the "Example content" section.
3. Disable the "Add example event …" setting with the checkbox

It can also be toggled from the command line:

```
sudo -E -u www-data php occ config:app:set dav create_example_event --value=no
```

To replace the built-in default event with a custom one:

1. Navigate to the Groupware settings in the admin settings.
2. Press the "Import calendar event" button.
3. Choose an ICS file to be imported.

> **Note**
>
> The start and end date will be overwritten with dates in the future when a custom event is supplied to ensure that the user sees the event.

It is also possible to revert to the default built-in event by pressing the "Reset to default" button next to the import button.

22.1.14 Data retention

Added in version 26.0.0.

You can configure how long Nextcloud keeps some of the calendar sync tokens.

Sync tokens

The CalDAV backend keeps track of any modifications of calendars. That is anything added, modified or removed. The data is used for differential synchronization of offline clients like Thunderbird. At a certain point in time, the data can be considered outdated assuming there will be no more client needing it. This can help keep the database table *calendarchanges* small:

```
sudo -E -u www-data php occ config:app:set totalNumberOfSyncTokensToKeep --value=30000
```

The default is keeping 10,000 entries. This option should be set adequate to the number of users. E.g. on an installation with 5000 active synced calendars the system would only keep an average of 10 changes per calendar. This will lead to premature data deletion and synchronization problems.

> **Warning**
>
> This setting will also influence *CardDAV data retention*.

22.2 Contacts / CardDAV

Nextcloud ships a CardDAV backend for users to store and share their address books and contacts.

22.2.1 System Address Book

Changed in version 27.0.0: The system address book is now accessible to all Nextcloud users

Nextcloud maintains a read-only address book containing contact information of all users of the instance.

Disabled users are removed from this address book.

You can disable or enable access to the system address book by using the administration interface or with a command line command.

Please note that this does not influence *Federated sharing*.

Command Line

Run `occ config:app:set dav system_addressbook_exposed --value="no"` to disable access to the system address book for all users.

Administration interface

Navigate to *Administration Settings -> Groupware -> System Address Book* section and toggle the *Enable system address book* option.

> **Warning**
>
> If clients have already connected to the CalDAV endpoint, the clients might experience sync issues after system address book access was disabled. This can often be remedied by choosing a different default address book on the client and forcing a resync.

Privacy and User Property Scopes

Contact information in the system address book is taken from users' *profile information*. Profile properties are only written to the system contact if the *scope* is set to *Local* or higher.

Users who set all their property scopes to *Private* are removed from the system address book and therefore not seen by other users.

File sharing settings controls the enumeration of other users.

- If username autocompletion is not allowed, the system address book will only show user's own system contact but no other contacts.
- If username autocompletion is allowed, users will see contact cards for all other users.
  - If autocompletion is limited to users within the same groups, users will see contact cards for other users in shared groups.
  - If autocompletion is limited to matching phone numbers, the system address book will only show user's own system contact but no other contacts.
  - If autocompletion is limited to users within the same groups **and** matching phone numbers, users will see contact cards for other users in shared groups.

Address Book Sync

The address book is updated automatically with every added, modified, disabled or removed user. Admins can also trigger a full rewrite of the address book *with occ*.

22.2.2 Shared items

Added in version 5.5.0: Nextcloud 25 or newer

For this feature, the shipped related resources app needs to be enabled.

22.2.3 Rate limits

Nextcloud rate limits the creation of address books and how many can be created in a short period of time. The default is 10 address books per hour. This can be customized as follows:

```
# Set limit to 15 items per 30 minutes
sudo -E -u www-data php occ config:app:set dav rateLimitAddressBookCreation --
↪type=integer --value=15
sudo -E -u www-data php occ config:app:set dav rateLimitPeriodAddressBookCreation --
↪type=integer --value=1800
```

Additionally, the maximum number of address books a user may create is limited to 10 items. This can be customized too:

```
# Allow users to create 50 addressbooks
sudo -E -u www-data php occ config:app:set dav maximumAdressbooks --type=integer --
↪value=50
```

or:

```
# Allow users to create address books without restriction
sudo -E -u www-data php occ config:app:set dav maximumAdressbooks --type=integer --
↪value=-1
```

22.2.4 Example contact

Added in version 32.0.0.

When a user logs in for the first time an example contact is created in the user's address book.

This feature is enabled by default and is controlled by the `enableDefaultContact` app config.

To disable the example contact feature:

1. Navigate to the Groupware settings in the admin settings.
2. Scroll down to the "Example content" section.
3. Disable the "Add example contact …" setting with the checkbox

It can also be toggled from the command line:

```
sudo -E -u www-data php occ config:app:set dav enableDefaultContact --value=no
```

If you want to set a specific contact that should be created.

4. Press the "Import contact" button.
5. Choose a vCard file (.vcf) that should be imported as an example contact.

Switching back to the default example contact provided by nextcloud is possible by pressing the "Reset to default" button next to the import button.

22.2.5 Data retention

Added in version 26.0.0.

You can configure how long Nextcloud keeps some of the contacts sync tokens.

Sync tokens

The CardDAV backend keeps track of any modifications of address books. That is anything added, modified or removed. The data is used for differential synchronization of offline clients like Thunderbird. At a certain point in time, the data can be considered outdated assuming there will be no more client needing it. This can help keep the database table *addressbookchanges* small:

```
sudo -E -u www-data php occ config:app:set totalNumberOfSyncTokensToKeep --value=30000
```

The default is keeping 10,000 entries. This option should be set adequate to the number of users. E.g. on an installation with 5000 active synced addressbooks the system would only keep an average of 10 changes per sync. This will lead to premature data deletion and synchronization problems.

Warning

This setting will also influence *CalDAV data retention*.

22.3 Contacts Interaction

The Contacts Interaction app automatically tracks which people a user has recently interacted with and provides this data as a read-only CardDAV address book called **Recently contacted**. This enables autocomplete suggestions in sharing dialogs, email composition, calendar invitations and other places that query the user's address books — even for people who are not saved as explicit contacts.

The app is shipped with Nextcloud and enabled by default. It can be disabled.

22.3.1 How interactions are tracked

The app listens for `ContactInteractedWithEvent` events dispatched by other Nextcloud apps. The following apps dispatch this event:

- **File sharing**: When a user creates a share with another local user, an email address or a federated remote user.
- **Calendar**: When a user shares a calendar with another user.
- **Mail**: When a user sends an email.

Any app can integrate by dispatching a `ContactInteractedWithEvent` with at least one identifier: a Nextcloud user ID, an email address or a federated cloud ID.

When an interaction is recorded, the app first checks whether the contacted person already exists in one of the user's regular address books. If a match is found, no entry is created in the recently contacted address book since the person is already a known contact. Self-interactions (where the user interacts with themselves) are also ignored.

For new contacts, a minimal vCard is generated containing:

- `FN` (display name): Resolved from the Nextcloud user profile if the person is a local user, falling back to the email address or federated cloud ID.
- `EMAIL`: Included when an email address is known.
- `CLOUD`: Included when a federated cloud ID is known.
- `CATEGORIES`: Set to `Recently contacted`, which allows the Contacts app to identify entries from this address book and offer users the option to copy them to a regular address book.

If the same person is contacted again, the existing entry's timestamp is updated rather than creating a duplicate.

22.3.2 The recently contacted address book

Each user's recently contacted address book is accessible via CardDAV at:

```
/remote.php/dav/addressbooks/users/{userId}/z-app-generated--contactsinteraction--
↪recent/
```

The `z-app-generated` prefix ensures the address book sorts after user-created address books. Users cannot create their own address books with this reserved prefix.

The address book is **read-only** and **not shareable**. Users cannot create, modify or delete entries. Entries are only removed automatically by the cleanup job or when a user account is deleted.

The address book is visible in:

- **CardDAV clients**: Any client syncing with the user's Nextcloud (e.g., Thunderbird, macOS Contacts, DAVx5) will see the address book.
- **Nextcloud Contacts**: The address book and its entries appear in the Contacts UI if the Contacts app is enabled. Contacts from this address book can be copied to a regular address book.
- **Autocomplete**: Entries are available for recipient suggestions in sharing dialogs and other places across the Nextcloud web interface.

22.3.3 Data retention

A background job runs every 24 hours and removes entries that have not been updated within the last 7 days. Both the retention period and the cleanup interval are fixed and cannot be configured.

> **Note**
>
> The cleanup job depends on the Nextcloud background job system. Make sure cron is configured correctly for your instance. See *Background jobs*.

22.3.4 User deletion

When a user account is deleted, all of that user's recently contacted entries are automatically removed from the database.

22.4 Mail

22.4.1 Configuration

Anti-abuse alerts

The app can write alerts to the logs when users send messages to a high number of recipients or sends a high number of messages for a short period of time. These events might indicate that the account is abused for sending spam messages.

To enable anti-abuse alerts, you'll have to set a few configuration options *via occ* .

```
# Turn alerts on
occ config:app:set mail abuse_detection --value=on
# Turn alerts off
occ config:app:set mail abuse_detection --value=off

# Alert when 50 or more recipients are used for one single message
occ config:app:set mail abuse_number_of_recipients_per_message_threshold --value=50

# Alerts can be configured for three intervals: 15m, 1h and 1d
# Alert when more than 10 messages are sent in 15 minutes
occ config:app:set mail abuse_number_of_messages_per_15m --value=10
# Alert when more than 30 messages are sent in one hour
occ config:app:set mail abuse_number_of_messages_per_1h --value=30
# Alert when more than 100 messages are sent in one day
occ config:app:set mail abuse_number_of_messages_per_1d --value=100
```

Attachment size limit

Admins can prevent users from attaching large attachments to their emails. Users will be asked to use link shares instead.

```
'app.mail.attachment-size-limit' => 3*1024*1024,
```

The unit is bytes. The example about with limit to 3MB attachments. The default is 0 bytes which means no upload limit.

Background sync interval

Configure how often Mail keeps users' mailboxes updated in the background in seconds. Defaults to 3600, minimum 300.

```
'app.mail.background-sync-interval' => 7200,
```

Disable TLS verification for IMAP/SMTP

Turn off TLS verification for IMAP/SMTP. This happens globally for all accounts and is only needed in edge cases like with email servers that have a self-signed certificate.

```
'app.mail.verify-tls-peer' => false
```

Google OAuth

This app can allow users to connect their Google accounts with OAuth. This makes it possible to use accounts without 2FA or app password.

1. Create authorization credentials. You will receive a client ID and a client secret.
2. Open the Nextcloud settings page. Navigate to *Groupware* and scroll down to *Gmail integration*. Enter and save the client ID and client secret.

Local IMAP and SMTP servers

By default, Nextcloud does not allow local hostnames and IP addresses as remote servers. This includes IMAP, SMTP and Sieve servers like `localhost`, `mx.local` and `10.0.0.3`. This check can be disabled with via `config/config.php`.

```
'allow_local_remote_servers' => true,
```

Timeouts

Depending on your mail host, it may be necessary to increase your IMAP and/or SMTP timeout threshold. Currently IMAP defaults to 5 seconds and SMTP defaults to 20 seconds. They can be changed as follows:

IMAP timeout

```
'app.mail.imap.timeout' => 5
```

SMTP timeout

```
'app.mail.smtp.timeout' => 20
```

Sieve timeout

```
'app.mail.sieve.timeout' => 5
```

Use php-mail for sending mail

Warning

Support for using php-mail was removed in version 4.4 of the mail app!

You can use the php-mail function to send mails. This is needed for some web hosters (1&1 (1und1)).

```
'app.mail.transport' => 'php-mail'
```

22.4.2 Account delegation

The Mail app supports account delegation if the delegation is handled by the mail server. That means the mail server has to accept emails sent from an alias address.

In mailcow, for example, the setting is called *Also allowed to send as user*.

Warning

Unless paired with shared *Sent* mailboxes or handled otherwise by the mail server, sent messages will be stored in the sender's personal *Sent* mailbox.

22.4.3 Snooze and scheduled sending

Note

If AJAX is selected for cron job execution in the admin settings, the snooze feature and scheduled sending are deactivated because of unreliable execution.

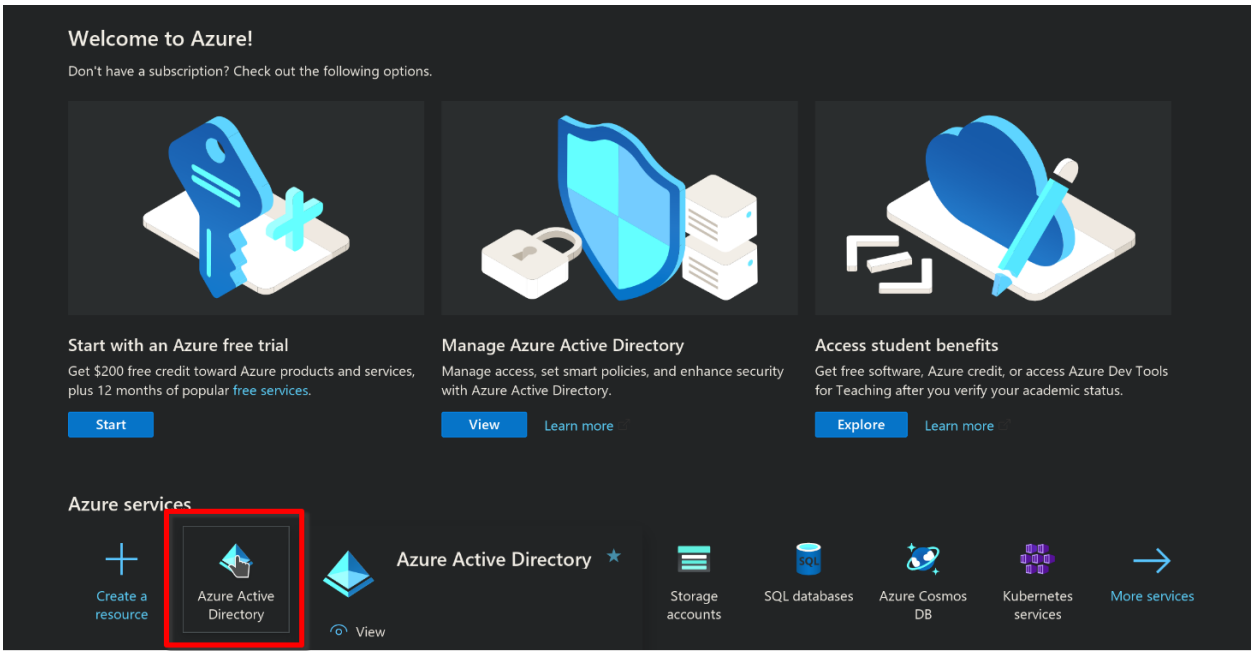
22.4.4 XOAUTH2 Authentication with Microsoft Azure AD

Added in version 3.0.0.

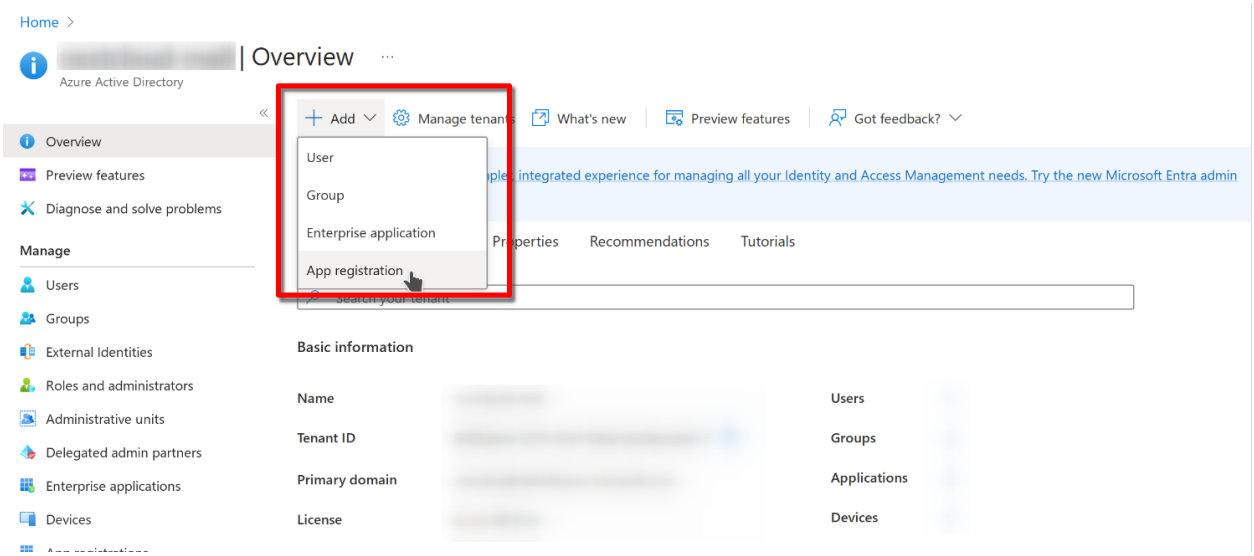
The Mail app supports XOAUTH2 authentication with hosted Microsoft Outlook accounts. An app has to be registered in the Microsoft Azure web interface and its credentials have to be supplied to the Nextcloud instance. You can find relevant settings in the Groupware section of the admin settings.

Step 1: Open the Azure AD Dashboard

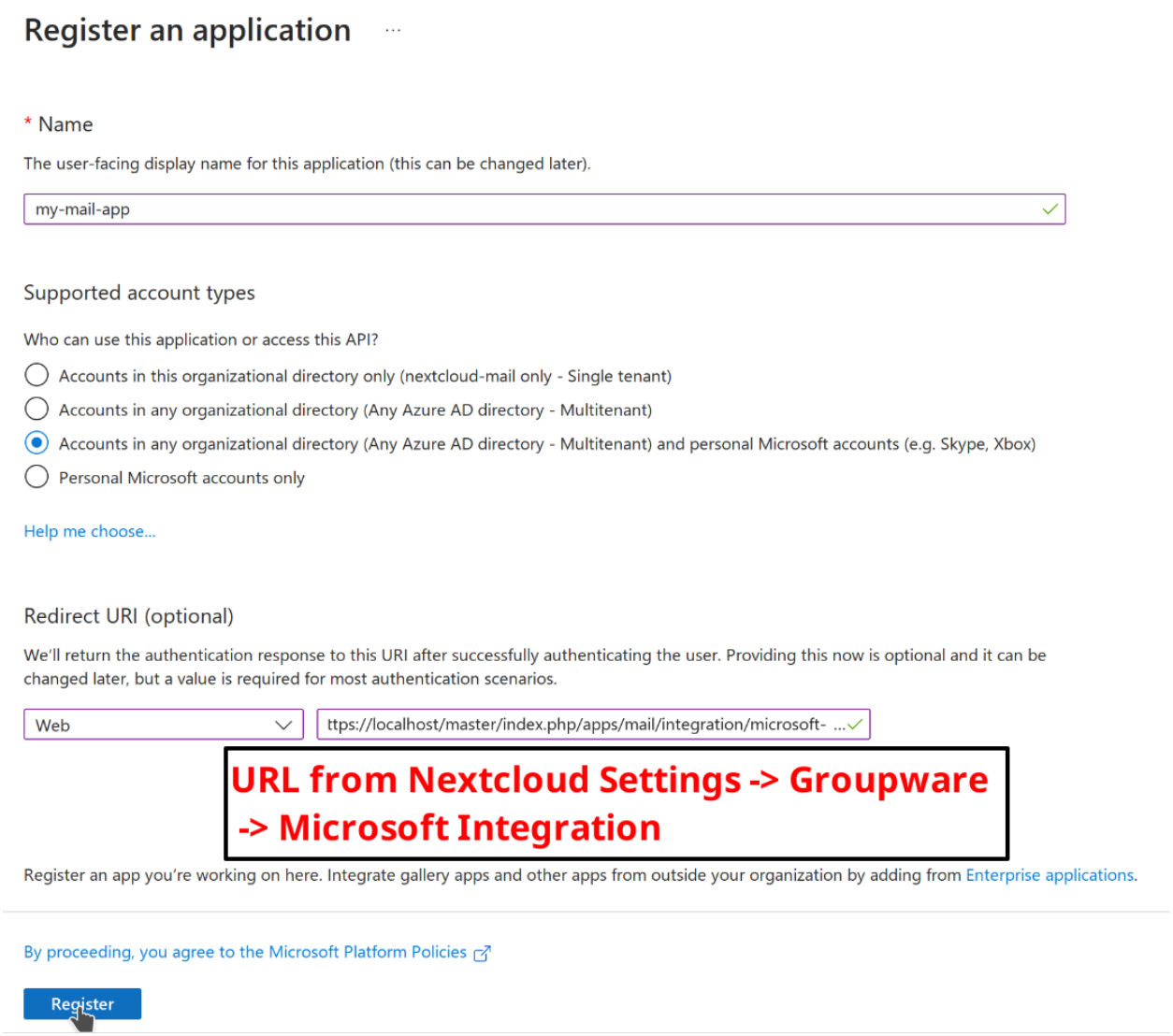
Visit the Azure portal and navigate to the Azure AD dashboard.



Step 2: Create a new app registration

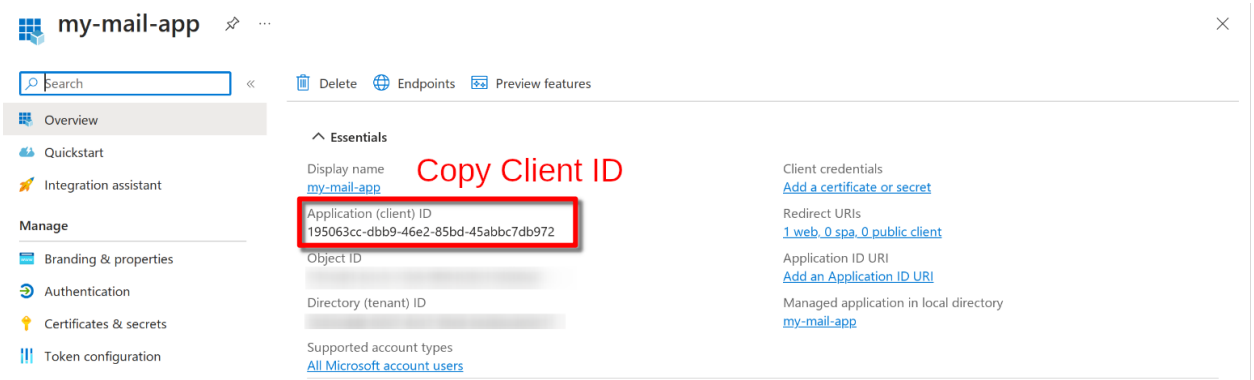


Chose a name, allow organizational and personal Microsoft accounts. Configure a web app and copy the redirect URI from the groupware settings of your Nextcloud instance. Have a look at step 8 on where to find the redirect URI. Finally, click on register to proceed.

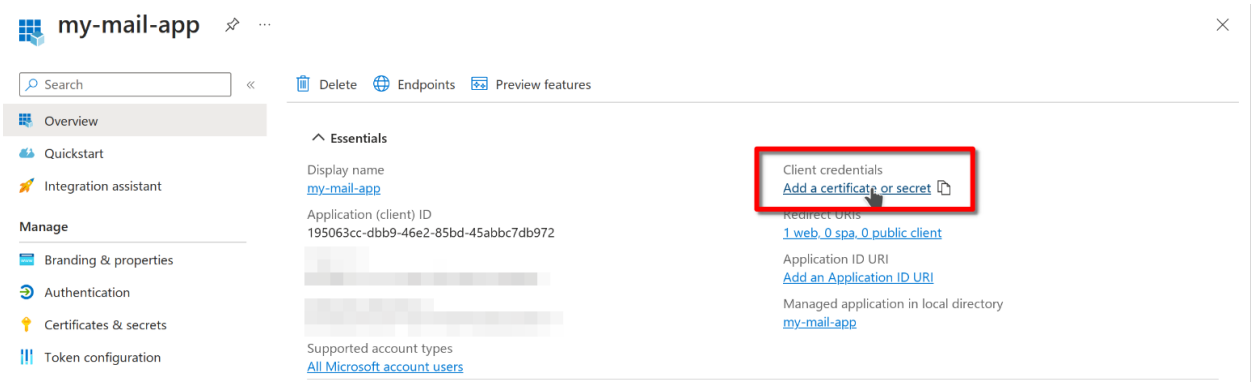


Step 3: Copy the client ID

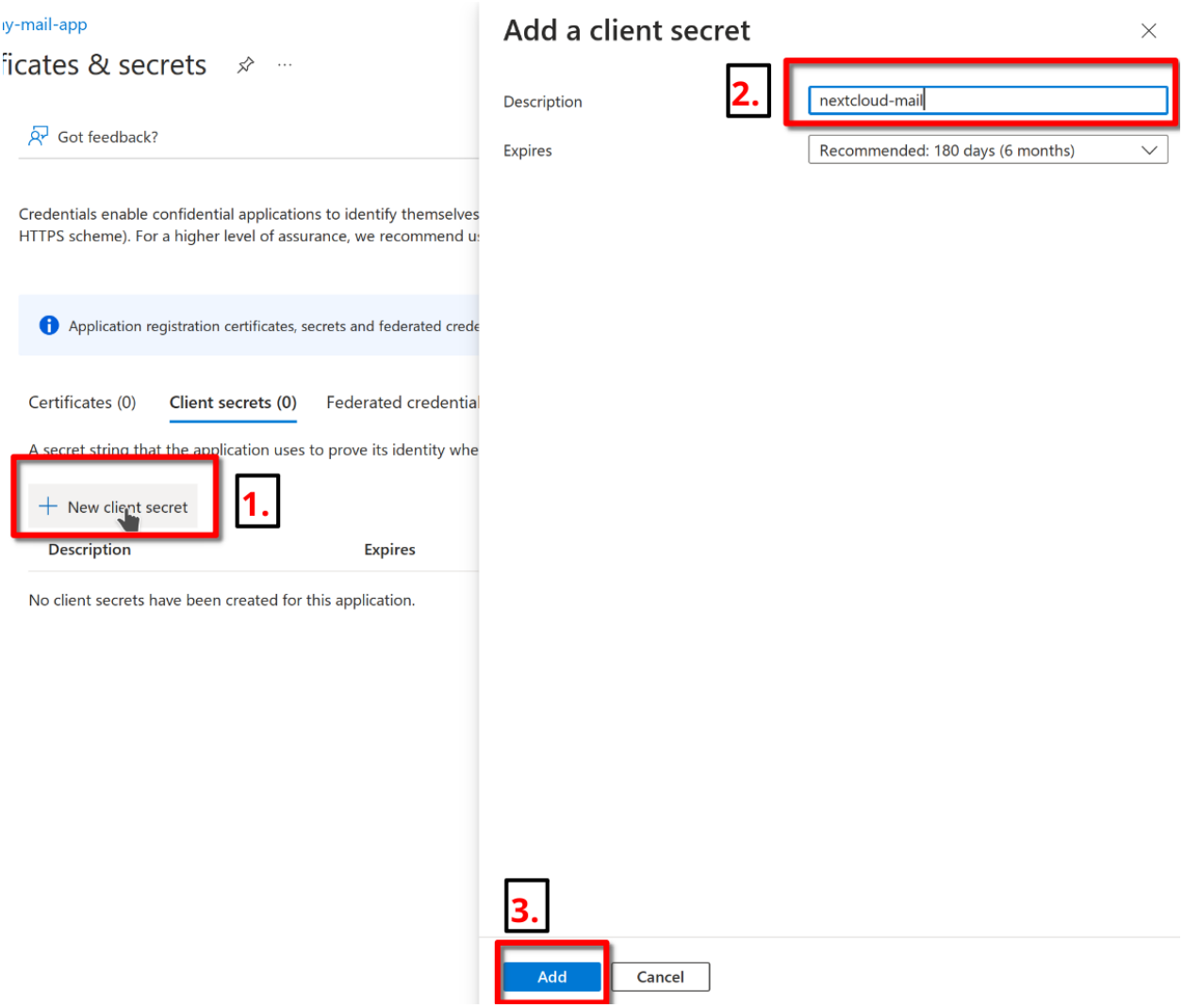
This ID will be needed later for the Nextcloud settings.



Step 4: Create a new client secret

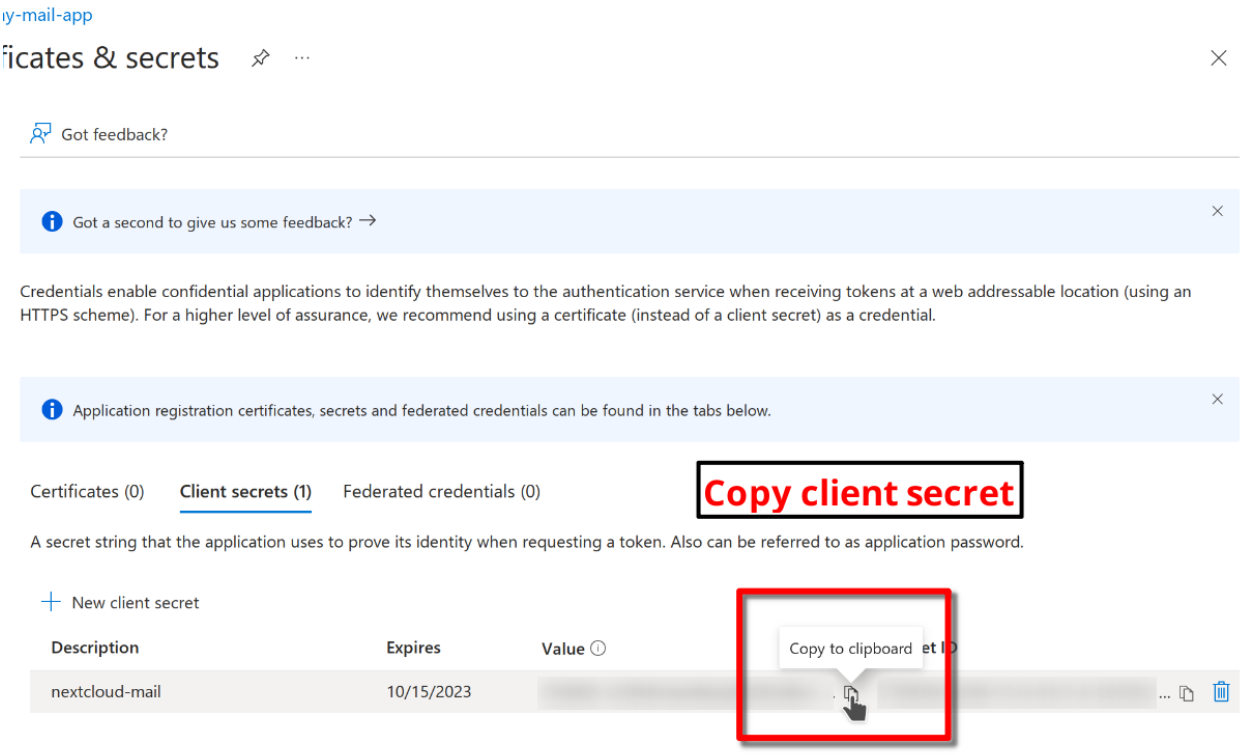


Chose a descriptive name for the secret and set the an appropriate expiration date. Click on add to create the secret.



Step 5: Copy the client secret

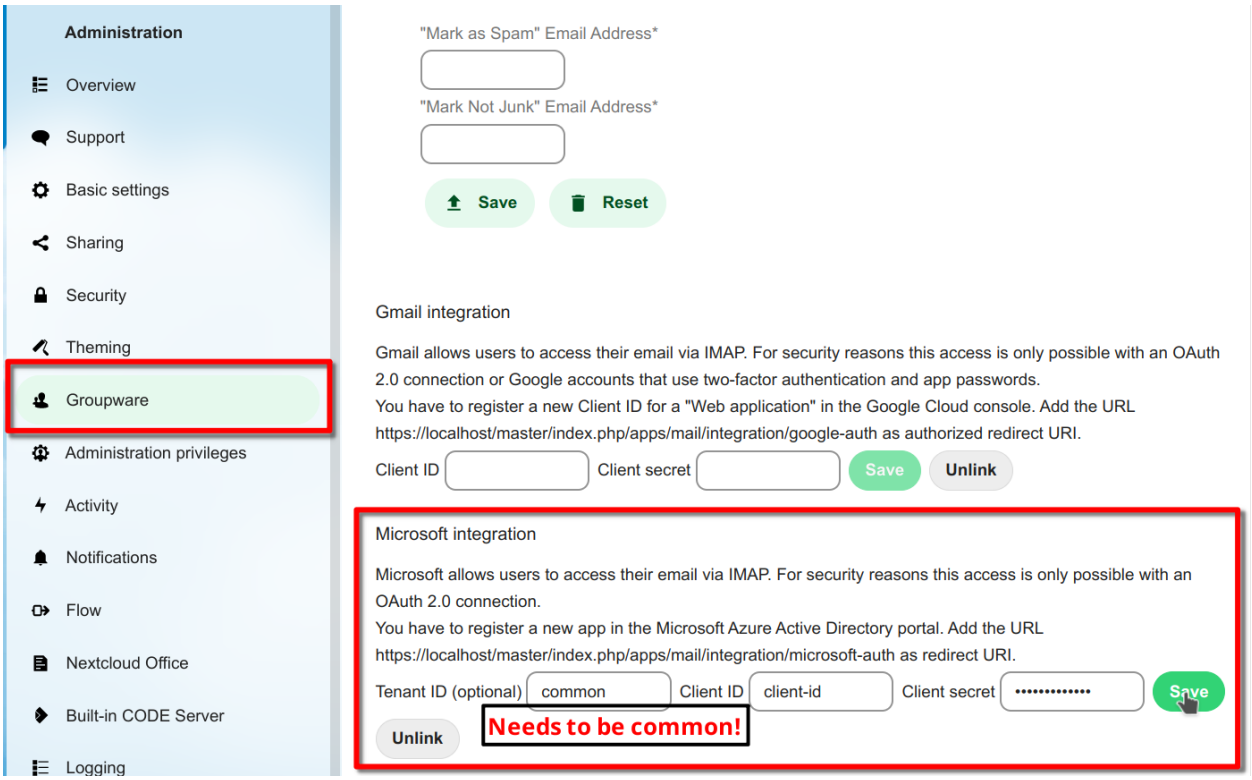
Copy the client secret manually or by clicking on the copy button. You can find it in the value column. The secret will also be needed later for the Nextcloud settings.



Step 6: Configure Nextcloud

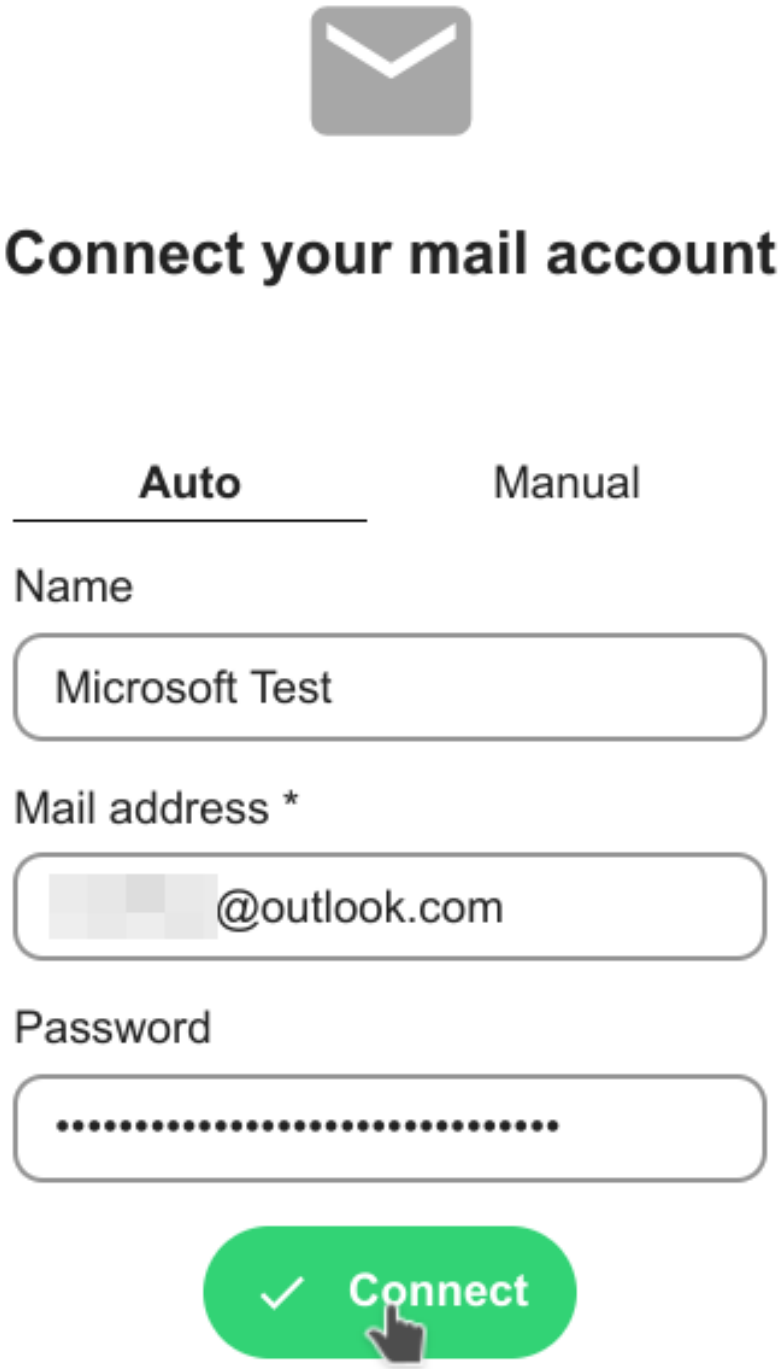
Open the groupware settings in the Nextcloud admin settings and fill in the client ID and client secret. Leave the tenant ID as is (common). You can also find the redirect URI here. Click on save to proceed.

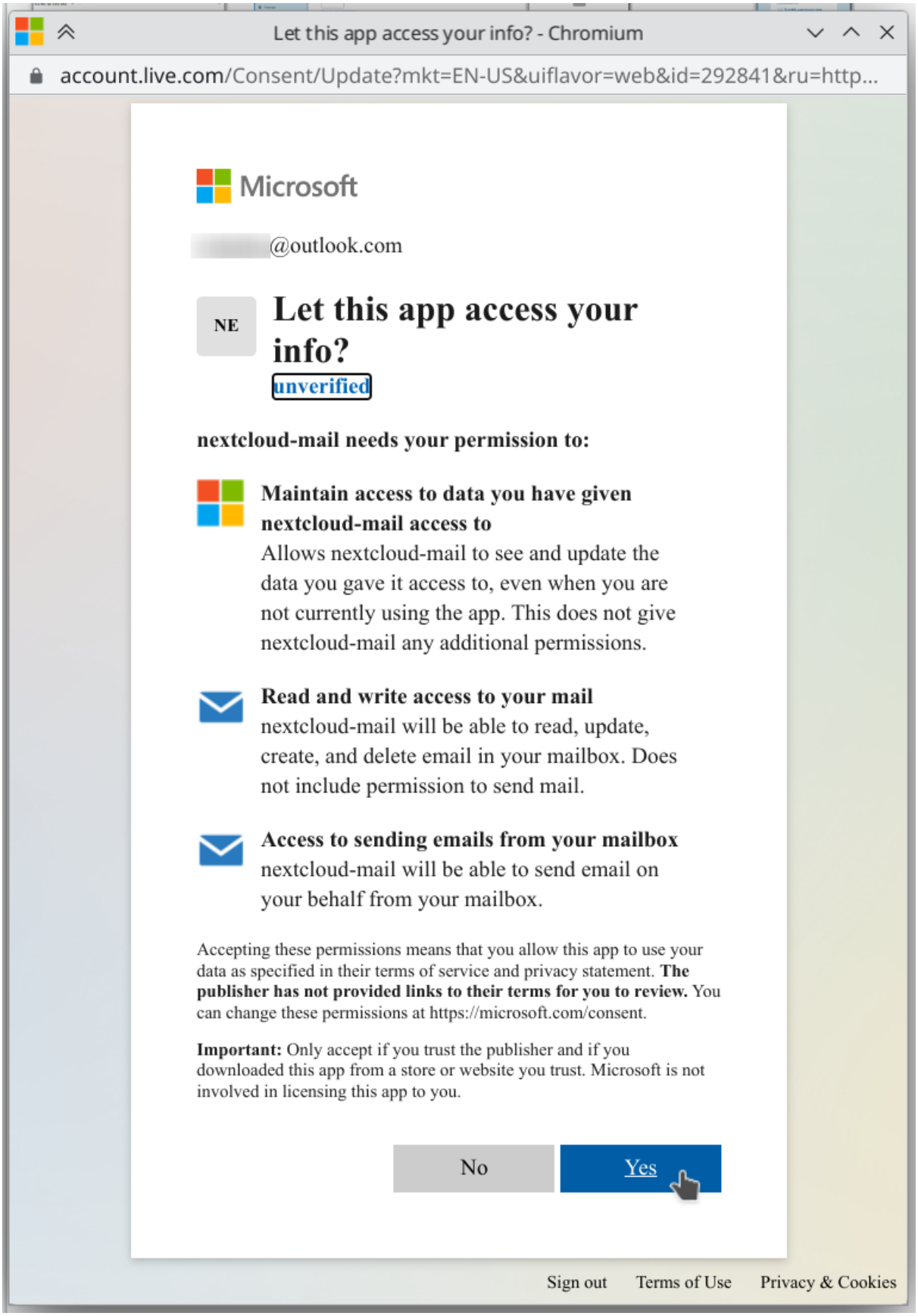
> **Warning**
>
> Using a custom tenant ID is not covered by this guide. Only configure it if you are an expert and changed the supported account types in step 2.



Step 7: Connect Microsoft Outlook accounts

Congratulations! You are now able to use hosted Microsoft Outlook accounts in the Mail app. Use your Microsoft account email and any password when adding your account. The password will be discarded and you will be prompted with a Microsoft consent popup to log in to your account.


Connect your mail account
Auto
Manual
Name
Microsoft Test
Mail address *
@outlook.com
Password
Connect


Let this app access your info? - Chromium
account.live.com/Consent/Update?mkt=EN-US&uiflavor=web&id=292841&ru=http...
Microsoft
@outlook.com
NE
Let this app access your info?
unverified
nextcloud-mail needs your permission to:
Maintain access to data you have given nextcloud-mail access to
Allows nextcloud-mail to see and update the data you gave it access to, even when you are not currently using the app. This does not give nextcloud-mail any additional permissions.
Read and write access to your mail
nextcloud-mail will be able to read, update, create, and delete email in your mailbox. Does not include permission to send mail.
Access to sending emails from your mailbox
nextcloud-mail will be able to send email on your behalf from your mailbox.
Accepting these permissions means that you allow this app to use your data as specified in their terms of service and privacy statement. The publisher has not provided links to their terms for you to review. You can change these permissions at https://microsoft.com/consent.
Important: Only accept if you trust the publisher and if you downloaded this app from a store or website you trust. Microsoft is not involved in licensing this app to you.
No
Yes
Sign out
Terms of Use
Privacy & Cookies

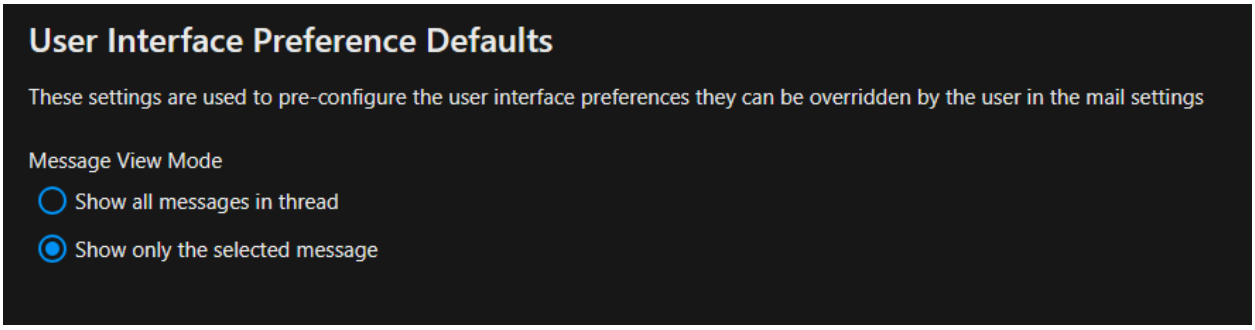
22.4.5 Mailbox Share

Users can share mailboxes with each other. So far, there is no UI for users to change the ACL in the Mail app, but if you want to use it, you need to enable it on the IMAP sever and configure the shares there.

22.4.6 User Interface Preference Defaults

Added in version 5.2: Nextcloud 30 or newer

The Mail app allows administrators to set default user interface preferences for all users, these preferences can be changed by the user afterwards. This can be useful to ensure a consistent experience across the application.



22.4.7 LLM Processing

The Mail app can optionally use large language models to process emails and offer assistance features like thread summaries, smart replies, event agendas and follow-up reminders.

> **Note**
>
> The supported languages depend on the used large language model.

> **Note**
>
> A fast text processing integration like https://apps.nextcloud.com/apps/integration_openai is required for best results.

The feature can be enabled in the Mail administration settings.

Administration settings > Groupware > Mail app > Enable text processing through LLMs

22.4.8 Thread Summary

Changed in version 3.6.0: Nextcloud 26 or newer This configuration option was merged into *LLM Processing*

The mail app supports summarizing message threads that contain 3 or more messages.

> **Warning**
>
> A text generation AI integration should be already in place to enable this feature.

The feature is opt-in, it is disabled by default and can be enabled in mail administration settings.

Administration settings > Groupware > Mail app > Enable thread summary

22.4.9 Follow-up reminders

Added in version 4.0: Nextcloud 30 or newer

The Mail app will automatically remind users when their outgoing emails remain unanswered for multiple days. Each sent email will be analyzed by an AI to check whether a reply is expected.

The feature can be enabled through the global *LLM Processing* setting.

22.4.10 Translation

Added in version 4.2: Nextcloud 30 or newer

The mail app can optionally provide translations for messages if the *translation API* is enabled.

22.4.11 User migration

Added in version 5.9: Nextcloud 30 or newer

Important

The User Migration App must be installed and enabled on both instances.

To migrate user data from one Nextcloud instance to another, use the User Migration App via the `occ user:export` or `occ user:import` command line tool.

The Mail app can migrate the following data:

- Manually added mail accounts and their respective account settings (e.g., display name, signature, quick actions, …)
- Configured internal addresses
- Configured trusted senders
- S/MIME certificates and private keys
- General app settings like the order of mail accounts

Note

Provisioned mail accounts are not migrated, as they are expected to be re-provisioned on the new instance.

Note

The User Migration App is **not** a backup solution, as existing data is not being removed before the import process.

See the User Migration App on the Nextcloud App Store for more information.

22.5 Out-of-office feature

Added in version 28.0.0.

The out-of-office feature allows users to schedule an out-of-office period including a status and message that is integrated with other apps such as Nextcloud Mail. Please refer to the user documentation for more information about the feature itself. It may be disabled globally by admins.

The feature relies on users to configure their preferred calendar time zones correctly. However, if a user does not configure their time zone, the default time zone of the server is used. It can be configured by setting `default_timezone` in the *config.php* file of your Nextcloud server. The configuration value accepts IANA identifiers like `Europe/Berlin` and defaults to `UTC`. Please refer to the *Nextcloud configuration* section for more information about the value.

To disable the out-of-office feature for all users the `hide_absence_settings` app configuration value of the *dav* app has to be set to *yes*. This can be achieved by running the following command on your server:

```
occ config:app:set --value=yes dav hide_absence_settings
```

Note

Out-of-office periods that were scheduled before the feature was disabled will not be deleted. Disabling it will only hide the feature from the user interface. If the feature is enabled again, the periods will be visible again.

Set the value for *hide_absence_settings* to *no* or delete the configuration option entirely to enable the feature again. The following command can be used to do so:

```
occ config:app:set --value=no dav hide_absence_settings
```

22.6 Troubleshooting

22.6.1 Calendar

Missing Shared Calendars

Problem:
A user should have access to a shared calendar, but the calendar is not displayed in Nextcloud Calendar or other CalDAV clients (e.g., DAVx[5] or Thunderbird).

Affected Versions:

- Nextcloud Server 31.0.5 and below
- Nextcloud Server 30.0.11 and below

Possible Reason:
A bug in previous versions of Nextcloud Server could mistakenly add a calendar unshare instead of removing the share permission. For example, a user has read access through a group membership, and the owner grants permission to a single user to modify a calendar. When the modify permission is removed, an unshare record is incorrectly created.

Troubleshooting Steps:

1. **Check for Hidden Calendars:**
   It's possible for a user to hide a calendar. Please check in Nextcloud Calendar if the missing calendar is listed in the "hidden" section. If it is, check the box in front of the calendar to enable it again.
2. **List Calendar Shares:**
   Run the command `occ dav:list-calendar-shares <uid>` to list all shares for a user. Look for lines with the Calendar URI/Calendar Name of the missing calendar and Permissions = Unshare. If such a line exists, but the user should have access, you have three options:

A. **Create a User Share and Remove It Again:**
   In most cases, sharing the calendar with the user again (as an individual/user share) will correct the state in the database.

B. **Remove All Calendar Unshares for a User:**
   Run `occ dav:clear-calendar-unshares <uid>`.

C. **Delete Specific Unshares:**
   Some users may have many calendar unshares, so it might be easier to delete only the unwanted unshare. The `Share Id` refers to the ID of a row in the `oc_dav_shares` database table. Delete the row with the matching ID to remove the unshare.

Why isn't there an automated migration to correct the problem?
Unsharing a calendar is a feature, and with the given information, we cannot determine if a calendar was unshared intentionally or as a result of the bug.

22.6.2 Contacts

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 because of a misconfigured web server. Please refer to *Troubleshooting WebDAV* for troubleshooting steps.

22.6.3 Mail

Autoconfig for your mail domain fails

If autoconfiguration for your domain fails, you can create an autoconfig file and place it as `https://autoconfig.yourdomain.tld/mail/config-v1.1.xml`. For more information please refer to Mozilla's documentation.

Database insert problems on MySQL

If the mail app fails to insert new rows for messages (*oc_mail_messages*), recipients (*oc_mail_recipients*) or similar tables, you are possibly not using the 4 byte support.

See *Enabling MySQL/MariaDB 4-byte support* for how to update your database configuration.

Export threading data

If you encounter an issue with threading, e.g. messages that belong to the same conversation thread don't show up as one, you can export the data the algorithm will use to build threads. We are dealing with sensitive data here, but the command will optionally redact the data with the `--redact` switch. The exported data will then only keep the original database IDs, the rest of the data is randomized. This format does not the export message details, it still contains metadata about how many messages you have and how they relate. Please consider this before posting the data online.

```
sudo -E -u www-data php occ mail:account:export-threads 1393
```

> **Note**
>
> 1393 represents the *account ID*.

The output will look similar to this:

```
[
    {
        "subject": "83379f9bc36915d5024de878386060b5@redacted",
        "id": "2def0f3597806ecb886da1d9cc323a7c@redacted",
        "references": [],
        "databaseId": 261535
    },
        {
        "subject": "Re: 1d4725ae1ac4e4798b541ca3f3cdce6e@redacted",
        "id": "ce9e248333c44a5a64ccad26f2550f95@redacted",
        "references": [
            "bc95cbaff3abbed716e1d40bbdaa58a0@redacted",
            "8651a9ac37674907606c936ced1333d7@redacted",
            "4a87e94522a3cf26dba8977ae901094d@redacted",
            "a3b30430b1ccb41089170eecbe315d3a@redacted",
            "8e9f60369dce3d8b2b27430bd50ec46d@redacted",
            "46cfa6e729ff329e6ede076853154113@redacted",
            "079e7bc89d69792839a5e1831b1cbc80@redacted",
            "079e7bc89d69792839a5e1831b1cbc80@redacted"
        ],
        "databaseId": 262086
    },
    {
        "subject": "Re: 1d4725ae1ac4e4798b541ca3f3cdce6e@redacted",
        "id": "8dd0e0ef2f7ab100b75922489ff26306@redacted",
        "references": [
            "bc95cbaff3abbed716e1d40bbdaa58a0@redacted",
            "8651a9ac37674907606c936ced1333d7@redacted",
            "4a87e94522a3cf26dba8977ae901094d@redacted",
            "a3b30430b1ccb41089170eecbe315d3a@redacted",
            "8e9f60369dce3d8b2b27430bd50ec46d@redacted",
            "46cfa6e729ff329e6ede076853154113@redacted",
            "079e7bc89d69792839a5e1831b1cbc80@redacted",
            "ce9e248333c44a5a64ccad26f2550f95@redacted",
            "ce9e248333c44a5a64ccad26f2550f95@redacted"
        ],
        "databaseId": 262087
    }
]
```


It's recommended practice to pipe the export into a file, which you can later share with the Mail app community and developers:

```
sudo -E -u www-data php occ mail:account:export-threads 1393 | gzip -c > /tmp/
↪nextcloud-mail-threads-1393.json.gz
```

Get account IDs

For many troubleshooting instructions you need to know the *id* of a mail account. You can acquire this through the database, but it's also possible to utilize the account export command of *occ* if you know the UID of the user utilizing the mail account:

```
sudo -E -u www-data php occ mail:account:export user123
```

The output will look similar to this:

```
Account 1393:
- E-Mail: christoph@domain.com
- Name: Christoph Wurst
- IMAP user: christoph
- IMAP host: mx.domain.com:993, security: ssl
- SMTP user: christoph
- SMTP host: mx.domain.com:587, security: tls
```

In this example, `1393` is the *account ID*.

Issues connecting to Outlook.com

If you can not access your Outlook.com account try to enable the Two-Factor Verification and set up an app password, which you then use for the Nextcloud Mail app.

Logging the IMAP/SMTP/Sieve connections

The Nextcloud Mail app offers an extensive logging system to make it easier identifying and tracking down bugs. As this may include sensitive data, be sure to remove or mask them before posting them publicly.

Per mail account

Added in version 5.1.0: Nextcloud 30 or newer

Starting with version 5.1.0 of the mail app, you can enable logging of outgoing IMAP/SMTP/Sieve connections limited to a specific mail account. As that saves a lot of resources on your system, this is the preferred method for debugging issues regarding IMAP/SMTP/Sieve connections.

First, you need to get the accountId for the mail account you like to enable debug logging on. Please see *Get account IDs* for more.

Once you know the accountId of the mail account in question, you can use it to enable debug logging by running the following command on the server:

```
sudo -E -u www-data php occ mail:account:debug <accountId> --on
```

All subsequent outgoing connections made by the mail app will then be written to the `data` directory. The file naming follows the following format: `mail-{{userId}}-{{accountId}}-{{protocol}}.log` (e.g., *mail-admin-49-imap.log*).

The debug logging for that specific account can be disabled once you've collected the necessary data by running the following command on the server:

```
occ mail:account:debug <accountId> --off
```

Globally

This enables logging of the IMAP/SMTP/Sieve connections for **all** mail accounts configured on the server. This should be used with caution as it can put a lot of strain on large environments.

Added in version 5.1.0: Nextcloud 30 or newer

To enable the global debug logging on versions 5.1.0 and above, just run the following command on the server:

```
sudo -E -u www-data php occ config:system:set app.mail.debug --value true --type bool
```

All subsequent outgoing connections made by the mail app will then be written to the `data` directory. The file naming follows the following format: `mail-{{userId}}-{{accountId}}-{{protocol}}.log` (e.g., *mail-admin-49-imap.log*).

The global debug logging can be disabled once you've collected the necessary data by running the following command on the server:

```
sudo -E -u www-data php occ config:system:set app.mail.debug --value false --type bool
```

Note

The following steps only apply to version 1.6.2 up to 5.0.8. Restrict logging of outgoing connections to a specific mail account is not available there.

Added in version 1.6.2: Nextcloud 20 or newer

To enable the global debug logging, it's necessary to enable both the debug mode as well as debug logging for the whole nextcloud instance by running the following commands on the server:

```
sudo -E -u www-data php occ config:system:set debug --value true --type bool
sudo -E -u www-data php occ config:system:set loglevel --value 0 --type int
```

All subsequent outgoing connections made by the mail app will then be written to the `data` directory. The file naming follows the following format: `horde_{{protocol}}.log` (e.g., *horde_imap.log*).

Once you've collected the necessary data, it's highly recommended to disable the debug mode as well as resetting the loglevel to the default value by running the following commands:

```
sudo -E -u www-data php occ config:system:set debug --value false --type bool
sudo -E -u www-data php occ config:system:set loglevel --value 2 --type int
```

Timeout and other connectivity issues

You can use OpenSSL to test and benchmark the connection from your nextcloud host to the IMAP/SMTP host.:

```
openssl s_time -connect imap.domain.tld:993
```

The output should look similar to this:

```
Collecting connection statistics for 30 seconds
********************************************************************************************************

483 connections in 0.94s; 513.83 connections/user sec, bytes read 0
483 connections in 31 real seconds, 0 bytes read per connection


Now timing with session id reuse.
starting
********************************************************************************************************

497 connections in 0.97s; 512.37 connections/user sec, bytes read 0
497 connections in 31 real seconds, 0 bytes read per connection
```

Manual account synchronization and threading

To troubleshoot synchronization or threading problems it's helpful to run the sync from the command line while the user does not use the web interface (reduces chances of a conflict):

```
sudo -E -u www-data php occ mail:account:sync -vvv 1393
```

Note

1393 represents the *account ID*.

The command offers a `--force` option. Use it wisely as it doesn't perform the same path a typical web triggered sync request would do.

The output will look similar to this:

```
[debug] Skipping mailbox sync for Archive
[debug] Skipping mailbox sync for Archive.2020
[debug] partial sync 1393:Drafts - get all known UIDs took 0s
[debug] partial sync 1393:Drafts - get new messages via Horde took 0s
[debug] partial sync 1393:Drafts - persist new messages took 0s
[debug] partial sync 1393:Drafts - get changed messages via Horde took 0s
[debug] partial sync 1393:Drafts - persist changed messages took 0s
[debug] partial sync 1393:Drafts - get vanished messages via Horde took 0s
[debug] partial sync 1393:Drafts - persist new messages took 0s
[debug] partial sync 1393:Drafts took 0s
[debug] partial sync 1393:INBOX - get all known UIDs took 0s
[debug] partial sync 1393:INBOX - get new messages via Horde took 0s
[debug] partial sync 1393:INBOX - classified a chunk of new messages took 1s
[debug] partial sync 1393:INBOX - persist new messages took 0s
[debug] partial sync 1393:INBOX - get changed messages via Horde took 1s
[debug] partial sync 1393:INBOX - persist changed messages took 0s
[debug] partial sync 1393:INBOX - get vanished messages via Horde took 0s
[debug] partial sync 1393:INBOX - persist new messages took 0s
[debug] partial sync 1393:INBOX took 2s
[debug] Skipping mailbox sync for Sent
[debug] Skipping mailbox sync for Sentry
[debug] Skipping mailbox sync for Trash
[debug] Account 1393 has 19417 messages for threading
[debug] Threading 19417 messages - build ID table took 1s
[debug] Threading 19417 messages - build root container took 0s
[debug] Threading 19417 messages - free ID table took 0s
[debug] Threading 19417 messages - prune containers took 0s
[debug] Threading 19417 messages - group by subject took 0s
[debug] Threading 19417 messages took 1s
[debug] Account 1393 has 9839 threads
[debug] Account 1393 has 0 messages with a new thread IDs
62MB of memory used
```

CHAPTER

TWENTYTHREE

OFFICE

Nextcloud supports two office solutions for collaborative document editing:

Nextcloud Office (powered by Euro-Office)
: The default office solution built into Nextcloud. Enables real-time collaborative editing of text documents, spreadsheets, and presentations with high-fidelity rendering. Requires a Euro-Office Document Server.

Collabora Online
: A fully supported alternative based on the Collabora Online Development Edition (CODE). Available as a self-hosted server or as a built-in app. Enterprise users can use the more stable Collabora Online Enterprise through a Nextcloud support subscription.

Both solutions support dozens of document formats including DOC, DOCX, PPT, PPTX, XLS, XLSX, ODF and many more.

23.1 Nextcloud Office

Nextcloud Office is Nextcloud's built-in office solution, powered by Euro-Office Document Server. It enables real-time collaborative editing of text documents, spreadsheets, and presentations directly in the browser, with high-fidelity rendering that preserves the layout and formatting of your files.

Nextcloud Office supports a wide range of formats including DOCX, XLSX, PPTX, ODT, ODS, ODP and many more.

> **Note**
>
> A dedicated Document Server is required for production use. Install the server first, then connect Nextcloud to it using the **Euro-Office** connector app.

23.1.1 Installation

Nextcloud Office requires a running Euro-Office Document Server that Nextcloud can reach over the network. Choose the installation method that fits your infrastructure:

Distribution	Architectures	Method
Ubuntu 24.04 LTS (Noble)	amd64, arm64	*Installing on Ubuntu 24.04 LTS*
Debian 12 (Bookworm)	amd64, arm64	*Installing on Debian 12*
Any Linux host	amd64, arm64	*Installing via Docker*

Minimum system requirements:

- 4 GB RAM (8 GB recommended for multi-user deployments)

- 10 GB disk space
- The Document Server and Nextcloud must be able to reach each other over HTTPS in production

> **Note**
>
> The Document Server does not need to run on the same host as Nextcloud. A reverse proxy (nginx or Apache) in front of the Document Server is required for production deployments to expose it over HTTPS.

Installing on Ubuntu 24.04 LTS

This guide covers installing Euro-Office Document Server on Ubuntu 24.04 LTS (Noble) from a GitHub release package.

System requirements

- Ubuntu 24.04 LTS (amd64 or arm64)
- 4 GB RAM minimum
- 10 GB disk space minimum

Step 1 — Install prerequisites

Euro-Office requires PostgreSQL, Redis, RabbitMQ, Nginx, and Supervisor. Install them before the package:

```
sudo apt-get update
sudo apt-get install -y postgresql redis-server rabbitmq-server nginx supervisor
```

Step 2 — Create the database

The post-install script connects to PostgreSQL during installation. Create the user and database first:

```
sudo -u postgres psql -c "CREATE USER ds WITH PASSWORD 'ds';"
sudo -u postgres psql -c "CREATE DATABASE ds OWNER ds;"
```

Step 3 — Pre-seed the installer answers

The package installer configures itself non-interactively using debconf. Pre-seed the database connection details so the post-install script can proceed without a prompt:

```
echo "ds ds/db-type select postgres
ds ds/db-host string localhost
ds ds/db-port string 5432
ds ds/db-user string ds
ds ds/db-pwd password ds
ds ds/db-name string ds" | sudo debconf-set-selections
```

Step 4 — Download the package

Download the latest release from GitHub Releases:

```
# Replace <version> and <arch> with your values, e.g. 9.3.1 and amd64 or arm64
wget "https://github.com/Euro-Office/DocumentServer/releases/download/v<version>/euro-
↪office-documentserver_<version>_<arch>.deb" \
  -O /tmp/euro-office-documentserver.deb
```

Step 5 — Install the package

```
sudo apt-get install -y /tmp/euro-office-documentserver.deb
```

The installer generates fonts, WOPI keys, and JS caches. This takes a minute or two. A successful install ends with:

```
Congratulations, the Euro-Office DocumentServer has been installed successfully!
```

Step 6 — Verify

Check that all services are running:

```
systemctl is-active ds-docservice ds-converter ds-metrics nginx
```

Expected output:

```
active
active
active
active
```

Run the health check:

```
curl http://localhost/healthcheck
```

Expected output: `true`

Updating

Download the new release package and reinstall:

```
wget "https://github.com/Euro-Office/DocumentServer/releases/download/v<new-version>/
↪euro-office-documentserver_<new-version>_<arch>.deb" \
  -O /tmp/euro-office-documentserver.deb
sudo apt-get install -y /tmp/euro-office-documentserver.deb
```

Uninstalling

```
sudo apt-get remove --purge euro-office-documentserver
sudo -u postgres psql -c "DROP DATABASE ds;"
sudo -u postgres psql -c "DROP USER ds;"
```

Installing on Debian 12

This guide covers installing Euro-Office Document Server on Debian 12 (Bookworm) from a GitHub release package.

System requirements

- Debian 12 (Bookworm) — amd64 or arm64
- 4 GB RAM minimum
- 10 GB disk space minimum

Step 1 — Enable the contrib component

The `ttf-mscorefonts-installer` package required by Euro-Office lives in Debian's `contrib` component, which is not enabled by default. Edit `/etc/apt/sources.list` and add `contrib` to each line:

```
deb http://deb.debian.org/debian bookworm main contrib
deb http://deb.debian.org/debian bookworm-updates main contrib
deb http://deb.debian.org/debian-security/ bookworm-security main contrib
```

Then update the package index:

```
sudo apt-get update
```

Step 2 — Install prerequisites

All prerequisites are available in the default Debian 12 repositories — no extra package sources are required:

```
sudo apt-get install -y postgresql redis-server rabbitmq-server nginx supervisor
```

Step 3 — Create the database

The post-install script connects to PostgreSQL during installation. Create the user and database first:

```
sudo -u postgres psql -c "CREATE USER ds WITH PASSWORD 'ds';"
sudo -u postgres psql -c "CREATE DATABASE ds OWNER ds;"
```

Step 4 — Pre-seed the installer answers

```
echo "ds ds/db-type select postgres
ds ds/db-host string localhost
ds ds/db-port string 5432
ds ds/db-user string ds
ds ds/db-pwd password ds
ds ds/db-name string ds" | sudo debconf-set-selections
```

Step 5 — Download the package

Download the latest release from GitHub Releases:

```
# Replace <version> and <arch> with your values, e.g. 9.3.1 and amd64 or arm64
wget "https://github.com/Euro-Office/DocumentServer/releases/download/v<version>/euro-
↪office-documentserver_<version>_<arch>.deb" \
  -O /tmp/euro-office-documentserver.deb
```

Step 6 — Install the package

```
sudo apt-get install -y /tmp/euro-office-documentserver.deb
```

The installer generates fonts, WOPI keys, and JS caches. This takes a minute or two. A successful install ends with:

```
Congratulations, the Euro-Office DocumentServer has been installed successfully!
```

Step 7 — Verify

Check that all services are running:

```
systemctl is-active ds-docservice ds-converter ds-metrics nginx
```

Expected output:

```
active
active
active
active
```

Run the health check:

```
curl http://localhost/healthcheck
```

Expected output: `true`

Updating

Download the new release package and reinstall:

```
wget "https://github.com/Euro-Office/DocumentServer/releases/download/v<new-version>/
↪euro-office-documentserver_<new-version>_<arch>.deb" \
  -O /tmp/euro-office-documentserver.deb
sudo apt-get install -y /tmp/euro-office-documentserver.deb
```

Uninstalling

```
sudo apt-get remove --purge euro-office-documentserver
sudo -u postgres psql -c "DROP DATABASE ds;"
sudo -u postgres psql -c "DROP USER ds;"
```

Installing via Docker

The quickest way to run Euro-Office Document Server is via the official Docker image.

Prerequisites

- Docker Engine 20.10 or later
- 4 GB RAM minimum
- 5 GB disk space for the image

Quick start

```
docker run -d \
  --name euro-office \
  --restart=unless-stopped \
  -p 80:80 \
  -e JWT_ENABLED=true \
  -e JWT_SECRET=your-secret \
  ghcr.io/euro-office/documentserver:latest
```

Warning

Replace `your-secret` with a strong random string. The JWT secret is shared between the Document Server and the Nextcloud connector app — both must use the same value.

The server is ready when the health check returns `true`:

```
curl http://localhost/healthcheck
```

Image tags

Tag	Use
`latest`	Most recent release — use in production
`nightly`	Nightly builds from `main` — not for production
`latest-dev`	Development image with build tools included

Pin to a specific version in production:

```
ghcr.io/euro-office/documentserver:9.3.1
```

Persistent data

By default, documents, configuration, and the secrets generated on first start are lost when the container is removed. Mount volumes to persist them:

```
docker run -d \
  --name euro-office \
  --restart=unless-stopped \
  -p 80:80 \
  -e JWT_ENABLED=true \
  -e JWT_SECRET=your-secret \
  -v /path/to/data:/var/lib/euro-office/documentserver \
  -v /path/to/private:/var/www/euro-office/Data \
  -v /path/to/logs:/var/log/euro-office/documentserver \
  -v /path/to/config:/etc/euro-office/documentserver \
  ghcr.io/euro-office/documentserver:latest
```

`/var/www/euro-office/Data` holds state the container creates at run time and is easy to overlook, because it is a separate tree from `/var/lib/euro-office/documentserver`:

- `runtime.json` — the administration panel password and any settings changed at run time

- `.private/jwt_secret` — only created when `JWT_SECRET` is not supplied
- `.private/secure_link_secret` — only created when `SECURE_LINK_SECRET` is not supplied
- `wopi_private.key` and `wopi_public.key` — only created when `WOPI_ENABLED=true`

Without this volume all of it is discarded when the container is recreated, and the generated values differ on the next start. Secrets you pass in as environment variables are not affected, but a regenerated JWT secret no longer matches the one configured in the Nextcloud connector app, and the connection stays broken until the new value is copied over.

Environment variables

The variables below are read by the container entrypoint and written into `local.json` and the nginx configuration at start-up. The four variables that limit file sizes are documented separately in *Size limits*.

Authentication

Variable	Default	Description
`JWT_ENABLED`	`true`	Enable JWT authentication
`JWT_SECRET`	generated	Shared secret. When unset, a random 32-character secret is generated and stored in `/var/www/euro-office/Data/.private/jwt_secret`. Set it explicitly in production
`JWT_HEADER`	`Authorization`	HTTP header carrying the JWT
`JWT_HEADER_INBOX` / `JWT_HEADER_OUTBOX`	`JWT_HEADER`	Separate headers per direction
`JWT_IN_BODY`	`false`	Accept the token in the request body
`JWT_SECRET_INBOX` / `JWT_SECRET_OUTBOX`	`JWT_SECRET`	Separate secrets per direction
`JWT_ENABLED_INBOX` / `JWT_ENABLED_OUTBOX`	`JWT_ENABLED`	Enable JWT per direction

Database

Variable	Default	Description
`DB_TYPE`	`postgres`	Database engine (standalone image supports `postgres` only)
`DB_HOST`	`localhost`	Database host
`DB_PORT`	`5432`	Database port
`DB_NAME`	`eurooffice`	Database name
`DB_USER`	`eurooffice`	Database user
`DB_PWD`	—	Database password

Note

`DB_PASSWORD` is a deprecated alias for `DB_PWD`. It still works, but the container logs a warning at start-up. `DB_PWD` wins if both are set.

Redis

Variable	Default	Description
REDIS_SERVER_HOST	localhost	Redis host (for external Redis)
REDIS_SERVER_PORT	6379	Redis port
REDIS_SERVER_USER	—	Redis username for ACL-based authentication. Only written to the configuration when set
REDIS_SERVER_PASS	—	Redis password. Only written to the configuration when set
REDIS_SERVER_DB	—	Redis database number. Only written to the configuration when set

RabbitMQ

Variable	Default	Description
AMQP_HOST	localhost	RabbitMQ host (for external RabbitMQ)
AMQP_PORT	5672	RabbitMQ port
AMQP_USER / AMQP_PWD	guest	RabbitMQ credentials
AMQP_VHOST	/	Virtual host name. A leading slash is added automatically if you omit it
AMQP_URI	—	Complete AMQP connection URI. Takes precedence over all other AMQP_* variables

Note

The container only points the Document Server at an external broker when AMQP_URI is set or AMQP_HOST differs from localhost. Otherwise the RabbitMQ instance bundled in the image is used.

WOPI

Variable	Default	Description
WOPI_ENABLED	false	Enable WOPI protocol support. An RSA key pair is generated in /var/www/euro-office/Data on first start

Outbound requests

These variables control how the Document Server fetches documents from storage such as Nextcloud.

Variable	Default	Description
ALLOW_PRIVATE_IP_ADDRESS	false	Allow the Document Server to fetch files from private IP ranges
ALLOW_META_IP_ADDRESS	false	Allow fetching documents from meta-private IPs (169.254.0.0/16)
USE_UNAUTHORIZED_STORAGE	false	Disable TLS certificate validation for outbound requests to storage

Warning

`USE_UNAUTHORIZED_STORAGE=true` sets `rejectUnauthorized: false` on every outbound HTTPS connection the Document Server makes. The certificate chain, the host name, and the expiry date of the storage server are no longer checked, so any machine on the network path can impersonate your Nextcloud instance and read or modify the documents in transit. It has no effect on plain HTTP connections, which never present a certificate.

Only use it for self-signed certificates in a trusted network, and prefer adding the certificate authority to the container's trust store instead.

HTTPS and TLS termination

Variable	Default	Description
`SSL_CERTIFICATE_PATH`	—	Path inside the container to the TLS certificate
`SSL_KEY_PATH`	—	Path inside the container to the matching private key
`SSL_DHPARAM_PATH`	—	Path to a Diffie-Hellman parameter file. When unset or unreadable, the `ssl_dhparam` directive is removed
`SSL_VERIFY_CLIENT`	`off`	Value for the nginx `ssl_verify_client` directive (client certificate verification)
`ONLYOFFICE_HTTPS_HSTS_ENABLED`	`true`	Send an HSTS header. When `false`, the `max-age` directive is removed
`ONLYOFFICE_HTTPS_HSTS_MAXAGE`	`31536000`	HSTS max-age in seconds (default 1 year)

Important

`SSL_DHPARAM_PATH`, `SSL_VERIFY_CLIENT`, and both `ONLYOFFICE_HTTPS_HSTS_*` variables are only applied when `SSL_CERTIFICATE_PATH` and `SSL_KEY_PATH` are both set **and** both files exist inside the container. Without them the container serves plain HTTP and these four variables have no effect at all.

Nginx

Variable	Default	Description
`NGINX_WORKER_PROCESSES`	`1`	Number of nginx worker processes
`NGINX_WORKER_CONNECTIONS`	`768`	Value for the nginx `worker_connections` directive. When unset, the value from the base image's `nginx.conf` is left unchanged
`NGINX_ACCESS_LOG`	`false`	Write an access log to `/var/log/euro-office/documentserver/nginx.access.log`
`NGINX_CLIENT_MAX_BODY_SIZE`	`100m`	Maximum size of an inbound request body — see *Size limits*
`SECURE_LINK_SECRET`	generated	Secret used for the nginx secure link URLs. When unset, a random 20-character secret is generated and stored in `/var/www/euro-office/Data/.private/secure_link_secret`

Metrics

Variable	Default	Description
METRICS_ENABLED	false	Enable StatsD metrics collection and start the metrics service
METRICS_HOST	localhost	StatsD host
METRICS_PORT	8125	StatsD port
METRICS_PREFIX	ds.	Prefix prepended to every metric name

Note

METRICS_HOST, METRICS_PORT, and METRICS_PREFIX are only written to the configuration when METRICS_ENABLED=true.

Logging and optional services

Variable	Default	Description
DS_LOG_LEVEL	WARN	log4js level for the default category, for example ERROR, WARN, INFO, or DEBUG
PLUGINS_ENABLED	true	Enable editor plugins
GENERATE_FONTS	true	Regenerate font cache on startup
ADMINPANEL_ENABLED	false	Start the administration panel service
EXAMPLE_ENABLED	false	Start the bundled example application at /example/. Do not enable it on a public instance

Size limits

Four variables limit file sizes, and they apply to two independent paths through the Document Server. Which variables matter depends on which path the file takes — raising the wrong pair has no effect.

Documents the server downloads

When a user opens or converts a document that is already stored in Nextcloud, the Document Server fetches the file itself over HTTP from Nextcloud. Nothing is uploaded from the browser, so neither the nginx request body limit nor the internal request body limit is involved. Two variables apply:

1. The server downloads the file — FILECONVERTER_MAX_DOWNLOAD_BYTES is the maximum number of bytes the Document Server will fetch. Default is 524288000 (500 MB). A larger document fails to open:

```
-e FILECONVERTER_MAX_DOWNLOAD_BYTES=838860800
```

2. The server unzips the archive — FILECONVERTER_INPUT_LIMIT_UNCOMPRESSED limits the *uncompressed* size of the XML inside the office file's ZIP container. Default is 500MB:

```
-e FILECONVERTER_INPUT_LIMIT_UNCOMPRESSED=800MB
```

For example, a user opens a **200 MB** .pptx. The 500 MB download limit is already sufficient, so no change is needed there. But a 200 MB presentation with embedded images, shapes, or animations can hold 800 MB of uncompressed XML, which exceeds the 500 MB default — so only FILECONVERTER_INPUT_LIMIT_UNCOMPRESSED has to be raised.

Note

`FILECONVERTER_MAX_DOWNLOAD_BYTES` must be a plain byte count with no unit suffix. A value such as `800MB` is rejected: the container writes a warning to its standard error stream at start-up, silently keeps the built-in default, and starts normally. Check the container log after changing it.

Note

`FILECONVERTER_INPUT_LIMIT_UNCOMPRESSED` accepts a size suffix and applies the same value to all four format groups the Document Server knows — `docx`, `xlsx`, `pptx`, and `vsdx` and their variants. It replaces the whole limit list rather than patching a single entry, so a format group added by a future release would lose its own default until this variable is updated.

Files posted to the server

These limits apply to requests that carry a file in the request body — inserting an image into an open document, saving a document back, and conversion or command requests posted to the Document Server. They do **not** apply to opening a stored document.

1. Nginx accepts the request body — `NGINX_CLIENT_MAX_BODY_SIZE` must be higher than the payload. Default is `100m`; a larger body is rejected with `413 Request Entity Too Large`:

```
-e NGINX_CLIENT_MAX_BODY_SIZE=250m
```

2. The Document Server parses the body — `MAX_FILE_SIZE` is the internal request body limit in bytes. Default is `104857600` (100 MB):

```
-e MAX_FILE_SIZE=268435456
```

Both have to be raised together. Nginx rejects the request first, so raising only `MAX_FILE_SIZE` changes nothing.

Summary

Variable	Default	Applies to
`FILECONVERTER_MAX_DOWNLOAD_BYTES`	`524288000`	Documents the server downloads from Nextcloud
`FILECONVERTER_INPUT_LIMIT_UNCOMPRE`	`500MB`	Uncompressed XML inside any office file the server opens
`NGINX_CLIENT_MAX_BODY_SIZE`	`100m`	Request bodies posted to the server (image inserts, save-back, conversion)
`MAX_FILE_SIZE`	`104857600`	Request bodies posted to the server (image inserts, save-back, conversion)

Updating

```
docker pull ghcr.io/euro-office/documentserver:latest
docker stop euro-office && docker rm euro-office
# re-run with the same docker run command used during installation
```

Uninstalling

```
docker stop euro-office
docker rm euro-office
docker rmi ghcr.io/euro-office/documentserver:latest
```

23.1.2 Configuration

After installing the Euro-Office Document Server, install the Nextcloud connector app and connect it to the server.

Installing the connector app

Install the **Euro-Office** app from the Nextcloud App Store:

1. Open *Settings* ▸ *Apps* in your Nextcloud admin interface.
2. Search for **Euro-Office**.
3. Click **Download and enable**.

Alternatively, install the app using the `occ` command:

```
sudo -E -u www-data php occ app:install eurooffice
```

Server settings

Open *Settings* ▸ *Administration* ▸ *Euro-Office* to configure the connection.

Nextcloud Office address
: The URL of the Euro-Office Document Server, accessible from both the Nextcloud server and end-user browsers:

  ```
  https://<documentserver>/
  ```

Disable certificate verification (insecure)
: Disables TLS certificate checks. Only enable this in test environments.

Secret key (leave blank to disable)
: The JWT shared secret. Must match the value configured on the Document Server:

  - **deb/rpm install:** `/etc/euro-office/documentserver/local.json` → `services.CoAuthoring.secret.browser.string`
  - **Docker install:** the `JWT_SECRET` environment variable

Click **Save** to apply. A green status indicator confirms a successful connection.

Advanced server settings

Expand **Advanced server settings** when the public Document Server URL is not directly reachable from your Nextcloud host (for example, behind a NAT or split-horizon DNS):

Authorization header (leave blank to use default header)
: The HTTP header used to carry the JWT token. Defaults to `Authorization`.

Nextcloud Office address for internal requests from the server
: The URL Nextcloud uses to contact the Document Server directly (server-to-server). Must be reachable from the Nextcloud host.

Server address for internal requests from Nextcloud Office
: The URL the Document Server uses to send callbacks back to Nextcloud. Must be reachable from the Document Server host.

Tip

Docker Compose example

When Nextcloud and Euro-Office run as services in the same Docker Compose network, containers can reach each other by service name. Given a `compose.yml` with services named `nextcloud` and `eurooffice`:

- **Nextcloud Office address** (public): `https://office.example.com/`
- **Nextcloud Office address for internal requests from the server**: `http://eurooffice/`
- **Server address for internal requests from Nextcloud Office**: `http://nextcloud/`

The public address is used by end-user browsers. The internal addresses bypass the public reverse proxy and let the two containers communicate directly over the shared Docker network.

Common settings

These settings become available after a successful server connection.

Setting	Description
Allow the following groups to access the editors	Restricts editor access to specified Nextcloud groups.
Use Nextcloud Office to generate a document preview	Generates thumbnail previews for Office files. Takes up additional disk space.
Open file in the same tab	Opens documents inside Nextcloud instead of a new browser tab.
Enable sharing	Allows users to share files from within the editor. May increase editor load time.
Provide advanced document permissions	Enables per-user fine-grained permissions (review-only, comment-only, etc.) on shared files via the sharing dialog.
Keep metadata for each version once the document is edited	Stores version metadata each time a document is saved. Takes up additional disk space.
Enable background connection check to the editors	Runs a periodic cron job to verify the Document Server is reachable.
Enable e-mail notifications	Sends e-mail notifications on comments and edits.
Unknown author display name	Display name shown for edits made by users not known to Nextcloud.

Default application for opening the format
: Checkboxes for each supported file format. When enabled for a format, Nextcloud Office becomes the default handler when that file type is clicked.

Open the file for editing
: Checkboxes for formats that can be edited natively. Enabling a format allows users to edit it directly. Note that saving back to some formats may result in data loss due to format restrictions.

Editor customization settings

Setting	Description
Keep intermediate versions when editing (forcesave)	Saves an intermediate version each time a user explicitly saves, in addition to the final version saved when the editor closes.
Enable live-viewing mode when accessing file by public link	Shows live updates to external viewers accessing a file via a public link.
Display Chat menu button	Shows or hides the in-editor chat panel.
Display the header more compact	Displays a more compact toolbar header.
Display Feedback & Support menu button	Shows or hides the Feedback & Support entry in the editor menu.
Display Help menu button	Shows or hides the Help entry in the editor menu.

REVIEW mode for viewing
: Default mode for displaying tracked changes: **Markup**, **Final**, or **Original**.

Default editor theme
: Color theme for the editor interface: **Same as system**, **Light**, or **Dark**.

Common templates

Administrators can upload shared document templates that are available to all users when creating new files. Templates are managed in the **Common templates** section of the admin settings page.

Security

Enable plugins
: Allows third-party plugins to run inside the editor.

Run document macros
: Allows macros embedded in documents to execute.

Enable document protection for
: Controls who can protect or unprotect documents:

- **All users** — any editor can protect or unprotect the document.
- **Owner only** — only the file owner can change protection.

Watermarks

Watermarks are overlaid on documents when they are opened by specified users or via shares. Enable **Enable watermarking** to configure the watermark.

Watermark text — the text overlaid on the document. Supported placeholders:

- `{userId}` — Nextcloud user ID
- `{userDisplayName}` — user's display name
- `{email}` — user's e-mail address
- `{date}` — current date
- `{themingName}` — Nextcloud instance name from theming settings

Watermark visibility can be configured per audience:

Option	When the watermark is shown
Show watermark on tagged files	Files with specific system tags (requires the System Tags app).
Show watermark for users of groups	Users belonging to selected groups.
Show watermark for all shares	Any user who accesses the file via a share.
Show watermark for read only shares	Users with read-only share access.
Show watermark for all link shares	Anyone accessing the file via a public link.
Show watermark for download hidden shares	Public links where file download is disabled.
Show watermark for read only link shares	Public links with read-only access.
Show watermark on link shares with specific system tags	Public links on files with specific system tags.

occ commands

Settings can also be managed via **occ**:

```
# Read a setting
sudo -E -u www-data php occ config:app:get eurooffice DocumentServerUrl

# Write a setting
sudo -E -u www-data php occ config:app:set eurooffice DocumentServerUrl --value=
↪"https://office.example.com/"
```

The `eurooffice:documentserver` command shows the configured server URL and optionally tests the connection:

```
# Print the currently configured Document Server URL
sudo -E -u www-data php occ eurooffice:documentserver

# Check the connection to the Document Server
sudo -E -u www-data php occ eurooffice:documentserver --check
```

config.php

Settings can also be set directly in `config/config.php` under the `eurooffice` key:

```
"eurooffice" => [
    "DocumentServerUrl" => "https://office.example.com/",
    "jwt_secret" => "your-secret",
    "verify_peer_off" => "true",
],
```

Note

`verify_peer_off` is read as a string — use `"true"` to disable certificate verification, not a PHP boolean.

Settings defined in `config.php` are used as fallback when no value is set via the admin UI or **occ**. The UI takes precedence.

23.2 Collabora Online

Nextcloud supports Collabora Online as a fully supported office solution. Collabora Online is available as a self-hosted server and as a built-in CODE (Collabora Online Development Edition) server app.

23.2.1 Installation

Nextcloud Office is built on Collabora Online which requires a dedicated service running next to the Nextcloud webserver stack. There are several ways to run the coolwsd service.

- **Nextcloud All In One:** Nextcloud Office comes preinstalled out of the box in the Nextcloud All In One setup and provides easy deployment and maintenance with most features included in this one Nextcloud instance.

For manual installations there are multiple options to get Nextcloud Office deployed:

- **Installation through distribution packages**
  There are packages for all major Linux distributions available which allow deploying a Collabora Online server through installing it through the regular package management. For an example installation guide on Ubuntu, see: *Installation example on Ubuntu 24.04*

  See also

  https://www.collaboraoffice.com/code/linux-packages/ https://sdk.collaboraonline.com/docs/installation/index.html

- **Installation through Docker**
  Docker images are available for deploying the Collabora Online server in container environments. For a detailed step by step guide, see: *Installation example with Docker*

  See also

  https://sdk.collaboraonline.com/docs/installation/CODE_Docker_image.html

- **Built-in CODE server**
  This app provides a built-in server with all of the document editing features of Collabora Online. Easy to install, for personal use or for small teams. A bit slower than a standalone server and without the advanced scalability features. Installation can be performed by enabling the according Nextcloud app. Further details can be found in the app documentation.

  Note

  This is the default option which works out of the box in most scenarios, however for improved performance it is highly recommended to switch to a dedicated Collabora Online installation using one of the other options.

Note

In most scenarios running a dedicated Collabora Online server will require some sort of reverse proxy to be setup in front of it. For more details see *Reverse proxy*.

Installation example on Ubuntu 24.04

Import signing keys:

```
cd /usr/share/keyrings && sudo wget https://collaboraoffice.com/downloads/gpg/
↪collaboraonline-release-keyring.gpg
```

Add repository:

```
sudo echo -e "Types: deb\nURIs: https://www.collaboraoffice.com/repos/CollaboraOnline/
↪CODE-deb\nSuites: ./\nSigned-By: /usr/share/keyrings/collaboraonline-release-
↪keyring.gpg" > /etc/apt/sources.list.d/collaboraonline.sources
```

Install packages

```
sudo apt update && sudo apt install coolwsd code-brand
```

Configuration

Edit /etc/coolwsd/coolwsd.xml. Collabora Online (coolwsd) service runs via systemd. After editing the configuration file, you have to restart the service:

```
sudo systemctl restart coolwsd
```

The default configuration is looking for an SSL certificate and key, which are not present, so probably it's the best to disable SSL, and optionally enable SSL termination, then set up the reverse proxy.

See also

Full configuration examples for reverse proxy setup can be found in the Collabora Online documentation: https://sdk.collaboraonline.com/docs/installation/Proxy_settings.html

```
sudo coolconfig set ssl.enable false
sudo coolconfig set ssl.termination true
sudo coolconfig set storage.wopi.host nextcloud.example.com
sudo coolconfig set-admin-password
sudo systemctl restart coolwsd
systemctl status coolwsd
```

Installation example with Docker

We'll describe how to get Nextcloud Office running on your server and how to integrate it into your Nextcloud using the docker image Nextcloud and Collabora built.

To install it the following dependencies are required:

- A host that can run a Docker container
- A subdomain or a second domain that the Collabora Online server can run on
- An Apache server with some enabled modules
- A valid SSL certificate for the domain that Collabora Online should run on

- A valid SSL certificate for your Nextcloud

Install the Collabora Online server

The following steps will download the Collabora Online docker. Make sure to replace "cloud.example.com" with the host that your own Nextcloud runs on. If you want to use the docker container with more than one Nextcloud, you can add another *-e aliasgroup2=https://cloud2.example.com:443*.

```
docker pull collabora/code
docker run -t -d -p 127.0.0.1:9980:9980 \
    -e 'aliasgroup1=https://cloud.example.com:443' \
    --restart always \
    --cap-add MKNOD \
    collabora/code
```

That will be enough. Once you have done that the server will listen on "localhost:9980". Now we just need to configure the locally installed Apache reverse proxy.

Install the Apache reverse proxy

On a recent Ubuntu or Debian this should be possible using:

```
apt-get install apache2
a2enmod proxy proxy_wstunnel proxy_http ssl
```

Afterward, configure one VirtualHost properly to proxy the traffic. For security reasons, we recommend using a subdomain such as office.example.com instead of running on the same domain. An example config can be found below:

```
########################################
# Reverse proxy for Collabora Online
########################################

AllowEncodedSlashes NoDecode
SSLProxyEngine On
ProxyPreserveHost On

# cert is issued for collaboraonline.example.com and we proxy to localhost
SSLProxyVerify None
SSLProxyCheckPeerCN Off
SSLProxyCheckPeerName Off

# static html, js, images, etc. served from coolwsd
# browser is the client part of Collabora Online
ProxyPass           /browser https://127.0.0.1:9980/browser retry=0
ProxyPassReverse    /browser https://127.0.0.1:9980/browser

# WOPI discovery URL
ProxyPass           /hosting/discovery https://127.0.0.1:9980/hosting/discovery␣
↪retry=0
ProxyPassReverse    /hosting/discovery https://127.0.0.1:9980/hosting/discovery

# Capabilities
ProxyPass           /hosting/capabilities https://127.0.0.1:9980/hosting/capabilities␣
↪retry=0
```

(continues on next page)

(continued from previous page)

```
ProxyPassReverse    /hosting/capabilities https://127.0.0.1:9980/hosting/capabilities

# Main websocket
ProxyPassMatch      "/cool/(.*)/ws$"      wss://127.0.0.1:9980/cool/$1/ws nocanon

# Admin Console websocket
ProxyPass           /cool/adminws wss://127.0.0.1:9980/cool/adminws

# Download as, Fullscreen presentation and Image upload operations
ProxyPass           /cool https://127.0.0.1:9980/cool
ProxyPassReverse    /cool https://127.0.0.1:9980/cool
# Compatibility with integrations that use the /lool/convert-to endpoint
ProxyPass           /lool https://127.0.0.1:9980/cool
ProxyPassReverse    /lool https://127.0.0.1:9980/cool
```

After configuring these do restart your apache using `systemctl restart apache2`.

See also

Full configuration examples for reverse proxy setup can be found in the Collabora Online documentation: https://sdk.collaboraonline.com/docs/installation/Proxy_settings.html

Configure the app in Nextcloud

- Go to the Apps section and choose “Office & text”
- Install the “Nextcloud Office” app
- Go to Admin -> Office -> Specify the server you have setup before (e.g. “https://office.example.com”)

Congratulations, your Nextcloud has Collabora Online Office integrated!

Updating

Occasionally, new versions of this docker image are released with security and feature updates. We will of course let you know when that happens! This is how you upgrade to a new version:

Update the docker image:

```
docker pull collabora/code
```

List running docker containers:

```
docker ps
```

Stop and remove the Collabora Online container with the container id of the running one:

```
docker stop CONTAINER_ID
docker rm CONTAINER_ID
```

Start the new container:

```
docker run -t -d -p 127.0.0.1:9980:9980 -e 'domain=cloud\\.example\\.com' \
    --restart always --cap-add MKNOD collabora/code
```

Reverse proxy

The server part of Nextcloud Office (coolwsd daemon) is listening on port 9980 by default, and clients should be able to communicate with it through port 9980. However on most setups it is common to use a reverse proxy to more easily handle SSL termination and have a unified entrypoint for HTTP requests.

The following rules should be in place to forward requests to the coolwsd daemon on port 9980:

- /browser
- /hosting/discovery
- /hosting/capabilities
- /cool/adminws
- /cool
- Web socket connections through /cool/(.*)/ws

See also

Full configuration examples can be found in the Collabora Online documentation: https://sdk.collaboraonline.com/docs/installation/Proxy_settings.html

23.2.2 Configuration

Nextcloud Office App Settings

Collabora Online Server

URL (and port) of the Collabora Online server that provides the editing functionality as a WOPI client. Collabora Online should use the same protocol (`http://` or `https://`) as the server installation. Naturally, `https://` is recommended.

Restrict usage to specific groups

By default the app is enabled for all. When this setting is active, only members of specified groups can use Nextcloud Office.

Restrict edit to specific groups

By default all users can edit documents with Nextcloud Office. When this setting is active, only the members of specified groups can edit, others can only view documents.

Use OOXML by default for new files

By default new files created by users are in OpenDocument Format (ODF). When this setting is active, new files will be created in Office Open XML (OOXML) format.

Enable access for external apps

Nextcloud internally passes an access token to Collabora Online that is used later by it to do various operations. By default, it's not possible to generate this token by 3rd parties; only Nextcloud can generate and pass it to Collabora Online.

In some applications, it might be necessary to generate the token by a 3rd party application. For this, one needs to add the 3rd party application (external apps) in this setting. You need to add an application identifier and a secret token. These credentials then can be used by the 3rd party application to make calls to *ajax/extapp/data/{fileId}* to fetch the access token and URL source for given fileId, both required to open a connection to Collabora Online.

Canonical webroot

Canonical webroot, in case there are multiple, for Collabora Online to use. Provide the one with least restrictions. E.g.: Use non-shibbolized webroot if this instance is accessed by both shibbolized and non-shibbolized webroots. You can ignore this setting if only one webroot is used to access this instance.

Additional configuration options

The coolwsd service allows additional configuration options which can be found in the Collabora Online documentation.

Previews

In order to allow Nextcloud to use the coolwsd conversion API to generate previews, the Nextcloud host IP needs to be added to the allow list:

```
sudo coolconfig set net.post_allow.host 10.0.0.4
```

Custom fonts

When you install coolwsd package, the post-install script will look for additional fonts on your system, and install them in the systemplate. If you install fonts to your system after installing coolwsd, you need to update the systemplate manually.

```
coolconfig update-system-template
```

See also

https://sdk.collaboraonline.com/docs/installation/Fonts.html

Secure view settings

The secure view settings enables Nextcloud to embed watermarks on your office files. The watermark may be set according to different rules:

- **Tags:** will watermark files for files containing the defined tags
- **Groups:** will watermark files when opened by users belonging to the defined groups.
- **All shares:** will watermark files accessed via a share.
- **Read-only shares:** will watermark files if they are accessed via a read-only share.

Warning

To enforce the confidentiality of your files it is crucial to restrict the ability to download the documents.

This includes ensuring that your *WOPI configuration* is configured to only serve documents between Nextcloud and Collabora.

Wopi configuration

It is highly recommended to restrict WOPI requests to the IP addresses of the Collabora servers that are expected to request files from the Nextcloud installation. This can be done by setting the `Allow list for WOPI requests` option from the Office admin settings.

Similarly, it is advised to configure Collabora's WOPI host configuration to only serve IPs from expected hosts.

23.2.3 Migration from Collabora Online

Nextcloud Office is based on Collabora Online so for enabling all Nextcloud Office functionality it would be enough to update to the most recent release. Nextcloud Office is available since CODE 21.11.

> **Note**
>
> This upgrade guide is aimed for upgrading from CODE 6.4 to CODE 21.11.

Update the reverse proxy configuration

Due to naming changes in the Collabora Online releases it may be required to adjust reverse proxy configurations that are already in use for previously existing setups.

- Paths with `lool` have been renamed to `cool`
- Paths with `loleaflet` have been renamed to `browser`

Fully detailed reverse proxy configuration guides for various solutions can be found at https://sdk.collaboraonline.com/docs/installation/Proxy_settings.html

Upgrade distribution packages

- The main service has been renamed from `loolwsd` to `coolwsd`
- The service rename also affects the location of the configuration file `/etc/coolwsd/coolwsd.xml`

Required upgrade steps:

- Stop the `loolwsd` service
- Backup `/etc/loolwsd/loolwsd.xml` configuration file.
- Remove `loolwsd` and `collaboraoffice*` packages.
- Change the version number in the repository URL, e.g. from 6.4 to 21.11
- Install the `coolwsd` package
- Adapt the new configuration file in `/etc/coolwsd/coolwsd.xml` to match your previous configuration
- Start and enable the `coolwsd` service

Upgrade the docker image

For upgrading the docker images it is enough to pull the latest CODE image from Docker Hub.

23.2.4 Troubleshooting

In case of connectivity issues, ensure that the following required connections are possible and not blocked by any firewall:

- The users browser can reach both the Nextcloud Server as well as the Collabora Online server through HTTP(S)
- The Nextcloud and the Collabora Online server are using the same protocol
- The Nextcloud server can reach the Collabora Online server through HTTP(S)
- The Collabora Online server can reach the Nextcloud server through HTTP(S)

Both the Nextcloud log as well as the Collabora Online server log may reveal more detailed error messages in case of connection issues.

- **Verify connectivity from the browser:**

  - https://office.example.com/hosting/capabilities
  - https://office.example.com/hosting/discovery

- **Verify connectivity from Nextcloud**
  - `curl https://office.example.com/hosting/capabilities`
  - `curl https://office.example.com/hosting/discovery`

- **Verify connection from the Collabora server**
  - `curl https://nextcloud.example.com/status.php`

Frequently asked questions

Issue: I get connection errors when trying to open documents
Be sure to check the error log from docker through `docker logs container-id`. If the logs note something like: `No acceptable WOPI hosts found matching the target host [YOUR NEXTCLOUD DOMAIN] in config.` Unauthorized WOPI host. Please try again later and report to your administrator if the issue persists. You might have started the docker container with the wrong URL. Be sure to triplecheck that you start it with the URL of your Nextcloud server, not the server where Collabora Online runs on.

Issue: Connection is not allowed errors.
It is possible your firewall is blocking connections. Try to start docker after you started the firewall, it makes changes to your iptables to enable Collabora Online to function.

Issue: We are sorry, this is an unexpected connection error. Please try again. error.
The Collabora Online app doesn't work at the moment, if you enable it only for certain groups. Remove the group filter in the App section.

Issue: Collabora Online doesn't handle my 100 users.
This docker image is designed for home usage. If you need a more scalable solution, consider a support subscription for a reliable, business-ready online office experience.

Issue: Collabora Online doesn't work with Encryption.
Yes, this is currently unsupported.

Issue: Nextcloud Office could not connect to the built-in Collabora Online - Built-in CODE server.
Make sure that the Nextcloud instance is able to reach itself using the same hostname that is used to access through the browser.

You might want to add your Nextcloud domain to /etc/hosts to ensure the connectivity if DNS resolution doesn't work for this: ` ``127.0.0.1 cloud.yourdomain.com`` `

If you continue to see a warning about the WOPI allow list, make sure to also add the IP to the allow list under Settings/Administration/Office.

CHAPTER

TWENTYFOUR

COLLECTIVES

24.1 Runtime Dependencies

Collectives requires the following apps to be enabled. They're all shipped and enabled by default with the Nextcloud server installation:

- **Teams** (with Nextcloud >= 29) or **Circles** (with Nextcloud <= 28)
- **Text**
- **Viewer**
- **files_versions**

24.2 Collectives and *group_everyone*

When using the group_everyone app, existing users will not see collectives with the "everyone" group as member. The group members need to be synced once in the circles app: `occ circles:sync --groups`

This only needs to be done once. New users that got created after the app was enabled will see the collectives straight away.

24.3 Collectives and guest users

In order to allow guest users (as provided by the guests app to access collectives, add the Collectives and Teams apps to the list of enabled apps for guest users in admin settings.

Please note that this enables guest users to create new collectives.

24.4 Public shares

WebDAV access to public shares must not be disabled (i.e. it must be enabled) for publicly shared collectives to work. Please make sure that the following admin option is enabled: "Allow users on this server to send shares to other servers (This option also enables WebDAV access to public shares)" under "Sharing -> Federated Cloud Sharing".

24.5 Configuration

24.5.1 Initial Content for new collectives

It's possible to create custom content for new collectives by putting files in the app skeleton directory at `data/app_<INSTANCE_ID>/collectives/skeleton`. New collectives start with the contents of this directory.

Create a `Readme.md` file to change the landing page that is opened automatically when entering a collective.

If the skeleton directory doesn't contain a `Readme.md`, the default landing page from `apps/collectives/skeleton/Readme.md` will be copied into the collectives directory instead.

24.6 Allow for groups in your collectives

You can configure the Teams app to allow adding groups to teams. Since the Collectives app relies on the Teams app for user management, this also allows adding entire groups to collectives.

Keep in mind thought that in contrast to teams, groups can only be managed by server admins.

24.7 Importing existing data

Import existing Markdown files is possible with the occ command `occ collectives:import:markdown`.

The command imports Markdown files from a directory as new pages into a collective. After importing all files, it processes relative links and referenced local attachments in the Markdown files. It tries to fix links to other pages and uploads referenced attachments when the source file is found in the import directory.

Please beware that the command is memory intensive. When importing a directory with many Markdown files, make sure to increase the PHP memory limit accordingly:

```
php -d memory_limit=<X>G ./occ collectives:import:markdown -c <collectiveId> -u
↪<userId> /path/to/markdown/files
```

24.7.1 Importing from Dokuwiki

The Markdown directory import command (see above) supports to import Markdown files generated from a Dokuwiki instance and tries to fix relative links to other pages and upload referenced attachments.

Importing is tested with Markdown files generated with the Dokuwiki2Markdown tool.

Here's an example how to import from a Dokuwiki instance:

```
/path/to/doku2md.py -d /path/to/dokuwiki/data/pages -T
php -d memory_limit=2G ./occ collectives:import:markdown -c 123 -u alice /path/to/
↪dokuwiki/data/pages
```

CHAPTER

TWENTYFIVE

MONITORING

25.1 OpenMetrics

Added in version 33.

Nextcloud exposes a `/metrics` endpoint. By default, it responds only on localhost. You can change this behaviour with `openmetrics_allowed_clients`

```
'openmetrics_allowed_clients' => [
  '192.168.0.0/16',
],
```

Warning

Ensure this endpoint is not accessible to everyone as it could lead to some load on your server.

You can view the content of this endpoint with the following command:

```
curl "https://your.domain/metrics"
```

If for some reason you want to disable some metrics (eg. if they take too long to generate), you can disable them by adding their class name into `openmetrics_skipped_classes`

```
'openmetrics_skipped_classes' => [
  'OC\OpenMetrics\Exporters\FilesByType',
],
```

See also

Check *Configuration Parameters* for more information

CHAPTER

TWENTYSIX

MAINTENANCE

26.1 Backup

To backup a Nextcloud installation there are five main things you need to retain:

1. The config folder
2. The custom apps folder (only if you have custom apps installed)
3. The data folder
4. The theme folder
5. The database

26.1.1 Maintenance mode

`maintenance:mode` locks the sessions of logged-in users and prevents new logins in order to prevent inconsistencies of your data. You must run `occ` as the HTTP user, like this example on Ubuntu Linux:

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

You may also put your server into this mode by editing `config/config.php`. Change `"maintenance" => false` to `"maintenance" => true`:

```
<?php

 "maintenance" => true,
```

Don't forget to change it back to `false` when you are finished.

26.1.2 Backup folders

Copy your config, data, and theme directories to a location outside your Nextcloud environment. Alternatively, copy your entire Nextcloud installation directory (including the data directory). You could use this command:

```
rsync -Aavx nextcloud/ nextcloud-dirbkp_`date +"%Y%m%d"`/
```

26.1.3 Backup database

Warning

Before restoring a backup see *Restoring backup*

MariaDB

MariaDB is the recommended database engine. To backup MariaDB using mariadb-dump:

```
mariadb-dump --single-transaction -h [server] -u [username] -p[password] [db_name] >␣
↪nextcloud-sqlbkp_`date +"%Y%m%d"`.bak
```

If you have enabled MariaDB 4-byte support (*Enabling MySQL/MariaDB 4-byte support*, needed for emoji), add `--default-character-set=utf8mb4`:

```
mariadb-dump --single-transaction --default-character-set=utf8mb4 -h [server] -u␣
↪[username] -p[password] [db_name] > nextcloud-sqlbkp_`date +"%Y%m%d"`.bak
```

MySQL

To backup MySQL:

```
mysqldump --single-transaction -h [server] -u [username] -p[password] [db_name] >␣
↪nextcloud-sqlbkp_`date +"%Y%m%d"`.bak
```

If you have enabled MySQL 4-byte support (*Enabling MySQL/MariaDB 4-byte support*, needed for emoji), add `--default-character-set=utf8mb4`:

```
mysqldump --single-transaction --default-character-set=utf8mb4 -h [server] -u␣
↪[username] -p[password] [db_name] > nextcloud-sqlbkp_`date +"%Y%m%d"`.bak
```

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
```

26.2 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 the database, data directory, and configuration files. You cannot complete restoration without all three.

26.2.1 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/
```

26.2.2 Restore database

> **Warning**
>
> Before restoring a backup you need to make sure to delete all existing database tables.

The easiest way to do this is to drop and recreate the database. SQLite does this automatically.

MariaDB

MariaDB is the recommended database engine. To restore MariaDB using the mariadb client:

```
mariadb -h [server] -u [username] -p[password] -e "DROP DATABASE nextcloud"
mariadb -h [server] -u [username] -p[password] -e "CREATE DATABASE nextcloud"
```

If you use UTF8 with multibyte support (e.g. for emojis in filenames), use:

```
mariadb -h [server] -u [username] -p[password] -e "DROP DATABASE nextcloud"
mariadb -h [server] -u [username] -p[password] -e "CREATE DATABASE nextcloud␣
↪CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci"
```

MySQL

To restore MySQL:

```
mysql -h [server] -u [username] -p[password] -e "DROP DATABASE nextcloud"
mysql -h [server] -u [username] -p[password] -e "CREATE DATABASE nextcloud"
```

If you use UTF8 with multibyte support (e.g. for emojis in filenames), use:

```
mysql -h [server] -u [username] -p[password] -e "DROP DATABASE nextcloud"
mysql -h [server] -u [username] -p[password] -e "CREATE DATABASE nextcloud CHARACTER␣
↪SET utf8mb4 COLLATE utf8mb4_general_ci"
```

PostgreSQL

```
PGPASSWORD="password" psql -h [server] -U [username] -d template1 -c "DROP DATABASE \
↪"nextcloud\";"
PGPASSWORD="password" psql -h [server] -U [username] -d template1 -c "CREATE DATABASE␣
↪\"nextcloud\";"
```

26.2.3 Restoring

> **Note**
>
> This guide assumes that your previous backup is called "nextcloud-sqlbkp.bak"

MariaDB

MariaDB is the recommended database engine. To restore MariaDB using the mariadb client:

```
mariadb -h [server] -u [username] -p[password] [db_name] < nextcloud-sqlbkp.bak
```

MySQL

To restore MySQL:

```
mysql -h [server] -u [username] -p[password] [db_name] < nextcloud-sqlbkp.bak
```

SQLite

```
rm data/owncloud.db
sqlite3 data/owncloud.db < nextcloud-sqlbkp.bak
```

PostgreSQL

```
PGPASSWORD="password" psql -h [server] -U [username] -d nextcloud -f nextcloud-sqlbkp.
↪bak
```

26.2.4 Synchronising with clients after data recovery

By default the Nextcloud server is considered the authoritative source for the data. If the data on the server and the client differs clients will default to fetching the data from the server.

If the recovered backup is outdated the state of the clients may be more up to date than the state of the server. In this case also make sure to run the *maintenance:data-fingerprint* command afterwards. It changes the logic of the synchronisation algorithm to try and recover as much data as possible. Files missing on the server are therefore recovered from the clients and in case of different content the users will be asked.

This can also help in rare scenarios when the database is newer than the data directory. The server will restore the data from the clients and preserve the shares. Until then the files would be visible but not accessible. A *files:scan* is required afterwards to update the database.

> **Note**
>
> The usage of *maintenance:data-fingerprint* can cause conflict dialogues and difficulties deleting files on the client. Therefore it's only recommended to prevent dataloss if the backup was outdated. This command does not require the server to be in maintenance mode.

If you are running multiple application servers you will need to make sure the config files are synced between them so that the updated *data-fingerprint* is applied on all instances.

26.3 How to upgrade

26.3.1 Overview

The approach used to upgrade your Nextcloud Server depends on your installation type. This manual mainly focuses on the methods that apply to an Archive based installation. If you installed using Snap, Docker, a pre-built VM, or a package management tool then refer to the installation and update instructions for that installation method for the most accurate upgrading instructions (generally located at the distribution point for the install method you chose).

There are two ways to upgrade an Archive based Nextcloud Server deployment:

- With the *Built-in Updater* (via the web or command-line interfaces).
- *Manually upgrading* (using a downloaded Archive file)

The Built-in Updater, in either Web or command-line mode, is the easiest choice for most environments. However some environments require the manual approach. Both approaches are covered fully here.

Important

Before upgrading, especially between major versions (e.g. v27.y.z -> v28.y.z) please review *critical changes* first. These are highlights of changes that may be required in your environment to accommodate changes in Nextcloud Server. These notes are periodically revised as needed so it is also a good idea to revisit them periodically, such as when proceeding with maintenance upgrades.

When an update is available for your Nextcloud server, by default you will receive a notification. You can also check for available updates by visiting the Update section under **Administration settings->Overview** in the Web UI.

Note

It is best to keep your Nextcloud server upgraded regularly. This means installing all maintenance/point releases and upgrading to new major releases before your current one reaches *end-of-life* status. Examples of major releases are 27, 28, or 29. Maintenance releases are intermediate releases for each major release that address critical functionality or security bugs. For example 28.0.4 and 29.0.2 are maintenance releases.

26.3.2 Approaching Upgrades

Nextcloud must be upgraded step by step:

- Before you can upgrade to the next major release, you need to upgrade to the latest point release of your current major version.
- Then run the upgrade again to upgrade to the next major release's latest point release.
- **You cannot skip major releases.** Please re-run the upgrade until you have reached the highest available (or applicable) release.
- Example: 18.0.5 -> 18.0.11 -> 19.0.5 -> 20.0.2

Wait for background migrations to finish after major upgrades. After upgrading to a new major version, some migrations are scheduled to run as a background job. If you plan to upgrade directly to another major version (e.g. 24 -> 25 -> 26) you need to make sure these migrations were executed before starting the next upgrade. To do so you should run the `cron.php` file 2-3 times, for example:

```
$ sudo -E -u www-data php -f /var/www/nextcloud/cron.php
```

For more information about background jobs see *Background jobs*.

Upgrading is disruptive. Your Nextcloud server will be put into maintenance mode, so your users will be locked out until the upgrade is completed. Large installations may take several hours to complete the upgrade. Nevertheless usual upgrade times even for bigger installations are in the range of a few minutes.

Warning

Downgrading is not supported and risks corrupting your data! If you want to revert to an older Nextcloud version, make a new, fresh installation and then restore your data from backup. Before doing this, file a support ticket (if you have paid support) or ask for help in the Nextcloud forums to see if your issue can be resolved without downgrading.

26.3.3 Update notifications

Nextcloud has an update notification app, that informs the administrator about the availability of an update. Then you decide which update method to use.

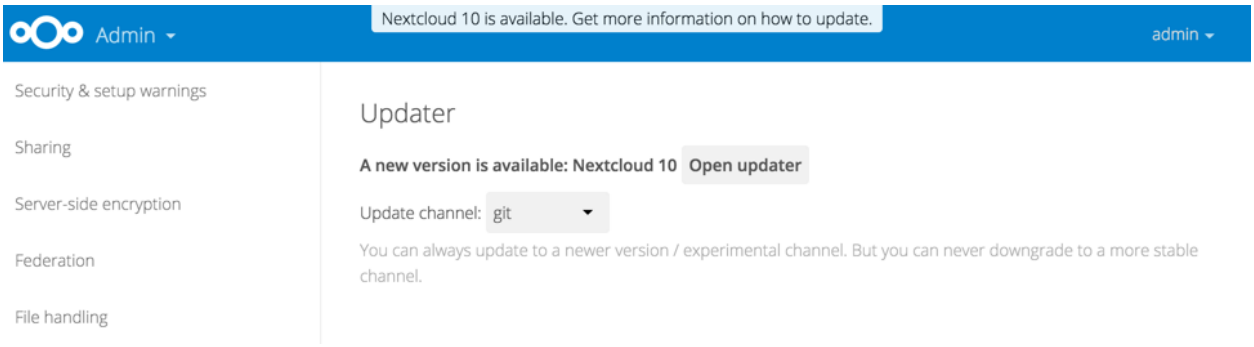


Fig. 1: *Figure 1: The top banner is the update notification that is shown on every page, and the Updates section can be found in the admin page*

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.

26.3.4 Prerequisites

See also

If you upgrade from a previous major version please see *critical changes* first.

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.

26.3.5 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 *built-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 -E -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.

26.3.6 Manual steps during upgrade

Some operation can be quite time consuming. Therefore we decided not to add them to the normal upgrade process. We recommend to run them manually after the upgrade was completed. Below you find a list of this commands.

Long running migration steps

From time to time we do some changes to the database layout that take a lot of time, but can be executed while Nextcloud stays online. Thus we moved them into a separate command that an administrator can execute on the CLI without the need to lock the instance into maintenance mode (at least for some of them). The instance will also work without those changes applied, but performance is improved significantly by them. There is also always an hint in the setup checks of the admin settings web interface.

Those include for example:

```
$ sudo -E -u www-data php occ db:add-missing-columns
$ sudo -E -u www-data php occ db:add-missing-indices
$ sudo -E -u www-data php occ db:add-missing-primary-keys
```

You can use the `--dry-run` option to output the SQL queries instead of executing them.

26.4 Upgrade via built-in updater

The built-in updater automates many of the steps of upgrading a Nextcloud installation. It is useful for installations that do not have root access, such as shared hosting, for installations with a smaller number of users and data, and it automates updating *manual installations*.

Warning

Downgrading is not supported and risks corrupting your data! If you want to revert to an older Nextcloud version, install it from scratch and then restore your data from backup. Before doing this, file a support ticket if you have paid support or ask for help in the Nextcloud forums to see if your issue can be resolved without downgrading.

Danger

You should maintain regular backups (see *Backup*), and make a backup before every update. The built-in updater does not backup your database or data directory.

26.4.1 What does the updater do?

Note

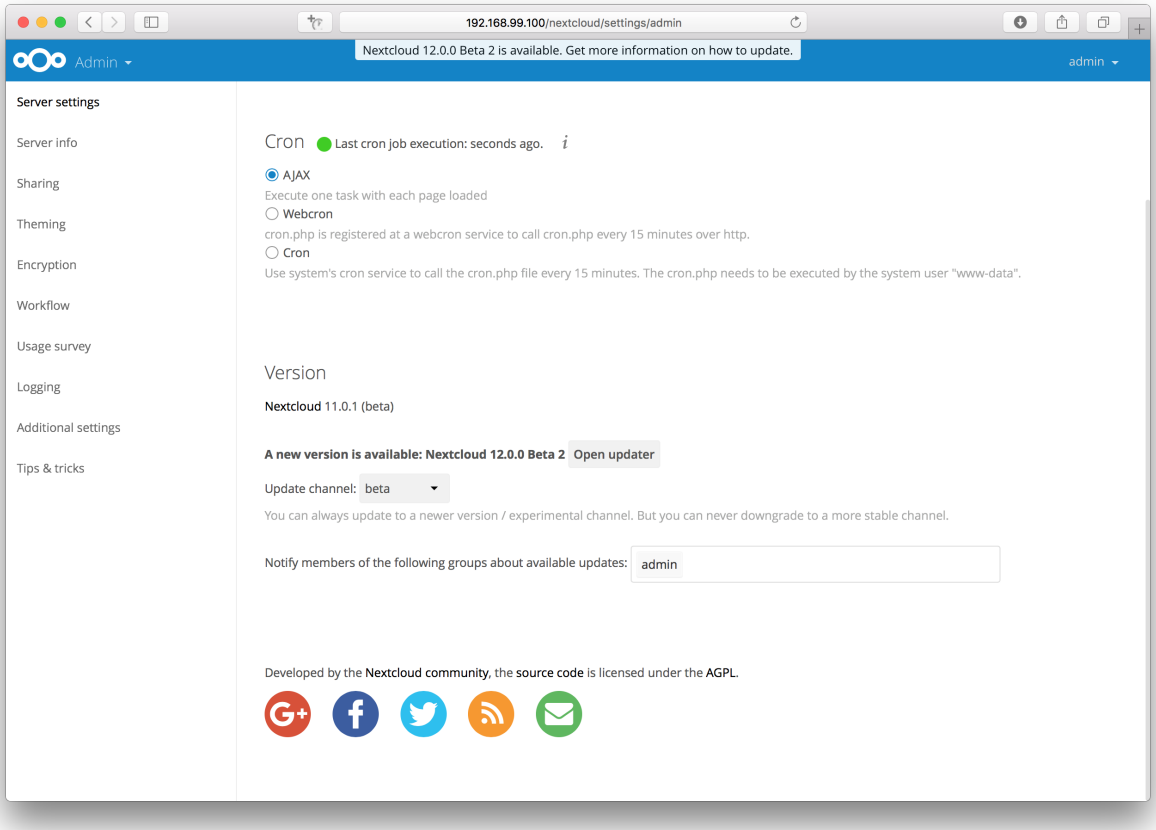
The built-in updater itself only replaces the existing files with the ones from the version it updates to. The migration phase, which upgrades your database and apps, needs to be executed afterwards. In command line mode, the updater offers to trigger this for you right after the code was successfully replaced by running `occ upgrade` for you. In web mode, the updater finishes and then offers to send you back to your instance's main URL to trigger the migration phase's web UI.

The built-in updater performs these operations:

- **Check for expected files:** checks if only the expected files of a Nextcloud installation are present, because it turned out that some files that were left in the Nextcloud directory caused side effects that risked the update procedure.
- **Check for write permissions:** checks if all files that need to be writable during the update procedure are actually writable.
- **Enable maintenance mode:** enables the maintenance mode so that no other actions are executed while running the update of the code.
- **Create backup:** creates a backup of the existing code base in `/updater-INSTANCEID/backups/nextcloud-CURRENTVERSION/` inside of the data directory (this does not contain the `/data` directory nor the database).
- **Downloading:** downloads the code in the version it should update to. This is also shown in the web UI before the update is started. This archive is downloaded to `/updater-INSTANCEID/downloads/`.
- **Extracting:** extracts the archive to the same folder.
- **Replace entry points:** replaces all Nextcloud entry points with dummy files so that when those files are replaced all clients still get the proper maintenance mode response. Examples for those endpoints are `index.php`, `remote.php` or `ocs/v1.php`.
- **Delete old files:** deletes all files except the above mentioned entry points, the data and config dir as well as non-shipped apps and themes. (And the updater itself of course)
- **Move new files in place:** moves the files from the extracted archive in place.
- **Keep maintenance mode active?:** asks you if the maintenance mode should be kept active. This allows the admin to use the web based updater but run the actual migration steps (`occ upgrade`) on the command line. If the maintenance mode is kept active command line access is required. To use the web based upgrade page disable the maintenance mode and click the link to get to the upgrade page. (This step is only available in the web based updater.)
- **Done** the update of the code is done and you either need to go to the linked page or to the command line to finish the upgrade by executing the migration steps.

26.4.2 Using the command line 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 notification 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 -E -u www-data php /var/www/nextcloud/updater/updater.phar`)

```
Nextcloud Updater - version: 1.0.3

Current version is 11.0.1.

Update to Nextcloud 12.0.0 Beta 2 available. (channel: "beta")
Following file will be downloaded automatically: https://download.nextcloud.com/server/prereleases/nextcloud-12.0.0beta2.zip

Steps that will be executed:
[ ] Check for expected files
[ ] Check for write permissions
[ ] Enable maintenance mode
[ ] Create backup
[ ] Downloading
[ ] Extracting
[ ] Replace entry points
[ ] Delete old files
[ ] Move new files in place
[ ] Done

Start update? [y/N]
```

3. Verify the information that is shown and enter “Y” to start the update.

```
Nextcloud Updater - version: 1.0.3

Current version is 11.0.1.

Update to Nextcloud 12.0.0 Beta 2 available. (channel: "beta")
Following file will be downloaded automatically: https://download.nextcloud.com/server/prereleases/nextcloud-12.0.0beta2.zip

Steps that will be executed:
[ ] Check for expected files
[ ] Check for write permissions
[ ] Enable maintenance mode
[ ] Create backup
[ ] Downloading
[ ] Extracting
[ ] Replace entry points
[ ] Delete old files
[ ] Move new files in place
[ ] Done

Start update? [y/N] y

Info: Pressing Ctrl-C will finish the currently running step and then stops the updater.

[✔] Check for expected files
[ ] Check for write permissions ...
```

```
Nextcloud Updater - version: 1.0.3

Current version is 11.0.1.

Update to Nextcloud 12.0.0 Beta 2 available. (channel: "beta")
Following file will be downloaded automatically: https://download.nextcloud.com/server/prereleases/nextcloud-12.0.0beta2.zip

Steps that will be executed:
[ ] Check for expected files
[ ] Check for write permissions
[ ] Enable maintenance mode
[ ] Create backup
[ ] Downloading
[ ] Extracting
[ ] Replace entry points
[ ] Delete old files
[ ] Move new files in place
[ ] Done

Start update? [y/N] y

Info: Pressing Ctrl-C will finish the currently running step and then stops the updater.

[✘] Check for expected files failed
The following extra files have been found:
    unexpectedFile.php

Update failed. To resume or retry just execute the updater again.
```

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

```
Nextcloud Updater - version: 1.0.3

Current version is 11.0.1.

Update to Nextcloud 12.0.0 Beta 2 available. (channel: "beta")
Following file will be downloaded automatically: https://download.nextcloud.com/server/prereleases/nextcloud-12.0.0beta2.zip

Steps that will be executed:
[✔] Check for expected files
[ ] Check for write permissions
[ ] Enable maintenance mode
[ ] Create backup
[ ] Downloading
[ ] Extracting
[ ] Replace entry points
[ ] Delete old files
[ ] Move new files in place
[ ] Done

Continue update? [y/N]
```

6. Once all steps are executed the updater will ask you a final question: “Should the “occ upgrade” command be executed?”. This allows you to directly execute the command line based upgrade procedure (`occ upgrade`). If

you select "No" then it will finish with *Please now execute "./occ upgrade" to finish the upgrade.*.

```
Info: Pressing Ctrl-C will finish the currently running step and then stops the updater.

[✔] Check for expected files
[✔] Check for write permissions
[✔] Enable maintenance mode
[✔] Create backup
[✔] Downloading
[✔] Extracting
[✔] Replace entry points
[✔] Delete old files
[✔] Move new files in place
[✔] Done

Update of code successful.

Should the "occ upgrade" command be executed? [Y/n]
```

7. Once the `occ upgrade` is done you get asked if the maintenance mode should be kept active.

```
Update of code successful.

Should the "occ upgrade" command be executed? [Y/n] y
Nextcloud or one of the apps require upgrade - only a limited number of commands are available
You may use your browser or the occ upgrade command to do the upgrade
Set log level to debug
Updating database schema
Updated database
Updating <federatedfilesharing> ...
Updated <federatedfilesharing> to 1.2.0
Updating <files_pdfviewer> ...
Updated <files_pdfviewer> to 1.1.1
Updated <systemtags> to 1.2.0
Updating <theming> ...
Updated <theming> to 1.3.0
Starting code integrity check...
Finished code integrity check
Update successful
Maintenance mode is kept active
Reset log level

Keep maintenance mode active? [y/N]
```

26.4.3 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 occurred during the `occ upgrade` or the replacement of the code.

To execute this, run the command with the `--no-interaction` option. (i.e. `sudo -E -u www-data php /var/www/nextcloud/updater/updater.phar --no-interaction`)

```
Nextcloud Updater - version: 1.0.3

Current version is 11.0.3.

Update to Nextcloud 12.0.0 Beta 2 available. (channel: "beta")
Following file will be downloaded automatically: https://download.nextcloud.com/server/prereleases/nextcloud-12.0.0beta2.zip

Updater run in non-interactive mode.

Start update

Info: Pressing Ctrl-C will finish the currently running step and then stops the updater.

[✔] Check for expected files
[✔] Check for write permissions
[✔] Enable maintenance mode
[✔] Create backup
[✔] Downloading
[✔] Extracting
[✔] Replace entry points
[✔] Delete old files
[✔] Move new files in place
[✔] Done

Update of code successful.
Updater run in non-interactive mode - will start "occ upgrade" now.

Nextcloud or one of the apps require upgrade - only a limited number of commands are available
You may use your browser or the occ upgrade command to do the upgrade
Set log level to debug
Updating database schema
Updated database
Updating <federatedfilesharing> ...
Updated <federatedfilesharing> to 1.2.0
Updating <files_pdfviewer> ...
Updated <files_pdfviewer> to 1.1.1
Updating <theming> ...
Updated <theming> to 1.3.0
Starting code integrity check...
Finished code integrity check
Update successful
Maintenance mode is kept active
Reset log level

Updater run in non-interactive mode - will disable maintenance mode now.

Maintenance mode is disabled
```

26.4.4 Using a custom download URL

The `--url` option lets you override the archive the updater downloads. Common use cases:

- **Pinning a specific version** — install an exact patch release rather than whatever the update check resolves to.
- **Internal mirrors** — serve the archive from a company mirror or proxy.
- **Offline / air-gapped environments** — stage the archive locally and supply it via a `file://` URL.

> **Warning**
>
> The normal update rules still apply when using `--url`:
>
> - **Downgrading is not supported**.
> - **Skipping major versions is not supported.** You must update one major version at a time (e.g. 28 → 29 → 30, not 28 → 30).
>
> Passing `--url` does not bypass these constraints.

Point the updater at any HTTP/HTTPS URL:

```
sudo -u www-data php /var/www/nextcloud/updater/updater.phar \
    --url https://download.nextcloud.com/server/releases/nextcloud-33.0.0.zip
```

Added in version 34.

For a locally staged archive, use a `file://` URL:

```
sudo -u www-data php /var/www/nextcloud/updater/updater.phar \
    --url file:///tmp/nextcloud-33.0.0.zip
```

Signature verification

When `--url` is used the updater cannot look up the official signature automatically. You have two options:

- **Provide the signature** — pass the base64-encoded signature with `--signature`. You can get the signature from https://nextcloud.com/changelog/ or from the same mirror that hosts the archive:

  ```
  sudo -u www-data php /var/www/nextcloud/updater/updater.phar \
      --url file:///tmp/nextcloud-33.0.0.zip \
      --signature "BASE64_SIGNATURE_HERE"
  ```

- **Skip verification** — pass `--no-verify` to disable integrity checking entirely. Only do this if you fully trust the source and transfer channel, or if you have already verified the archive yourself (e.g. by checking the SHA-512 checksum):

  ```
  sha512sum -c nextcloud-33.0.0.zip.sha512
  ```

  Then run the updater without signature checking:

  ```
  sudo -u www-data php /var/www/nextcloud/updater/updater.phar \
      --url file:///tmp/nextcloud-33.0.0.zip \
      --no-verify
  ```

  Warning

  `--no-verify` removes the integrity check that protects against corrupted or tampered archives. Always verify the archive through an independent channel before using this option.

These options can be combined with `--no-interaction` for fully automated runs:

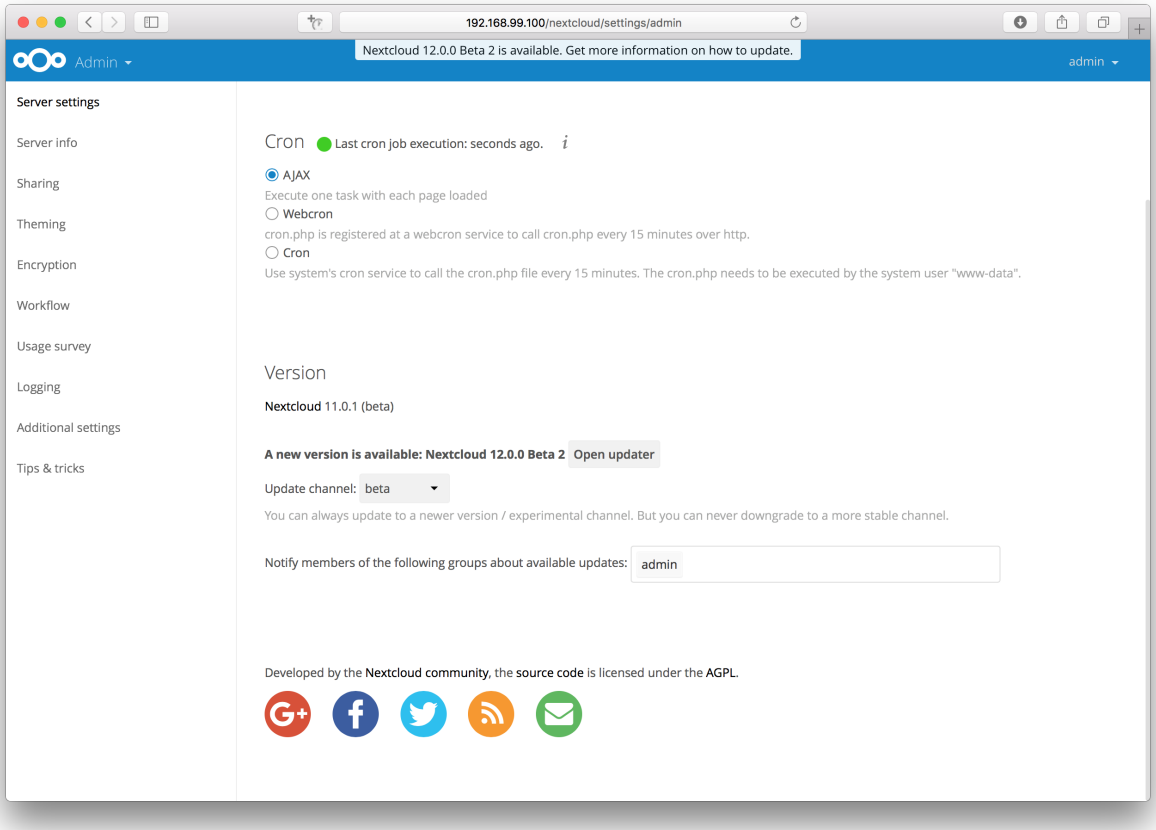
```
sudo -u www-data php /var/www/nextcloud/updater/updater.phar \
    --url file:///tmp/nextcloud-33.0.0.zip \
    --signature "BASE64_SIGNATURE_HERE" \
    --no-interaction
```

26.4.5 Using the web based updater

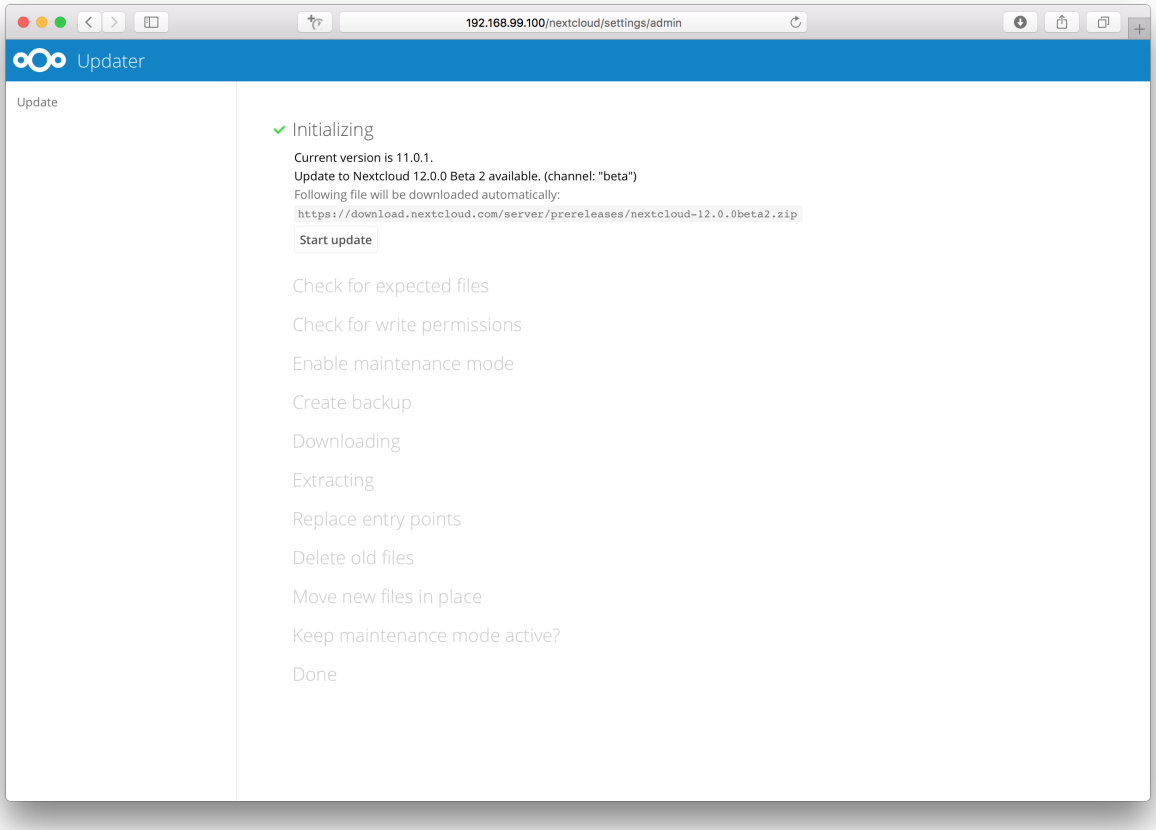
The web based updater performs the same steps and checks as the command line based updater.

Using the built-in updater to update your Nextcloud installation is just a few steps:

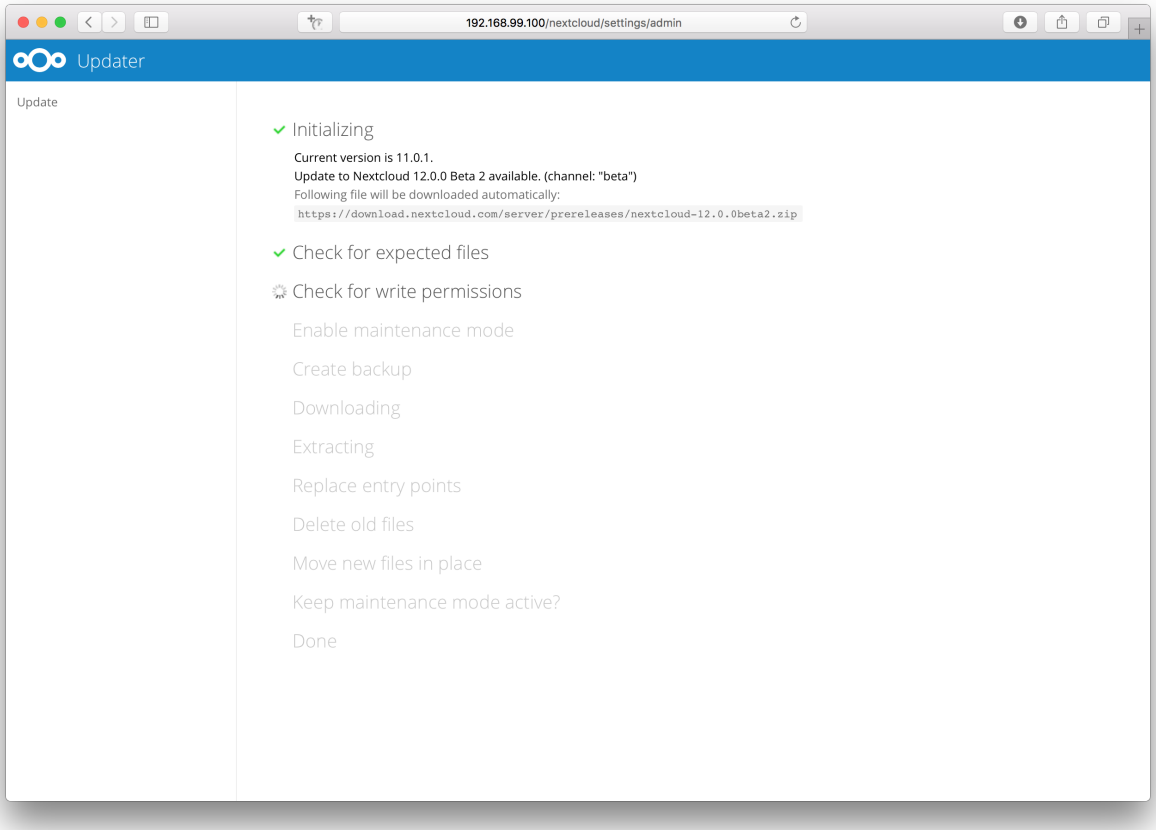
1. You should see a notification at the top of any Nextcloud page when there is a new update available. Go to the admin settings page and scroll to the section “Version”. This section has a button to open the updater. This section as well as the update notification is only available if the update notification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



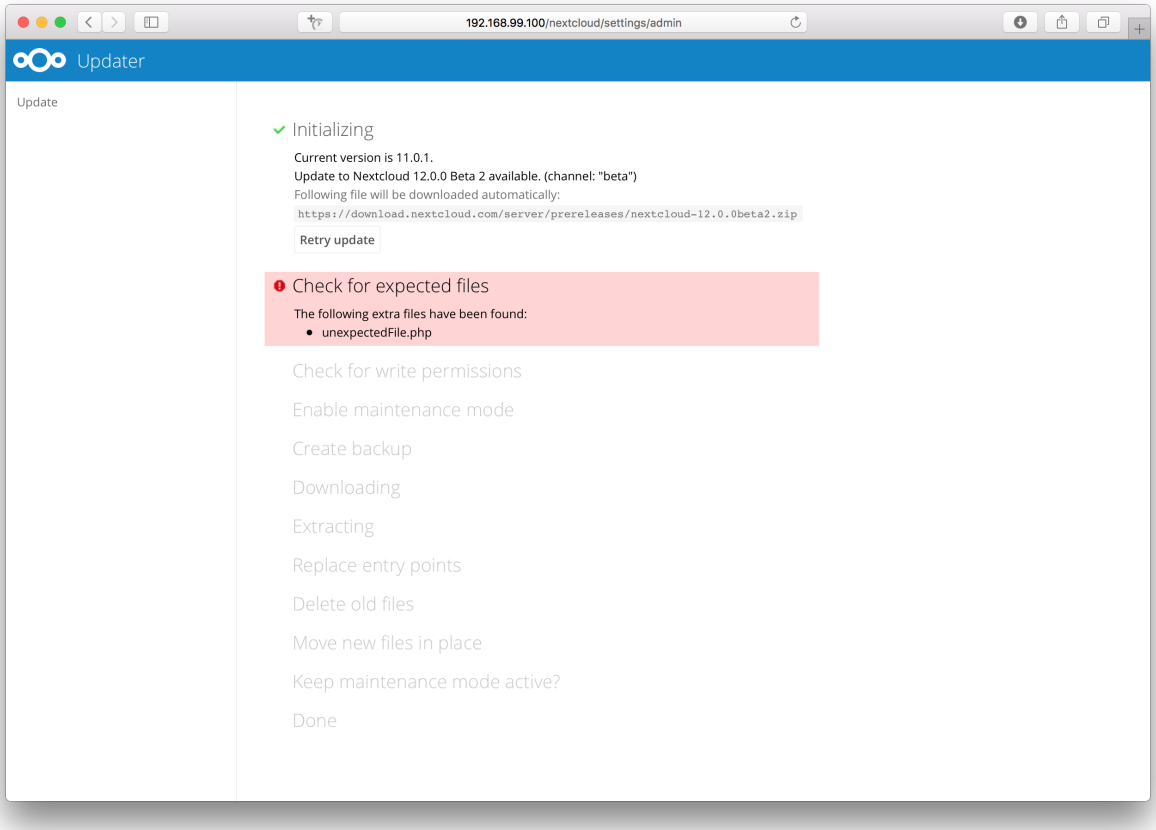
2. Click the button “Open updater”.



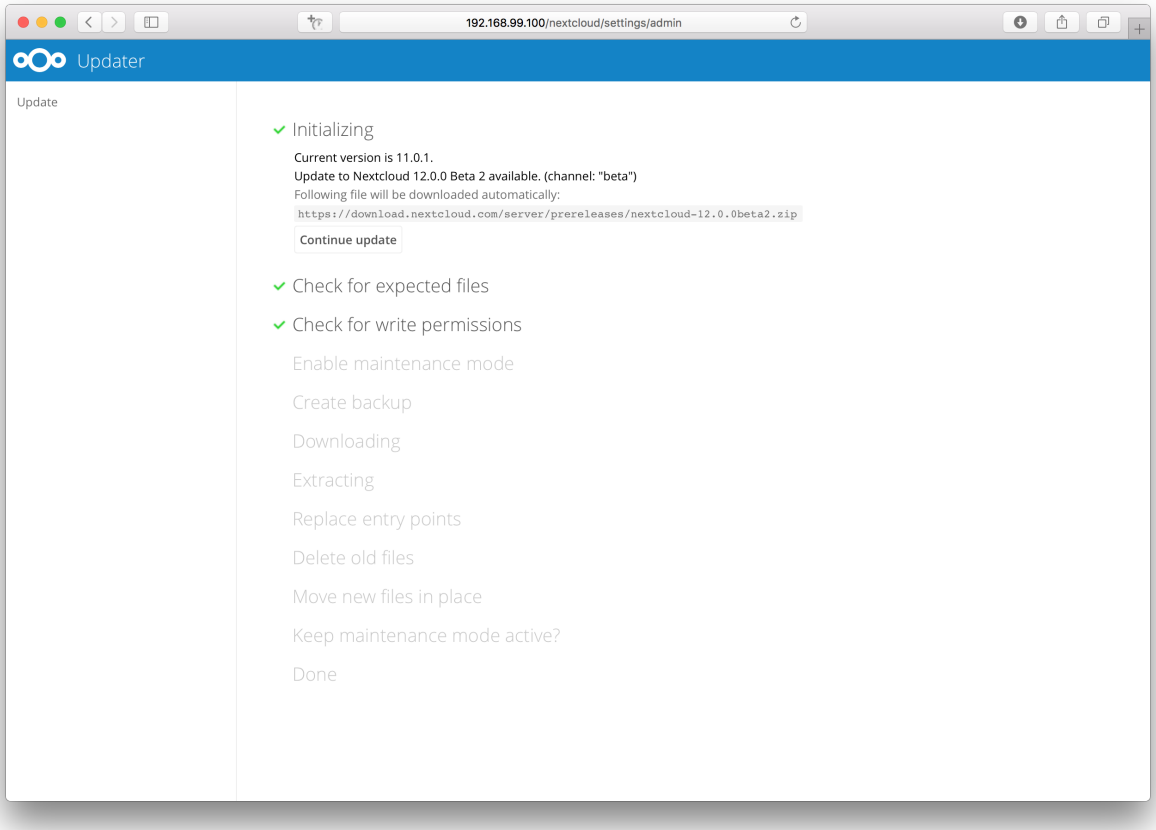
3. Verify the information that is shown and click the button “Start update” to start the update.



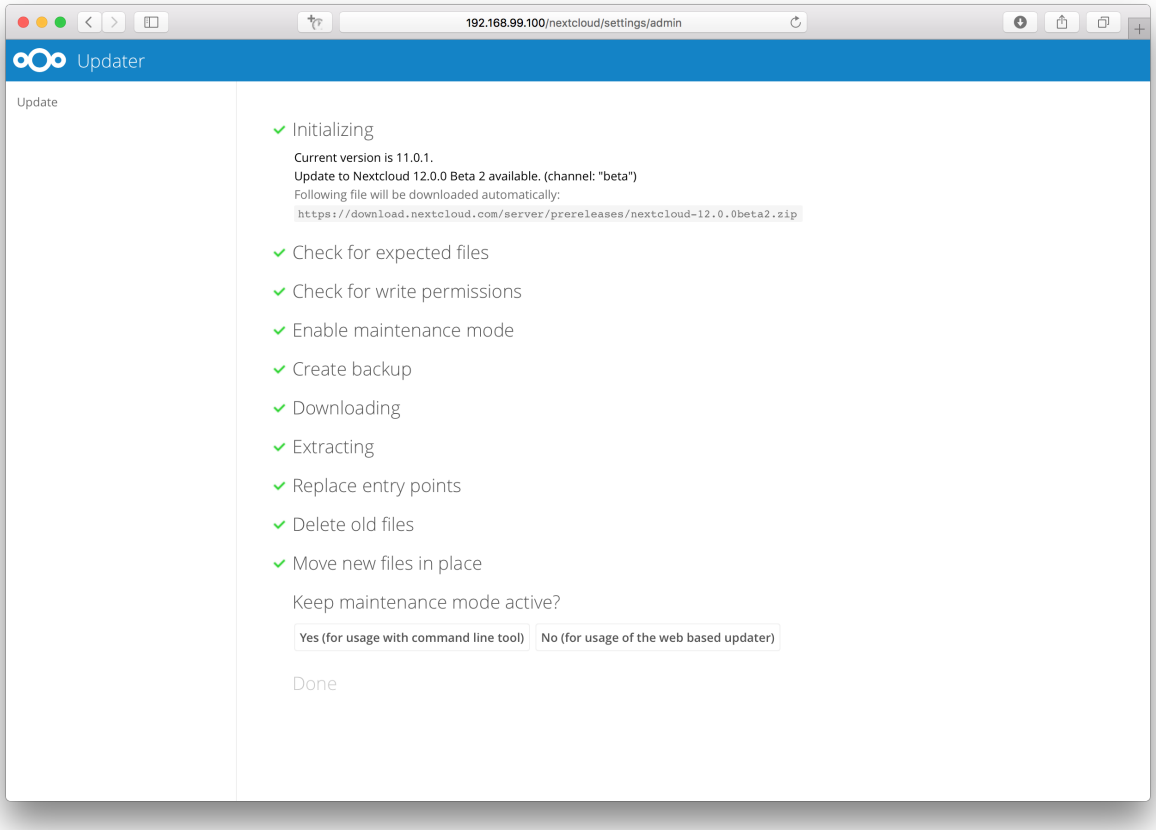
4. In case an error happens or the check failed the updater stops processing and gives feedback. You can now try to solve the problem and click the “Retry update” button. This will continue the update and re-run the failed step. It will not re-run the previous succeeded steps.



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.



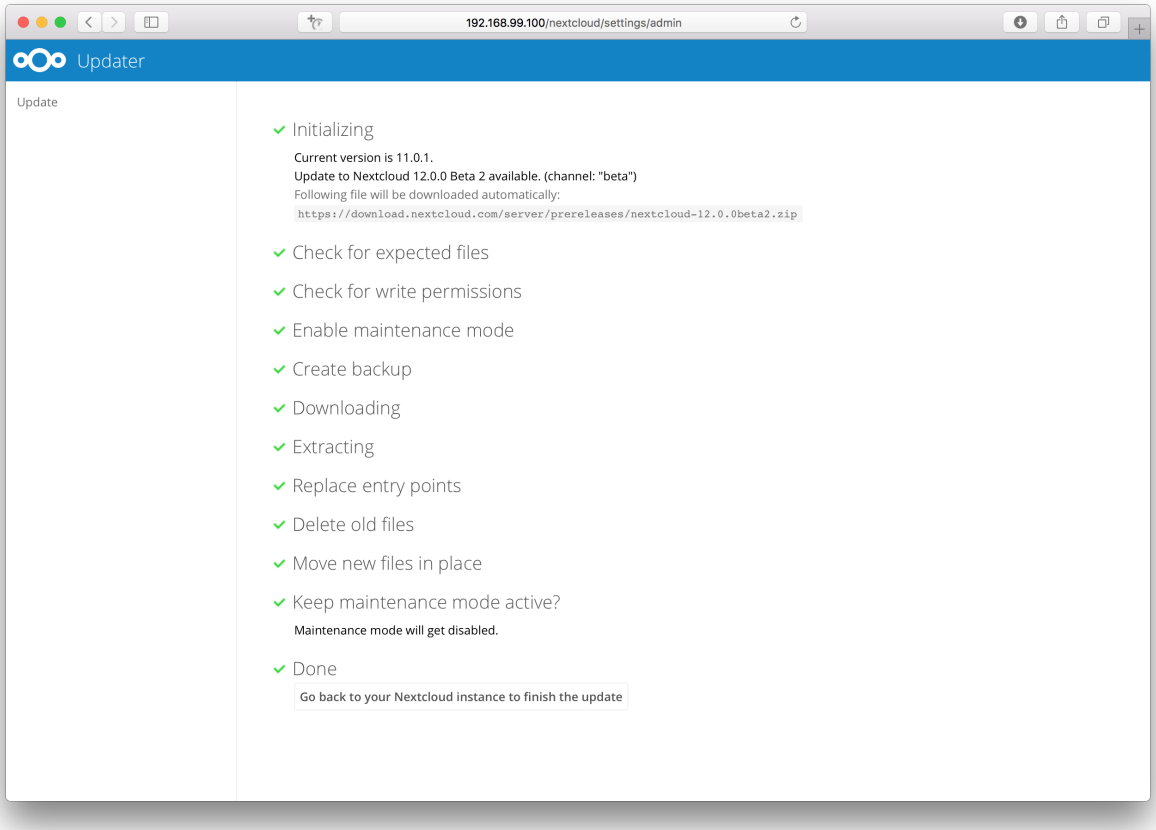
6. Once all steps are executed the updater will ask you a final question: “Keep maintenance mode active?”. This allows you to use either the web based upgrade page or the command line based upgrade procedure (`occ upgrade`). Command line access is required if the maintenance mode is kept active.

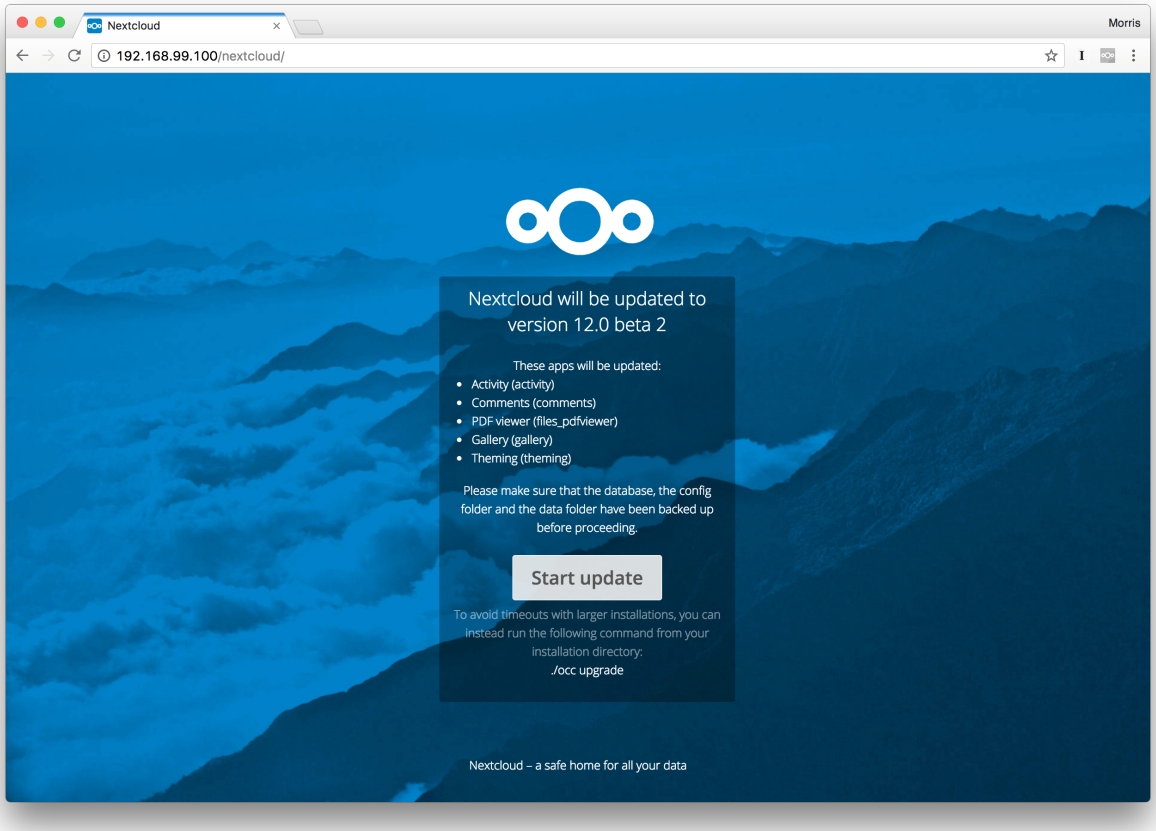


7. Done. You now can continue either to the web based upgrade page or run `occ upgrade`. The two examples "Web based upgrade" and "Command line based upgrade" shows how the screens then look like.

Web based upgrade

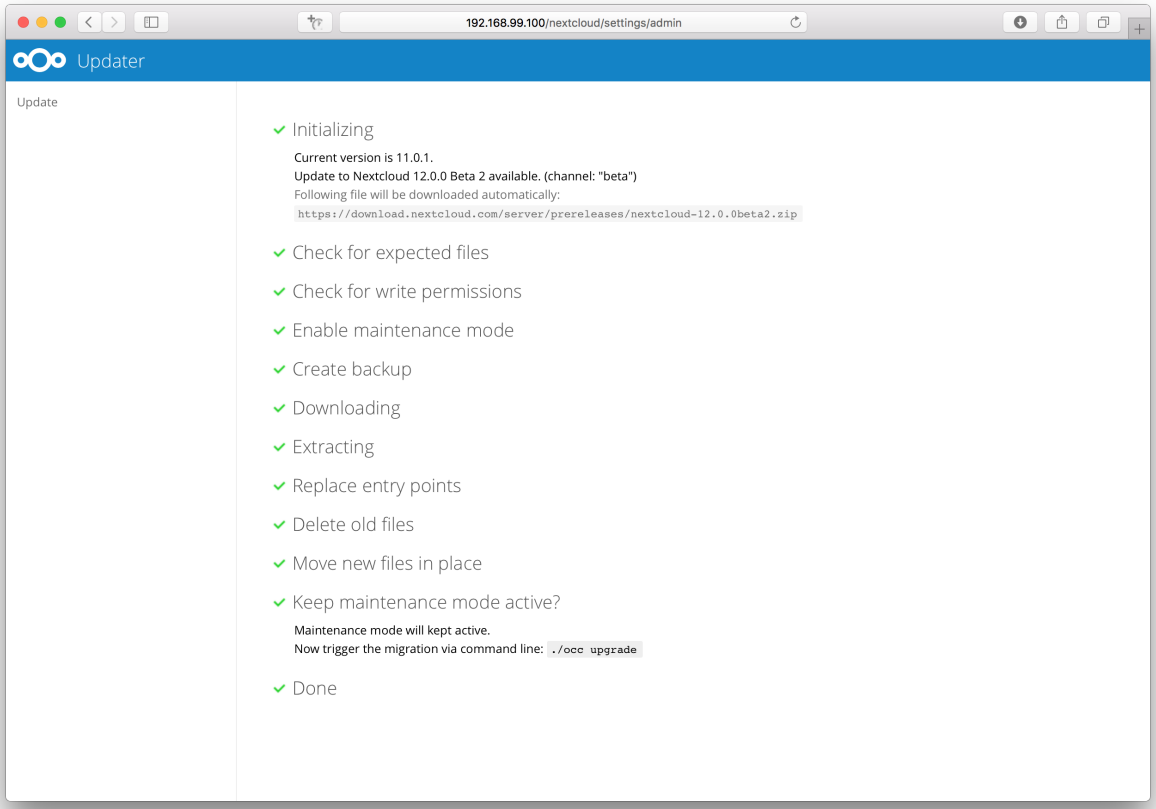
This is how the web based update would continue:


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
```

26.4.6 Troubleshooting

- The built-in updater logs all of its actions to a dedicated log file called `updater.log` located in your configured `datadirectory` (e.g. `/var/www/html/data/updater.log`). This file can be helpful in isolating where things are failing. It will also be needed if you reach out for assistance on the community help forum (https://help.nextcloud.com).
- If you are having problems using the Updater in web-mode, you should try using command-line mode (if it's an option in your environment). Command-line avoids issues with web server timeouts, which can be problematic since sometimes the Updater can take a long time to complete certain steps.
- If the problem seems to be during the backup step, you can try disabling the backups the updater automatically creates of the installation files. Keep in mind these backups do **not** include your data (which you are already hopefully doing). The backup step can only be disabled while in command-line mode. Append the option `--no-backup` to the `updater.phar` command.
- If you accidentally say no when the command-line mode of the updater asks if you'd like to run `occ upgrade`, you can safely execute `occ upgrade` manually or simply visit the URL of your instance to complete the database migrations and app upgrade phase.
- Reach out to the community help forum for assistance (https://help.nextcloud.com)

26.5 Upgrade manually

26.5.1 Overview

In some environments using the Built-in Updater in Web mode is not reliable (such as due to web server timeouts) and running it in command-line mode is not an option (such as in some shared hosting environments). In these cases a manual upgrade may be the best approach.

A manual upgrade consists of downloading and unpacking the Nextcloud Archive file either to your PC or host. Then deleting your existing Nextcloud Server installation files and folders, **except** `data/` and `config/`, on your host. Then moving the new Nextcloud Server installation files into the appropriate place on your host, again preserving your existing `data/` and `config/` files. And doing a few other housekeeping items, such as making sure your installed apps are transferred into the new installation and adjusting permissions. That may sound like a lot, but detailed instructions are below.

Important

Before upgrading, especially between major versions (e.g. v27.y.z -> v28.y.z) please review *critical changes* first. These are highlights of changes that may be required in your environment to accommodate changes in Nextcloud Server. These notes are periodically revised as needed so it is a good idea to revisit them even when proceeding with minor and maintenance upgrades just in case.

Warning

When upgrading manually, you must confirm your system meets the *System requirements* of the new version as well as that you are following the standard *upgrade requirements* (such as upgrading to the latest maintenance release *before* upgrading to a new major release).

26.5.2 Step-by-Step Manual Upgrade

Important

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. Choose a target Nextcloud Server release from https://nextcloud.com/changelog/ and download the Archive file (tarball or zip archive) into an empty directory outside of your current installation.

   Warning

   You cannot jump more than one major version forward at a time (i.e. 27->28 is okay, but 27->29 is not).

3. Unpack the the downloaded tarball or zip archive - e.g.:

   ```
   unzip nextcloud-[version].zip
   (or)
   tar -xjf nextcloud-[version].tar.bz2
   ```

4. Stop your Web server.
5. In case you are running a cron-job for nextcloud's house-keeping disable it by commenting the entry in the crontab file:

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

   (Put a # at the beginning of the corresponding line.)
6. Rename your current Nextcloud directory, for example `nextcloud-old`.
7. Unpacking the new archive creates a new `nextcloud` directory populated with your new server files. Move this directory and its contents to the original location of your old server. For example `/var/www/`, so that once again you have `/var/www/nextcloud`.
8. Copy the `config/config.php` file from your old Nextcloud directory to your new Nextcloud directory.
9. If you keep your `data/` directory in your `nextcloud/` directory, move 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.
10. If you are using 3rd party application, it may not always be available in your upgraded/new Nextcloud instance. To check this, compare a list of the apps in the new `nextcloud/apps/` folder to a list of the of the apps in your backed-up/old `nextcloud/apps/` folder. If you find 3rd party apps in the old folder that needs to be in the new/upgraded instance, simply copy them over and ensure the permissions are set up as shown below.
11. If you have additional apps folders like for example `nextcloud/apps-extras` or `nextcloud/apps-external`, make sure to also transfer/keep these in the upgraded folder.
12. If you are using 3rd party theme make sure to copy it from your `themes/` directory to your new one. It is possible you will have to make some modifications to it after the upgrade.
13. 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 {} \;
```

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

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

    (!) this MUST be executed from within your nextcloud installation directory
16. 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.
17. Re-enable the nextcloud cron-job. (See step 4 above.)

    crontab -u www-data -e

    (Delete the # at the beginning of the corresponding line in the crontab file.)

Login and take a look at the bottom of your Admin page to verify the version number. Check your other settings to make sure they're correct. Go to the Apps page and review the core apps to make sure the right ones are enabled. Re-enable your third-party apps.

26.5.3 Previous Nextcloud releases

You'll find previous Nextcloud releases in the Nextcloud Server Changelog.

26.5.4 Troubleshooting

Occasionally, *files do not show up after an upgrade*. A rescan of the files can help:

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

Then start the manual process:

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

If this does not work properly, try the repair function:

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

26.6 Upgrade via snap packages

26.6.1 Upgrade quickstart

Nextcloud snap is an unofficial Nextcloud designed to be easy to install and simple to maintain. The ideal Nextcloud snap is an "install and forget" Nextcloud instance that works on most architectures and updates itself without needing

administrative skills. Combining Nextcloud with snapd makes it a perfect fit for IoT or scalable environments. Snapd is a secure and robust technology which the Nextcloud snap team has embraced.

However, the snap is opinionated.

- Nextcloud snap uses recommended Apache.
- Nextcloud snap uses recommended MySQL.
- Nextcloud snap uses recommended PHP.

26.6.2 Installation

On Ubuntu

- https://snapcraft.io/nextcloud
- Install Nextcloud `sudo snap install nextcloud`

All other distros be warned

By default the latest stable Nextcloud snap release will be installed and it will automatically update to subsequent stable releases, but there are other releases available as well and you have full control of automatic updates.

After installation, Nextcloud will start automatically. Assuming you and the device on which it was installed are on the same network, you will reach the Nextcloud installation by visiting `<hostname>.local` or the IP address of the instance in your browser. If your hostname is `localhost` or `localhost.localdomain`, like on an Ubuntu Core device, `nextcloud.local` will be used instead.

26.6.3 1st login

Upon visiting the Nextcloud installation for the first time, you will be prompted to enter an admin username and password before Nextcloud is initialised. This may take a while depending on resources and the device. After you provide that information you will be logged in and able to install apps, create users, and upload files.

26.6.4 Upgrade tips

By default the Nextcloud snap will automatically update to subsequent stable releases. You may however upgrade manually too by issuing the command:

```
sudo snap refresh nextcloud
```

If the upgrade fails you can easily revert to the last working version by issuing the command:

```
sudo snap revert nextcloud
```

Further documentation, an extensive Wiki and FAQ's can be found on the developers GitHub.

26.7 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 unforeseen happen you can go back to your existing installation and provide your users with a running Nextcloud while debugging the problem.

1. Set up the new machine with the desired OS, install and configure the Web server as well as PHP for Nextcloud (e.g. permissions or file upload size limits) and make sure the PHP version matches Nextcloud supported configuration and all relevant PHP extensions are installed. Also set up the database and make sure it is a Nextcloud supported configuration. If your original machine was installed recently just copying that base configuration is a safe best practice.

   Important

   Before beginning the migration, **check the ``config/config.php`` file on your ORIGINAL system** to identify any optional services that are configured, such as:

   - Redis or Memcached (for caching/sessions)
   - External object storage (S3, etc.)
   - LDAP
   - Mail server settings
   - Full-text search backends (Elasticsearch, etc.)

   You must also install and configure these same services on the NEW machine before copying the Nextcloud files. Failure to do so may cause errors such as "Redis server went away" or connection failures during Nextcloud startup.

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 is in maintenance mode stop the application and/or Web server that serves Nextcloud.
3. Create a dump from the database and copy it to the new machine. There import it in the database (See *Backup* and *Restoring backup*).
4. Copy all files from your Nextcloud instance, the Nextcloud program files, the data files, the log files and the configuration files, to the new machine (See *Backup* and *Restoring backup*). The data files should keep their original timestamp (can be done by using `rsync` with `-t` option) otherwise the clients will re-download all the files after the migration. Depending on the original installation method and the OS the files are located in different locations. On the new system make sure to pick the appropriate locations. If you change any paths, make sure to adapt the paths in the Nextcloud config.php file.

   Note

   This step might take several hours, depending on your installation.

Warning

Changing the location of the data directory might cause a corruption of relations in the database and is not supported.

Important

After copying `config/config.php` to the new machine, **review and update every server-specific value** before starting Nextcloud. At minimum check:

- `datadirectory` — keep this path identical to the original server wherever possible. Changing the data directory path requires a database update and is strongly discouraged; see *Troubleshooting data-directory* if you have no choice.
- `dbhost`, `dbname`, `dbuser`, `dbpassword` — update if the new server uses a different database host or credentials.
- `trusted_domains` — add or replace the new server's hostname or IP address.
- `overwrite.cli.url`, `overwritehost`, `overwriteprotocol`, `overwritewebroot` — update to reflect the new server's URL and any reverse-proxy setup.
- `memcache.local`, `memcache.distributed`, `memcache.locking` and the associated connection parameters — update if the cache/session backend address changed on the new machine. Depending on the backend in use, check `redis` or `redis.cluster` (for Redis / Redis Cluster) or `memcached_servers` (for Memcached).
- `objectstore` — if external object storage (S3, Swift, etc.) is configured, verify that the endpoint, bucket, and credentials are still valid and reachable from the new server.
- `mail_smtphost`, `mail_smtpport` — update if the SMTP relay differs on the new server.
- `logfile` — update if the log path differs on the new server.
- `tempdirectory` — if set to a custom path, make sure that path exists and is writable on the new server.
- `serverid` — if set, keep the same value. It identifies the server in multi-PHP-server setups and must not change. If you override it per server via the `NC_serverid` environment variable, configure the same override on the new server.

The `secret` and `instanceid` values are generated once at install time and tied to all encrypted data and user sessions. **Do not change or regenerate them.** They must be copied verbatim from the original `config.php`.

Leaving stale values (especially database credentials or `datadirectory`) will cause startup errors or data corruption.

5. Check the config.php file of the **ORIGINAL** system to see if it has the `data-fingerprint` set to a non-empty value. If this is the case, make sure to also run the `maintenance:data-fingerprint` command on the **NEW** system, similarly to how it is required when performing a backup restoration (See *Restoring backup* for details).
6. While still having Nextcloud in maintenance mode (confirm!) and **BEFORE** changing the `CNAME` record in the DNS start up the database, Web server / application server on the new machine and point your web browser to the migrated Nextcloud instance. Confirm that you see the maintenance mode notice, that a logfile entry is written by both the Web server and Nextcloud and that no error messages occur. Then take Nextcloud out of maintenance mode and repeat. Log in as admin and confirm normal function of Nextcloud.
7. Change the `CNAME` entry in the DNS to point your users to the new location.

26.8 Migrating from ownCloud

Note

Especially when migrating from ownCloud to Nextcloud you should create a backup of the config, database and the data directory, in case something goes wrong.

Currently migrating from ownCloud is like performing a manual update. So it is quite easy, to migrate from one ownCloud version to at least one Nextcloud version. However this does only work with versions that are sufficiently compatible in database schema and codebase. See the table below for a version map, where migrating is easily possible:

ownCloud	Nextcloud
10.13.x 10.14.x 10.15.x 10.16.x	34.0.0 34.0.0 34.0.0 34.0.0

Note

Since ownCloud does not and will not support PHP 8.0 or higher, you need to upgrade your PHP instance first, after which, ownCloud won't be accessible anymore. See *System requirements* about PHP versions supported by Nextcloud. We urge you to migrate to Nextcloud since PHP versions prior PHP 8 are end of life, see https://www.php.net/supported-versions.php.

1. First download the correct version of Nextcloud from our releases page,
2. Make sure to have do a *backup* before migrating.
3. Upgrade PHP to a version supported by the downloaded Nextcloud version.
4. Follow the upgrade instructions described in the *Upgrade manually* manual.
5. Run the following commands after `occ upgrade`:

- `occ db:convert-filecache-bigint`
- `occ db:add-missing-columns`
- `occ db:add-missing-indices`
- `occ db:add-missing-primary-keys`

6. If system cron was used, please verify if crontab entry was using the command `occ system:cron`. If yes, please adjust it to use the `php` command instead according to *the background jobs configuration documentation*
7. In case, use the *Nextcloud built-in updater* to update your instance to the newest version. This must be done for every major version, since updates between multiple major versions are not supported.
8. If multiple major version upgrades are done, it might be needed to upgrade PHP again in between.
9. Make sure to also verify the "Security & setup warnings" in the "Overview" section on the settings page.
10. In some cases, apps installed from the ownCloud Market might have been disabled as incompatible (ex: calendar and contacts), so you should reinstall the Nextcloud ones using `occ app:enable calendar`, `occ app:enable contacts`, etc

CHAPTER

TWENTYSEVEN

ISSUES AND TROUBLESHOOTING

27.1 General troubleshooting

If you have trouble installing, configuring or maintaining Nextcloud, please refer to our community support channels:

- **The Nextcloud Forums**
  The Nextcloud forums have a FAQ page where each topic corresponds to typical mistakes or frequently occurring issues

Please understand that all these channels essentially consist of users like you helping each other out. Consider helping others out where you can, to contribute back for the help you get. This is the only way to keep a community like Nextcloud healthy and sustainable!

If you are using Nextcloud in a business or otherwise large scale deployment, note that Nextcloud GmbH offers commercial support options.

27.1.1 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.

27.1.2 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.

Internal Server Errors

An Internal Server Error, sometimes called a "500 error", indicates that the web server encountered an unexpected condition that prevented it from fulfilling the request.

This error response is a generic "catch-all" response. To find out the source of the error you will need to check your Nextcloud log (located in *data/nextcloud.log* by default) and possibly your web server's error log (depending on where the failure is occurring).

> **Tip**
>
> Whenever possible, Nextcloud will include the "Request id" in the error. This request ID can be searched for in your Nextcloud log file to find entries associated with the failing transaction.

Nextcloud log files

The Nextcloud log file is located in the data directory by default - e.g. `data/nextcloud.log`. If the Web UI is still reachable, it is also available via *Administration settings->Logging*.

> **Tip**
>
> When asking for help, the entire raw log entry is generally required.

For some situations you may need to adjust the log level in your `config.php` file. Please see *Logging* for more information on these log levels.

Some logging - for example JavaScript console logging - needs debugging enabled. Edit `config/config.php` and change `'debug' => false,` to `'debug' => true,` Be sure to change it back when you are finished.

For JavaScript issues you will also need to view the javascript console. All major browsers have developer tools for viewing the console, and you usually access them by pressing F12.

PHP version and information

You will need to know the PHP version and configuration that is in-use on your Nextcloud server. This will not necessarily be the same version and configuration as can be reached from the command-line. The simplest way to gather this information is by using what's commonly referenced as `phpinfo()`.

The most accurate - and easiest - way to access `phpinfo` is by checking it from within Nextcloud itself. Of course, this requires that Nextcloud is functioning enough that you can log in as an administrator and access the **Administration settings -> System** menu. If so, you can enable the exposure of `phpinfo` data by toggling it on via `occ`:

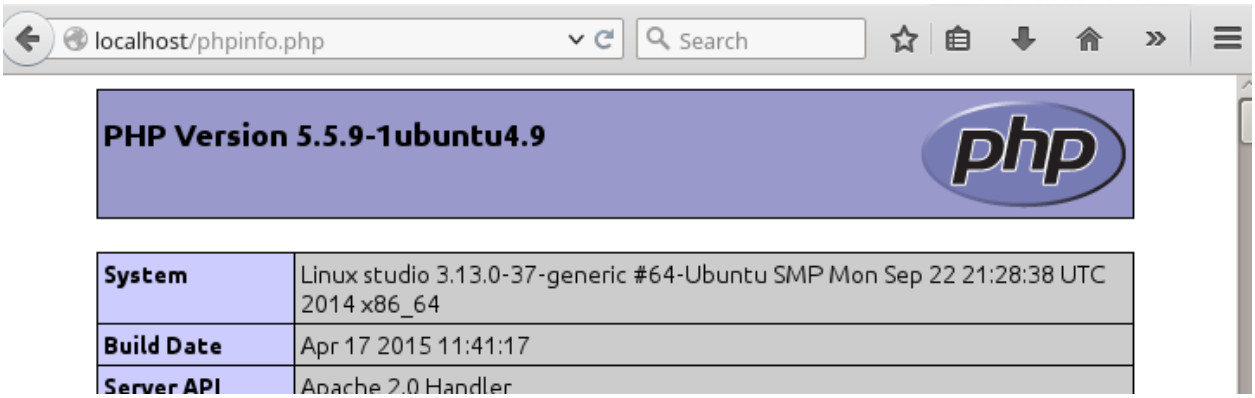
```
./occ config:app:set --value=yes serverinfo phpinfo
```

From then on a new button labeled **Show phpinfo** will be visible in the web interface under **Administration settings -> System**. Clicking it will expose just about everything you may want to know about your PHP environment.

If accessing the Nextcloud web interface is not an option, you may create a plain-text file named **phpinfo.php** and place it in your Web root, for example `/var/www/html/phpinfo.php`. (Your Web root may be in a different location; your Linux distribution documentation will tell you where.) This file contains just this line:

```
<?php phpinfo(); ?>
```

Open this file in a Web browser by pointing your browser to `localhost/phpinfo.php`:



Your PHP version is at the top, and the rest of the page contains abundant system information such as active modules, active `.ini` files, and much more. When you are finished reviewing your information you must delete `phpinfo.php`, or move it outside of your Web directory, because it is a security risk to expose such sensitive data.

Debugging sync issues

Warning

The data directory on the server is exclusive to Nextcloud and must not be modified manually.

Disregarding this can lead to unwanted behaviors like:

- Problems with sync clients
- Undetected changes due to caching in the database

If you need to directly upload files from the same server please use a WebDAV command line client like `cadaver` to upload files to the WebDAV interface at:

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

Common problems / error messages

Some common problems / error messages found in your logfiles as described above:

- `SQLSTATE[HY000] [1040] Too many connections` -> You need to increase the connection limit of your database, please refer to the manual of your database for more information.
- `SQLSTATE[HY000]: General error: 5 database is locked` -> You're using `SQLite` which can't handle a lot of parallel requests. Please consider converting to another database like described in *Convert to a different 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.

27.1.3 Troubleshooting Web server and PHP problems

Logfiles

When having issues the first step is to check the logfiles provided by PHP, the Web server and Nextcloud itself.

> **Note**
>
> In the following the paths to the logfiles of a default Debian installation running Apache2 with mod_php is assumed. On other Web servers, Linux distros or operating systems they can differ.

- The logfile of Apache2 is located in `/var/log/apache2/error.log`.
- The logfile of PHP can be configured in your `/etc/php/8.3/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 uploads/downloads. The following shows a draft overview of these modules:

1. Apache

- mod_pagespeed
- mod_evasive
- mod_security
- mod_reqtimeout
- mod_deflate
- mod_spdy
- 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

- Tideways

- eAccelerator

27.1.4 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.

27.1.5 Service discovery

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

If you want to use CalDAV or CardDAV clients or other clients that require service discovery together with Nextcloud it is important to have a correct working setup of the following URLs:

```
https://example.com/.well-known/carddav
https://example.com/.well-known/caldav
```

Those need to be redirecting your clients to the correct endpoints. If Nextcloud is running at the `DocumentRoot` of your Web server the correct URL is `https://example.com/remote.php/dav` for CardDAV and CalDAV and if running in a subfolder like `nextcloud`, then the correct URL is `https://example.com/nextcloud/remote.php/dav`.

> **Note**
>
> On Debian/Ubuntu systems, the Apache `DocumentRoot` defaults to `/var/www/html`. On other distributions, this path may differ. Adjust paths in the examples below accordingly.

For the first case, the `.htaccess` file shipped with Nextcloud should do this work for you when you're running Apache. You need to make sure that your Web server is using this file. Additionally, you need the mod_rewrite Apache module installed and `AllowOverride All` set in your `apache2.conf` or vHost-file to process these redirects. When running Nginx please refer to *NGINX configuration*.

For the second case, you need to add the following to `/etc/apache2/apache2.conf`, replacing `/var/www/html` with your actual `DocumentRoot` if it differs:

```
<Directory /var/www/html>
    AllowOverride FileInfo
</Directory>
```

`AllowOverride FileInfo` is sufficient for the rewrite directives used by service discovery. If you already have `AllowOverride All` set for this directory, no change is needed. Next, create or edit the `.htaccess` file within the `DocumentRoot` of your Web server and add the following lines:

```
<IfModule mod_rewrite.c>
  RewriteEngine on
  RewriteRule ^\.well-known/carddav /nextcloud/remote.php/dav [R=301,L]
  RewriteRule ^\.well-known/caldav /nextcloud/remote.php/dav [R=301,L]
  RewriteRule ^\.well-known/webfinger /nextcloud/index.php/.well-known/webfinger␣
↪[R=301,L]
  RewriteRule ^\.well-known/nodeinfo /nextcloud/index.php/.well-known/nodeinfo [R=301,
↪L]
</IfModule>
```

Make sure to change /nextcloud to the actual subfolder your Nextcloud instance is running in.

Note

If you put the above directives directly into an Apache configuration file (usually within `/etc/apache2/`) instead of `.htaccess`, you need to prepend the first argument of each `RewriteRule` option with a forward slash `/`, for example `^/\.well-known/carddav`. This is because Apache normalizes paths for the use in `.htaccess` files by dropping any number of leading slashes, but it does not do so for the use in its main configuration files.

If you are running NGINX, make sure `location = /.well-known/carddav {` and `location = /.well-known/caldav {` are properly configured as described in *NGINX configuration*, adapt to use a subfolder if necessary.

Now change the URL in the client settings to just use:

`https://example.com`

instead of e.g.

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

There are also several techniques to remedy this, which are described extensively at the Sabre DAV website.

27.1.6 Troubleshooting sharing

Users' Federated Cloud IDs not updated after a domain name change

1. run Database query

```
DELETE FROM oc_cards_properties WHERE name = 'CLOUD' AND addressbookid = (select id from
oc_addressbooks where principaluri = 'principals/system/system' AND uri = 'system');
```

2. run occ commands

```
occ dav:sync-system-addressbook
occ federation:sync-addressbooks
```

27.1.7 Troubleshooting contacts & calendar

Tip

Please also refer to the troubleshooting article in the groupware section: *Troubleshooting*.

27.1.8 Troubleshooting data-directory

Moving the data directory / changing the `datadirectory` path

For local storage, Nextcloud identifies storage by its absolute path on disk. Ideally, the location of the data directory should not change after deployment. If this is a new installation, consider reinstalling with your preferred directory location before moving into production.

Danger

If you must change the `datadirectory` path – unless the transition is handled carefully – Nextcloud will treat its content as "new storage." This can trigger duplicate or orphaned files, lost file metadata, and the loss of previously shared links.

For safely moving the data directory, the recommended actions are:

1. Make sure no cron jobs are running and, if using system cron, that the Nextcloud crontab entry is disabled.
2. Stop web/app server(s).
3. Move `/data` to the new location (ensure you also move hidden/dot files such as `.ncdata`).
4. Create a symlink from the original location to the new location.
5. Ensure permissions are still correct (including for any parent folders).
6. Restart web/app server(s).
7. Re-enable Nextcloud's system crontab entry (if applicable).

Note

You may need to configure your web server to support symlinks.

It is also possible to move the data directory without using symlinks, but this requires manually modifying the internal `oc_storages` database table:

1. Make sure no cron jobs are running and, if using system cron, that the Nextcloud crontab entry is disabled.
2. Stop web/app server(s).
3. Move `/data` to the new location (ensure you also move hidden/dot files such as `.ncdata`).
4. Update the value of `datadirectory` in your `config.php`.
5. Edit the database: In the `oc_storages` table, update the path portion of the `id` field of the entry beginning with `local::/old-data-dir/` (e.g., change `local::/old-data-dir/` to `local::/new-data-dir/`).
6. Ensure permissions are still correct (including for any parent folders).
7. Restart web/app server(s).
8. Re-enable Nextcloud's system crontab entry (if applicable).

Warning

This method is not supported and you risk breaking your database. Always make sure you have up-to-date backups – including your database – and a working (tested) restore process **before** attempting this.

27.1.9 Troubleshooting quota or size issues

Sometimes it can happen that the used space reported in the web UI or with `occ user:info $userId` does not match the actual data stored in the user's `data/$userId/files` directory.

Note

Metadata, versions, trashbin and encryption keys are not counted in the used space above. Please refer to the quota documentation for details.

Running the following command can help fix the sizes and quota for a given user:

```
sudo -E -u www-data php occ files:scan -vvv <user-id>
```

If **encryption was enabled earlier on the instance and disabled later on**, it is likely that some size values in the database did not correctly get reset upon decrypting. You can run the following SQL query to reset those after **backing up the database**:

```
UPDATE oc_filecache SET unencrypted_size=0 WHERE encrypted=0;
```

27.1.10 Troubleshooting encrypted files

Tip

Please also refer to the troubleshooting section in the encryption chapter: *Server-side Encryption*.

27.1.11 Fair Use Policy

Nextcloud is open source and you can host it for free on your own server or at a provider.

Nextcloud recommends Using Nextcloud Enterprise for deploying instances with more than 500 users. With that size, issues like a broken server or a data leak become very serious.

If there is an issue with the server, 500 people can't work. A data leak would risk the data of many users. In short, the server should be considered mission-critical. We believe you and your users would have a better experience with Nextcloud Enterprise.

Nextcloud Enterprise is pre-configured and optimised for the needs of professional organisations rather than home users. It comes with support, security and scaling benefits, compliance expertise, and access to our knowledge about running a successful Nextcloud, to get the best possible experience for users and admins. This also reduces the load on our home user forum http://help.nextcloud.com from issues unique to big deployments.

Nextcloud provides some infrastructure components needed for Nextcloud servers to run reliably. This includes notification, our app store and more. To ensure these resources do not get overloaded by administrators who run Nextcloud for thousands of users without providing financial resources to Nextcloud in return, these components are limited and will not work for more than 500 users.

We believe all organisations who run Nextcloud for hundreds of users should be officially supported. We know there can be financial restrictions for non-profit organisations and, as we want everybody to have a chance to get the most out of Nextcloud, we have special offers for NGOs, small schools and other non-profits. Please reach out to talk to us about what is possible through the contact form on our site or ask your system administrator to reach out.

27.1.12 Other issues

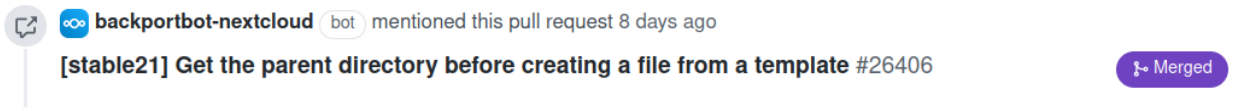
Some services like *Cloudflare* can cause issues by minimizing JavaScript and loading it only when needed. When having issues like a not working login button or creating new users make sure to disable such services first.

27.2 Patching Nextcloud

27.2.1 Obtaining a patch

If you found a related pull request on GitHub that solves your issue, or you want to help developers and verify a fix works, you can get a patch for the pull request.

1. Using https://github.com/nextcloud/server/pull/26396 as an example.
2. Append `.diff` to the URL: https://github.com/nextcloud/server/pull/26396.diff
3. Download the patch to your server e.g. via `wget https://github.com/nextcloud/server/pull/26396.diff` (this will place `26396.diff` in the local directory)
4. Follow the *Applying a patch* steps.
5. If you are on an older Nextcloud version, you might first need to go to the correct backported patch for your version.



6. You can find the appropriate version by looking for a link posted by `backportbot-nextcloud` to the backport pull request for your release, or by checking for a developer comment with a manual backport link. Use the `.diff` URL of that backport PR.

27.2.2 Applying a patch

Patching server

1. Navigate to your Nextcloud server's root directory (the one that contains the `status.php` file).
2. Download the patch to your server e.g. via `wget https://github.com/nextcloud/server/pull/26396.diff` (this will place `26396.diff` in the local directory)
3. Apply the patch with the following command:

```
patch -p 1 < ./26396.diff
```

4. Alternatively, if the patch command is not available, use:

```
git apply --check ./26396.diff
git apply ./26396.diff
```

Patching apps

1. Navigate to the root of the app (usually `apps/[APPID]/`). If you cannot find the app there, use the `sudo -E -u www-data php occ app:getpath APPID` command to find the path.
2. Download the patch to your server e.g. via `wget https://github.com/nextcloud/<app>/pull/26396.diff` (this will place `26396.diff` in the local directory)
3. Apply the patch with the same command as in *Patching server*.

27.2.3 Reverting a patch

1. Navigate to the directory where you applied the patch.
2. Revert the patch with the `-R` option:

```
patch -R -p 1 < ./26396.diff
```

3. Alternatively, if the patch command is not available, use:

```
git apply --reverse ./26396.diff
```

27.2.4 Notes and troubleshooting

Note

You may see errors about files not being found, especially when applying patches from GitHub. Patches can include development or test files (for example, files under `build/` or `tests/`) that are not present on your installation. These messages are expected and can be ignored if they refer only to such files.

27.3 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.

27.3.1 FAQ

Why did Nextcloud add code signing?

By supporting Code Signing we add another layer of security by ensuring that nobody other than authorized persons can push updates for applications, and ensuring proper upgrades.

Do we lock down Nextcloud?

The Nextcloud project is open source and always will be. We do not want to make it more difficult for our users to run Nextcloud. Any code signing errors on upgrades will not prevent Nextcloud from running, but will display a warning on the Admin page. For applications that are not tagged “Official” the code signing process is optional.

Not open source anymore?

The Nextcloud project is open source and always will be. The code signing process is optional, though highly recommended. The code check for the core parts of Nextcloud is enabled when the Nextcloud release version branch has been set to stable.

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

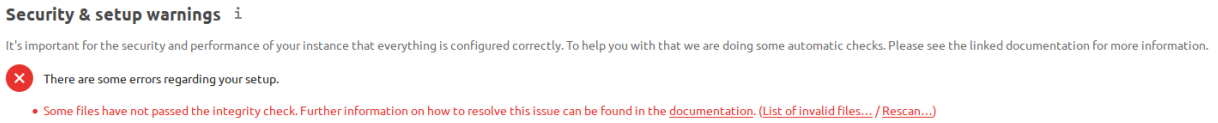
Is code signing mandatory for apps?

Code signing is required for all applications on apps.nextcloud.com.

27.3.2 Fixing invalid code integrity messages

A code integrity error message ("Some files have not passed the integrity check…") appears on your Nextcloud admin page under "Overview", which provides the following options:

1. Link to this documentation entry.
2. Show a list of invalid files.
3. Trigger a rescan.



To debug issues caused by the code integrity check click on "List of invalid files…", and you will be shown a text document listing the different issues. The content of the file will look similar to the following example:

```
Technical information
=====================
The following list covers which files have failed the integrity check. Please read
the previous linked documentation to learn more about the errors and how to fix
them.

Results
=======
- core
  - INVALID_HASH
      - /index.php
      - /version.php
  - EXTRA_FILE
      - /test.php
- calendar
  - EXCEPTION
      - OC\IntegrityCheck\Exceptions\InvalidSignatureException
      - Signature data not found.

Raw output
==========
Array
(
    [core] => Array
        (
```

(continues on next page)

(continued from previous page)

```
            [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.
                )
```

(continues on next page)

(continued from previous page)

```
        )

)
```

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.

27.3.3 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`.

27.3.4 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`.