ownCloud Administration Manual

The ownCloud Team

Version 10.4, May 13, 2020

Table of Contents

Introduction. 1
Release Notes 1
What's New in ownCloud 46
Frequently Asked Questions 46
Installation. 47
Configuration 120
Maintenance. 479
What is the Appliance? 514
Enterprise Edition 569
Document Classification and Policy Enforcement 625
Troubleshooting. 633
Have You Found a Mistake In The Documentation? 634

Introduction

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

Current editions of ownCloud manuals are always available online at doc.owncloud.org and doc.owncloud.com.

ownCloud server is available in three editions:

- The free community-supported server. This is the core server for all editions.
- The Standard Subscription for customers who want paid support for the core Server, without Enterprise applications.
- The Enterprise Subscription provides paid support for the Enterprise Edition. This includes the core Server and Enterprise apps.

See What's New in ownCloud for more information on the different ownCloud editions.

ownCloud Videos and Blogs

See the official ownCloud channel and ownClouders community channel on YouTube for tutorials, overviews, and conference videos. Visit ownCloud Planet for news and developer blogs.

Target Audience

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

- ownCloud User Manual
- ownCloud Desktop Client
- ownCloud Android App
- ownCloud iOS App

Release Notes

- Changes in 10.4.1
- Changes in 10.4.0
- Changes in 10.3.2
- Changes in 10.3.1
- Changes in 10.3.0
- Changes in 10.2.1
- Changes in 10.2.0
- Changes in 10.1.1
- Changes in 10.1.0
- Changes in 10.0.10
- Changes in 10.0.9
- Changes in 10.0.8

- Changes in 10.0.7
- Changes in 10.0.6
- Changes in 10.0.5
- Changes in 10.0.4
- Changes in 10.0.3
- Changes in 10.0.1
- Changes in 10.0.0
- Changes in 9.1
- Changes in 9.0
- Changes in 8.2
- Changes in 8.1
- Changes in 8.0
- Changes in 7.0

Changes in 10.4.1

ownCloud Server 10.4.1 is a bug fix and maintenance release. You can read the full ownCloud Server changelog for further details on what has changed.

Notable changes

- The 10.4.0 known issue between Password Policy and user/group share expiration is fixed. Server 10.4.1 and Password Policy 2.1.2 are required to resolve it. #37135
- Reshared public links are now shown to the share owner. #36865
- Externally encrypted files can now be downloaded. #36921
- Improvements have been added to make long-running downloads more stable. #36978
- Pending federated shares are now also shown in the "*Shared with you*" tab and can be accepted/declined there. #37022
- The files:transfer-ownership occ command can now also be executed for users who have never logged in. #37038
- File download for files without a file extension from Google Drive external storages now works. #37044
- The calculation of the remaining upload time in public links has been improved. #37053
- E-mail notifications (e.g., for sharing) now respect the default_language config.php option. #37039
- A new occ command (files:check-cache) is now available. It checks if a target file can be read from the storage and cleans up stored information in ownCloud's filecache, in case a file disappears from the primary storage. This is mainly important for object stores and should only be utilized in rare cases. #37067

Known issues

Sharing with Numeric UIDs

With Server 10.4.0 and 10.4.1, sharing resources with users that have numeric user ids (e.g., "123") does not work in some cases. #37324

Apart from this patch release, please consider the ownCloud Server 10.4.0 release

notes, below.

Changes in 10.4.0

Dear ownCloud administrator, please find below the changes and known issues in ownCloud Server 10.4 that need your attention. You can also read {owncloud-changelog-url}[the full ownCloud Server changelog] for further details on what has changed.

Migrations

Upgrading from ownCloud Server 10.3.x to 10.4.0 does not involve database migrations. The upgrade duration is, therefore, expected to be short.

PHP 7.0 Support Discontinued

As announced, in the previous release of ownCloud Server, from version 10.4 onward, ownCloud **no longer supports PHP 7.0**. If you're running on PHP 7.0, it is necessary to upgrade PHP **prior** to conducting the upgrade to Server 10.4. We strongly recommend upgrading to PHP 7.2 or 7.3. See the system requirements for more information.



If you're using the official Docker containers or the Univention appliance, this has been taken care of already.

PHP 7.1 Deprecation Note

PHP 7.1 recently reached its end of life and is not maintained anymore. ownCloud Server will, therefore, drop support in one of the next minor versions as well. If you're running on PHP < 7.2, please make sure to schedule an upgrade to PHP 7.2 or 7.3 as soon as possible. See the system requirements for more information.

Expiration Date for User and Group Shares

To give users and administrators more control of access to resources, Server 10.4 introduces an expiration date for user and group shares, just like in public links. With this new feature, users can control the lifetime of shares with other users or groups. Administrators can choose to set a default maximum lifetime and to enforce it. To integrate this change, the UI in the user/group sharing tab of the sidebar has been adapted. When a resource is shared, the user and group entries are expandable and collapsible using the cogwheel next to the trash bin icon to show/hide the permissions and the expiration date field to maintain an overview. Additionally, to allow users to recognize expiring shares at a glance, a new clock indicator will be shown next to the cogwheel.

Administrators can configure the feature in the '*Sharing*' section of the admin settings.

Sharing Information in Subfolders

ownCloud Server 10.4 puts the focus on user awareness for shared areas to prevent accidentally sharing data or changing other users' data, as well as to make it easier for users to recognize who has access to shared areas. Practically, users are better able to recognize shared resources using a new share overlay indicator on file and folder icons. The indicators are also applied to resources that are not directly shared but are part of a share (when working in a shared folder).

Apart from that, the sharing sidebar panels have been improved to also show users/groups and public links which have access through shares on parent folders. These will be shown as static entries with a "*via*" indicator that allows users to jump to

the parent folder and to change the share properties, if desired.

This sharing information is only shown to share owners (users that created shares) as other share recipients are not entitled to get detailed information about who else has access.

MariaDB 10.4 and PostgreSQL 10 Support

The discontinuation of PHP 7.0 enables support for MariaDB up to version 10.4 and PostgreSQL 10. Server 10.4 is thoroughly tested against these database versions and proven to work stable.

Other Notable Changes

- External storages can now be mounted in read-only mode. #36397
- Filter options (--enabled and --disabled) have been added to the occ app:list command to only show enabled or disabled apps, respectively. #36520
- Support for Oracle DB connection strings has been added to be able to use Oracle-specific configuration settings like failover. #36489
- Two new config.php options (blacklisted_files_regex and excluded_directories_regex) have been added to allow excluding files and folders from ownCloud using regular expressions (e.g., to prevent creating/renaming/scanning certain file types like ".pst"). See config.sample.php for more information. #36360
- Previously, when a settings section did not have any panels to display, an "Error" was shown. This has been changed to be more user-friendly. #36776
- The memory consumption of the occ files:checksums:verify command has been optimized, and the command will now show progress information. #36787
- The memory consumption of the trash bin expiration background job has been optimized. #36565

Solved Known Issues

- Folder download via the web interface now works in macOS Catalina. #36722
- User creation now allows "" characters in the user id (e.g., to invite guests with mail addresses containing ""). For the change to take effect, you also need to upgrade the guests and/or user_ldap apps to the latest version. #36613
- File locking actions are not available for public link endpoints anymore. #36402
- occ files:transfer-ownership now works in S3 multi-bucket setups. #36464
- The "*Notify by email*" button in users/groups sharing now also works when the initiator does not have an email address set. #36505
- Remaining .part files from unfinished uploads via public links will now be cleaned up. #36761
- The quota usage calculation of the trash bin retention has been fixed. Previously, it mistakenly counted the space usage of incoming shares toward the user's quota usage leading to undesired behavior, e.g., when trashbin_retention_obligation was set to auto, the user had a quota set and incoming shares exceeded 50% of this quota. #36494
- The command to sync single users from external user backends like LDAP (occ user:sync -u 'username') does not abort anymore if multiple users matching the search term are returned (e.g., 'alice' could return 'alice' and 'alice1'). It will only abort if none of the results matches the search term (e.g., 'alice' returns 'alice1' and 'alice2'). #36576
- When sharing with both a user and a group with the same name, adjusting the

permissions of the second entry works again. #36813

For Developers

- The WebDAV Trash bin API and the WebDAV endpoint for public links (introduced with 10.3.0) have left the tech preview state. They are considered stable and are enabled by default.
- The config.php option to enable/disable tech preview APIs ('dav.enable.tech_preview' ⇒ true) has been removed as it's obsolete. #36815
- A new OCS User Sync API to trigger user sync from external user backends has been added. This allows external user provisioning systems to push new users to ownCloud on demand and removes the necessity to do full user sync. #36428

Known Issues

Password Policy App

If the public link expiration policy "*days maximum until link expires if password is not set*" is enabled, sharing with users and groups will not work. A fix for this issue is currently being developed. If you have already upgraded to ownCloud 10.4.0, please disable this option until the fix is available and deployed on your system. #287

This issue has been resolved with ownCloud Server 10.4.1.

Sharing with Numeric UIDs

With Server 10.4.0 and 10.4.1, sharing resources with users that have numeric user ids (e.g., "123") does not work in some cases. #37324

Changes in 10.3.2

ownCloud Server 10.3.2 is a bug fix and maintenance release. You can read the full ownCloud Server changelog for further details on what has changed.

Notable changes

- Guest sharing works now even when the sharing restriction Restrict users to only share with users in their groups is enabled #36384
- When creating a public link on a received shared resource (reshare), users can now send the public link via mail using the web interface if the feature is enabled #36386
- occ system:cron now only shows output when errors occur or when the --progress option is added #36298
- occ files:transfer-ownership does not collect shares outside of the given path anymore preventing errors #36222
- The accounts.enable_medial_search config.php setting now also respects federated user search #36225
- When using multi-bucket object storage, versioning information is now stored on the object storage instead of the database #36329
- When using the share_folder config.php option, the defined folder and its parents can't be shared anymore #36241
- Files/folders can again be shared when a user and a group have the same name #35488
- occ files_external:list can now list mount options by adding --mount-options #36420

Apart from this patch release, please consider the ownCloud Server 10.3.0 release

notes, below.

Changes in 10.3.1

ownCloud Server 10.3.1 is a bug fix and maintenance follow-up release. You can read the full ownCloud Server changelog for further details on what has changed. It is recommended to schedule an upgrade to this version soon.

Apart from this patch release, please consider the ownCloud Server 10.3.0 release notes, below.

Changes in 10.3.0

Dear ownCloud administrator, please find, below, the changes and known issues in ownCloud Server 10.3 that need your attention. You can also read the full ownCloud Server changelog for further details on what has changed.

Migrations

- For improved compliance with the OpenCloudMesh protocol specification (Federation) a migration step will convert the fields of the remoteId column of the federated_reshares and share_external tables from int to string. This migration might increase the upgrade duration depending on the number of federated shares.
- A repair step has been added that drops the deprecated contacts_cards_properties table. This migration is not expected to increase the upgrade duration significantly.
- A housekeeping repair step for the oc_properties table removes existing entries which have fileid with value null and restrict the further creation of such. This repair step is not expected to increase the upgrade duration significantly.

Official PHP 7.3 support

ownCloud Server 10.3 officially supports PHP 7.3. The Server Core and all apps maintained by ownCloud have received a full QA cycle and are proven to work reliably with PHP 7.3. If you are still using version 5.6, you must upgrade PHP before upgrading ownCloud Server as it's not supported anymore since ownCloud Server 10.2. If you are still running PHP 7.0 or 7.1, please plan an upgrade soon as these versions are or will soon be unsupported, respectively. See the system requirements in the ownCloud Documentation for more information.

PHP 7.0 deprecation note

As announced with ownCloud Server 10.0.8 and 10.2.0, support for PHP 7.0 is discontinued. The next minor version of ownCloud Server (around the end of 2019) **no longer supports PHP 7.0**. If you are still running on PHP 7.0, please make sure to plan an upgrade to PHP >= 7.2 to stay compatible.

Changes to background job execution

For code cleanup reasons, the execution of background jobs (e.g., for public link expiration, trash bin emptying, cleanup of old file versions) has been changed.

The following changes require manual interaction in your installation:

- If you're using System cron to trigger background job execution, there is a new occ command (occ system:cron) which executes the background jobs. To make use of it, you have to change the entry in crontab. Instead of executing cron.php (e.g., /usr/bin/php -f /path/to/your/owncloud/cron.php), cron should use occ system:cron (e.g., {occ-command-example-prefix} system:cron). As a fallback, cron.php will continue to work with Server 10.3 but will be removed in a later version.

- If you're using Webcron to trigger background job execution you now have to call the new route ../cron instead of ../cron.php. As a fallback, ../cron.php will continue to work with Server 10.3 but will be removed in a later version.



See the occ System documentation for more information.



In a later version of ownCloud Server, cron.php will be removed. Please apply the changes to ensure that background jobs continue to work.



If your ownCloud deployment is based on the official Docker images or the Univention appliance, these changes have already been applied for you.

Media Viewer replaces Gallery and Video Player

The Media Viewer app has recently been released. The Media Viewer is the next generation image and video file viewer for ownCloud. It provides a foundation based on new technologies and officially supersedes the former gallery and files_videoplayer apps. ownCloud Server 10.3 does not bundle gallery and files_videoplayer anymore. Instead, it bundles files_mediaviewer. With this change, support and maintenance for gallery and files_videoplayer are discontinued. More details on the Media Viewer can be found in the release blog post.

- For a clean transition to Media Viewer, it is necessary to **disable both deprecated apps before the upgrade** using either the admin "Apps" panel in the web interface or via occ (e.g., occ app:disable gallery followed by occ app:disable files_videoplayer).
- After the upgrade, enable the Media Viewer app via the admin panel or occ app:enable files_mediaviewer.
- It is not recommended to continue with the deprecated apps. However, if you want to do so, you can copy over the files_videoplayer directory from the apps/ folder of the previous ownCloud Server directory and obtain gallery from the ownCloud Marketplace.



Please do not enable gallery/files_videoplayer and files_mediaviewer simultaneously, as these apps are mutually exclusive.

For more information on the Media Viewer app, visit the ownCloud Documentation.

OAuth2 and session handling improvements

Server 10.3 comes with improvements for session handling with Redis. These are designed to cope with issues encountered around duplicate session tokens, which make the ownCloud Clients lose their OAuth2 authorization from time to time, and force users to re-authenticate.

The session handling in ownCloud 10.3.0 has been generally improved, making user and client sessions more stable. If Redis is used for session handling, it is necessary to enable Session Locking to ensure that the mentioned issues no longer occur.



You can find out if Redis session handling is configured in your web server if you generate an ownCloud Configreport in the web interface. You will find the value session.save_handler set to redis.

- It is recommended to use Redis Session Locking if Redis is used for session handling (minimum required version for php-redis is 4.1.0)

- Enable Redis Session Locking by setting redis.session.locking_enabled = 1 in php.ini

	If Redis is just used as a memory cache or not in use at all, you do not have to apply changes.

	Please note that Redis Session Locking is not supported in clustered Redis environments.

	If your ownCloud deployment is based on the official Docker images or the Univention appliance, you do not have to apply changes as Redis is not used for session handling (unless you configured it differently using ENV variables).

Restructured user/group sharing autocompletion

To cope for long user names or additional user information and to provide a better overview for users, the user/group sharing autocompletion dropdown has been restructured. The available information is now distributed vertically to improve space usage and user experience. Screenshots are available in the pull request. Other ownCloud clients will align with this behavior with the next releases.

SWIFT object storage as primary & secondary storage removed

Following the deprecation announcement with ownCloud Server 10.0.9, support for primary and secondary storage via the OpenStack SWIFT protocol has been removed. Please get in contact with ownCloud Support if you're still using OpenStack SWIFT and want to upgrade.

S3 object storage as secondary storage is now a separate app

The extension to integrate S3 object storages as secondary storages (files_external_s3) has been updated, unbundled from ownCloud Server (was previously part of files_external) and released to the ownCloud Marketplace. If you're using S3 external storage mounts, you have to conduct some steps to ensure continuous operation after upgrading to Server 10.3:

- After the upgrade to Server 10.3 has finished successfully, keep the maintenance mode activated and install/enable files_external_s3 (either manually or via the Market app) as the app is not bundled with ownCloud Server anymore.
- If users were allowed to configure personal mount points before the upgrade, switch from maintenance mode into single user mode (occ maintenance:singleuser --on) and enable the option again by ticking the respective checkbox (*Amazon S3*) below "*Allow users to mount external storage*" (in Admin settings ⇒ Storage).
- Existing storage mount points will remain and do not have to be touched.
- Make sure that everything works and disable maintenance/single-user mode to put the installation back into normal operations (occ maintance:mode --off / occ maintenance:singleuser --off).

New HTTP APIs

ownCloud Server is being prepared for Phoenix, the upcoming web frontend for ownCloud. As Phoenix is separated from the backend and communicates only via HTTP APIs, it is necessary to complete the API coverage.

The following new HTTP APIs have been added with Server 10.3:

- WebDAV Trash bin API.
- OCS API for public link share email notifications
- WebDAV endpoint for public links.

All new endpoints are currently in tech preview state and are mainly used for Phoenix development. For this reason, they are disabled by default and have to be explicitly enabled using the new config.php option: 'dav.enable.tech_preview' ⇒ true,.

Other notable changes

- The previews_path config option has been added to allow customization of the thumbnail storage path (by default those reside in the user storage). #35131
- An Activity entry is now shown when a share receiver unshares a share. #35193
- The WebUI experience on mobile devices has been improved. #35919 #35813 #35347 #34803
- The config.php options proxy and proxyuserpwd will now be respected to enable federation when an instance needs to go through an authenticated proxy to reach a federated instance. See config.sample.php and the Federated Cloud Sharing Configuration documentation for more information.
- The occ files:scan command is now case-insensitive for the userid. #35324
- A new config.php option (dav.enable.tech_preview) has been added to disable tech preview APIs by default. #36124
- [PHOENIX] Support for redirecting links to ownCloud Phoenix frontend has been added by introducing a new config.php option which stores the address where Phoenix is reachable (e.g., 'phoenix.baseUrl' ⇒ 'http://phoenix.example.tld:port'). #35819
- The performance when loading groups of users has been improved. #35822
- Memory handling for the trashbin expiry background job has been improved. #35708

Solved known issues

- A new occ command, encryption:fix-encrypted-version, has been introduced to address issues related to encrypted files no longer being accessible. This originated from a security measure to avoid that encrypted files with the same content look identical. In some cases, users get a Bad Signature error when trying to access files. The new command corrects this behavior, making files accessible again. The command only needs to be run if users report the mentioned error. #115
- If an instance uses the share_folder config.php option to gather incoming user shares in a specific folder, this folder cannot be deleted by users anymore. #35998
- The share_folder config.php option now also respects federated shares. #35396
- The user.min_search_length config.php option now also respects federated users. #35977
- Issues with database conversions using the db:convert-type occ command (e.g., SQLite to MySQL) have been fixed. This is still in an experimental state and should be tested thoroughly. Please provide feedback if you encounter issues. #35390
- File integrity checking has been improved to prevent issues: If a checksum mismatch occurs after uploading a file, the uploaded file and its checksum is deleted to prepare for a clean re-upload. #35294
- User/group sharing permission handling
  - When a share recipient shared a resource with a group the resource owner was a member of (reshare), the resource owner was unable to increase the

permissions of the initial share. This has now been fixed. #35884

  - When a user shared a resource with a group, share recipients (members of the group) were able to remove the share altogether (instead of just unsharing from themselves). This has been fixed. #36120

- External storages now return StorageNotAvailable correctly on temporary network failures to prevent associated issues (e.g., Desktop clients will not delete local folders anymore when the storage is temporarily not available). #35707
- External storage: Multiple Google Drive external storages can be added again. #34987
- The input fields in user administration are not captured by password manager autocompletion anymore. #35931
- Storage encryption with a master key in an HSM: Recreating a master key works again. #128

For developers

- Tech preview for WebDAV Trash bin API (disabled by default). #35716 #35879
- Tech preview for OCS API for public link share email notifications (disabled by default). #36063
- Tech preview DAV endpoint for public shares (disabled by default). #35932
- Two-factor providers may now display custom challenge messages. #34848
- The theming capabilities have been improved by allowing HTML for Name and LogoClaim. Please check the changes to owncloud/theme-example if you are interested in making use of this in your theme. #35273
- A new Roles API has been added to allow clients to query the server for available permissions/roles for user/group sharing and public links. In future client releases, this endpoint will be used to dynamically display roles/permissions depending on the server's capabilities. You can find out more about it in the Roles API documentation.
- A new, improved version of the "*Advanced Sharing Permissions*" JavaScript API (v2) has been added to allow ownCloud apps to register additional permissions/restrictions in user/group sharing. Version 1 of the API is still available in parallel. #35863

Known issues



This section will be updated if further issues become known.

- **WebDAV Locks:** When a file in a folder is locked, exclusively locking the parent folder currently still works ("conflicting lock"; divergent from RFC 4918))

Changes in 10.2.1

ownCloud Server 10.2.1 is a bug fix and maintenance release taking care of several bugs and known issues. Please find, below, the changes in ownCloud Server 10.2.1 that need your attention. You can also read the full ownCloud Server changelog for further details on what has changed. It is recommended to schedule an upgrade to this version soon, especially if you're running 10.2.0 already.



No occ upgrade is required when upgrading from 10.2.0.

Improved Performance For Storage Encryption With Master Key

ownCloud Server offers two ways for key management with storage encryption. Either a central master key pair or individual user key pairs are used to encrypt/decrypt data. Previously both modes used the same mechanisms which resulted in potentially significant overhead when master key encryption was used as user key encryption relies on so-called share keys which are necessary to allow share recipients to decrypt shared files.

With master key encryption, share keys are redundant as you have one central key that can be used to decrypt all files. Version 10.2.1 corrects this behavior by dropping share keys for master key encryption, thereby increasing the performance dramatically, especially when sharing folders with many files as said keys do not have to be generated anymore for each file.

Solved Issues

- **Fixed reshare permission issue**
  An issue in the Sharing API allowed users to increase sharing permissions beyond their own permissions in a reshare scenario: When user A shares a folder "*Project*" with user B, granting only read and share permission, then the Sharing API allowed user B to reshare a subfolder of "*Project*" with user C granting full permissions or to create a public link on the shared folder, respectively. This undesired behavior is fixed with 10.2.1.
- **Fixed issue with Sharing API and enforced public link expiration dates**
  An issue in the Sharing API caused the ownCloud clients to prevent users from creating public links when the option "Enforce expiration date" for public links is in use. This is now solved.
- **Fixed known issue with user avatar paths**
  Version 10.2.0 accidentally changed the location of user avatars making them unavailable and storing uploaded avatar images in the wrong location. 10.2.1 restores the earlier behavior and provides a repair step to move back the avatar images uploaded with 10.2.0 to the right location. As it is not necessary nor possible to run occ upgrade when upgrading from 10.2.0 to this patch release, if you are already running 10.2.0 then after installing 10.2.1 **you need to run `occ maintenance:repair -s 'OC\Repair\MoveAvatarIntoSubFolder'` manually to trigger the repair step**.
- **Fixed known issue with "Password changed" HTML emails rendered in plain text**
- **Fixed use of invalid token on password reset**
  Password reset links sent to a user were invalid, if the user attempted to login using their e-mail address and an invalid password prior to filling out and submitting the Reset Password form.
- **Fixed issue when removing a user from a group**
  Removing a user from a group using the user management UI resulted in an error that required the page to be refreshed to show the changes. This has been corrected.
- **Added -y option to `occ encryption:encrypt-all` command**
  The occ command encryption:encrypt-all now offers a -y option that can be used to automatically answer potential questions with "yes" which is particularly important for automated deployments with Ansible or similar tools.
- **Fixed an issue with loading JS files when multiple apps folders are in use**
  Previously ownCloud would have taken the files from the apps/ folder even though there might be newer versions in e.g. apps-external. This has been changed so that ownCloud will always take the files from the most recent app version.



Apart from this patch release, please consider the ownCloud Server 10.2.0 release notes.

Changes in 10.2.0

Dear ownCloud administrator, please find, below, the changes and known issues in ownCloud Server 10.2 that need your attention. You can also read the full ownCloud Server changelog for further details on what has changed.

Migrations

Please note that this minor release contains database migrations which impact the upgrade duration. Specifically:

- The oc_share table has a new column. The time the upgrade takes for this change depends on the number of shares in your ownCloud installation.
- The oc_authtoken table's login name column size has been increased. The time the upgrade takes for this change depends on the number of recently logged in users, and the number of app passwords that have been created.

PHP 5.6 Deprecation

Following up the PHP 5.6/7.0 deprecation notice in the ownCloud Server 10.0.8 releases ownCloud Server 10.2 **does not support PHP 5.6** and some apps no longer support older PHP versions. Additionally, PHP 7.3 support will be available in an upcoming version.

If you're still running PHP 5.6, **you must upgrade to PHP 7 before upgrading to ownCloud Server 10.2**. Please be aware that apps that do not support outdated PHP versions will not upgrade.



See the system requirements in the ownCloud documentation.

To allow for additional upgrade time, version 10.2 still supports PHP 7.0, because some of the major Linux distributions continue to support it. However, support for PHP 7.0 will be discontinued in an upcoming version of ownCloud 10, to enhance both security and performance. To prepare for this change, we strongly encourage you to begin planning an upgrade as soon as possible.

Advanced Sharing Permissions

The new server version introduces the means for extensions to implement additional, advanced permissions for user and group sharing. This feature increases sharing flexibility and opens the doors for extension developers to introduce new functionality based on sharing permissions.

Especially, considering collaborative editing solutions, this addition provides the foundation for mode-based document sharing, such as "view-only", "comments-only" or "enforce change tracking". In the future, such advanced permissions should significantly improve the security as well as the usability of review processes, working on Office documents collaboratively, or exchanging information securely.

Based on the new capabilities a set of features has been developed together with Collabora Online, called *Secure View*. Secure View is designed to enable information distribution processes for sensitive data, meaning that information can be provided securely yet can — **under no circumstances** — leave the platform.

Practically, it enables users to share documents (such as docx, xlsx, pptx, and PDF files) in such a way that the recipient can't edit, download, copy and paste, nor print

them. Additional protection for screenshots and photos is provided by watermarks which display user information. What's more, users can decide to allow printing and exporting of documents protected by watermarks as well.

More Granular Permissions for Public Links on Folders

With ownCloud Server 10.2, the former "Download / View / Upload" permission has been renamed to "Download / View / Edit", as this better reflects its behavior (full permissions). Additionally, a new permission ("Download / View / Upload") has been introduced which allows recipients to view, download, and upload contents but not to make any changes to existing content (e.g., rename, move, delete, update). Another way of looking at it is as a public file drop folder for distributing and gathering information with a single link, yet which prevents recipients from altering the existing content.

Storage Encryption with Master Key in HSM

With version 10.2, ownCloud Server officially supports storage encryption with master keys stored in hardware security modules (HSM). In contrast to regular master key-based storage encryption, which stores the keys on the storage, storage encryption with keys in an HSM allows administrators to completely prevent anyone with access to the storage from accessing the data stored in ownCloud.

As a result, the bundled encryption app has been updated to support HSM, and a standalone service (hsmdaemon) that connects ownCloud Server and the HSM device is now available within ownCloud Enterprise Edition. To get started with storage encryption and HSM, please get in touch with us. For more information around the different encryption types ownCloud offers, consider this whitepaper.

Background Job for Change Detection of Nested Federated Shares

When using federation to share data across ownCloud instances, deeply nested folders (e.g., folders with many sub-items) are not discovered automatically for performance reasons. This leads to several issues such as the ownCloud Desktop Client not being able to synchronize newly added or changed content unless the user navigates down the hierarchy using the web interface, which manually triggers content discovery.

Also, the size of such folders can't be calculated, showing "Pending" instead, until the discovery is manually triggered. To help alleviate this problem, a new occ command has been introduced. It can be executed regularly as a background job to discover federated shares (occ incoming-shares:poll). This is aimed at handling this issue while providing the means for administrators to control resource usage.

When using federation, it is recommended to execute occ incoming-shares:poll regularly using Cron jobs. The time interval to choose between executions is a trade-off between the availability of changes in federated shares and resource consumption, which naturally depends a lot on the number of federated shares and the frequency of changes within those shares.

Executing the command once per 12 hours should be safe enough for any instance. However, the interval could be reduced to once per 2 hours for instances with a low number of federated shares.

Depending on the desired resource consumption this value should be lowered or increased based on individual expectations. To find a value that fits a specific setup, it is recommended to execute the command once, measure the execution time and set the interval so that the background job can finish before the next execution is triggered.

New Option to Automatically Accept Federated Shares from Trusted Servers

ownCloud Server 10.0.9 introduced the **Pending Shares** feature which allows users to decide whether or not they want to accept local user shares instead of just making the decision for them, giving more control thereby. In contrast, Federated shares always had to be accepted as they can originate from external, potentially untrusted, sources.

ownCloud Server 10.2 introduces a global option to automatically accept federated shares originating from trusted servers. This option enables providers of several instances (e.g., an external and an internal instance) to facilitate or automate data exchange between them, not requiring users to accept shares.



For security reasons, federated shares from untrusted servers will never be accepted automatically.

New Privacy and Self-Service Options for Users

In the spirit of self-service, ownCloud Server 10.2 introduces new options for users that previously were reserved for global admin settings:

- As discussed in the section above, there are global options for **Pending Shares** regarding federated as well as regular user/group shares. To give users more control over the sharing behavior in the scope of their account, user-based override options were introduced that allow users to enable/disable **Pending Shares** for themselves if the instance's global setting is disabled (when "*Automatically accept new incoming local user shares*" is enabled). The two new checkboxes can be found in the 'Sharing' settings panel of personal settings.
- In addition to the option "*Allow username autocompletion in share dialog*" in the global 'Sharing' settings, users can now autonomously decide to opt-out of autocompletion to protect their privacy. When enabled, other users need to enter a user's full identifier to be able to share with them. This option is not a general override but an opt-out, meaning it can only be used when "*Allow username autocompletion in share dialog*" is enabled. The new checkbox is available in the 'Sharing' settings panel of personal settings.

Other Notable Changes

- **Added email footer with motto in email for changing passwords.** If you use customized email templates, it is necessary to adapt those to incorporate the footer. Please compare the original templates with your custom templates (core/templates/lostpassword/notify.php and core/templates/lostpassword/altnotify.php).
- **Repair steps can now be executed individually in case one would need to be run again.** Repair steps are employed to clean up and resolve issues from former versions. Usually, they run during upgrades, but some scenarios make it necessary to rerun them. To save time when only specific steps need to be taken, administrators can now individually execute them using occ maintenance:repair --list and occ maintenance:repair --single "<repair step>".
- **Command for the first run wizard to reset for all users.** In some cases, administrators customize the First Run Wizard in order to distribute information to users. Using occ firstrunwizard:reset-all you can reset the popup so that it will appear for each user upon their next login.
- **Added checkboxes to hide quota and password in user management.** The columns in user management have been made more flexible. Using the bottom left cog wheel you can now show/hide the columns for *Quota* and *Password*.
- **By default, the "apps-external" directory is included in config.php during**

installation. For new installations, there will be two apps directories so that the bundled apps are distinguishable from the apps that were installed or updated by the administrator. Existing installations will not change but, generally, this separation is recommended in all scenarios, as it makes upgrading easier and less error-prone.

- **Update the occ files:scan --group and --groups options.** The occ files:scan command is used to scan resources on the storage and make them available in ownCloud. While previously it could only be used for all or single users and groups of users, you can now also execute it for groups where the group name contains a comma.
- **Allow administrators to enable/disable medial search for users and groups.** Medial search is used to get search results when typing keys within a search term in autocomplete fields (e.g. when typing "*ter*" you'll find "Peter"). Depending on the configuration of available search terms (e.g., attributes from LDAP), search results can deliver better results without medial search. For these reasons medial search can now be enabled/disabled for user ('accounts.enable_medial_search') and group ('groups.enable_medial_search') search. See config.sample.php for more information.
- Added a new occ command, background:queue:execute, for running cron jobs manually.
- Added two new occ background:queue commands: status and delete.
  - status lists the current background job queue status
  - delete removes a single background job, identified by its id.

Solved Known Issues

- Fixed public link share default expiration behavior #34971. Previously, when a default expiration date for public links had been set by an administrator (without enforcement option), the default value has been applied upon link creation even when a user removed it. The only way to create a link without expiration date was to subsequently edit it and remove the expiration date. This has been fixed to work as expected.
- Better support for international email addresses after Swiftmailer update #34759
- Improved speed of apps list settings page by caching integrity check results #34584
- Improved upgrade speed when migrating avatars from oC < 10 #34592
- Improved performance and memory usage of account sync service #34546
- Store quota overrides in the oc_preferences table #34467. In former versions, functionality has been introduced to preserve quota values either imported via LDAP attributes against manual changes by the administrator in ownCloud user management, or via the provisioning API. This functionality works again properly. If you sync accounts from LDAP and have a quota attribute specified in LDAP, each user:sync run will set the quota values to the ones from LDAP, no matter if they were changed manually.
- Images are again properly rotated now based on EXIF rotation, also affects gallery app #34356
- An exception is logged when a background job class is not found - 34723

Known Issues



This section will be updated if further issues become known.

- Server 10.2 accidentally changes the location of user avatars on the storage from

data/avatars/.. to data/.., making existing avatars unavailable and storing uploaded avatar images in the wrong location. The next release will correct the behavior.

- The HTML email that confirms a successful password change is rendered in plain text. Please apply this patch to fix the issue.
- WebDAV Locks: When a file in a folder is locked, exclusively locking the parent folder currently still works ("conflicting lock"; divergent from RFC 4918))

For Developers

- It is now possible for apps to specify extra permissions for shares #34951
- Add before-after share link auth events #34399
- Add events for user preference changes #34820
- Added CORS headers for many existing API calls, required for Phoenix #34476
- Remove classes that were deprecated since OC 8.0.0: OCP\Config, OCP\PERMISSION_XXX, OCP\Template #34927
- A capability has been added to the Capabilities API to allow clients to check whether the server supports the details parameter for private links, e.g., as a direct link to a resource's sharing or versions tab in the web interface #35104

Changes in 10.1.1

ownCloud Server 10.1.1 is a hotfix follow-up release that takes care of an issue with loading updated apps. Instead of updating the app versions to their new values in the database, the old version value is written causing the process to repeat with every request.

This issue can cause high load on the database, especially in large installations. If you have already upgraded to 10.1.0, we strongly recommend upgrading to 10.1.1. You can expect minimal downtime for the upgrade to this patch release.

Apart from this patch release, please consider the ownCloud Server 10.1.0 release notes.

Changes in 10.1.0

Dear ownCloud administrator, please find below the changes and known issues in ownCloud Server 10.1 that need your attention. You can also read the full ownCloud Server changelog for further details on what has changed.

Semantic Versioning

Starting with this release, ownCloud Server and the app ecosystem will follow the principles of Semantic Versioning. This step was taken to benefit operators by clearly indicating the contents and upgrade procedures of new releases via version numbers. Practically, the versioning scheme will follow the "Major.Minor.Patch" (or "Breaking.Feature.Fix") format. App developers need to re-release their apps to make them compatible with the new version. For details, please refer to this blog post.

Change Management for Server Updates

occ upgrade pulls app updates from the ownCloud Marketplace to make sure that not only the Server itself but also the installed apps are kept up-to-date. In line with the new versioning principles occ upgrade as well as the Market App now make a difference between major and minor app updates. Practically, this means that during a minor Server upgrade only new minor app versions will be installed. This is to make sure that apps with breaking changes will not be automatically installed when upgrading the Server. The --major option for occ upgrade and occ market:install

provides the means for administrators to force installing new major app versions. Additionally, the Market App now includes a version picker to enable administrators to choose which version of an app they want to install or upgrade to.

MS Office Online Server Compatibility

Version 10.1 delivers all the prerequisites to be compatible with the Microsoft Office Online Server Integration (WOPI) that is about to become available. This enables providers to integrate ownCloud Server with Microsoft's Office Online Server which brings users the benefits of working on Office documents in the browser as well as collaboratively with other users. The integration will work with MS Office Online Server (on-premise) out-of-the-box. We kindly ask you to get in touch with us if you want to make use of the Office 365 (cloud) version of Office Online.

WebDAV Locks

ownCloud Server 10.1 introduces WebDAV Locks that allow clients to lock and unlock resources to prevent other users from making changes. The feature has been implemented as a prerequisite for manual file locking and MS Office Online Server compatibility. In the current state, file locking is only available via API. Users can recognize locked files via the "lock" icon in the file list. Additionally a lock owner (the user who locked the file) can manually unlock them via the "Locks" tab in the right sidebar. The "Locks" tab will only appear for files that have active locks.

Foreign Keys in Database

Please note that foreign keys have been added with the :WebDAV Locks feature. This is the first time ownCloud implements foreign keys.

How Does This Affect Each Database

MySQL

MySQL supports foreign keys. They are enabled by default.

MariaDB

MariaDB supports foreign keys. They are enabled by default.

PostgreSQL

PostgreSQL supports foreign keys. They are enabled by default.

SQLite

Foreign keys are, by default, disabled in SQLite. You must ensure that foreign keys are enabled in your SQLite installation. Here is what the current documentation says about enabling foreign key support:

> To enable foreign key support, the library must be compiled with neither SQLITE_OMIT_FOREIGN_KEY or SQLITE_OMIT_TRIGGER defined. If SQLITE_OMIT_TRIGGER is defined but SQLITE_OMIT_FOREIGN_KEY is not, then SQLite behaves as it did prior to version 3.6.19 (2009-10-14) - foreign key definitions are parsed and may be queried using PRAGMA foreign_key_list, but foreign key constraints are not enforced. The PRAGMA foreign_keys command is a no-op in this configuration. If SQLITE_OMIT_FOREIGN_KEY is defined, then foreign key definitions cannot even be parsed (attempting to specify a foreign key definition is a syntax error).
>
> — Enabling Foreign Key Support



SQLite is not recommended for production deployments.

Oracle

Oracle supports foreign keys. They are enabled by default.

Federation: Compliance with proposed OpenCloudMesh 1.0 specification

Federation enables instances of ownCloud and other supporting platforms to exchange information. It allows users to share data across installations building a worldwide collaboration network of decentralized nodes - each under the full control of it's provider. Together with the other vendors the underlying OpenCloudMesh API specification has been shifted to a new level to clean up the interface, improve its stability and to set the foundation for future feature improvements. ownCloud Server 10.1 is compliant with the new specification proposal. The introduction of the new specification does not involve changes in functionality for users.

New Collaborative Tags Scope: Static Tags

Version 10.1 comes with a new scope for Collaborative Tags called "Static Tags". In addition to the other tag scopes, these tags are intended to be supplied by administrators and linked with policies in the File Firewall, Document Classification or Workflows, for example. Every user will be able to see these tags assigned to files but only users in specified groups have the permission to assign or unassign them. This makes it possible to equip certain users with the means to impose pre-defined policies upon files. To create such tags administrators need to use the Collaborative Tags Management extension.

Other notable changes

- The user/group deletion in the users page now has a confirmation dialog to prevent unintentional user deletion
- The default public link share name has been changed to be "Public link" instead of formerly the file or folder's name
- Allow loading JSON files in setups with pretty URLs. Please check that the .htaccess file has updated automatically. If not, see https://github.com/owncloud/core/pull/32718/files for the required change.

Solved known issues

- LDAP users can upload avatars again #33369

- Versions list performance improvements #33291
- Improved compatibility with third party WebDAV applications (fixed PROPFIND with depth infinity requests through Sabre update) #28341
- Fixed occ encrypt-all command to not attempt re-encrypting already encrypted files #33206

Known issues

- WebDAV Locks: When a file in a folder is locked, exclusively locking the parent folder currently still works ("conflicting lock"; divergent from RFC 4918))

For developers

- Added "getBucket" method to HomeObjectStore to fix S3 issue #33513
- Public JS utility function for email validation #33699
- If only the patch level of an app's version changes no migrations will run when updating #33218
- Deprecated Sharing 1.0 PHP APIs which will be removed in ownCloud 11 #33220
- Add "uid" argument to Symfony login events for consistency #33470

Changes in 10.0.10

Dear ownCloud administrator, please find below the changes and known issues in ownCloud Server 10.0.10 that need your attention. You can also read the full ownCloud Server changelog for further details on what has changed.

Official PHP 7.2 Support

After announcing the future deprecation of PHP 5.6 and 7.0 with the 10.0.8 release, ownCloud Server now follows up by officially adding PHP 7.2 support. The Server Core and all apps maintained by ownCloud have received a full QA cycle and are proven to work reliably with PHP 7.2. ownCloud Server is also being prepared for PHP 7.3, which is scheduled to become available by the end of 2018.

If you are still using versions 5.6 or 7.0, please plan an upgrade to 7.2 soon. See the system requirements in the ownCloud Documentation.



With PHP 7.2 some extensions have changed. If you have not yet upgraded, you need to install php-openssl. See #30337 for more information.

New Local User Creation Flow

In previous versions, administrators created local users by entering a username and a password. In many cases this is undesirable, as administrators set the password for new users and need to provide it via a second communication channel. For this reason the local user creation flow has been changed to expect a username and an email address, which will be used to send an activation link to new users.

This way user creation is easier and more secure as new users are informed automatically and can choose a password in self-service. For cases where administrators want to set the initial password, it's possible to deviate from the default by setting the option "**Set a password for new users**" on the bottom left settings cog. The former option "**Send email to new users**" has been removed, as this change made it obsolete.

HTTP API for Search

ownCloud Server 10.0.10 introduces an HTTP API for search functionality. It enables the use of search terms to query the server and the delivery of search results via HTTP (WebDAV). In upcoming releases, ownCloud clients will make use of it to search content on the server, without the need to have them available locally.

In combination with the Full-Text Search integration, which is soon to be released as an ownCloud Server extension (Community Edition), HTTP API for Search will boost usability and productivity for users. For example, they will be able to search through all the content which they store in their account and quickly find files on their smartphones.

Native Brute-Force Protection

Together with the new server version, another security-enhancing extension is available, Brute Force Protection. This extension is tasked with preventing attackers from guessing user passwords (brute-force attack) by delaying subsequent failed login attempts for a user account from the same IP address.

While in the past similar functionality was only achievable via third party applications, such as **Fail2Ban**, this extension provides the functionality natively, configurable by ownCloud administrators on the Security settings section.

The new extension supersedes the former Security extension together with the new Password Policy extension, which has been released with ownCloud Server 10.0.9. This community-contributed extension is well-tested, but out of ownCloud's general support scope. However, individual support can be obtained on request.

Improved Reliability for Uploads Via Web Interface on Unreliable Connections

The reliability of the file upload feature in the ownCloud web interface has been improved. When uploading larger amounts of data on unreliable connections (e.g., on the train or with mobile data) you have to deal with interruptions and timeouts, which in the past required users to restart stalled uploads from the beginning in the worst case.

On top of ownCloud's chunking mechanism, which splits large files into pieces and uploads them separately, there's new logic that takes care of retrying stalled chunks. With this, uploads can now continue from the point they froze when a connection becomes available again.

New Option to Prevent Sharing With Specific System Groups

System groups in ownCloud can have many purposes. They can be used for sharing with many users at once, for feature and access restrictions, or for storage mounts to specific users - just to name a few. In some cases, especially in larger deployments, it's undesirable that groups which are used for other purposes are also available for sharing. To prevent users from sharing with such groups, administrators can now blacklist the respective system groups using the option "**Exclude groups from receiving shares**" in the administration settings "**Sharing**" section.

New Options for the occ Command to Reset User Passwords

The occ command user:resetpassword allows system administrators to reset or change user passwords. It has been extended to provide the additional options --send-email and --output-link, which can be used to send a password reset link to the user via mail and output the password reset link to the command line, respectively. This change is in line with the new local user creation flow, which is explained above, and can also be used for further processing with scripts. See the ownCloud Documentation and the

--help option for more information.

New Default Minimum Supported Desktop Client Version

To ensure clean and reliable operation of the ownCloud platform it is important to stay up-to-date with the latest releases for the server as well as the clients. To take care of compatibility between the server and desktop clients, the minimum version the server will accept connections from has been raised to version 2.3.3.

While it's recommended to keep up with later versions, this is the new default value. It can be changed by altering the config.php parameter 'minimum.supported.desktop.version' ⇒ '2.3.3', if absolutely necessary.

New Option to Configure the Language of Mail Notifications for Public Links

Usually ownCloud renders mail notifications in the language of the recipients, when they are known. For the recently improved feature to send public links with a personal note directly from the user interface, the recipients' language can't be determined automatically, it just knows the recipients' mail addresses.

ownCloud therefore uses the language of the user who sent the notification, which can have the drawback that recipients can't understand them. This is still the default behavior but administrators can now change it via a dropdown menu **"Language used for public mail notifications for shared files"** in the settings **"Sharing"** section.

Theming Changes

Mail templates for share notifications do not strip line breaks from the personal note anymore. This affects the HTML (core/templates/mail.php) and plain text (core/templates/altmail.php) mail templates. The default templates shipped with ownCloud Server 10.0.10 have been modified to accommodate these changes. If your custom theme overrides these templates, you have to follow up with the changes:

- Replace the following line of the HTML template p($l→t("Personal note from the sender: %s.", [$*['personal_note']])); with print_unescaped($l→t("Personal note from the sender:
 %s.", $['personal_note']));.
- Replace the following line of the plain text template print_unescaped($l→t("Personal note from the sender: %s.", [$*['personal_note']])); with print_unescaped($l→t("Personal note from the sender: \n %s.", $['personal_note']));.

Other Notable Changes

- Allow automated SSL certificate verifications for CAs other than Let's Encrypt. See #31858 for further details.
- "/" and "%" are now valid characters in group names. See #31109 for further details.
- New audit events for login action with token or Apache. See #31985 for further details.
- Log entries for exceeding user quota: Loglevel changed to "debug" (Insufficient storage exception is now logged with "debug" log level).
- The app for embedding external sites to the app launcher (**"external"**) now supports icons that originate from theme apps.
- The occ command to deactivate storage encryption (occ encryption:decrypt-all) has received stability improvements and can now read the required recovery key from an environment variable which is very helpful for a scripted per-user decryption process.

Solved Known Issues

ownCloud Server 10.0.10 takes care of 10.0.9 known issues and provides remedies for several others:

- The Password Policy extension now works with two- or multi-factor authentication extensions. See #32058 for further details.
- The Versions feature now works also when the Comments app is disabled. See #32208 for further details.
- E-mail addresses with subdomains with hyphens are now also accepted for public link emails. See #32281 for further details.
- Allow null in "Origin" header for third party clients that send it with WebDAV. See #32189 for further details.
- Properly log failed message when token based authentication is enforced (Fail2Ban). See #31948 for further details.
- Deleting a user now also properly deletes their external storages and storage assignations. See #32069 for further details.
- Lockout issues with wrong passwords for Windows Network Drives are mitigated: Fixed mount config in front-end to only load once to avoid side effects. See #32095 for further details.
- Fixed update issue related to oc_jobs when automatically enabling market app to assist for update in OC 10. See #32573 for further details.
- Fixed missing migrations in files_sharing app and add indices to improve performance. See #32562 for further details.
- Fixed issue with spam filters when sending public link emails. See #32542 for further details.

Known Issues

Currently there are no known issues with ownCloud Server 10.0.10. This section will be updated in the case that issues become known.

For Developers

- Search API for files using WebDAV REPORT and an underlying search provider. See #31946 and #32328 for further details.
- Add information whether user can share to capabilities API. See #31824 for further details.
- Hook loadAdditionalScripts now also available for public link page. See #31944 for further details.
- Added URL parameter to files app which opens a specific sidebar tab. See #32202 for further details.
- Allow slashes in generated resource routes in app framework. See #31939 for further details.
- The app for embedding external sites to the app launcher ("**external**") has been moved to a separate repository. It is still bundled with ownCloud Server releases and can be used normally.

Changes in 10.0.9

Dear ownCloud administrator, please find below the changes and known issues in ownCloud Server 10.0.9 that need your attention. You can also read the full ownCloud Server changelog for further details on what has changed.

New Features

Pending Shares

ownCloud Server 10.0.9 introduces new features to close usability gaps and to give users more control over incoming shares. Previously, shared contents would appear, unannounced, in the receiving user's file hierarchy, and clients would start synchronizing.

Incoming shares can now have a pending state, offering the ability to accept or decline (as known from federated sharing). We anticipate that this will provide a better user experience.

In addition, the recently introduced notifications framework is being used to inform users via mail.

The bell icon in the web interface and the ownCloud Desktop Client can additionally be used to take action. To switch to the new behavior administrators need to disable the configuration option Automatically accept new incoming local user shares in the *Sharing* settings section. By default the option will be enabled to preserve the known behavior.

Mail notifications do not, currently, support asynchronous batch processing. For this reason, ownCloud will send notification emails directly when initiating shares between users. Due to this limitation, sharing with large groups (> 50 users) can take some time and might cause load peaks. When operating installations with large groups, it is, therefore, not yet recommended to enable the feature.

Overview of pending & rejected shares

In addition to the "*Pending Shares*" feature, ownCloud Server now provides the means to view "*accepted*", "*pending*" and **"rejected**" incoming shares. Leveraging the "*Shared with you*" filter in the left sidebar of the files view users can now list all incoming shares, their respective states and have the ability to switch between the states easily.

This improvement not only empowers users to accept rejected shares subsequently but also to restore shares that have been unshared before without requiring the owner to share it again.

Password history and expiration

To prepare ownCloud Server for new capabilities in the authentication process, we have introduced an authentication middleware, and a new major version of the Password Policy extension is now available.

The Authentication Middleware

It:

- Offers a defined way of inserting mandatory functionality between user authentication and user account access. For example, forcing users to accept legal agreements.
- Affords the ability to interact with the user during the login process, such as retrieving user details like their email address.



The authentication middleware is currently focused on offering new features for the Password Policy extension.

The Password Policy Extension

The Password Policy Extension has got a new major release and has been relicensed (OCL ⇒ GPLv2) to be available for community and standard subscription users as well. It now supports password expiration and history policies for user accounts.



These features don’t apply to users imported from LDAP or other backends but only for local users created by administrators or the Guests extension.

Imposing password expiration and history policies enhances security for a number of reasons. For example, by forcing users to choose a new password, they can be prevented from using one or more of their previous passwords. In doing this, it encourages them to not use a previous password, which may be known to attackers.

Two further examples are manually expiring passwords and configuring the number of days that have to pass since the last change before the password expires. These help ensure that users change their passwords on a semi-regular basis, making them harder to crack.

However, we encourage administrators to always consider the implication of their password policies, so that they strike an appropriate balance between security and usability. For example, a high frequency of password changes, for instance, might increase security but could also decrease user satisfaction.

To help ensure a good user experience it is possible to configure:

- Email notifications.
- Internal notifications (they appear on the web interface and clients).
- The password history count.
- The days before reminder notification are sent.

Users will always be informed when passwords have expired.



Although the above two password practices are discouraged by NIST, ownCloud is now fully compliant with common password guidelines in enterprise scenarios.



When users employ tokens for client authentication, which can be configured on the user settings page ("App passwords"), those are not affected from password policies.



When imposing password expiration policies on an existing installation it is necessary to take some further actions. Please consult `the ownCloud documentation`_ for guidance.

Technology preview for new S3 Objectstore implementation

ownCloud Server 10.0.9 comes with the prerequisites to be ready for the new S3 Objectstore implementation "*files_primary_s3*", which will massively improve performance, reliability and protocol-related capabilities. The new extension is available as a technology preview via the ownCloud Marketplace and will supersede the current Objectstore extension.

It has received extensive testing and is in very good shape. However, there is no out-of-the-box migration from the current *Objectstore* to *files_primary_s3* as this will require individual guidance.

Due to changes to the Versioning API, the ownCloud Ransomware Protection is not yet compatible with *files_primary_s3*. For now the Objectstore extension will continue to work as usual. Once the new implementation leaves the technology preview state and migrations have been taken care of, the current implementation will be deprecated.

SWIFT Objectstore Deprecation

As the markets are moving in the direction of the S3 protocol to communicate with object storages, ownCloud will follow this path with a clear focus. To do this, it will be a necessity to deprecate object storage via the OpenStack SWIFT protocol.

The extension will still be available as part of ownCloud Server, but it will neither be maintained nor developed any further by ownCloud, and support will be discontinued. Please make sure to move to the S3 protocol to use object storage as primary storage with future ownCloud Server versions.

New options to display Imprint and Privacy Policy

To enable GDPR and legal compliance in various jurisdictions for ownCloud providers, it is now possible to specify links to Imprint and Privacy Policy:

- In the "*General*" Administration settings section
- Via the following OCC commands:
  - {occ-command-example-prefix} config:app:set core legal.imprint_url <link>
  - {occ-command-example-prefix} config:app:set core legal.privacy_policy_url <link>

These links can be displayed on all pages of the ownCloud web interface and in the footer of mail notifications. When using one of the default themes provided by ownCloud, as well as the default mail templates, configured links will be automatically included.

For customized themes or mail templates, actions are required to include the links. These are:

Add the following at the end of each HTML template to add the footer:

```
<?php print_unescaped($this→inc('html.mail.footer', ['app' ⇒ 'core'])); ?>
```

Add the following at the end of each plain text template to add the footer:

```
<?php print_unescaped($this→inc('plain.mail.footer', ['app' ⇒ 'core'])); ?>
```

In a custom theme, change getShortFooter and getLongFooter in defaults.php without links to include the links

Changed behavior of "Exclude groups from sharing" option

The option "*Exclude groups from sharing*", in the administration settings "*Sharing*" section, enables administrators to exclude groups of users from the ability to initiate file shares. In previous versions this restriction only applied to users who were members of exactly these groups (membership of one or more non-excluded groups bypassed the restriction).

This behavior has been changed to be both more restrictive and to better cover the expectations of administrators. With ownCloud Server 10.0.9, it will apply to all users who are members of at least one of the excluded groups.

Changes to the sharing autocomplete mechanism

In ownCloud Server 10.0.8, the value for minimum characters to trigger the sharing autocomplete mechanism <min-chars-for-sharing-autocomplete-label> has been made configurable and set to 4 by default. As this security-enhancing change came at the expense of usability, and might only be required in special scenarios, the default value has been reverted to 2.

For increased security requirements, the config.php option 'user.search_min_length' ⇒ 2 can be adjusted. To further improve usability, a hint has been added to inform users about the required character count, to get suggestions.

Improvements for *occ user:list*

To improve the usability of the occ user:list command, the output has been made configurable by using the -a option, for including certain attributes. This change has mainly been introduced to facilitate automation tasks. Check the --help option for more information.

Additional events for audit logging

New events are available for audit logging, among others. These include:

- Changes in user specific settings
- Sending public links via mail; and
- Accepting and rejecting shares

When logs are forwarded to external analyzers, like Splunk, administrators can check to add the new events. The latest version of the Auditing extension (admin_audit) is required.

Theming improvements and changes

- HTML templates for lost password mails have been added. This is important in case a custom theme is used and it needs manual adjustments.
- The mail notifications framework, introduced with ownCloud Server 10.0.8 <new-mail-notifications-feature-label>, has been extended to provide a basic framework and notification structure, which can be used by ownCloud features and third party extensions. To support this, mail template wording and structure have been updated. Please review the templates in apps/notifications/templates/mail/ to align them with your needs.
- Mail templates can now include a footer for HTML (core/templates/html.mail.footer.php) and plain text mails (core/templates/plain.mail.footer.php). The default templates shipped with ownCloud Server 10.0.9 contain the respective references. For customized mail templates, it is necessary to manually add the references. To do so:
- Add the following at the end of each HTML template: :

```
<?php print_unescaped($this->inc('html.mail.footer', ['app' => 'core'])); ?>
```

- Add the following at the end of each plain text template: :

```
<?php print_unescaped($this->inc('plain.mail.footer', ['app' => 'core'])); ?>
```

- The ownCloud example theme (theme-example), which can be used as a solid base

to create custom themes, is no longer bundled with ownCloud Server. It now lives in it's own repository on GitHub.

Solved known issues

ownCloud Server 10.0.9 takes care of 10.0.8 known issues, and provides remedy for several others:

- Issues with multiple theme apps and the Mail Template Editor #31478
- OCC command to transfer data between users (occ transfer:ownership) works as expected again. Previously, public link shares were not transferred. See #31176 for further details.
- OCC commands to encrypt (occ encryption:encrypt-all) and decrypt (occ encryption:decrypt-all) user data work correctly again. Previously, shares might have been lost during the encryption process. See #31600 and #31590 for further details.
- Files larger than 10 MB can now properly be uploaded by guest users. See #31596 for further details.
- Issues with public link dialog when collaborative tags app is disabled has been resolved. See #31581 for further details.
- Enabling/disabling of users by group administrators in the web UI works again. See #31489 for further details.
- Issues with file upload using Microsoft EDGE are now circumvented (hard memory limit of 5 GB causing uploads to fail randomly as garbage collection for file chunks did not work properly). See #31884 for further details.

Known issues

The new Password Policy feature "Password Expiration":

- Does not work together with Multi-Factor Authentication (e.g. twofactor_totp, twofactor_privacyidea). Please do not deploy expiration policies yet when having Two- or Multi-Factor Authentication extensions in place. This issue will be solved with the next ownCloud Server release. See #32059 for more information.
- The new Password Policy feature "Password Expiration" includes an **occ** command to manually force password expiration. Please run it directly after imposing expiration policies on an instance with existing users. Currently the command will only work when the policy **X days until user password expires** has been enabled. This might be confusing and will be solved with the next release of the extension. See #66 for more information.

For developers

- The symfony event for logging has been extended to include the original exception when applicable: #31623
- Added Symfony event for whenever user settings are changed #31266
- Added Symfony event for whenever a public link share is sent by email #31632
- Added Symfony event for whenever local shares are accepted or rejected #31702
- Added public WebDAV API for versions using a new meta DAV endpoint #31729 #29637
- Added support for retrieving file previews using WebDAV endpoint #29319 #30192

Changes in 10.0.8

Dear ownCloud administrator, please find below the changes and known issues in ownCloud Server 10.0.8 that need your attention. You can also read the full ownCloud Server changelog for further details on what has changed.

PHP 5.6 deprecation

PHP 5.6/7.0 active support has ended on January 19th 2017 / December 3rd 2017 and security support will be dropped by the end of 2018. Many libraries used by ownCloud (including the QA-Suite *PHPUnit*) will therefore not be maintained actively anymore which forces ownCloud to drop support in one of the next minor server versions as well. Please make sure to upgrade to PHP 7.1 as soon as possible. See the system requirements in the ownCloud documentation.

Personal note for public link mail notification

One of the usability enhancements of ownCloud Server 10.0.8 is the possibility for users to add a personal note when sending public links via mail. When using customized mail templates it is necessary to either adapt the shipped original template to the customizations or to add the code block for the personal note to customized templates in order to display the personal note in the mail notifications.

New mail notifications feature

ownCloud Server 10.0.8 introduces a new extensible notification framework. Apart from technical changes under the hood the Notifications app can now also send mails for all notifications that previously were only displayed within the web interfaces (notification bell) or on the Desktop client (notifications API) like incoming federated share or Custom Group notifications, for example. In the "*General*" settings section users can configure whether they want to receive mails for all notifications, only for those that require an action or decide not to get notifications via mail (by default users will only receive notifications when an action is required).

LDAP-related improvements

- When disabling or deleting user accounts in LDAP, the administrator can choose to either *delete* or *disable* respective accounts in ownCloud when executing occ user:sync (-m, --missing-account-action=MISSING-ACCOUNT-ACTION). User accounts that are disabled in ownCloud can now be re-enabled automatically when running occ user:sync if they are enabled in LDAP. When this behavior is desired administrators just need to add the -r, --re-enable option to their cron jobs or when manually executing occ user:sync.
- Furthermore it is now possible to execute occ user:sync only for **single** (-u, --uid=UID) or **seen** (-s, --seenOnly) users (users that are present in the database and have logged in at least once). These new options provide more granularity for administrators in terms of managing occ user:sync performance.
- Another notable change in behavior of occ user:sync is that administrators now have to explicitly specify the option -c, --showCount to display the number of users to be synchronized.

New events for audit logging

New events have been added to be used for audit logging, among others. These include *configuration changes* by administrators and users, *file comments* (*add/edit/delete*) and *updating existing public links*. When logs are forwarded to external analyzers like Splunk, administrators can check to add the new events. The latest version of the Auditing extension (*admin_audit*) is required.

New command to verify and repair file checksums

With ownCloud 10 file integrity checking by computing and matching checksums has been introduced to ensure that transferred files arrive at their target in the exact state as their origin. In some rare cases wrong checksums can be written to the database leading to synchronization issues with e.g. the Desktop Client. To mitigate such situations a new command occ files:checksums:verify has been introduced. The command recalculates checksums either for all files of a user or for files within a specified path, and compares them with the values in the database. Naturally the command also offers an option to repair incorrect checksum values (-r, --repair). Please check the available options by executing occ files:checksums:verify --help. Note: Executing this command might take some time depending on the file count.

New config setting to specify minimum characters for sharing autocomplete

For security reasons the default value for minimum characters to trigger the sharing autocomplete mechanism has been set to "4" (previously it was set to "2"). This is to prevent people from easily downloading lots of email addresses or user names by requesting their first letters through the API. As it is a trade-off between security and usability for some scenarios this high security level might not be desirable. Therefore the value now is configurable via the *config.php* option 'user.search_min_length' ⇒ 4,. Please check which value fits your needs best.

New option to granularly configure public link password enforcement

With ownCloud 10 the File Drop feature has been merged with public link permissions. This kind of public link does not give recipients access to any content, but it gives them the possibility to drop files. As a result, it might not always be desirable to enforce password protection for such shares. Given that, passwords for public links can now be enforced based on permissions (*read-only, read & write, upload only/File Drop*). Please check the administration settings `*Sharing*` section and configure as desired.

New option to exclude apps from integrity check

By verifying signature files the *integrity check* ensures that the code running in an ownCloud instance has not been altered by third parties. Naturally this check can only be successful for code that has been obtained from official ownCloud sources. When providing custom apps (like theme apps) that do not have a signature, the integrity check will fail and notify the administrator. These apps can now be excluded from the *integrity check* by using the *config.php* option 'integrity.ignore.missing.app.signature' ⇒ ['app_id1', 'app_id2', 'app_id3'],. See *config.sample.php* for more information.

New occ command to modify user details

It is now possible to modify user details like display names or mail addresses via the command occ user:modify. Please append --help for more information.

occ files:scan can now be executed for groups

Apart from using the occ files:scan command for *single users* and *whole instances* it can now be executed for *groups* using -g, --groups=GROUPS. Please append --help for more information.

New configurable default format for syslog

When using syslog as the log type ('log_type' ⇒ 'syslog', in *config.php*) the default format has been changed to include *request IDs* for easier debugging. Additionally the log format has been made configurable using 'log.syslog.format' in *config.php*. If you require a certain log format, please check the new format and *config.sample.php* on

how to change it.

New config option to enable fallback to HTTP for federated shares

For security reasons federated sharing (sharing between different ownCloud instances) strictly requires HTTPS (SSL/TLS). When this behavior is undesired the insecure fallback to HTTP needs to be enabled explicitly by setting 'sharing.federation.allowHttpFallback' ⇒ false, to true in *config.php*.

Migration related to auth_tokens (app passwords)

Upgrading to 10.0.8 includes migrations related to *auth_tokens* (*app passwords*). When users have created *app passwords* as separate passwords for their clients the upgrade duration will increase depending on user count. Please consider this when planning the upgrade.

Changed behavior of e-mail autocomplete for public link share dialog

When the "*Sharing*" settings option Allow users to send mail notifications for shared files for public links is enabled, users can send public links via mail from within the web interface. The behavior of the autocomplete when entering mail addresses in the public link share dialog has been changed. Previously the autocomplete queried for local users, users from federated address books and contacts from CardDAV/Contacts App. As public links are not intended for sharing between ownCloud users (local/federated), those have been removed. Contacts synchronized via CardDAV or created in the Contacts app will still appear as suggestions.

Notifications sent by *occ* can now include links

The command occ notifications:generate can be used to send notifications to individual users or groups. With 10.0.8 it is also capable of including links to such notifications using the -l, --link=LINK option. Please append --help for more information. There is also Announcement center to conduct such tasks from the web interface but it is currently limited to send notifications to all users. For now administrators can use the `occ command if more granularity is required.

Global option for CORS domains

For security reasons ownCloud has a *Same-Origin-Policy* that prevents requests to ownCloud resources from other domains than the domain the backend server is hosted on. If ownCloud resources should be accessible from other domains, e.g. for a separate web frontend operated on a different domain, administrators can now globally specify policy exceptions via *CORS (Cross-Origin Resource Sharing)* using 'cors.allowed-domains' in *config.php*. Please check *config.sample.php* for more information.

Mail Template Editor is now unbundled

The Mail Template Editor has been unbundled from the default apps and is not shipped with the Server anymore. When upgrading ownCloud will try to automatically install the latest version from the ownCloud Marketplace in case the app was installed before.

If this is not possible (e.g. no internet connection or clustered setup) you will either need to disable the app (occ app:disable templateeditor) or download and install it manually.

Solved known issues

- Bogus Login failed log entries have been removed (see 10.0.7 known issues)
- The *Provisioning API* can now properly set default or zero quota

- User quota settings can be queried through *Provisioning API*
- A regression preventing a user from setting their e-mail address in the settings page has been fixed
- File deletion as a guest user works correctly (trash bin permissions are checked correctly)

Known issues

- Issues with multiple theme apps and Mail Template Editor

As of ownCloud Server 10.0.5 it is only possible to have one theme app enabled simultaneously. When a theme app is enabled and the administrator attempts to enable a second one this will result in an error. However, when also having the Mail Template Editor enabled in this scenario the administrators "*General*" settings section will be displayed incorrectly. As a remedy administrators can either uninstall the second theme app or disable the Mail Template Editor app.

- occ transfer:ownership does not transfer public link shares if they were created by the target user (reshare).

For developers

- The global JS variable oc_current_user was removed. Please use the public method OC.getCurrentUser() instead.
- Lots of new Symfony events have been added for various user actions, see changelog for details, or the documentation ticket.
- When requesting a private link there is a new HTTP response header Webdav-Location that contains the WebDAV path to the requested file while the Location still points at the frontend URL for viewing the file.

Changes in 10.0.7

ownCloud Server 10.0.7 is a hotfix follow-up release that takes care of an issue regarding OAuth authentication.

Please consider the ownCloud Server 10.0.5 release notes.

Known issues

- When using application passwords, log entries related to Login Failed will appear and can be ignored. For people using fail2ban or other account locking tools based on log parsing, please apply this patch with patch -p1 < 50c78a4bf4c2ab4194f40111b8a34b7e9cc17a14.patch (original pull request here).

Changes in 10.0.6

ownCloud Server 10.0.6 is a hotfix follow-up release that takes care of an issue during the build process (https://github.com/owncloud/core/pull/30265). Please consider the ownCloud Server 10.0.5 release notes.

Changes in 10.0.5

Dear ownCloud administrator, please find below the changes and known issues in ownCloud Server 10.0.5 that need your attention. You can also read the full ownCloud Server changelog for further details on what has changed.

Technology preview for PHP 7.2 support

ownCloud catches up with new web technologies. This has mainly been introduced for the open-source community to test and give feedback. PHP 7.2 is not yet supported nor recommended for production scenarios. ownCloud is going to fully support PHP 7.2 with the next major release.

php-intl now is a hard requirement

Please make sure to have the PHP extension installed before upgrading.

Changed: Only allow a single active theme app

The theming behavior has been changed so that only a single theme can be active concurrently. This change ensures that themes can not interfere in any way (e.g., override default theming in an arbitrary order). Please make sure to have the desired theme enabled after upgrading.

Removed old Dropbox external storage backend (Dropbox API v1)

Please switch to the new *External Storage: Dropbox* app with Dropbox API v2 support to continue providing Dropbox external storages to your users.

Fixed: Only set CORS headers on WebDAV endpoint when Origin header is specified

ownCloud Server 10.0.4 known issue is resolved.

Fixes and improvements for the Mail Template Editor

- Known issues are resolved: Mail Template Editor works again, got support for app themes and additional templates were added for customization.
- Mail Template Editor is still bundled with ownCloud Server but will soon be released as a separate app to ownCloud Marketplace.
- Changelog: https://github.com/owncloud/templateeditor/blob/master/CHANGELOG.md#02---2018-02-28

Known issues

- When using application passwords, log entries related to Login Failed will appear, please upgrade to 10.0.7 and check the fix mentionned in its release notes.

Changes in 10.0.4

Dear ownCloud administrator, please find below the changes and known issues in ownCloud Server 10.0.4 that need your attention. You can also read the full ownCloud Server 10.0.4 changelog for further details on what has changed.

More granular sharing restrictions

The "*Restrict users to only share with users in their groups*" option, in the Sharing settings, restricts users to only share with groups which they are a member of, while simultaneously prohibiting sharing with single users that do not belong to any of the users' groups.

To make this more granular, we split this option into two parts and added "*Restrict users to only share with groups they are member of*", which differentiates between users and groups. Doing so makes it possible to restrict users from sharing with all users of an installation, limiting them to only being able to share with groups which they are a member of, and vice versa.

Configurable solution for indistinguishable user display names

The ownCloud sharing dialog displays users according to their display name. As users can choose their display name in self-service (which can be disabled in config.php) and display names are not unique, it is possible that a user can't distinguish sharing results.

To cover this case the displayed user identifiers are now configurable. In the Sharing settings administrators can now configure the display of either mail addresses or user ids.

Added occ files:scan repair mode to repair filecache inconsistencies

We recommend to use this command when directed to do so in the upgrade process. Please refer to the occ command's files:scan –repair documentation for more information.

Detailed mode for occ security:routes

Administrators can use the output of this command when using a network firewall, to check the appropriateness of configured rules or to get assistance when setting up.

Added mode of operations to differentiate between single-instance or clustered setup

As ownCloud needs to behave differently when operating in a clustered setup versus a single instance setup, the new config.php option operation.mode has been added. It can take one of two values: single-instance and clustered-instance. For example: 'operation.mode' ⇒ 'clustered-instance',.

Currently the Market App (ownCloud Marketplace integration) does not support clustered setups and can do harm when used for installing or updating apps. The new config setting prevents this and other actions that are undesired in cluster mode.

When operating in a clustered setup, it is mandatory to set this option. Please check the config_sample_php_parameters documentation for more information.

Added occ dav:cleanup-chunks command to clean up expired uploads

When file uploads are interrupted for any reason, already uploaded file parts (chunks) remain in the underlying storage so that the file upload can resume in a future upload attempt. However, resuming an upload is only possible until the partial upload is expired and deleted, respectively.

To clean up chunks (expire and delete) originating from unfinished uploads, administrators can use this newly introduced command. The default expiry time is two days, but it can be specified as a parameter to the command.



It is recommended to configure CRON to execute this background job regularly.

It is not included in the regular ownCloud background jobs so that the administrators have more flexibility in scheduling it. Please check the background jobs configuration documentation for more information.

Administrators can now exclude files from integrity check in config.php

When administrators did intentional changes to the ownCloud code they now have the ability to exclude certain files from the integrity checker. Please check config.sample.php for the usage of 'integrity.excluded.files'.

Modification time value of files is now 64 bits long

When upgrading to 10.0.4 migrations may increase update duration dependent on number of files.

Updated minimum supported browser versions

Users with outdated browsers might get warnings. See the list of supported browser versions.

Known issues

- When using application passwords, log entries related to Login Failed will appear, please upgrade to 10.0.7 and check the fix mentioned in its release notes.

10.0.3 resolved known issues

- SFTP external storages with key pair mode work again
- Added support for MariaDB 10.2.7+
- Encryption panel in admin settings fixed to properly detect current mode after upgrade to ownCloud 10
- Removed double quotes from boolean values in status.php output

Known issues

- Impersonate app 0.1.1 does not work with ownCloud Server 10.0.4. Please update to Impersonate 0.1.2 to be able to use the feature with ownCloud 10.0.4.
- Mounting ownCloud storage via davfs does not work

Changes in 10.0.3

Dear ownCloud administrator, please find below the changes and known issues of ownCloud Server 10.0.3 that need your attention:

The full ownCloud Server 10.0.3 changelog can be found here: https://github.com/owncloud/core/blob/stable10/CHANGELOG.md

- It is now possible to directly upgrade from 8.2.11 to 10.0.3 in a single upgrade process.
- Added occ command to list routes which can help administrators setting up network firewall rules.
- occ upgrade is now verbose by default. Administrators may need to adjust scripts for automated setup/upgrade procedures that rely on `occ upgrade' outputs.
- Reenabled medial search by default::
  - Enables partial search in sharing dialog autocompletion (e.g. a user wants to share with the user "Peter": Entering "pe" will find the user, entering "ter" will only find the user if the option is enabled)
  - New default is set to enabled as there is no performance impact anymore due to the introduction of the user account table in ownCloud Server 10.0.1.
  - Please check the setting. You need to disable it explicitly if the functionality is undesired.
- All database columns that use the fileid have been changed to bigint (64-bits). For large instances it is therefore highly recommended to upgrade in order to avoid reaching limits.
- Upgrade and Market app information::

- Removed appstoreenabled setting from config.php. If you want to disable the app store / Marketplace integration, please disable the Market app.
- Added setting `upgrade.automatic-app-update' to config.php to disable automatic app updates with `occ upgrade' when Market app is enabled
- On upgrade from OC < 10 the Market app won't be enabled if appstoreenabled was false in config.php.

- Clustering: Better support of read only config file and apps folder
- Default minimum desktop client version in config.php is now 2.2.4.

Known issues

- Added quotes in boolean result values of yourdomain/status.php output
- Setting up SFTP external storages with keypairs does not work. https://github.com/owncloud/core/issues/28669
- If you have storage encryption enabled, the web UI for encryption will ask again what mode you want to operate with even if you already had a mode selected before. The administrator must select the mode they had selected before. https://github.com/owncloud/core/issues/28985
- Uploading a folder in Chrome in a way that would overwrite an existing folder can randomly fail (race conditions). https://github.com/owncloud/core/issues/28844
- Federated shares can not be accepted in WebUI for SAML/Shibboleth users
- For **MariaDB users**: Currently, Doctrine has no support for the breaking changes introduced in MariaDB 10.2.7, and above. If you are on MariaDB 10.2.7 or above, and have encountered the message 1067 Invalid default value for `lastmodified, please apply this patch to Doctrine. We expect this bug to be fixed in ownCloud 10.0.4. For more information on the bug, check out the related issue.
- When updating from ownCloud < 9.0 the CLI output may hang for some time (potentially up to 20 minutes for big instances) whilst sharing is updated. This can happen in a variety of places during the upgrade and is to be expected. Please be patient as the update is performed and the output will continue as normal.

Changes in 10.0.1

Hello ownCloud administrator, please read carefully to be prepared for updates and operations of your ownCloud setup.

- **A new update path:** ownCloud 10.0.1 contains migration logic to allow upgrading directly from 9.0 to 10.0.1.
- **Marketplace:** Please create an account for `the new marketplace`_. Access to optional ownCloud extensions and enterprise apps will be provided by the marketplace from now on. Currently some apps are still shipped with the tarballs / packages and will be moved to the marketplace in the near future.
- **Apps:** *LDAP*, *gallery*, *activity*, *PDF viewer*, and *text editor* were moved to the marketplace.
- **Updates with marketplace:** During the upgrade, enabled apps are also updated by fetching new versions directly from the marketplace. If during an update, sources for some apps are missing, and the ownCloud instance has no access to the marketplace, the administrator needs to disable these apps or manually download and provide the apps before updating.
- **App updates:** Third party apps are not disabled anymore when upgrading.
- **Upgrade migration test:** The upgrade migration test, --skip-migration-tests, has been removed.



The template editor app is not included in the 10.0.1 release due to technical reasons, but will be distributed via the marketplace. However, you can still edit template files manually.

Settings

- **Settings design:** Admin, personal pages, and app management are now merged together into a single "Settings" entry.
- **Disable users:** The ability to disable users in the user management panel has been added.
- **Password Policy:** Rules now apply not only to link passwords but also to user passwords.

Infrastructure

- **Client:** You need to update to the latest desktop client version.
- **Cron jobs:** The user account table has been reworked. As a result the Cron job for syncing user backends, e.g., LDAP, needs to be configured.
- **Logfiles:** App logs, e.g., auditing and owncloud.log, can now be split, see: xref:configuration/server/config_sample_php_parameters.adoc#logging.

Known Issues

Converting the Database Type doesn't work

Converting a Database from e.g. SQLite to MySQL or PostgreSQL with the occ db:convert-type currently doesn't work. See https://github.com/owncloud/core/issues/27075 for more info.

Installing the LDAP user backend will trigger the installation twice

This causes an SQL error such as the following:

```
{occ-command-example-prefix} market:install user_ldap

user_ldap: Installing new app ...
user_ldap: An exception occurred while executing 'CREATE TABLE
`ldap_user_mapping` (`ldap_dn` VARCHAR(255) DEFAULT '' NOT NULL,
`owncloud_name` VARCHAR(255) DEFAULT '' NOT NULL, `directory_uuid`
VARCHAR(255) DEFAULT '' NOT NULL, UNIQUE INDEX ldap_dn_users (`ldap_dn`),
PRIMARY KEY(`owncloud_name`)) DEFAULT CHARACTER SET utf8mb4 COLLATE
utf8mb4_bin ENGINE = InnoDB ROW_FORMAT = compressed':

SQLSTATE[42S01]: Base table or view already exists: 1050 Table
'ldap_user_mapping' already exists
```

This can be safely ignored. And the app can be used after enabling it. Please be aware that when upgrading an existing ownCloud installation that already has user_ldap this error will not occur. It was fixed by https://github.com/owncloud/core/pull/27982. However, this could happen for other apps as well that use database.xml. If it does please use the same workaround.

SAML authentication only works for users synced with occ user:sync

We will re-enable SSO for LDAP users with an update of the app in the market after completing internal testing.

The web UI prevents uninstalling apps marked as shipped, e.g., user_ldap

To uninstall, disable the app with occ and rm the app directory.

Moving files around in external storages outside of ownCloud will invalidate the metadata

All shares, comments, and tags on the moved files will be lost.

Existing LDAP users only show up in the user management page and the share dialog after being synced

The account table introduced in ownCloud 10.0.0 significantly reduces LDAP communication overhead. Password checks are yet to be accounted for. LDAP user metadata in the account table will be updated when users log in or when the administrator runs occ user:sync "OCA\User_LDAP\User_Proxy". We recommend setting up a nightly Cron job to keep metadata of users not actively logging in up to date.

Error pages will not use the configured theme but will instead fall back to the community default

Changes in 10.0.0

- PHP 7.1 support added (supported PHP versions are 5.6 and 7.0+)
- The upgrade migration test has been removed; (Option "--skip-migration-tests" removed from update command)
- Requires to use the latest desktop client version 2.3
- Third party apps are not disabled anymore when upgrading
- User account table has been reworked. CRON job for syncing with e.g., LDAP needs to be configured (see Syncing User Accounts for more information)
- LDAP app is not released with ownCloud 10.0.0 and will be released on the marketplace after some more QA
- files_drop app is not shipped anymore as it’s integrated with core now. Since migrations are not possible you will have to reconfigure your drop folders (in the `Public Link' section of the sharing dialog of the respective folders).
- SAML/Shibboleth with device-specific app passwords: No migration possible; Users need to regenerate device-specific app passwords in the WebUI and enter those in their clients.
- For security reasons status.php can now be configured in config.php to not return server version information anymore (`version.hide'; default `false'). As clients still depend on version information this is not yet recommended. The default will change to `true' with 10.0.2 once clients are ready.
- Order of owncloud.log entries changed a bit, please review any application (e.g. fail2ban rules) relying on this file
- External storages::
  - FTP external storage moved to a separate app (https://marketplace.owncloud.com/apps/files_external_ftp)
  - "Local" storage type can now be disabled by sysadmin in config.php (to prevent users mounting the local file system)

Full changelog: https://github.com/owncloud/core/wiki/ownCloud-10.0-Features

Changes in 9.1

General

- Background jobs (cron) can now run in parallel
- Update notifications in client via API - You can now be notified in your desktop client about available updates for core and apps. The notifications are made available via the notifications API.
- Multi-bucket support for primary objectstore integration
- Support for Internet Explorer below version 11 was dropped
- Symlinks pointing outside of the data directory are disallowed. Please use the configuration/files/external_storage_configuration_gui with the configuration/files/external_storage/local storage backend instead.
- Removed dav:migrate-calendars and dav:migrate-addressbooks commands for occ. Users planning to upgrade from ownCloud 9.0 or below to ownCloud 9.1 needs to make sure that their calendars and address books are correctly migrated **before** continuing to upgrade to 9.1.

Authentication

- Pluggable authentication: plugin system that supports different authentication schemes
- Token-based authentication
- Ability to invalidate sessions
- List connected browsers/devices in the personal settings page. Allows the user to disconnect browsers/devices.
- Device-specific passwords/tokens, can be generated in the personal page and revoked
- Disable users and automatically revoke their sessions
- Detect disabled LDAP users or password changes and revoke their sessions
- Log in with email address
- Configuration option to enforce token-based login outside the web UI
- Two Factor authentication plug-in system
- OCC command added to (temporarily) disable/enable two-factor authentication for single users



The current desktop and mobile client versions do not support two-factor yet, this will be added later. It is already possible to generate a device specific password and enter that in the current client versions.

Files app

- Ability to toggle displaying hidden files
- Remember sort order
- Permalinks for internal shares
- Visual cue when dragging in files app
- Autoscroll file list when dragging files
- Upload progress estimate

Federated sharing

- Ability to create federated shares with CRUDS permissions
- Resharing a federated share does not create a chain of shares any more but connects the share owner's server to the reshare recipient

External storage

- UTF-8 NFD encoding compatibility support for NFD file names stored directly on external storages (new mount option in external storage admin page)
- Direct links to the configuration pages for setting up a GDrive or Dropbox application for use with ownCloud
- Some performance and memory usage improvements for GDrive, stream download and chunk upload
- Performance and memory usage improvements for Dropbox with stream download
- GDrive library update provides exponential backoff which will reduce rate limit errors

Shibboleth

- The WebDAV endpoint was changed from /remote.php/webdav to /remote.php/dav. You need to check your Apache configuration if you have exceptions or rules for WebDAV configured.

Minor additions

- Support for print style sheets
- Command line based update will now be suggested if the instance is bigger to avoid potential timeouts
- Web updater will be disabled if LDAP or shibboleth are installed
- DB/application update process now shows better progress information
- Added occ files:scan --unscanned to only scan folders that haven't yet been explored on external storages
- Chunk cache TTL can now be configured
- Added warning for wrongly configured database transactions, helps prevent database is locked issues
- Use a capped memory cache to reduce memory usage especially in background jobs and the file scanner
- Allow login by email
- Respect CLASS property in calendar events
- Allow addressbook export using VCFExportPlugin
- Birthdays are also generated based on shared addressbooks

For developers

- New DAV endpoint with a new chunking protocol aiming to solve many issues like timeouts (not used by clients yet)
- New webdav property for share permissions
- Background repair steps can be specified info.xml
- Background jobs (cron) can now be declared in info.xml
- Apps can now define repair steps to run at install/uninstall time
- Export contact images via Sabre DAV plugin

- Sabre DAV's browser plugin is available in debug mode to allow easier development around webdav

Technical debt

- PSR-4 autoloading forced for OC\ and OCP\, optional for OCA\ docs at xref:developer_manual/app/classloader.adoc
- More cleanup of the sharing code (ongoing)

Changes in 9.0

9.0 requires .ico files for favicons. This will change in 9.1, which will use .svg files. See Changing favicon in the Developer Manual.

Home folder rule is enforced in the user_ldap application in new ownCloud installations; see configuration/user/user_auth_ldap. This affects ownCloud 8.0.10, 8.1.5 and 8.2.0 and up.

The Calendar and Contacts apps have been rewritten and the CalDAV and CardDAV backends of these apps were merged into ownCloud core. During the upgrade existing Calendars and Addressbooks are automatically migrated (except when using the IMAP user backend). As a fallback for failed upgrades, when using the IMAP user backend or as an option to test a migration dav:migrate-calendars and/or dav:migrate-addressbooks scripts are available (**only in ownCloud 9.0**) via the occ command. See configuration/server/occ_command.



After upgrading to ownCloud 9.0 and **before** continuing to upgrade to 9.1 make sure that all of your and your users Calendars and Addressbooks are migrated correctly. Especially when using the IMAP user backend (other user backends might be also affected) you need to manually run the mentioned occ migration commands described above.

Updates on systems with large datasets will take longer, due to the addition of checksums to the ownCloud database. See https://github.com/owncloud/core/issues/22747.

Linux packages are available from our official download repository. New in 9.0: split packages. owncloud installs ownCloud plus dependencies, including Apache and PHP. owncloud-files installs only ownCloud. This is useful for custom LAMP stacks, and allows you to install your own LAMP apps and versions without packaging conflicts with ownCloud. See installation/linux_installation.

New option for the ownCloud admin to enable or disable sharing on individual external mountpoints (see External Storage GUI Mount Options). Sharing on such mount points is disabled by default.

Enterprise 9.0

owncloud-enterprise packages are no longer available for CentOS 6, RHEL6, Debian 7, or any version of Fedora. A new package, owncloud-enterprise-files, is available for all supported platforms, including the above. This new package comes without dependencies, and is installable on a larger number of platforms. System administrators must install their own LAMP stacks and databases. See https://owncloud.org/blog/time-to-upgrade-to-owncloud-9-0/.

Changes in 8.2

New location for Linux package repositories; ownCloud admins must manually change to the new repos. See maintenance/upgrade

PHP 5.6.11+ breaks the LDAP wizard with a `Could not connect to LDAP' error. See https://github.com/owncloud/core/issues/20020.

filesystem_check_changes in config.php is set to 0 by default. This prevents unnecessary update checks and improves performance. If you are using external storage mounts such as NFS on a remote storage server, set this to 1 so that ownCloud will detect remote file changes.

XSendFile support has been removed, so there is no longer support for serving static files from your ownCloud server.

LDAP issue: 8.2 uses the memberof attribute by default. If this is not activated on your LDAP server your user groups will not be detected, and you will see this message in your ownCloud log: Error PHP Array to string conversion at /var/www/html/owncloud/lib/private/template/functions.php#36. Fix this by disabling the memberof attribute on your ownCloud server with the occ command, like this example on Ubuntu Linux:

```
{occ-command-example-prefix} ldap:set-config "s01"
useMemberOfToDetectMembership 0
```

Run {occ-command-example-prefix} ldap:show-config to find the correct sNN value; if there is not one then use empty quotes, "". (See configuration/server/occ_command.)

Users of the Linux Package need to update their repository setup as described in this blogpost.

Changes in 8.1

Use APCu only if available in version 4.0.6 and higher. If you install an older version, you will see a APCu below version 4.0.6 is installed, for stability and performance reasons we recommend to update to a newer APCu version warning on your ownCloud admin page.

SMB external storage now based on php5-libsmbclient, which must be downloaded from the ownCloud software repositories (installation instructions).

Download from link feature has been removed.

The .htaccess and index.html files in the data/ directory are now updated after every update. If you make any modifications to these files they will be lost after updates.

The SabreDAV browser at /remote.php/webdav has been removed.

Using ownCloud without a trusted_domain configuration will not work anymore.

The logging format for failed logins has changed and considers now the proxy configuration in config.php.

A default set of security and privacy HTTP headers have been added to the ownCloud .htaccess file, and ownCloud administrators may now customize which headers are sent.

More strict SSL certificate checking improves security but can result in cURL error 60: SSL certificate problem: unable to get local issuer certificate errors with certain broken PHP versions. Please verify your SSL setup, update your PHP or contact your vendor if you receive these errors.

The persistent file-based cache (e.g. used by LDAP integration) has been dropped and

replaced with a memory-only cache, which must be explicitly configured. See configuration/user/user_auth_ldap. Memory cache configuration for the ownCloud server is no longer automatic, requiring installation of your desired cache backend and configuration in config.php (see configuration/server/caching_configuration.)

The OC_User_HTTP backend has been removed. Administrators are encouraged to use the user_webdavauth application instead.

ownCloud ships now with its own root certificate bundle derived from Mozilla's root certificates file. The system root certificate bundle will not be used anymore for most requests.

When you upgrade from ownCloud 8.0, with encryption enabled, to 8.1, you must enable the new encryption backend and migrate your encryption keys.

Encryption can no longer be disabled in ownCloud 8.1. It is planned to re-add this feature to the command line client for a future release.

It is not recommended to upgrade encryption-enabled systems from ownCloud Server 8.0 to version 8.1.0 as there is a chance the migration will break. We recommend migrating to the first bugfix release, ownCloud Server 8.1.1.

Due to various technical issues, by default desktop sync clients older than 1.7 are not allowed to connect and sync with the ownCloud server. This is configurable via the minimum.supported.desktop.version switch in config.php.

Previews are now generated at a maximum size of 2048 x 2048 pixels. This is configurable via the preview_max_x and preview_max_y switches in config.php.

The ownCloud 8 server is not supported on any version of Windows.

The 8.1.0 release has a minor bug which makes application updates fail at first try. Reload the apps page and try again, and the update will succeed.

The forcessl option within the config.php and the Enforce SSL option within the Admin-Backend was removed. This now needs to be configured like described in Hardening and Security Guidance.

WebDAV file locking was removed in ownCloud 8.1 which causes Finder on macOS to mount WebDAV read-only.

Enterprise 8.1

The SharePoint Drive application does not verify the SSL certificate of the SharePoint server or the ownCloud server, as it is expected that both devices are in the same trusted environment.

Changes in 8.0

Manual LDAP Port Configuration

When you are configuring the LDAP user and group backend application, ownCloud may not auto-detect the LDAP server's port number, so you will need to enter it manually.

No Preview Icon on Text Files

There is no preview icon displayed for text files when the file contains fewer than six characters.

Remote Federated Cloud Share Cannot be Reshared With Local Users

When you mount a Federated Cloud share from a remote ownCloud server, you cannot re-share it with your local ownCloud users. (See Federated Cloud Sharing Configuration to learn more about federated cloud sharing)

Manually Migrate Encryption Keys after Upgrade

If you are using the Encryption application and upgrading from older versions of ownCloud to ownCloud 8.0, you must manually migrate your encryption keys.

Windows Server Not Supported

Windows Server is not supported in ownCloud 8.

PHP 5.3 Support Dropped

PHP 5.3 is not supported in ownCloud 8, and PHP 5.4 or better is required.

Disable Apache Multiviews

If Multiviews are enabled in your Apache configuration, this may cause problems with content negotiation, so disable Multiviews by removing it from your Apache configuration. Look for lines like this:

```
<Directory /var/www/owncloud>
Options Indexes FollowSymLinks Multiviews
```

Delete Multiviews and restart Apache.

ownCloud Does Not Follow Symlinks

ownCloud's file scanner does not follow symlinks, which could lead to infinite loops. To avoid this do not use soft or hard links in your ownCloud data directory.

No Commas in Group Names

Creating an ownCloud group with a comma in the group name causes ownCloud to treat the group as two groups.

Hebrew File Names Too Large on Windows

On Windows servers Hebrew file names grow to five times their original size after being translated to Unicode.

Google Drive Large Files Fail with 500 Error

Google Drive tries to download the entire file into memory, then write it to a temp file, and then stream it to the client, so very large file downloads from Google Drive may fail with a 500 internal server error.

Encrypting Large Numbers of Files

When you activate the Encryption application on a running server that has large numbers of files, it is possible that you will experience timeouts. It is best to activate encryption at installation, before accumulating large numbers of files on your ownCloud server.

Enterprise 8.0

Sharepoint Drive SSL Not Verified

The SharePoint Drive application does not verify the SSL certificate of the SharePoint server or the ownCloud server, as it is expected that both devices are in the same trusted environment.

No Federated Cloud Sharing with Shibboleth

Federated Cloud Sharing (formerly Server-to-Server file sharing)does not work with Shibboleth .

Direct Uploads to SWIFT do not Appear in ownCloud

When files are uploaded directly to a SWIFT share mounted as external storage in ownCloud, the files do not appear in ownCloud. However, files uploaded to the SWIFT mount through ownCloud are listed correctly in both locations.

SWIFT Objectstore Incompatible with Encryption App

The current SWIFT implementation is incompatible with any application that uses direct file I/O and circumvents the ownCloud virtual filesystem. Using the Encryption application on a SWIFT object store incurs twice as many HTTP requests and increases latency significantly.

application Store is Back

The ownCloud application Store has been re-enabled in ownCloud 8. Note that third-party apps are not supported.

Changes in 7.0

Manual LDAP Port Configuration

When you are configuring the LDAP user and group backend application, ownCloud may not auto-detect the LDAP server's port number, so you will need to enter it manually.

LDAP Search Performance Improved

Prior to 7.0.4, LDAP searches were substring-based and would match search attributes if the substring occurred anywhere in the attribute value. Rather, searches are performed on beginning attributes. With 7.0.4, searches will match at the beginning of the attribute value only. This provides better performance and a better user experience.

Substring searches can still be performed by prepending the search term with *. For example, a search for te will find Terri, but not Nate:

```
occ ldap:search "te"
```

If you want to broaden the search to include Nate, then search for *te:

```
occ ldap:search "*te"
```

Refine searches by adjusting the User Search Attributes field of the Advanced tab in your LDAP configuration on the Admin page. 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.

Protecting ownCloud on IIS from Data Loss

Under certain circumstances, running your ownCloud server on IIS could be at risk of data loss. To prevent this, follow these steps.

- In your ownCloud server configuration file, owncloud\config\config.php, set config_is_read_only to true.
- Set the config.php file to read-only.
- When you make server updates config.php must be made writeable. When your updates are completed re-set it to read-only.

Antivirus Application Modes

The Antivirus application offers three modes for running the ClamAV anti-virus scanner: as a daemon on the ownCloud server, a daemon on a remote server, or an executable mode that calls clamscan on the local server. We recommend using one of the daemon modes, as they are the most reliable.

Enable Only for Specific Groups Fails

Some ownCloud applications have the option to be enabled only for certain groups. However, when you select specific groups they do not get access to the app.

Changes to File Previews

For security and performance reasons, file previews are available only for image files, covers of MP3 files, and text files, and have been disabled for all other filetypes. Files without previews are represented by generic icons according to their file types.

4GB Limit on SFTP Transfers

Because of limitations in phpseclib, you cannot upload files larger than 4GB over SFTP.

Not Enough Space Available on File Upload

Setting user quotas to unlimited on an ownCloud installation that has unreliable free disk space reporting– for example, on a shared hosting provider– may cause file uploads to fail with a Not Enough Space Available error. A workaround is to set file quotas for all users instead of unlimited.

No More Expiration Date On Local Shares

In older versions of ownCloud, you could set an expiration date on both local and public link shares. Now you can set an expiration date only on public link shares, and local shares do not expire when public link shares expire.

Zero Quota Not Read-Only

Setting a user's storage quota should be the equivalent of read-only, however, users can still create empty files.

Enterprise 7.0

No Federated Cloud Sharing with Shibboleth

Federated Cloud Sharing (formerly Server-to-Server file sharing) does not work with Shibboleth .

Windows Network Drive

Windows Network Drive runs only on Linux servers because it requires the Samba client, which is included in all Linux distributions.

php5-libsmbclient is also required, and there may be issues with older versions of libsmbclient; see Using External Storage > Installing and Configuring the Windows Network Drive application in the Enterprise Admin manual for more information.

By default CentOS has activated SELinux, and the httpd process can not make outgoing network connections. This will cause problems with curl, LDAP and samba libraries. Again, see Using External Storage > Installing and Configuring the Windows Network Drive application in the Enterprise Admin manual for instructions.

Sharepoint Drive SSL

The SharePoint Drive application does not verify the SSL certificate of the SharePoint server or the ownCloud server, as it is expected that both devices are in the same trusted environment.

Shibboleth and WebDAV Incompatible

Shibboleth and standard WebDAV are incompatible, and cannot be used together in ownCloud. If Shibboleth is enabled, the ownCloud client uses an extended WebDAV protocol

No SQLite

SQLite is no longer an installation option for ownCloud Enterprise Edition, as it not suitable for multiple-user installations or managing large numbers of files.

No Application Store

The application Store is disabled for the Enterprise Edition.

LDAP Home Connector Linux Only

The LDAP Home Connector application requires Linux (with MySQL, MariaDB, or PostgreSQL) to operate correctly.

What's New in ownCloud

See the ownCloud 10.0 Features page on Github for a comprehensive list of new features and updates.

Frequently Asked Questions

I want to upgrade from Community Version to Enterprise Version. What are the changes?

In ownCloud Enterprise you will get access to new apps and features, mainly targeted towards enterprises; apps and features that ensure security, for example.

You can upgrade to the Enterprise version without concern, as your existing files, shares, and users remain as they are.

How do I transfer files from one user to another?

See transferring files to another user.

How do I deal with problems caused by using self-signed SSL certificates?

See the security section of the OCC command.

I'm the admin and I lost my password! What do I do now!

See the reset admin password documentation.

What is a Federated System?

A Federated System is another ownCloud or OpenCloudMesh supporting cloud service.

Platform-wide known limitations, excluded files

There are known file names that can not be synced with ownCloud, these are:

- Folders and files with a trailing space.
- .htaccess.
- *.part files.
- File names that exceed 253 characters.
- client/sync-exclude.list.
- Desktop.ini in the root directory.
- UNIX/Linux hidden files (files whose names have a leading dot, e.g., .12345.pdf). Users must activate "*sync hidden files*" to sync them.

Installation

In this section, you will find all the information you need for installing ownCloud.

Deployment Considerations

Hardware

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

Single Machine / Scale-Up Deployment

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

Pros:

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

Cons:

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

Scale-Out Deployment

Provider setup:

- DNS round robin to HAProxy servers (2-n, SSL offloading, cache static resources)
- Least load to Apache servers (2-n)
- Memcached/Redis for shared session storage (2-n)
- Database cluster with single Master, multiple slaves and proxy to split requests accordingly (2-n)
- GPFS or Ceph via phprados (2-n, 3 to be safe, Ceph 10+ nodes to see speed benefits under load)
- In case of clustering, your cluster nodes must have the same ownCloud configuration including an identical config.php to avoid any potential issues.

Pros:

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

Cons:

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

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

When master fails another slave can become master. However, the increased complexity carries some risks: Multi-master has the risk of split brain, and deadlocks. ownCloud tries to solve the problem of deadlocks with high-level file locking.

Software

Operating System

We are dependent on distributions that offer an easy way to install the various components in up-to-date versions. ownCloud has a partnership with RedHat and SUSE for customers who need commercial support. Canonical, the parent company of Ubuntu Linux, also offers enterprise service and support. Debian and Ubuntu are free of cost, and include newer software packages. CentOS is the community-supported free-of-cost Red Hat Enterprise Linux clone. openSUSE is community-supported, and includes many of the same system administration tools as SUSE Linux Enterprise Server.

Web server

Apache with mod_php is currently the best option. Mod_php is recommended instead

of PHP_FPM, because in scale-out deployments separate PHP pools are not necessary.

Relational Database

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

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

What about the other DBMS?

- Sqlite is adequate for simple testing, and for low-load single-user deployments. It is not adequate for production systems.
- Microsoft SQL Server is not a supported option.
- Oracle DB is the de facto standard at large enterprises and is fully supported with ownCloud Enterprise Edition only.

File Storage

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

Session Storage

- Redis is required for transactional file locking Transactional File Locking, provides session persistence, and graphical inspection tools available.
- If you need to scale out Shibboleth you must use Memcached, as Shibboleth does not provide an interface to Redis. Memcached can also be used to scale-out shibd session storage (see Memcache StorageService).

Deployment Recommendations

Introduction

What is the best way to install and maintain ownCloud? The answer to that is, as always: '*it depends*'.

This is because every ownCloud customer has their own particular needs and IT infrastructure. However, both ownCloud and the LAMP stack are highly-configurable. Given that, in this document we present a set of general recommendations, followed by three typical scenarios, and finish up with making best-practice recommendations for both software and hardware.



The recommendations presented here are based on a standard ownCloud installation, one without any particular *apps*, *themes*, or *code changes*. But, server load is dependent upon the number of *clients*, *files*, and *user activity*, as well as other usage patterns. Given that, these recommendations are only a rule of thumb based on our experience, as well as that of one of our customers.

General Recommendations

- Operating system: Linux.
- Web server: Apache 2.4.
- Database: MySQL/MariaDB with InnoDB storage engine (MyISAM is not supported, see: MySQL / MariaDB storage engine)
- And a recent PHP Version. See System Requirements
- Consider setting up a scale-out deployment, or using Federated Cloud Sharing to keep individual ownCloud instances to a manageable size.



Whatever the size of your organization, always keep one thing in mind: *The amount of data stored in ownCloud will only grow. So plan ahead.*

ownCloud Administrators Must Have Command Line or Cron Access

We only recommend using hosts that provide command-line or Cron access (ideally both) *to ownCloud administrators*, for three key reasons:

1. Without command-line access, OCC commands, required for administrative tasks such as repairs and upgrades, are not available.
2. Without Crontab access, you cannot run background jobs reliably. ajax/cron.php is available, but it is not reliable enough, because it only runs when people are using the web UI. Additionally, ownCloud relies heavily on background jobs especially for long-running operations, which will likely cause PHP timeouts.
3. PHP timeout values are often low. Having low timeout settings can break long-running operations, such as moving a huge folder.

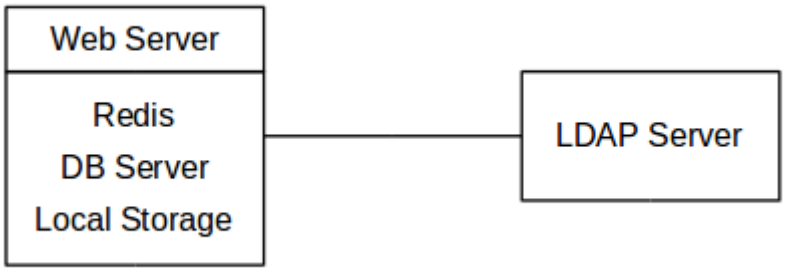
Scenario 1: Small Workgroups and Departments

This recommendation applies if you meet the following criteria:

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

Recommended System Requirements

One machine running the application, web, and database server, as well as local storage. Authentication via an existing LDAP or Active Directory server.



Components

One server with at least 2 CPU cores, 16GB RAM, and local storage as needed.

Operating system

Enterprise-grade Linux distribution with full support from an operating system vendor. We recommend both RedHat Enterprise Linux and SUSE Linux Enterprise Server 12.

SSL Configuration

The SSL termination is done in Apache. A standard SSL certificate is required to be installed according to the official Apache documentation.

Load Balancer

None.

Database

MySQL, MariaDB, or PostgreSQL. We currently recommend MySQL / MariaDB, as our customers have had good experiences when moving to a Galera cluster to scale the DB. If using either MySQL or MariaDB, you must use the InnoDB storage engine as MyISAM is not supported, see: MySQL / MariaDB storage engine



If you are using MaxScale/Galera, then you need to use at least version 1.3.0. In earlier versions, there is a bug where the value of last_insert_id is not routed to the master node. This bug can cause loops within ownCloud and corrupt database rows. You can find out more information in the issue documentation.

Backup

Install ownCloud, the ownCloud data directory, and database on a Btrfs filesystem. Make regular snapshots at desired intervals for zero downtime backups. Mount DB partitions with the "nodatacow" option to prevent fragmentation.

Alternatively, you can make nightly backups — with service interruption — as follows:

1. Shut down Apache.
2. Create database dump.
3. Push data directory to backup.
4. Push database dump to backup.
5. Start Apache.

After these steps have been completed, then, optionally, rsync the backup to either an external backup storage or tape backup. See the Maintenance section of the

Administration manual for tips on backups and restores.

Authentication

User authentication via one or several LDAP or Active Directory (AD) servers. See User Authentication with LDAP for information on configuring ownCloud to use LDAP and AD.

Session Management

Local session management on the application server. PHP sessions are stored in a temporary filesystem, mounted at the operating system-specific session storage location. You can find out where that is by running grep -R 'session.save_path' /etc/php* and then add it to the /etc/fstab file, for example:

```
# Retrieve the session save path setting (default or explicit value) for PHP
{recommended-php-version}
# Please change the file path to match your server configuration
session_path=$(\
   awk 'match($0, /^;?session.save_path = "(.*)"/, a) { print a[1] }' \
   /etc/php/{recommended-php-version}/**/php.ini \
   | uniq )

# Set the session save path in /etc/fstab
echo "tmpfs ${session_path} tmpfs defaults,noatime,mode=1777 0 0" >>
/etc/fstab
```

Memory Caching

A memory cache speeds up server performance, and ownCloud supports four of them. Refer to Configuring Memory Caching for information on selecting and configuring a memory cache.

Storage

Local storage.

ownCloud Edition

Standard Edition. See ownCloud Server or Enterprise Edition for comparisons of the ownCloud editions.

Scenario 2: Mid-Sized Enterprises

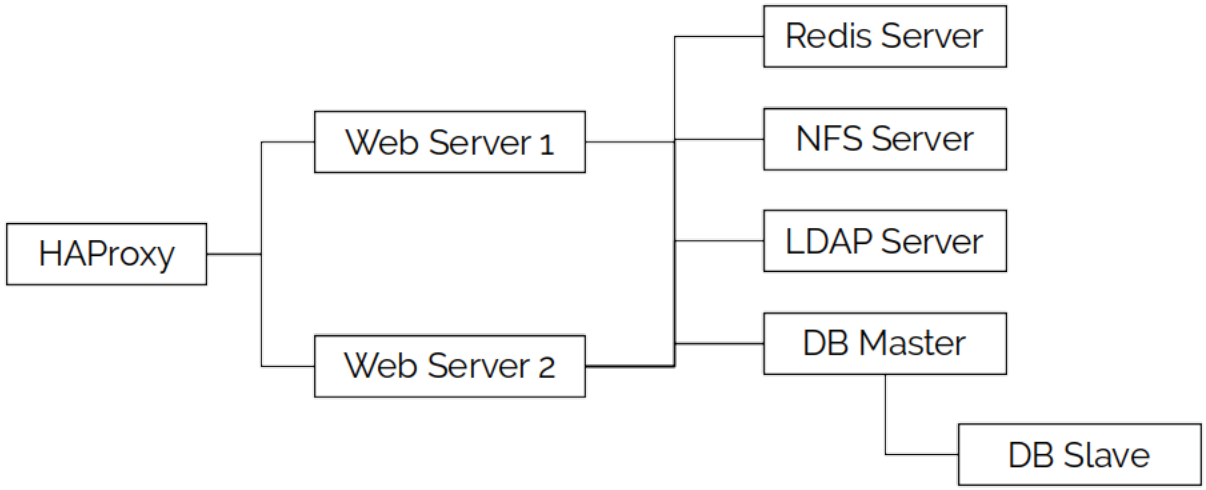
These recommendations apply if you meet the following criteria:

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

Option	Value
	service interruption

Recommended System Requirements

- 2 to 4 application servers.
- A cluster of two database servers.
- Storage on an NFS server.
- Authentication via an existing LDAP or Active Directory server.
- A Redis server for file locking



Components

- 2 to 4 application servers with four sockets and 32GB RAM.
- 2 DB servers with four sockets and 64GB RAM.
- 1 HAproxy load balancer with two sockets and 16GB RAM.
- NFS storage server as needed.

Operating System

Enterprise grade Linux distribution with full support from an operating system vendor. We recommend both RedHat Enterprise Linux and SUSE Linux Enterprise Server 12.

SSL Configuration

The SSL termination is done in the HAProxy load balancer. A standard SSL certificate is needed, installed according to the HAProxy documentation.

Load Balancer

HAProxy running on a dedicated server in front of the application servers. Sticky session needs to be used because of local session management on the application servers.

Database

MySQL/MariaDB Galera cluster with master-slave replication. InnoDB storage engine, MyISAM is not supported, see: MySQL / MariaDB storage engine. For mariadb consider: MariaDB Monitor to configure your setup for a failover Scenario.

Backup

Minimum daily backup without downtime. All MySQL/MariaDB statements should be replicated to a backup MySQL/MariaDB slave instance.

- Create a snapshot on the NFS storage server.
- At the same time stop the MySQL replication.
- Create a MySQL dump of the backup slave.
- Push the NFS snapshot to the backup.
- Push the MySQL dump to the backup.
- Delete the NFS snapshot.
- Restart MySQL replication.

Authentication

User authentication via one or several LDAP or Active Directory servers. See User Authentication with LDAP for information on configuring ownCloud to use LDAP and AD.

Session Management

Session management on the application server. PHP sessions are stored in a temporary filesystem, mounted at the operating system-specific session storage location. You can find out where that is by running grep -R 'session.save_path' /etc/php* and then add it to the /etc/fstab file, for example:

```
# Retrieve the session save path setting (default or explicit value) for PHP
{recommended-php-version}
# Please change the file path to match your server configuration
session_path=$(\
   awk 'match($0, /^;?session.save_path = "(.*)"/, a) { print a[1] }' \
   /etc/php/{recommended-php-version}/**/php.ini \
   | uniq )

# Set the session save path in /etc/fstab
echo "tmpfs ${session_path} tmpfs defaults,noatime,mode=1777 0 0" >>
/etc/fstab
```

Memory Caching

A memory cache speeds up server performance, and ownCloud supports four memory cache types. Refer to Configuring Memory Caching for information on selecting and configuring a memory cache.

Storage

For accessing a backend storage system via NFS, you can use a dedicated storage system like NetApp Hybrid Flash Storage Systems, or other systems like IBM Elastic Storage based on their Power8 servers or RedHat Ceph with their NFS-Ceph gateway.

You may take a look on the NetApp NFS Best Practice and Implementation Guide for best NFS configuring practices, especially section *9.4 Mount Option Best Practices with NFS* on page 111 and MySQL Database on NetApp ONTAP which also includes performance measurements.

ownCloud Edition

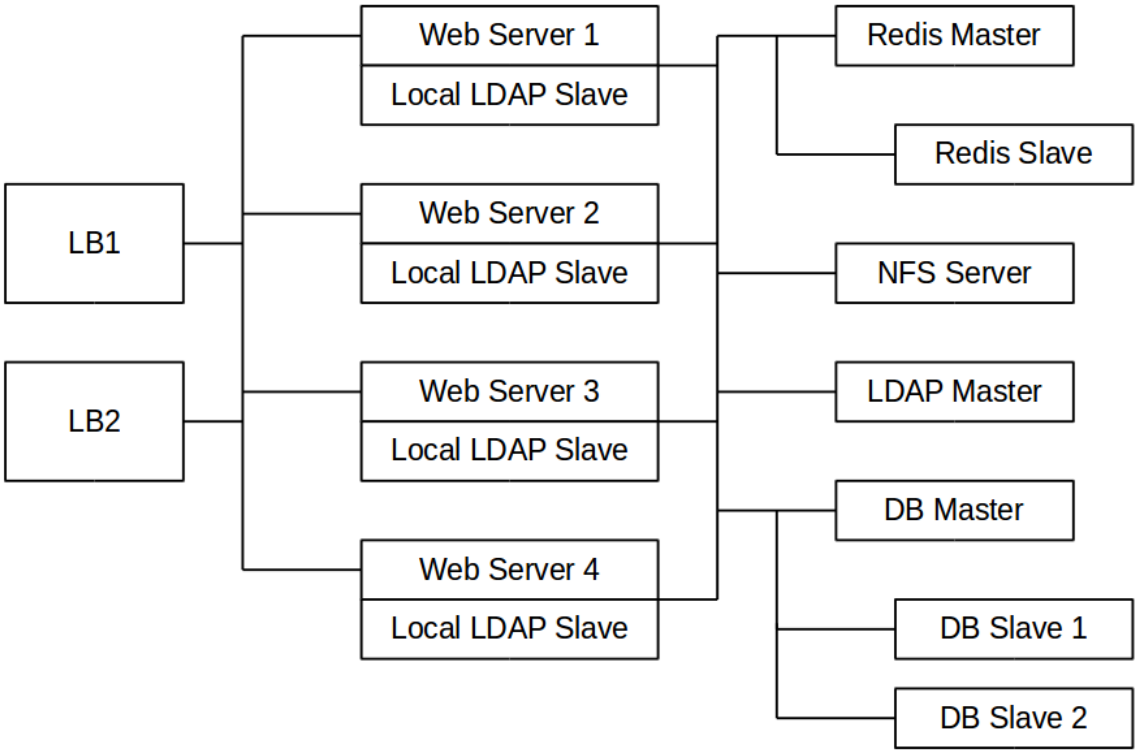
Enterprise Edition. See ownCloud Server or Enterprise Edition for comparisons of the ownCloud editions.

Scenario 3: Large Enterprises and Service Providers

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

Recommended System Requirements

- 4 to 20 application/Web servers.
- A cluster of two or more database servers.
- Storage is an NFS server or an object store that is S3 compatible.
- Cloud federation for a distributed setup over several data centers.
- Authentication via an existing LDAP or Active Directory server, or SAML.



Components

- 4 to 20 application servers with four sockets and 64GB RAM.
- 4 DB servers with four sockets and 128GB RAM. 2 Hardware load balancer, for

example, BIG IP from F5.

- NFS storage server as needed.

Operating system

RHEL 7 with latest service packs.

SSL Configuration

The SSL termination is done in the load balancer. A standard SSL certificate is needed, installed according to the load balancer documentation.

Load Balancer

A redundant hardware load-balancer with heartbeat, for example, F5 Big-IP. This runs two load balancers in front of the application servers.

Database

MySQL/MariaDB Galera Cluster with Master-Slave replication. InnoDB storage engine, MyISAM is not supported, see: MySQL / MariaDB storage engine. For mariadb consider: MariaDB Monitor to configure your setup for a failover Scenario.

Backup

Minimum daily backup without downtime. All MySQL/MariaDB statements should be replicated to a backup MySQL/MariaDB slave instance. To do this, follow these steps:

1. Create a snapshot on the NFS storage server.
2. At the same time stop the MySQL replication.
3. Create a MySQL dump of the backup slave.
4. Push the NFS snapshot to the backup.
5. Push the MySQL dump to the backup.
6. Delete the NFS snapshot.
7. Restart MySQL replication.

Authentication

User authentication via one or several LDAP or Active Directory servers, or SAML/Shibboleth. See User Authentication with LDAP and Shibboleth Integration.

LDAP

Read-only slaves should be deployed on every application server for optimal scalability.

Session Management

Redis should be used for the session management storage.

Caching

Redis for distributed in-memory caching.

Storage

For accessing a backend storage system via NFS, you can use a dedicated storage system like NetApp Hybrid Flash Storage Systems, or other systems like IBM Elastic Storage based on their Power8 servers or RedHat Ceph with their NFS-Ceph gateway.

Optionally, an S3 compatible object store can also be used.

You may take a look on the NetApp NFS Best Practice and Implementation Guide for best NFS configuring practices, especially section *9.4 Mount Option Best Practices with NFS* on page 111 and MySQL Database on NetApp ONTAP which also includes performance measurements.

ownCloud Edition

Enterprise Edition. See ownCloud Server or Enterprise Edition for comparisons of the ownCloud editions.

Redis Configuration

Redis in a master-slave configuration is a hot failover setup, and is usually sufficient. A slave can be omitted if high availability is provided via other means. And when it is, in the event of a failure, restarting Redis typically occurs quickly enough. Regarding Redis cluster, we don't, usually, recommend it, as it requires a greater level of both maintenance and management in the case of failure. A single Redis server, however, just needs to be rebooted, in the event of failure.

Known Issues

Deadlocks When Using MariaDB Galera Cluster

If you're using MariaDB Galera Cluster with your ownCloud installation, you may encounter deadlocks when you attempt to sync a large number of files. You may also encounter database errors, such as this one:

```
SQLSTATE[40001]: Serialization failure: 1213 Deadlock found when trying to get lock; try restarting transaction
```

The issue, identified by Michael Roth, is caused when MariaDB Galera cluster sends write requests to all servers in the cluster; here is a detailed explanation. The solution is to send all write requests to a single server, instead of all of them.

Set wsrep_sync_wait to 1 on all Galera Cluster nodes

What the parameter does

When enabled, the node triggers causality checks in response to certain types of queries. During the check, the node blocks new queries while the database server catches up with all updates made in the cluster to the point where the check begun. Once it reaches this point, the node executes the original query.

Why enable it

A Galera Cluster write operation is sent to the master while reads are retrieved from the slaves. Since Galera Cluster replication is, by default, not strictly synchronous it could happen that items are requested before the replication has actually taken place.



This setting is disabled by default. See the Galera Cluster WSREP documentation for more details.

References

- Database High Availability
- Performance enhancements for Apache and PHP

- How to Set Up a Redis Server as a Session Handler for PHP on Ubuntu 18.04

Network File System (NFS) Deployment Recommendations

ownCloud recommends using NFS for any scenario other than local storage. It has solid performance and is very stable. This document contains ownCloud's official deployment recommendations.

There can be different scenarios where ownCloud's storage is located on an NFS mount (primary/secondary). In some scenarios, multiple application servers can use the same NFS mount point.



It is advised to use network storage like NFS only in un-routed, switched Gigabit or higher environments.



This guide only covers the NFS client side where ownCloud runs. Follow the storage vendors recommendations to configure the NFS server (storage backend).

General Performance Considerations

Please consider that a network stack runs in ranges of µs while a storage backend usually runs in ranges of ms. Any tuning considerations should therefore first be attempted on the backend storage layout side, especially under high loads.

NFS Version Comparison Overview

NFSv3	
Exports	All exports are mounted separately
Protocol	Numerous protocols for different aspects collected together. MOUNT, LOCK, STATUS…
Locking	Permanent locks in yet another protocol
Security	UNIX based. SecureNFS. Mode Bit Locking
Communication	One operation per RPC
I18N	All locales must match
Parallel high bandwidth access	None native. (Addition such as MPFS)

NFSv4	
Exports	All exports can be mounted together in a directory tree structure as part of a pseudo-filesystem
Protocol	A single protocol with the addition of OPEN and CLOSE for security auditing
Locking	Lease based locking in the same protocol
Security	Kerberos and ACL based
Communication	Multiple operations per RPC. (Improves performance)
I18N	UTF-8
Parallel high bandwidth access	pNFS

NFSv4

ownCloud recommends using NFSv4 over previous versions for a number of key reasons. These are:

- **Improved Security:** It mandates a strong security architecture. It does not require rpc.statd or lockd. As a result, it only uses port 2049.
- **Improved Reliability:** Uses TCP by default.
- **Improved Performance:** It uses Multi-Component Messages, which reduce network traffic. It is capable of using a 32KB page size, compared to the default, 1024 bytes.
- Use of Read/Write Delegations.

NFS Mount Options

Please see the Ubuntu man pages for a detailed description of the NFS mount options.

Dependent on the NFS version used, consider following mount options:

_netdev

Use this option to ensure that the network is enabled, before NFS attempts to mount these filesystem. This setting is essential when database files are located on an NFS storage. The database could error or not start correctly, if the mount is not ready before attempting to access its data files.



You can also use autofs, to ensure that mounts are always available before attempting to access them.

bg

ownCloud recommends using this option. Determines how the mount command behaves if an attempt to mount an export fails. If the bg option is specified, a timeout or failure triggers the mount command to fork a child, which will continue to attempt mounting the export. The parent immediately returns with a zero exit code. This is known as a "background" mount. This option is useful for continuous operation without manual intervention if the network connectivity is temporarily down or the storage backend must be rebooted.

hard

Default value is *hard*. For business-critical NFS exports, ownCloud recommends using *hard* mounts. ownCloud strongly discourages the use of *soft* mounts.

retrans

Default value is 3. This option can be tuned when using option *soft*.

timeo

Default value is 600 (60 seconds). This option can be tuned when using option *soft*.

sync/async

With the default value of *async*, the NFS client may delay sending application writes to the NFS server. In other words, under normal circumstances, data written by an application may not immediately appear on the server that hosts the file. **sync** provides greater data cache coherence among clients, but at a **significant performance cost**. Having the database like MySQL or Mariadb on NFS, the default

database option value for innodb_flush_method is *fsync*, even if it is not explicitly set. This database option forces the mount to immediately write to the NFS server without generally setting the mount *sync* option and avoiding this performance penalty. You may consider further tuning when using clustederd server environments.

tcp

ownCloud recommends using this option. Force using TCP as transport protocol. Alternatively you can use *proto=tcp*.

Tune the Read and Write Block Sizes

The allowed block sizes are the packet chunk sizes that NFS uses when reading and writing data. The smaller the size, the greater the number of packets need to be sent to send or receive a file. Conversely, the larger the size, the fewer the number of packets need to be sent to send or receive a file. With NFS Version 3 and 4, you can set the rsize and wsize values as high as 65536, when the network transport is TCP. The default value is 32768 and must be a multiple of 4096.



Read and write size must be identical on the NFS server and client.

You can find the set values by working with the output of the mount command on a standard server, as in the example below.

```
#root@server:~# mount | egrep -o rsize=[0-9]*
rsize=65536

#root@server:~# mount | egrep -o wsize=[0-9]*
wsize=65536
```

The information can also be retrieved using the command set of your dedicated storage backend. Once you've determined the best sizes, set them permanently by passing the (rsize and wsize) options when mounting the share or in the share's mount configuration.

Listing 1. Specifying the read and write block sizes when calling mount

```
mount 192.168.0.104:/data  /mnt -o rsize=65536,wsize=65536
```

Listing 2. Example for a set of NFS mount options:

```
bg,nfsvers=3,wsize=65536,rsize=65536,tcp,_netdev
```

Ethernet Configuration Options

MTU (Maximum Transmission Unit) Size

The MTU size dictates the maximum amount of data that can be transferred in one Ethernet frame. If the MTU size is too small, then regardless of the read and write block sizes, the data must still be fragmented across multiple frames. Keep in mind that MTU = payload (packetsize) + 28.

Get the Current Set MTU Size

You can find the current MTU size for each interface using *netstat*, *ifconfig*, *ip*, and

cat, as in the following examples:

Listing 3. Retrieve interface MTU size with netstat

```
netstat -i

Kernel Interface table
Iface      MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
lo       65536   363183      0      0 0       363183      0      0      0 LRU
eth0      1500  3138292      0      0 0      2049155      0      0      0 BMR
```

Listing 4. Retrieve interface MTU size with ifconfig

```
ifconfig| grep -i MTU

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
```

Listing 5. Retrieve interface MTU size with ip

```
ip addr | grep mtu

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
group default qlen 1000
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
group default qlen 1000
```

Listing 6. Retrieve interface MTU size with cat

```
cat /sys/class/net/<interface>/mtu
```

Check for MTU Fragmentation

To check if a particular packet size will be fragmented on the way to the target, run the following command:

```
ping <your-storage-backend> -c 3 -M do -s <packetsize>
```

Get the Optimal MTU Size

To get the optimal MTU size, run following command:

```
tracepath <your-storage-backend>
```

You can expect to see output like the following:

```
1?: [LOCALHOST]                 pmtu 1500 ①
1:  <your-storage-backend>                 0.263ms reached ②
1:  <your-storage-backend>                 0.224ms reached ③
   Resume: pmtu 1500 hops 1 back 1
```

① The first line with localhost shows the given MTS size.

② The last line shows the optimal MTU size.

③ If both are identical, nothing needs to be done.

Change Your MTU Value

In case you need or want to change the MTU size, under Ubuntu:

- If NetworkManager is managing all devices on the system, then you can use nmtui or nmcli to configure the MTU setting.
- If NetworkManager is not managing all devices on the system, you can set the MTU to 1280 with Netplan, as in the following example.

```
network:
  version: 2
  ethernets:
    eth0:
      mtu: 1280
```

Refer to the Netplan documentation for further information.



NetworkWorld has an excellent overview of MTU size issues.

Software Requirements

Officially Recommended Environment

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

Platform	Options
Operating System	Ubuntu 18.04 LTS
Database	MariaDB 10+
Web server	Apache 2.4 with prefork and mod_php
PHP Runtime	{recommended-php-version}

Officially Supported Environments

For *best performance*, *stability*, *support*, and *full functionality* we officially support:

Server

Platform	Options
Operating System (64bit)	• Debian 9 and 10 • Fedora 30 and 31 • Red Hat Enterprise Linux/Centos 7.5 and 8 • SUSE Linux Enterprise Server 12 with SP4 and 15 • Ubuntu 16.04 and 18.04 • openSUSE Leap 42.3 and 15
Database	• MySQL 8+ or MariaDB 10+ (**Recommended**) • Oracle 11 and 12 • PostgreSQL 9 and 10 • SQLite (**Not for production**)
Web server	• Apache 2.4 with prefork and mod_php
PHP Runtime	• {supported-php-versions}



For Linux distributions, we support, if technically feasible, the latest two versions per platform and the previous LTS Version.

Hypervisors

- Hyper-V
- VMware ESX
- Xen
- KVM

Desktop Sync Client

We always recommend to use the newest sync client with the latest server release. The latest stable client supports the platforms listed below:

- Linux
  - CentOS 6 & 7 & 8 (**64-bit only**)
  - Debian 8.0 & 9.0 & 10.0
  - Fedora 27 & 28 & 29 & 30 & 31
  - Ubuntu 18.04 & 19.04 & 19.10
  - openSUSE Leap 42.2 & 42.3 & 15.0 & 15.1 & 15.2
- macOS X 10.10+ (**64-bit only**)
- Microsoft Windows 7+

Web Browser

- Edge (current version on Windows 10)
- IE11 or newer (except Compatibility Mode)
- Firefox 60 ESR+

- Chrome 66+
- Safari 10+

Mobile Apps

We always recommend to use the newest mobile apps with the latest server release. The latest stable mobile apps support the platforms listed below:

- iOS 9.0+
- Android 4.4+



You can find out more in the changelog.

Database Requirements

The following database settings are currently required if you're running ownCloud together with a MySQL or MariaDB database:

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

Memory Requirements

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

Installation

In this section, you will find all the information you need for installing ownCloud.

Installing with Docker

ownCloud can be installed using Docker, using the official ownCloud Docker image. This official image works standalone (e.g. for a quick evaluation with docker run -p{std-port-http}:{std-port-http} owncloud/server) but it is designed to work with a data volume in the host filesystem and with separate *MariaDB* and *Redis* containers.

The configuration:

- exposes ports {std-port-http}, allowing for HTTP connections.
- mounts the data and MySQL data directories on the host for persistent storage.

Installation on a Local Machine

To use it, first create a new project directory and download ownCloud's docker-compose.yml file into that new directory. Next, create a .env configuration file, which contains the required configuration settings. Only a few settings are required, these are:

Setting Name	Description	Example
OWNCLOUD_VERSION	The ownCloud version	latest

Setting Name	Description	Example
OWNCLOUD_DOMAIN	The ownCloud domain	localhost
ADMIN_USERNAME	The admin username	admin
ADMIN_PASSWORD	The admin user's password	admin
HTTP_PORT	The HTTP port to bind to	{std-port-http}

Then, you can start the container, using your preferred Docker command-line tool. The example below shows how to use Docker Compose.

```
# Create a new project directory
mkdir owncloud-docker-server

cd owncloud-docker-server

# Copy docker-compose.yml from the GitHub repository
wget
https://raw.githubusercontent.com/owncloud/docs/master/modules/admin_manual/e
xamples/installation/docker/docker-compose.yml

# Create the environment configuration file
cat << EOF > .env
OWNCLOUD_VERSION={latest-version}
OWNCLOUD_DOMAIN=localhost
ADMIN_USERNAME=admin
ADMIN_PASSWORD=admin
HTTP_PORT={std-port-http}
EOF

# Build and start the container
docker-compose up -d
```

When the process completes, then check that all the containers have successfully started, by running docker-compose ps. If they are all working correctly, you should expect to see output similar to that below:

```
Name              Command                  State          Ports
_____________________________________________________________________________
_____
server_db_1        /usr/bin/entrypoint/bin/s ...   Up              {std-port-mysql}/tcp
server_owncloud_1   /usr/local/bin/entrypoint ...   Up              0.0.0.0:{std-port-
http}->{std-port-http}/tcp
server_redis_1      /bin/s6-svscan /etc/s6        Up              {std-port-redis}/tcp
```

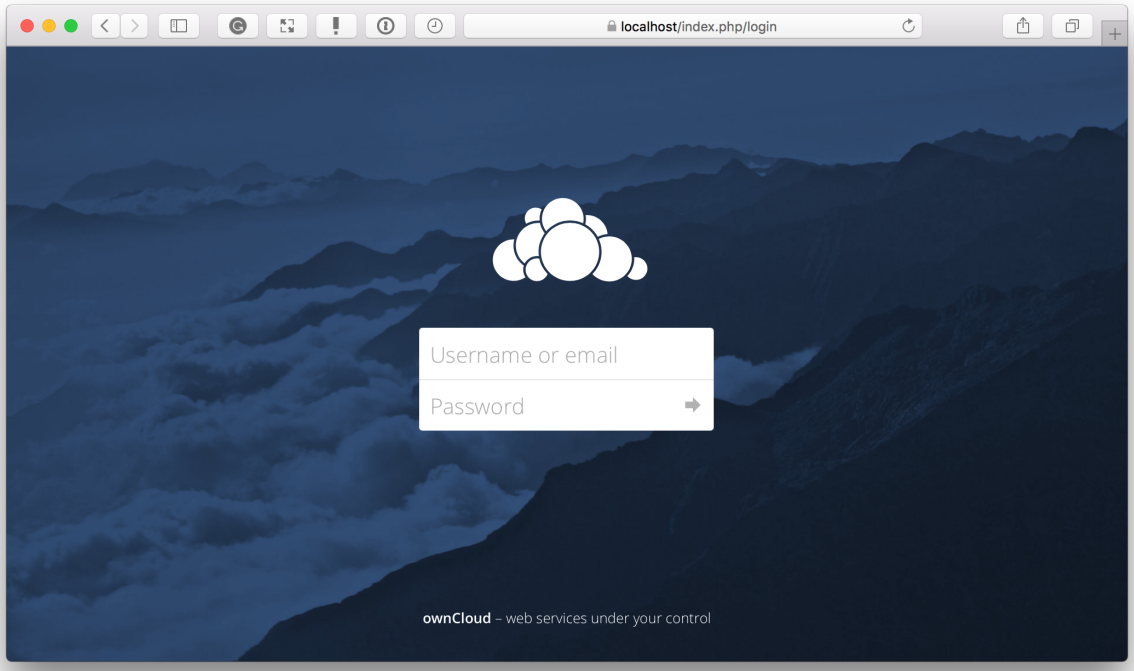
In it, you can see that the database, ownCloud, and Redis containers are running, and that ownCloud is accessible via port {std-port-http} on the host machine.



Just because all the containers are running, it takes a few minutes for ownCloud to be fully functional. If you run docker-compose logs --follow owncloud and see a significant amount of information logging to the console, then please wait until it slows down to attempt to access the web UI.

Logging In

To log in to the ownCloud UI, open http://localhost:{std-port-http} in your browser of choice, where you see the standard ownCloud login screen, as in the image below.



The username and password are the admin username and password which you stored in .env earlier.

Stopping the Containers

Assuming you used docker-compose, as in the previous example, to stop the containers use docker-compose stop. Alternatively, use docker-compose down --rmi all --volumes to stop and remove containers, along with the related networks, images, and volumes.

Upgrading ownCloud on Docker

When a new version of ownCloud gets released, you should update your instance. To do so, follow these simple steps.

First, go to your docker directory where your .yaml or .env file exists. Second, put ownCloud into maintenance mode; you can do so using the following command:

```
docker-compose exec owncloud occ maintenance:mode --on
```

Third, create a backup in case something goes wrong during the upgrade process, using the following command:

```
docker-compose exec db backup
```



This assumes that you are using the default database container from Webhippie.

Fifth, shutdown the containers.

```
docker-compose down
```

Sixth, update the version number of ownCloud in your .env file or the YAML file. You can use sed for it, as in the following example.

```
# Make sure that you adjust the example to match your installation.
sed -i 's/^OWNCLOUD_VERSION=.*$/OWNCLOUD_VERSION=<newVersion>/'
/compose/*/.env
```

Seventh, view the file to ensure the changes has been implemented.

```
cat .env
```

Eighth, start your docker instance again.

```
docker-compose up -d
```

Now you should have the current ownCloud running with docker-compose. Please note that the container will automatically run occ upgrade when starting up. If you notice the container starting over and over again, you can check the update log with the following command:

```
docker-compose logs --timestamp owncloud
```

Docker Compose YAML File



If you are an enterprise customer and are already registered on portal.owncloud.com, replace image: owncloud/server with image: registry.owncloud.com/owncloud/enterprise to be able to download our enterprise docker image. Then, login to our registry by running docker login registry.owncloud.com, along with your portal credentials.

```
version: '2.1'

volumes:
  files:
    driver: local
  mysql:
    driver: local
```

```
  backup:
    driver: local
  redis:
    driver: local

services:
  owncloud:
    image: owncloud/server:${OWNCLOUD_VERSION}
    restart: always
    ports:
      - ${HTTP_PORT}:8080
    depends_on:
      - db
      - redis
    environment:
      - OWNCLOUD_DOMAIN=${OWNCLOUD_DOMAIN}
      - OWNCLOUD_DB_TYPE=mysql
      - OWNCLOUD_DB_NAME=owncloud
      - OWNCLOUD_DB_USERNAME=owncloud
      - OWNCLOUD_DB_PASSWORD=owncloud
      - OWNCLOUD_DB_HOST=db
      - OWNCLOUD_ADMIN_USERNAME=${ADMIN_USERNAME}
      - OWNCLOUD_ADMIN_PASSWORD=${ADMIN_PASSWORD}
      - OWNCLOUD_MYSQL_UTF8MB4=true
      - OWNCLOUD_REDIS_ENABLED=true
      - OWNCLOUD_REDIS_HOST=redis
    healthcheck:
      test: ["CMD", "/usr/bin/healthcheck"]
      interval: 30s
      timeout: 10s
      retries: 5
    volumes:
      - files:/mnt/data

  db:
    image: webhippie/mariadb:latest
    restart: always
    environment:
      - MARIADB_ROOT_PASSWORD=owncloud
      - MARIADB_USERNAME=owncloud
      - MARIADB_PASSWORD=owncloud
      - MARIADB_DATABASE=owncloud
      - MARIADB_MAX_ALLOWED_PACKET=128M
      - MARIADB_INNODB_LOG_FILE_SIZE=64M
    healthcheck:
      test: ["CMD", "/usr/bin/healthcheck"]
      interval: 30s
      timeout: 10s
      retries: 5
    volumes:
```

```
    - mysql:/var/lib/mysql
    - backup:/var/lib/backup

  redis:
    image: webhippie/redis:latest
    restart: always
    environment:
      - REDIS_DATABASES=1
    healthcheck:
      test: ["CMD", "/usr/bin/healthcheck"]
      interval: 30s
      timeout: 10s
      retries: 5
    volumes:
      - redis:/var/lib/redis
```

Troubleshooting

If you have issues logging in to the registry, make sure the .docker file is in your home directory. If you installed Docker via snap, create a symbolic link to your home directory with the following command:

```
ln -sf snap/docker/384/.docker
```

The version 384 might differ from yours. Please adjust it accordingly.

Manual Installation on Linux

Install the Required Packages

When Are Stable Channel Packages Updated?

Packages in the supported distributions' stable channels are not immediately updated following a release. This is because we need to make sure that the release is sufficiently stable, as many people use automatic updates. By waiting a number of business days after a tarball has been released, we are able to make this assessment, based on a number of criteria which include the submitted bug reports from systems administrators.

If you are planning on running additional apps, keep in mind that you might require additional packages. See the prerequisites list for details.

Ubuntu 18.04 LTS Server

To prepare your Ubuntu 18.04 server for the use with ownCloud, follow the Ubuntu 18.04 preparation guide.

Install ownCloud

Now download the archive of the latest ownCloud version:

- Go to the ownCloud Download Page.
- Go to **Download ownCloud Server > Download > Archive file for server owners** and download either the tar.bz2 or .zip archive.
- This downloads a file named owncloud-x.y.z.tar.bz2 or owncloud-x.y.z.zip (where

x.y.z is the version number).

- Download its corresponding checksum file, e.g., owncloud-x.y.z.tar.bz2.md5, or owncloud-x.y.z.tar.bz2.sha256.
- Verify the MD5 or SHA256 sum:

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

- You may also verify the PGP signature:

```
wget https://download.owncloud.org/community/owncloud-x.y.z.tar.bz2.asc
wget https://owncloud.org/owncloud.asc
gpg --import owncloud.asc
gpg --verify owncloud-x.y.z.tar.bz2.asc owncloud-x.y.z.tar.bz2
```

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

```
tar -xjf owncloud-x.y.z.tar.bz2
unzip owncloud-x.y.z.zip
```

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

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

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

```
cp -r owncloud /var/www
```

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

Configure the Web Server

Configure Apache

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

```
Alias /owncloud "/var/www/owncloud/"

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

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

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

```
ln -s /etc/apache2/sites-available/owncloud.conf /etc/apache2/sites-enabled/owncloud.conf
```

Additional Apache Configurations

- For ownCloud to work correctly, we need the module mod_rewrite. Enable it by running: a2enmod rewrite. Additionally recommended modules are mod_headers, mod_env, mod_dir, mod_mime, and mod_unique_id. To enable them, run the following commands:

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

  If you want to use the OAuth2 app, then mod_headers must be installed and enabled.

- You must disable any server-configured authentication for ownCloud, 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 disable the authentication specifically for the ownCloud 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 ownCloud to be reachable via the internet, then set both of these to the domain you want to reach your ownCloud server.
- Now restart Apache

  ```
  service apache2 restart
  ```

- If you're running ownCloud in a sub-directory and want to use CalDAV or CardDAV clients make sure you have configured the correct Service Discovery URLs.

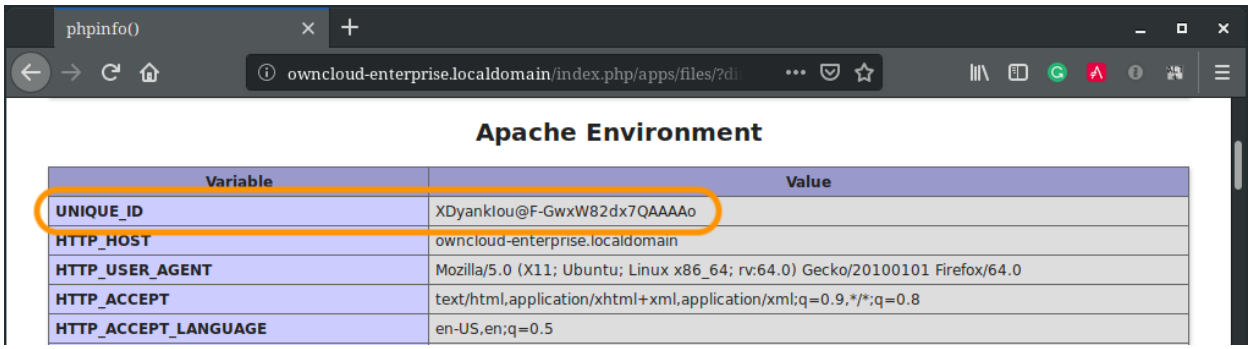
Apache Mod_Unique_Id Configuration

mod_unique_id:

> Provides a magic token for each request which is guaranteed to be unique across "all" requests under very specific conditions.

If you enable the module, there is nothing else that you have to do, as ownCloud automatically includes the UNIQUE_ID environment variable, which the module makes available, in ownCloud's log file.

To confirm that it's working though, check that the UNIQUE_ID environment variable is being set, by running phpinfo() (as in the screenshot below).



Next, compare the value set for UNIQUE_ID in the output of phpinfo() with the value in ownCloud's log file, to ensure that they're the same. In the example below, you can see an example log entry, where ownCloud is logging the unique id provided by Apache, as the value for the first key reqId in the record.


```
{
   "reqId": "XDyankIou@F-GwxW82dx7QAAAAo",
   "level": 3,
   "time": "2019-01-14T14:20:14+00:00",
   "remoteAddr": "127.0.0.1",
   "user": "--",
   "app": "PHP",
   "method": "GET",
   "url": "\/index.php\/apps\/files\/?dir=\/Documents&fileid=26",
   "message": "..."
}
```


Enable SSL



You can use ownCloud 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
```



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

Multi-Processing Module (MPM)

Apache prefork has to be used. Don't use a threaded MPM like event or worker with mod_php, because PHP is currently not thread safe.

Run the 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, temporarily change the ownership on your ownCloud directories to your HTTP user



Refer to the Set Strong Directory Permissions section to learn how to find your HTTP user):

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



Admins of SELinux-enabled distributions may need to write new SELinux rules to complete their ownCloud installation; see the SELinux guide for a suggested configuration.

To use occ refer to the command-line installation details. To use the graphical Installation Wizard refer to the installation_wizard.



Please know that ownCloud's data directory **must be exclusive to ownCloud** and not be modified manually by any other process or user.

Headers



ownCloud has a mechanism to set headers programmatically. These headers are set with the always directive to avoid errors when there are additional headers set in the web servers configuration file like http.conf. More information on headers can be found in the mod_headers documentation.

Set Strong Directory Permissions

After completing the installation, you must immediately set the directory permissions in your ownCloud installation as strictly as possible for stronger security. After you do so, your ownCloud server will be ready to use.

Managing Trusted Domains

All URLs used to access your ownCloud server must be white-listed in your config.php

file, under the trusted_domains setting. Users are allowed to log into ownCloud only when they point their browsers to a URL that is listed in the trusted_domains setting.



This setting is important when changing or moving to a new domain name. You may use IP addresses and domain names.

A typical configuration looks like this:

```
'trusted_domains' => [
  0 => 'localhost',
  1 => 'server1.example.com',
  2 => '192.168.1.50',
],
```

The loopback address, 127.0.0.1, is automatically white-listed, so as long as you have access to the physical server you can always log in. In the event that a load-balancer is in place, there will be no issues as long as it sends the correct X-Forwarded-Host header.



For further information on improving the quality of your ownCloud installation, please see the configuration notes and tips guide.



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

Prerequisites

The ownCloud tar archive contains all of the required third-party PHP libraries. As a result, no extra ones are, strictly, necessary. However, ownCloud does require that PHP has a set of extensions installed, enabled, and configured.

This section lists both the required and optional PHP extensions. If you need further information about a particular extension, please consult the relevant section of the extensions section of the PHP manual.

If you are using a Linux distribution, it should have packages for all the required extensions. You can check the presence of a module by typing php -m | grep -i <module_name>. If you get a result, the module is present.

Required

PHP Version

PHP {supported-php-versions}



ownCloud recommends the use of PHP 7.2 in new installations. Sites using a version earlier than PHP 7.2 are **strongly encouraged** to migrate to PHP 7.2.

PHP Extensions

Name	Description
Ctype	For character type checking

Name	Description
cURL	Used for aspects of HTTP user authentication
DOM	For operating on XML documents through the DOM API
GD	For creating and manipulating image files in a variety of different image formats, including GIF, PNG, JPEG, WBMP, and XPM.
HASH Message Digest Framework	For working with message digests (hash).
iconv	For working with the iconv character set conversion facility.
intl	Increases language translation performance and fixes sorting of non-ASCII characters
JSON	For working with the JSON data-interchange format.
libxml	This is required for the DOM, libxml, SimpleXML, and XMLWriter extensions to work. It requires that libxml2, version 2.7.0 or higher, is installed.
Multibyte String	For working with multibyte character encoding schemes.
OpenSSL	For symmetric and asymmetric encryption and decryption, PBKDF2, PKCS7, PKCS12, X509 and other crypto operations.
PDO	This is required for the pdo_msql function to work.
Phar	For working with PHP Archives (.phar files).
POSIX	For working with UNIX POSIX functionality.
SimpleXML	For working with XML files as objects.
XMLWriter	For generating streams or files of XML data.
Zip	For reading and writing ZIP compressed archives and the files inside them.
Zlib	For reading and writing gzip (.gz) compressed files.



The *Phar*, *OpenSSL*, and *cUrl* extensions are mandatory if you want to use Make to setup your ownCloud environment, prior to running either the web installation wizard, or the command line installer.

Database Extensions

Name	Description
pdo_mysql	For working with MySQL & MariaDB.
pgsql	For working with PostgreSQL. It requires PostgreSQL 9.0 or above.
sqlite	For working with SQLite. It requires SQLite 3 or above. This is, usually, not recommended for performance reasons.

Required For Specific Apps

Name	Description
ftp	For working with FTP storage
sftp	For working with SFTP storage
imap	For IMAP integration
ldap	For LDAP integration
smbclient	For SMB/CIFS integration



SMB/Windows Network Drive mounts require the PHP module smbclient version 0.8.0+. See SMB/CIFS.

Optional

Extension	Reason
Bzip2	Required for extraction of applications
Fileinfo	Highly recommended, as it enhances file analysis performance
Mcrypt	Increases file encryption performance
OpenSSL	Required for accessing HTTPS resources
imagick	Required for creating and modifying images and preview thumbnails

Recommended

For Specific Apps

Extension	Reason
Exif	For image rotation in the pictures app
GMP	For working with arbitrary-length integers

For Server Performance

For enhanced server performance consider installing one of the following cache extensions:

- apcu
- memcached
- redis (>= 2.2.6+, required for transactional file locking)

See Caching Configuration to learn how to select and configure Memcache.

For Preview Generation

- avconv or ffmpeg
- OpenOffice or LibreOffice

For Command Line Processing

Extension	Reason
PCNTL	Enables command interruption by pressing ctrl-c



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

For MySQL/MariaDB

The InnoDB storage engine is required, and MyISAM is not supported, see MySQL / MariaDB storage engine for more information.

The Installation Wizard

Introduction

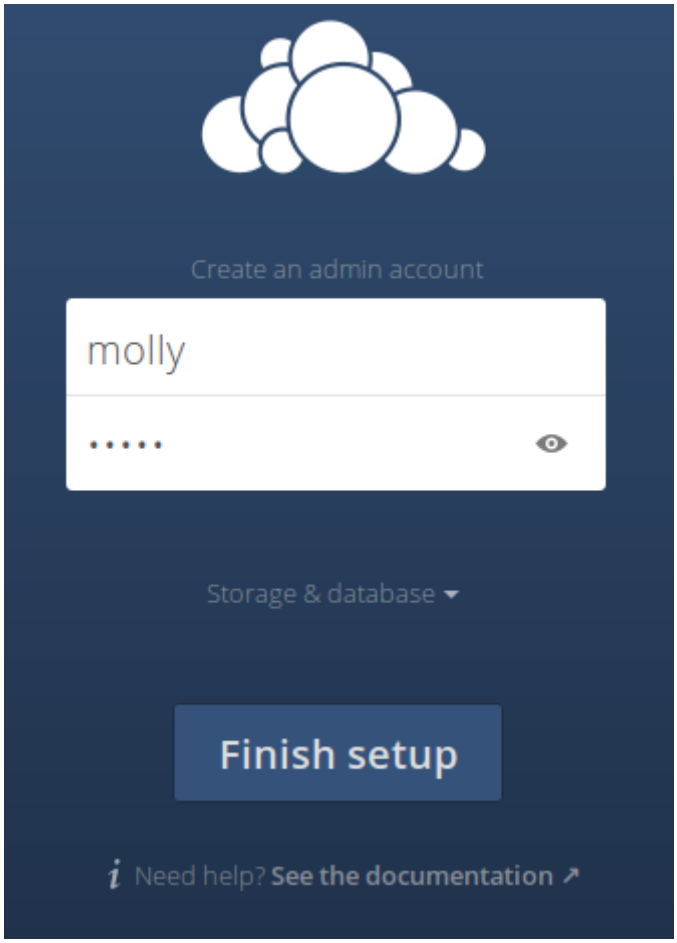


If you are planning to use the installation wizard, we **strongly** encourage you to protect it, through some form of password authentication, or access control. If the installer is left unprotected when exposed to the public internet, there is the possibility that a malicious actor could finish the installation and block you out — or worse. So please ensure that only you — or someone from your organization — can access the web installer.

Quick Start

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

1. Point your web browser to http://localhost/owncloud
2. Enter your desired administrator's username and password
3. Click btn:[Finish Setup]



You’re now finished and can start using your new ownCloud server. Of course, there is much more that you *can* do to set up your ownCloud server for best performance and security. In the following sections we will cover important installation and post-installation steps. Note that you must follow the instructions in Setting Strong Permissions in order to use the occ command.

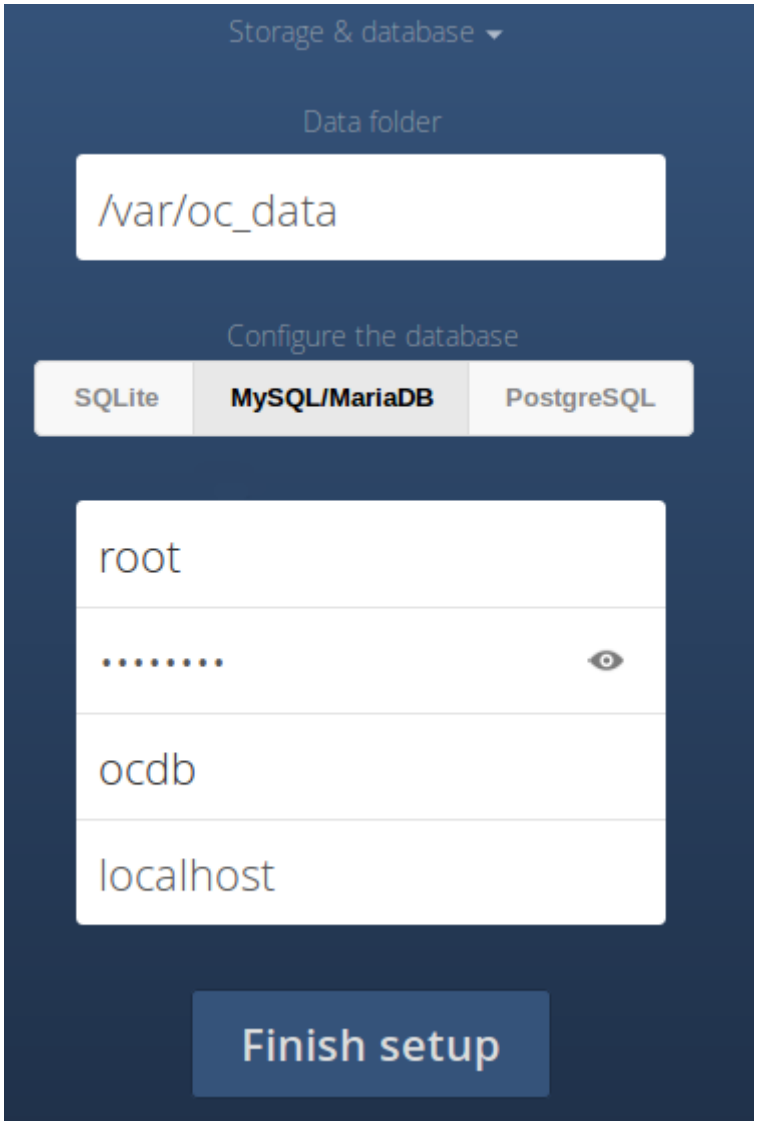
In-Depth Guide

This section provides a more detailed guide to the installation wizard. Specifically, it is broken down into three steps:

1. Data Directory Location
2. Database Choices
3. Post-Installation Steps

Data Directory Location

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



You should locate your ownCloud data directory outside of your Web root if you are using an HTTP server other than Apache, or you may wish to store your ownCloud data in a different location for other reasons (e.g., on a storage server).

ownCloud's data directory **must be exclusive to ownCloud** and not be modified manually by any other process or user.

It is best to configure your data directory location at installation, as it is difficult to move after installation. You may put it anywhere; in this example is it located in /var/oc_data. This directory must already exist, and must be owned by your HTTP user.

Database Choices

When installing ownCloud Server & ownCloud Enterprise editions the administrator may choose one of 4 supported database products. These are:

- SQLite
- MYSQL/MariaDB
- PostgreSQL
- Oracle 11g (Enterprise-edition only)

SQLite



SQLite is the default database for ownCloud Server — but is not supported by the ownCloud Enterprise edition.

SQLite is only good for testing and lightweight single user setups. It has no client synchronization support, so other devices will not be able to synchronize with the data stored in an ownCloud SQLite database.

SQLite will be installed by the ownCloud package and all the necessary dependencies will be satisfied. If you used the package manager to install ownCloud, you may "Finish Setup" with no additional steps to configure ownCloud using the SQLite database for limited use.

MYSQL/MariaDB

MariaDB is the ownCloud recommended database. It may be used with either ownCloud Server or ownCloud Enterprise editions. To install the recommended MySQL/MariaDB database, use the following command:

```
sudo apt-get install mariadb-server
```

If you have an administrator login that has permissions to create and modify databases, you may choose "Storage & Database". Then, enter your database administrator username and password, and the name you want for your ownCloud database. Alternatively, you can use these steps to create a temporary database administrator account.

```
sudo mysql --user=root mysql
CREATE USER 'dbadmin'@'localhost' IDENTIFIED BY 'Apassword';
GRANT ALL PRIVILEGES ON *.* TO 'dbadmin'@'localhost' WITH GRANT OPTION;
FLUSH PRIVILEGES;
exit
```

For more detailed information, see MySQL/MariaDB <system_requirements>.

PostgreSQL

PostgreSQL is also supported by ownCloud. To install it, use the following command (or that of your preferred package manager):

```
sudo apt-get install postgresql
```

In order to allow ownCloud access to the database, create a known password for the default user, postgres, which was added when the database was installed.

```
sudo -i -u postgres psql
postgres=# \password
Enter new password:
Enter it again:
postgres=# \q
exit
```

Oracle 11g

Oracle 11g is only supported for the ownCloud Enterprise edition.

Database Setup By ownCloud

Your database and PHP connectors must be installed before you run the Installation Wizard by clicking the btn:[Finish setup] button. After you enter your temporary or root administrator login for your database, the installer creates a special database user with privileges limited to the ownCloud database.

Following this, ownCloud needs only this special ownCloud database user and drops the temporary or root database login. This new user is named from your ownCloud admin user, with an oc_ prefix, and given a random password. The ownCloud database user and password are written into config.php:

For MySQL/MariaDB:

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

For PostgreSQL:

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

Click Finish Setup, and you're ready to start using your new ownCloud server.

Post-Installation Steps

Now we will look at some important post-installation steps. For hardened security we recommend setting the permissions on your ownCloud directories as strictly as possible, and for proper server operations. This should be done immediately after the initial installation and before running the setup.

Your HTTP user must own the config/, data/, apps/ respectively the apps-external/ directories so that you can configure ownCloud, create, modify and delete your data files, and install apps via the ownCloud Web interface.

You can find your HTTP user in your HTTP server configuration files, or you can use label-phpinfo (Look for the **User/Group** line).

- 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.



When using an NFS mount for the data directory, do not change its ownership from the default. The simple act of mounting the drive will set proper permissions for ownCloud to write to the directory. Changing ownership as above could result in some issues if the NFS mount is lost.

The easy way to set the correct permissions is to copy and run the script, below. The script sets proper permissions and ownership including the handling of necessary directories. The script also prepares for an apps-external directory, for details see

config.sample.php:

- Replace the ocpath variable with the path to your ownCloud directory.
- Replace the ocdata variable with the path to your ownCloud data directory.
- Replace the apps_external variable with the path to your ownCloud apps-external directory.

In case use want to use links for the data and apps-external directory:

- Replace the linkdata variable with the path to your ownCloud linked data directory.
- Replace the linkapps-external variable with the path to your ownCloud linked apps-external directory.

Set the correct HTTP user and group according your needs:

- Replace the htuser and htgroup variables with your HTTP user and group.

In case of upgrading using tar:

- Replace the oldocpath variable with the path to your old ownCloud directory.

If you have customized your ownCloud installation and your file paths are different than the standard installation, modify this script accordingly.

This summary lists the recommended modes and ownership for your ownCloud directories and files:

- All files should be read-write for the file owner, read-only for the group owner, and zero for the world
- All directories should be executable (because directories always need the executable bit set), read-write for the directory owner, and read-only for the group owner
- The apps/ directory should be owned by [HTTP user]:[HTTP group]
- The apps-external/ directory should be owned by [HTTP user]:[HTTP group]
- The config/ directory should be owned by [HTTP user]:[HTTP group]
- The data/ directory should be owned by [HTTP user]:[HTTP group]
- The updater/ directory should be owned by [HTTP user]:[HTTP group]
- The [ocpath]/.htaccess file should be owned by root:[HTTP group]
- The data/.htaccess file should be owned by root:[HTTP group]
- Both .htaccess files are read-write file owner, read-only group and world

These strong permissions prevent upgrading your ownCloud server; see Setting Permissions for Updating for a script to quickly change permissions to allow upgrading.

```
#!/bin/bash

# To setup this script for your environment, adopt the following variables to your
needs:
#
# ocname    the name of your directory containing the owncloud files
# ocroot    the path to ocname, usually /var/www (no trailing slash)
# linkroot  the path to your source directory for linking data and apps-external (no
```

```
trailing slash)
# htuser    the webserver user
# htgroup   the webserver group
# rootuser  the root user

# Short description for paramters used in find
#
# -L       ... Follow symbolic links. Needed in case if links are used or present
# -path    ... The path to process
# -prune   ... If the file is a directory, do not descend into it (used to exclude
directories)
# -o       ... OR (to add more parameters)
# -type    ... File is of type [d ... directory, f ... file]
# -print0  ... Print the full file name on the standard output, followed by a null
character
# xargs -0 ... Reads items from the standard input, input items are terminated by a
null character


ocname='owncloud'
ocroot='/var/www'
ocpath=$ocroot/$ocname
ocdata=$ocroot/$ocname/'data'
ocapps_external=$ocpath/'apps-external'
oldocpath=$ocroot/$ocname'_'$(date +%F-%H.%M.%S)

linkroot='/var/mylinks'
linkdata=$linkroot/'data'
linkapps_external=$linkroot/'apps-external'

htuser='www-data'
htgroup='www-data'
rootuser='root'

# Because the data directory can be huge or on external storage, an automatic
chmod/chown can take a while.
# Therefore this directory can be treated differently.
# If you have already created an external "data" and "apps-external" directory
which you want to link,
# set the paths above accordingly. This script can link and set the proper rights and
permissions
# depending what you enter when running the script.

# When the instance is setup, either post a fresh install or, after an upgrade, run
this script again - but
# only for securing ".htaccess files". This sets the appropriate ownership and
permissions for them.

# In case you upgrade an existing installation, your original directory will be
renamed, including a timestamp.
```

```
# Example input (without securing your .htaccess files)
# New install using mkdir:      n/n/n/n (create possible missing directories, setup
permissions and ownership)
# Upgrade using mkdir:          y/n/n/n (you move/replace data, apps-external and
config.php manually, set setup permissions and ownership)
# New install using links:      n/y/y/n (link existing directories, setup permissions and
ownership)
# Upgrade using links:          y/y/n/y (link existing directories, copy config.php,
permissions and ownership are already ok)
# Post installation/upgrade:    either n/n/n/n or n/y/y/n
# Reset all perm & own:         either n/n/n/n or n/y/y/n

echo
read -p "Do you want to secure your .htaccess files post installing/upgrade (y/N)? "
-r -e answer
(echo "$answer" | grep -iq "^y") && do_secure="y" || do_secure="n"

if [ "$do_secure" = "y" ]; then
  printf "\nSecuring .htaccess files with chmod/chown\n"
  if [ -f ${ocpath}/.htaccess ]; then
    chmod 0644 ${ocpath}/.htaccess
    chown ${rootuser}:${htgroup} ${ocpath}/.htaccess
  fi
  if [ -f ${ocdata}/.htaccess ];then
    chmod 0644 ${ocdata}/.htaccess
    chown ${rootuser}:${htgroup} ${ocdata}/.htaccess
  fi
  echo
  exit
fi


read -p "Do you want to upgrade an existing installation (y/N)? " -r -e answer
(echo "$answer" | grep -iq "^y") && do_upgrade="y" || do_upgrade="n"

read -p "Do you want to use ln instead of mkdir for creating directories (y/N)? " -r -e
answer
(echo "$answer" | grep -iq "^y") && uselinks="y" || uselinks="n"

read -p "Do you also want to chmod/chown these links (y/N)? " -r -e answer
(echo "$answer" | grep -iq "^y") && chmdir="y" || chmdir="n"

if [ "$do_upgrade" = "y" ]; then
  read -p "Do you want to copy an existing config.php file (y/N)? " -r -e answer
  (echo "$answer" | grep -iq "^y") && upgrdcfg="y" || upgrdcfg="n"
fi


# check if upgrading an existing installation
```

```
if [ "$do_upgrade" = "y" ]; then
  read -p "Is the instance in maintenance mode? (y/N)? " -r -e answer
  (echo "$answer" | grep -iq "^y") && mmode="y" || mmode="n"
  if [ "$mmode" = "n" ]; then
    echo "Please enable maintenance mode first: sudo -uwww-data ./occ
maintenance:mode --on"
    echo
    exit
  fi

  read -p "Please specify the tar file to extract with full path: " -r -e tarFile
  if [ ! -f "$tarFile" ]; then
    echo "tar file to extract not found. Exiting."
    echo
    exit
  fi

  if [ -d ${ocpath} ]; then
    mv $ocpath $oldocpath
  fi

  mkdir -p $ocpath
  tar xvf "$tarFile" -C $ocpath --strip-components=1

  if [ $? != 0 ]; then
    echo
    echo "tar extract failed, please check !"
    echo
    exit
  fi
fi

# create / link missing directories
printf "\nCreating or linking possible missing directories \n"
mkdir -p $ocpath/updater
# check if directory creation is possible and create if ok
if [ "$uselinks" = "n" ]; then
  if [ -L ${ocdata} ]; then
    echo "Symlink for $ocdata found but mkdir requested. Exiting."
    echo
    exit
  else
    echo "mkdir $ocdata"
    echo
    mkdir -p $ocdata
  fi
  if [ -L ${ocapps_external} ]; then
    echo "Symlink for $ocapps_external found but mkdir requested. Exiting."
    echo
    exit
```

```
  else
    printf "mkdir $ocapps_external \n"
    mkdir -p $ocapps_external
  fi
else
  if [ -d ${ocdata} ]; then
    echo "Directory for $ocdata found but link requested. Exiting."
    echo
    exit
  else
    printf "ln $ocdata \n"
    mkdir -p $linkdata
    ln -sfn $linkdata $ocdata
  fi
  if [ -d ${ocapps_external} ]; then
    echo "Directory for $ocapps_external found but link requested. Exiting."
    echo
    exit
  else
    printf "ln $ocapps_external \n"
    mkdir -p $linkapps_external
    ln -sfn $linkapps_external $ocapps_external
  fi
fi

# copy if requested an existing config.php
if [ "$upgrdcfg" = "y" ]; then
  if [ -f ${oldocpath}/config/config.php ]; then
    printf "\nCopy existing config.php file \n"
    cp ${oldocpath}/config/config.php ${ocpath}/config/config.php
  else
    printf "Skip to copy old config.php, file not found: ${oldocpath
}/config/config.php \n"
  fi
fi

printf "\nchmod files and directories excluding data and apps-external directory \n"

# check if there are files to chmod/chown available. If not exiting.
# chmod
if [ ! "$(find $ocpath -maxdepth 1 -type f)" ]; then
  echo "Something is wrong. There are no files to chmod. Exiting."
  exit
fi

find -L ${ocpath} -path ${ocdata} -prune -o -path ${ocapps_external} -prune
-o -type f -print0 | xargs -0 chmod 0640
find -L ${ocpath} -path ${ocdata} -prune -o -path ${ocapps_external} -prune
-o -type d -print0 | xargs -0 chmod 0750
```

```
# no error messages on empty directories
if [ "$chmdir" = "n" ] && [ "$uselinks" = "n" ]; then

  printf "chmod data and apps-external directory (mkdir) \n"

  if [ -n "$(ls -A $ocdata)" ]; then
    find ${ocdata}/ -type f -print0 | xargs -0 chmod 0640
  fi
  find ${ocdata}/ -type d -print0 | xargs -0 chmod 0750
  if [ -n "$(ls -A $ocapps_external)" ]; then
    find ${ocapps_external}/ -type f -print0 | xargs -0 chmod 0640
  fi
  find ${ocapps_external}/ -type d -print0 | xargs -0 chmod 0750
fi

if [ "$chmdir" = "y" ] && [ "$uselinks" = "y" ]; then

  printf "chmod data and apps-external directory (linked) \n"

  if [ -n "$(ls -A $ocdata)" ]; then
    find -L ${ocdata}/ -type f -print0 | xargs -0 chmod 0640
  fi
  find -L ${ocdata}/ -type d -print0 | xargs -0 chmod 0750
  if [ -n "$(ls -A $ocapps_external)" ]; then
    find -L ${ocapps_external}/ -type f -print0 | xargs -0 chmod 0640
  fi
  find -L ${ocapps_external}/ -type d -print0 | xargs -0 chmod 0750
fi

#chown
printf "\nchown files and directories excluding data and apps-external directory \n"

find  -L $ocpath  -path ${ocdata} -prune -o -path ${ocapps_external} -prune -o
-type d -print0 | xargs -0 chown ${rootuser}:${htgroup}
find  -L $ocpath  -path ${ocdata} -prune -o -path ${ocapps_external} -prune -o
-type f -print0 | xargs -0 chown ${rootuser}:${htgroup}

# do only if directories are present
if [ -d ${ocpath}/apps/ ]; then
  printf "chown apps directory \n"
  chown -R ${htuser}:${htgroup} ${ocpath}/apps/
fi
if [ -d ${ocpath}/config/ ]; then
  printf "chown config directory \n"
  chown -R ${htuser}:${htgroup} ${ocpath}/config/
fi
if [ -d ${ocpath}/updater/ ]; then
  printf "chown updater directory \n"
  chown -R ${htuser}:${htgroup} ${ocpath}/updater
fi
```

```
if [ "$chmdir" = "n" ] && [ "$uselinks" = "n" ]; then
  printf "chown data and apps-external directories (mkdir) \n"
  chown -R ${htuser}:${htgroup} ${ocapps_external}/
  chown -R ${htuser}:${htgroup} ${ocdata}/
fi
if [ "$chmdir" = "y" ] && [ "$uselinks" = "y" ]; then
  printf "chown data and apps-external directories (linked) \n"
  chown -R ${htuser}:${htgroup} ${ocapps_external}/
  chown -R ${htuser}:${htgroup} ${ocdata}/
fi

printf "\nchmod occ command to make it executable \n"
if [ -f ${ocpath}/occ ]; then
  chmod +x ${ocpath}/occ
fi


# Tell the user to remove the old instance, do an upgrade, and to end maintenance
mode etc., if all is fine.
if [ "$do_upgrade" = "y" ]; then
  echo "Please manually remove the directory of the old instance: $oldocpath"
  echo "Please manually run: sudo -uwww-data ./occ upgrade"
  echo "Please manually run: sudo -uwww-data ./occ maintenance:mode --off"
  echo "When all is done successfully, re-run this script and secure your .htaccess
files"
  echo
fi
```

Command Line Installation

ownCloud can be installed entirely from the command line. This is convenient for scripted operations and for systems administrators who prefer using the command line over a GUI. It involves five steps:

1. Ensure your server meets the ownCloud prerequisites
2. Download and unpack the source
3. Install using the occ command
4. Set the correct owner and permissions
5. Optional post#.installation considerations

Let's begin. To install ownCloud, first download the source (whether community or enterprise) directly from ownCloud, and then unpack (decompress) the tarball into the appropriate directory.

With that done, you next need to set your webserver user to be the owner of your unpacked owncloud directory, as in the example below.

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

With those steps completed, next use the occ command, from the root directory of the

ownCloud source, to perform the installation. This removes the need to run the Graphical Installation Wizard. Here's an example of how to do it

```
# Assuming you've unpacked the source to /var/www/owncloud/
$ cd /var/www/owncloud/
$ {occ-command-example-prefix} maintenance:install \
  --database "mysql" --database-name "owncloud" \
  --database-user "root" --database-pass "password" \
  --admin-user "admin" --admin-pass "password"
```



You must run occ as your HTTP user.

If you want to use a directory other than the default (which is data inside the root ownCloud directory), you can also supply the --data-dir switch. For example, if you were using the command above and you wanted the data directory to be /opt/owncloud/data, then add --data-dir /opt/owncloud/data to the command.

When the command completes, apply the correct permissions to your ownCloud files and directories.



This is extremely important, as it helps protect your ownCloud installation and ensure that it will operate correctly.

Linux Package Manager Installation

Add the ownCloud Repository

Before you can install owncloud-files, you need to add ownCloud's repository to your distribution's package manager.



Package managers should only be used for single-server setups. For production environments, we recommend installing from the tar archive.

Available Packages

The recommended package to use is owncloud-files. It only installs ownCloud, and does not install Apache, a database, or any of the required PHP dependencies.

Avoid Automatic Upgrades

If you are installing ownCloud using one of the various Linux package managers, we **strongly** recommend that you avoid automatically updating the owncloud-files package, when running a system update or upgrade and when upgrading other packages. That way, there are no surprise changes (whether positive or negative) to your ownCloud installation.

Here are the ways to do so for APT, Yum, and Zypper.

APT

If you are using APT, use apt-mark hold to mark the owncloud-files package as held. Here's an example of how to do so:

```
apt-mark hold owncloud-files
```

To see if owncloud-files has already been held, use the showhold command, as in the following example. If it's printed out to the console, then it's being held.

```
apt-mark showhold owncloud-files
```

To unset owncloud-files as held back, use the unhold command, as in the example below.

```
apt-mark unhold owncloud-files
```

Yum

If you are using Yum, there are two options that you can take to lock packages from being upgraded. You can:

1. Add exclude=owncloud-files to /etc/yum.conf
2. Use the versionlock plugin for Yum.

The VersionLock Plugin

If the versionlock plugin is not installed, install it by running:

```
yum install yum-plugin-versionlock
```

When it is installed, you can lock owncloud-files run:

```
yum versionlock add owncloud-files
```

To confirm that it is locked, run:

```
yum versionlock list
```

To unlock owncloud-files, run:

```
yum versionlock delete owncloud-files
```

Zypper

If you are using Zypper, use the addlock or al commands. Similar to apt-mark hold these add a package lock that prevents the package from being modified. The example below shows how to use the command to lock owncloud-files.

```
zypper addlock owncloud-files
```

To see if the package has already been locked, use the locks command. If owncloud-files is already locked, then you will see output similar to the below example.

```
# | Name          | Type    | Repository
--+---------------+---------+-----------
1 | owncloud-files | package | (any)
```

To unlock owncloud-files, if it is already locked, use the removelocks or rl commands, as in the example below.

```
zypper removelock owncloud-files
```

Installing ownCloud Community Edition

First, install your own LAMP stack, as doing so allows you to create your own custom LAMP stack without dependency conflicts with the ownCloud package. Then, update package manager's configuration.

Configurations are available for the following Linux distributions:

- Ubuntu 14.04 & 16.04
- Debian 7 & 8
- RHEL 6 & 7
- CentOS 7.2 & 7.3
- SLES 11SP4 & 12SP2
- openSUSE Leap 42.2 & 42.3



Repositories for Fedora, openSUSE Tumbleweed, and Ubuntu 15.04 have been dropped. If you use Fedora, use the tar archive with your own LAMP stack. openSUSE users can rely on LEAP packages for Tumbleweed.

Once your package manager has been updated, follow the rest of the instructions on the download page to install ownCloud. Once ownCloud's installed, run the Installation Wizard to complete your installation.



See the system_requirements for the recommended ownCloud setup and supported platforms.



Do not move the folders provided by these packages after the installation, as this will break updates.

What is the Correct Version?

Package versions are composed of a major, a minor, and a patch number, such as 9.0, 9.1, 10.0, 10.0.1, and 10.0.2. The second number represents a major release, and the third number represents a minor release.

Major Releases

If you want to follow either of the most recent major releases, then substitute version with either 9.0 or 10.0.

Minor Releases

If you want to follow any of the four most recent patch releases, then substitute version with one of 10.0.1, 10.0.2, 10.0.3, or 10.0.4. Following a minor release avoids you accidentally upgrading to the next major release before you're ready.

The Latest Stable Version

Alternatively you can use stable for the latest stable version. If you do, you never have to change it as it always tracks the current stable ownCloud version through all major releases.

Installing ownCloud Enterprise Edition

See the enterprise installation guide for instructions on installing ownCloud Enterprise edition.

Downgrading

Downgrading is not supported and risks corrupting your data! If you want to revert to an older ownCloud 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 ownCloud forums to see if your issue can be resolved without downgrading.

Additional Guides and Notes

See installation_wizard for important steps, such as choosing the best database and setting correct directory permissions. See the SELinux guide for a suggested configuration for SELinux-enabled distributions such as *Fedora* and *CentOS*.

If your distribution is not listed, your Linux distribution may maintain its own ownCloud packages or you may prefer to install from source.

Archlinux

The current client stable version is in the official community repository, more packages are in the Arch User Repository.

Mageia

The Mageia Wiki has a good page on installing ownCloud from the Mageia software repository.

Note for MySQL/MariaDB environments

Please refer to MySQL / MariaDB with Binary Logging Enabled on how to correctly configure your environment if you have binary logging enabled.

Running ownCloud in a sub-directory

If you're running ownCloud in a sub-directory and want to use CalDAV or CardDAV clients, make sure you have configured the correct service discovery URLs.

Install ownCloud on Ubuntu 18.04

This is an ultra-short guide to installing ownCloud on a fresh installation of Ubuntu 18.04. Run the following commands in your terminal to complete the installation.

Prerequisites

- A fresh install of Ubuntu 18.04 with SSH enabled.

- This guide assumes that you are connected as the root user.

Preparation

First, ensure that all of the installed packages are entirely up to date, and that PHP is available in the APT repository. To do so, follow the instructions below:

```
apt update && \
  apt upgrade -y
```

Create the occ Helper Script

Create a helper script to simplify running occ commands.

```
FILE="/usr/local/bin/occ"
/bin/cat <<EOM >$FILE
#! /bin/bash

cd /var/www/owncloud
sudo -u www-data /usr/bin/php /var/www/owncloud/occ "\$@"
EOM
```

Make helper script executable:

```
chmod +x /usr/local/bin/occ
```

Install the Required Packages

```
apt install -y \
  apache2 \
  libapache2-mod-php \
  mariadb-server \
  openssl \
  php-imagick php-common php-curl \
  php-gd php-imap php-intl \
  php-json php-mbstring php-mysql \
  php-ssh2 php-xml php-zip \
  php-apcu php-redis redis-server \
  wget
```

Install the Recommended Packages

```
apt install -y \
  ssh bzip2 rsync curl jq \
  inetutils-ping smbclient\
  php-smbclient coreutils php-ldap
```



Ubuntu 18.04 includes smbclient 4.7.6, which has a known limitation of only using version 1 of the SMB protocol.

Installation

Configure Apache

Change the Document Root

```
sed -i "s#html#owncloud#" /etc/apache2/sites-available/000-default.conf

service apache2 restart
```

Create a Virtual Host Configuration

```
FILE="/etc/apache2/sites-available/owncloud.conf"
sudo /bin/cat <<EOM >$FILE
Alias /owncloud "/var/www/owncloud/"

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

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

 SetEnv HOME /var/www/owncloud
 SetEnv HTTP_HOME /var/www/owncloud
</Directory>
EOM
```

Enable the Virtual Host Configuration

```
a2ensite owncloud.conf
service apache2 reload
```

Configure the Database

```
mysql -u root -e "CREATE DATABASE IF NOT EXISTS owncloud; \
GRANT ALL PRIVILEGES ON owncloud.* \
  TO owncloud@localhost \
  IDENTIFIED BY 'password'";
```

Enable the Recommended Apache Modules

```
echo "Enabling Apache Modules"

a2enmod dir env headers mime rewrite setenvif
service apache2 reload
```

Setup ownCloud

Download ownCloud

```
cd /var/www/
wget https://download.owncloud.org/community/owncloud-10.4.1.tar.bz2 && \
tar -xjf owncloud-10.4.1.tar.bz2 && \
chown -R www-data. owncloud
```

Install ownCloud

```
occ maintenance:install \
   --database "mysql" \
   --database-name "owncloud" \
   --database-user "owncloud" \
   --database-pass "password" \
   --admin-user "admin" \
   --admin-pass "admin"
```

Configure ownCloud's Trusted Domains

```
myip=$(hostname -I|cut -f1 -d ' ')
occ config:system:set trusted_domains 1 --value="$myip"
```

Set Up a Cron Job

```
echo "*/15  *  *  *  * /usr/bin/php /path/to/your/owncloud/occ system:cron" \
  > /var/spool/cron/crontabs/www-data
chown www-data.crontab /var/spool/cron/crontabs/www-data
chmod 0600 /var/spool/cron/crontabs/www-data
```



If you need to sync your users from an LDAP or Active Directory Server, add this additional Cron job.

```
echo "*/15  *  *  *  * /usr/bin/php /path/to/your/owncloud/occ
system:cron" > /var/spool/cron/crontabs/www-data
chown www-data.crontab  /var/spool/cron/crontabs/www-data
chmod 0600  /var/spool/cron/crontabs/www-data
```

Configure Caching and File Locking

Execute these commands:

```
occ config:system:set \
  memcache.local \
  --value '\OC\Memcache\APCu'

occ config:system:set \
  memcache.locking \
  --value '\OC\Memcache\Redis'

occ config:system:set \
  redis \
  --value '{"host": "127.0.0.1", "port": "6379"}' \
  --type json
```

Configure Log Rotation

Execute this command to set up log rotation.

```
FILE="/etc/logrotate.d/owncloud"
sudo /bin/cat <<EOM >$FILE
/var/www/owncloud/data/owncloud.log {
  size 10M
  rotate 12
  copytruncate
  missingok
  compress
  compresscmd /bin/gzip
}
EOM
```

Finalise the Installation

Make sure the permissions are correct

```
cd /var/www/
chown -R www-data. owncloud
```

ownCloud is now installed. You can confirm that it is ready to use by pointing your web browser to your ownCloud installation.

Configuration Notes and Tips

SELinux

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

php.ini

Several core PHP settings must be configured correctly, otherwise ownCloud may not work properly. Known settings causing issues are listed here. Please note that, there might be other settings which cause unwanted behavior. In general, however, it is recommended to keep the php.ini settings at their defaults, except when you know exactly why the change is required, and its implications.



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

php.ini - Used by the Web server

For PHP version {minimum-php-version} onward, replace php_version with the version number installed, e.g., {minimum-php-version} in the following examples.

```
/etc/php/[php_version]/apache2/php.ini
```

or

```
/etc/php/[php_version]/fpm/php.ini
```

or

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

```
/etc/php/[php_version]/cli/php.ini
```

session.auto_start && enable_post_data_reading

Ensure that session.auto_start is set to 0 or Off and enable_post_data_reading to 1 or On in your configuration. If not, you may have issues logging in to ownCloud via the WebUI, where you see the error: "*Access denied. CSRF check failed*".

session.save_path

In addition to setting session.auto_start and enable_post_data_reading correctly, ensure that, if session.save_handler is set to files, that session.save_path is set to a path on the filesystem which **only** the web server process (or process which PHP is running as) can read from and write to.

This is especially important if your ownCloud installation is using a shared-hosting arrangement. In these situations, session poisoning can occur if all of the session files are stored in the same location. Session poisoning is where one web application can manipulate data in the $_SESSION superglobal array of another.

When this happens, the original application has no way of knowing that this corruption has occurred and may not treat the data with any sense of suspicion. You can read through a thorough discussion of local session poisoning if you'd like to know more.

post_max_size

Please ensure that you have post_max_size configured with *at least* the minimum amount of memory for use with ownCloud, which is 512 MB.



Please be careful when you set this value if you use the byte value shortcut as it is very specific. Use K for kilobyte, M for megabyte and G for gigabyte. KB, MB, and GB **do not work!**

realpath_cache_size

This determines the size of the realpath cache used by PHP. This value should be increased on systems where PHP opens many files, to reflect the number of file operations performed. For a detailed description see realpath-cache-size. This setting has been available since PHP 5.1.0. Prior to PHP 7.0.16 and 7.1.2, the default was 16 KB.

To see your current value, query your phpinfo() output for this key. It is recommended to set the value if it is currently set to the default of 16 KB. A good reading about the background can be found at tideways.io.

How to get a working value

With the assumption of 112 bytes per file path needed, this would allow the cache to hold around 37.000 items with a cache size of 4096K (4M), but only about a hundred entries for a cache size of 16 KB.



It's a good rule of thumb to always have a realpath cache that can hold entries for all your files paths in memory. If you use symlink deployment, then set it to double or triple the amount of files.

The easiest way to get the quantity of PHP files is to use cloc, which can be installed by running sudo apt-get install cloc. The cloc package is available for nearly all distributions.

```
sudo cloc /var/www/owncloud --exclude-dir=data --follow-links
   12179 text files.
   11367 unique files.
   73126 files ignored.

http://cloc.sourceforge.net v 1.60  T=1308.98 s (6.4 files/s, 1283.5 lines/s)
-------------------------------------------------------------------------------
Language                     files       blank       comment          code
-------------------------------------------------------------------------------
PHP                           4896       96509        285384        558135
...
```

Taking the math from above and assuming a symlinked instance, using factor 3. For example: 4896 * 3 * 112 = 1.6MB This result shows that you can run with the PHP setting of 4M two instances of ownCloud.

Having the default of 16 KB means that only 1/100 of the existing PHP file paths can be cached and need continuous cache refresh slowing down performance. If you run more web services using PHP, you have to calculate accordingly.

PHP-FPM

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:

Ubuntu/Mint	CentOS/Red Hat/Fedora
/etc/php/[php_version]/fpm/	/etc/php-fpm.d/

In both examples, the ini/config file is called www.conf, and depending on the distribution or customizations which you have made, it may be in a sub-directory.

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

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

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

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

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

When you are using shared hosting or a control panel to manage your ownCloud virtual machine 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 label-phpinfo.

Maximum Upload Size

If you want to increase the maximum upload size, you will also have to modify your php-fpm configuration and increase the upload_max_filesize and post_max_size values. You will need to restart php5-fpm and your HTTP server in order for these changes to be applied.

.htaccess Notes for Apache

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

No basic authentication headers were found

This error is shown in your data/owncloud.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 ownCloud via WebDAV, CalDAV and CardDAV clients is failing. Information on how to correctly configure your environment can be found in the forums but we generally recommend against the use of these modules and recommend mod_php instead.

Other Web Servers

- Other HTTP servers
- Univention Corporate Server installation

Troubleshooting

If your ownCloud installation fails and you see the following error in your ownCloud log please refer to MySQL / MariaDB with Binary Logging Enabled for how to resolve it.

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

Changing Your ownCloud URL

This admin manual assumes that the ownCloud server is already accessible under the route /owncloud (which is the default, e.g. https://example.com/owncloud). If you like, you can change this in your web server configuration, for example by changing it from https://example.com/owncloud/ to https://example.com/.

To do so on Debian/Ubuntu Linux, you need to edit these files:

- /etc/apache2/sites-enabled/owncloud.conf
- /var/www/owncloud/config/config.php

Edit the Alias directive in /etc/apache2/sites-enabled/owncloud.conf to alias your ownCloud directory to the Web server root:

```
Alias / "/var/www/owncloud/"
```

Edit the overwrite.cli.url parameter in /var/www/owncloud/config/config.php:

```
'overwrite.cli.url' => 'http://localhost/',
```

When the changes have been made and the file saved, restart Apache. Now you can access ownCloud from either https://example.com/ or https://localhost/.



You will not be able to run any other virtual hosts, as ownCloud is aliased to your web root. On CentOS/Fedora/Red Hat, edit /etc/httpd/conf.d/owncloud.conf and /var/www/html/owncloud/config/config.php, then restart Apache.

Installing and Managing Apps

Introduction

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

Installing and Managing Apps

Installing Apps Via the ownCloud Marketplace

[ownCloud top navigation menu showing the Market app]

To add an app, use the *Market* app, which is accessible from the top-level navigation bar, on the left-hand side of the page. Once in the Market app, click an app's name to view more details about it. Once you have done this, you can also install it by clicking btn:[Install]. Clicking *Install*, downloads it from the ownCloud Marketplace, installs, and enables it.

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



If you would like to create or add (your own) ownCloud app, please refer to the developer manual.

Installing Apps Manually

To install an app manually, instead of by using the Market app, extract the app tarball into your ownCloud installation's default app folder (</path/to/owncloud>/apps) or, ideally, into a custom app directory.

Once the tarball has been extracted into place:

1. Enable the app via either:
   - Navigating to menu:Settings[Admin > Apps] and clicking btn:[Enable]; or
   - Using the occ app command.

Managing Apps

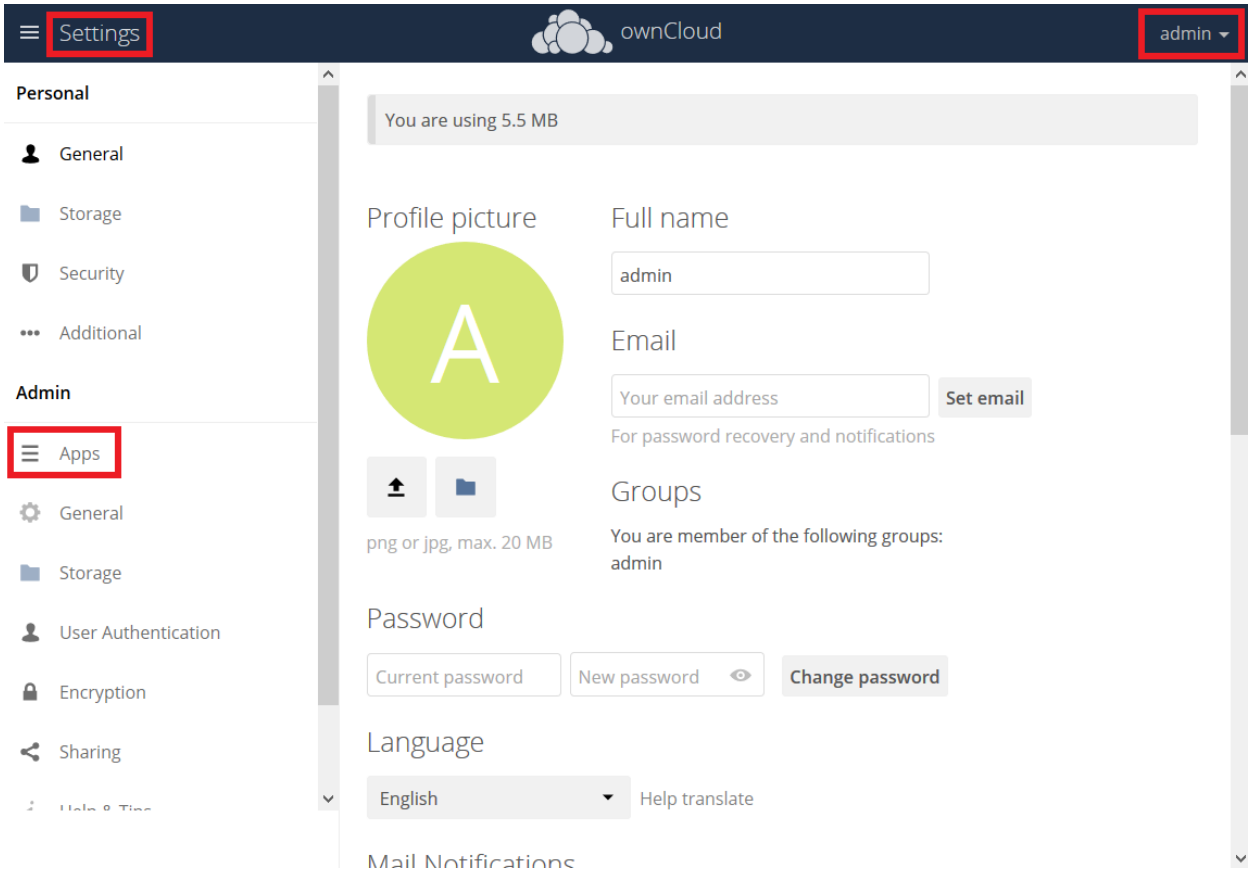
Some apps are installed and enabled during ownCloud installation, while other apps can be installed and enabled post-installation.



Supported Enterprise Apps

See supported apps for a list of supported Enterprise edition apps.

View App Status



To see the status of your installation's applications, go to your Apps page, via menu:Settings[Admin > Apps]. There, you will see which apps are currently: *enabled*, *not enabled*, and *recommended*. You'll also see additional filters, such as *Multimedia*, *Productivity*, and *Tool* for finding more apps quickly.

Enabling and Disabling Apps

On the Apps page (menu:Settings[Admin > Apps]), you can enable or disable applications. Be default, enabled apps are displayed. To disable an app, click btn:[Disable] under its details.

To display disabled apps, click btn:[Show disabled apps]. To enable an app, click btn:[Enable] under its details.

Configuring Apps

Some apps have configurable options on the Apps page, such as **Enable only for specific groups**. However, this is the exception. Apps are mainly configured from your ownCloud Personal or Admin settings page, or in config.php.

Using Custom App Directories

There are several reasons for using custom app directories instead of ownCloud's default. These are:

1. It separates ownCloud's core apps from user or admin downloaded apps. Doing so distinguishes which apps are core and which aren't, simplifying upgrades.
2. It eases manual upgrades. Having non-core apps in a directory separate to the core app directory makes them simpler to manage.
3. ownCloud may gain new core apps in newer versions. Doing so orphans deprecated apps, but doesn't remove them.

If you want to store apps in a custom directory, instead of ownCloud's default (/app),

you need to modify the apps_paths element in config/config.php. There, you need to add a new associative array that contains three elements. These are:

path
: The absolute file system path to the custom app folder.

url
: The request path to that folder relative to the ownCloud webroot, prefixed with /.

writable
: Whether users can install apps in that folder.

After adding the configuration, ownCloud only installs apps in directories where writable is set to true. The configuration example below shows how to add a second directory, called apps-external.

```
<?php
$CONFIG = [
   'apps_paths' => [
      [
         'path' => OC::$SERVERROOT.'/apps',
         'url' => '/apps',
         'writable' => false,
      ],
      [
         'path' => OC::$SERVERROOT.'/apps-external',
         'url' => '/apps-external',
         'writable' => true,
      ],
   ],
   // remainder of the configuration
];
```

After you add a new directory configuration, you can then move apps from the original app directory to the new one. To do so, follow these steps:

1. Enable maintenance mode.
2. Disable the apps that you want to move.
3. Create a new apps directory and assign it the same user and group, and ownership permissions as the core apps directory.
4. Move the apps from the old apps directory to the new apps directory.
5. Add a new app directory in config/config.php.
6. If you're using a cache, such as Redis or Memcached, ensure that you clear the cache.
7. Re-enable the apps.
8. Disable maintenance mode.

Supported Apps in ownCloud

AGPL Apps

- Activity
- Anti-Virus
- Collaborative Tags
- Comments
- Encryption
- External Sites
- External Storage
- ownCloud WebDAV Endpoint (handles old and new webdav endpoints)
- Federated File Sharing (allows file sharing across ownCloud instances)
- Federation (allows usernname auto-complete across ownCloud instances)
- Files (cannot be disabled)
- Files Media Viewer



The *Gallery* and *Files VideoPlayer* apps need to be uninstalled before installing the Media Viewer app.

- Files PDF Viewer
- Files Sharing
- Files TextEditor
- Files Trashbin
- Files Versions
- First Run Wizard
- Notifications
- Provisioning API
- Template Editor (for notification emails)
- Update Notifications
- User External
- User LDAP

Enterprise-Only Apps

- Auditing
- Collaborative Tags Management
- Enterprise License Key
- File Firewall
- LDAP Home Connector
- Object Storage Support
- Password Policy
- External Storage: SharePoint
- SAML/Shibboleth User Backend
- Windows Network Drives (requires External Storage)
- Workflows
- ownCloud X Enterprise Theme

Media Viewer App

The Media Viewer App is a simple viewer for pictures and videos integrated into the files app, which supersedes the former Gallery and Video Player apps.



- Before installing the Media Viewer app, the Gallery and Video Player apps need to be removed, or at the very least disabled. When removing or disabling, ensure that any gallery link shares are redirected.
- Gallery and Video Player are no longer supported and will not receive any further security or bug fixes.
- Users are strongly encouraged to switch to Media Player.

Add Support For More Media Types

To add support for additional media types, in addition to the default set, ensure that ImageMagick and its PECL extension are installed and enabled. Next, add new entries to the enabledPreviewProviders in config/config.php. Below, is an example of how to configure it.

```
'enabledPreviewProviders' => [
  'OC\\Preview\\PNG',
  'OC\\Preview\\JPEG',
  'OC\\Preview\\GIF',
  'OC\\Preview\\Illustrator',
  'OC\\Preview\\Postscript',
  'OC\\Preview\\Photoshop',
  'OC\\Preview\\TIFF'
],
```



Support for playing Apple QuickTime (*.mov) does not work in Chrome - however it is supported in Safari and Mozilla.



Look at the sample configuration (config.sample.php) in your config folder, for more information about this configuration key.

SELinux Configuration

Introduction

Security-Enhanced Linux (SELinux) is a Linux kernel security module that provides a mechanism for supporting access control security policies, including mandatory access controls (MAC).

Preparation

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

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

```
semanage fcontext -a -t httpd_sys_rw_content_t
'/var/www/html/owncloud/data(/.*)?'
semanage fcontext -a -t httpd_sys_rw_content_t
'/var/www/html/owncloud/config(/.*)?'
semanage fcontext -a -t httpd_sys_rw_content_t
'/var/www/html/owncloud/apps(/.*)?'
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/html/owncloud/apps-
external(/.*)?'
semanage fcontext -a -t httpd_sys_rw_content_t
'/var/www/html/owncloud/.htaccess'
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/html/owncloud/.user.ini'

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

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

```
semanage fcontext -d '/var/www/html/owncloud/data(/.*)?'
semanage fcontext -d '/var/www/html/owncloud/config(/.*)?'
semanage fcontext -d '/var/www/html/owncloud/apps(/.*)?'
semanage fcontext -d '/var/www/html/owncloud/apps-external(/.*)?'
semanage fcontext -d '/var/www/html/owncloud/.htaccess'
semanage fcontext -d '/var/www/html/owncloud/.user.ini'

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

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/owncloud/data
/var/www/html/owncloud/config
/var/www/html/owncloud/apps
/var/www/html/owncloud/apps-external
```

Enable updates via the web interface

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

```
setsebool httpd_unified on
```

When the update is completed, disable write access:

```
setsebool -P  httpd_unified  off
```

Disallow write access to the whole web directory

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

```
setsebool -P  httpd_unified  off
```

Allow access to a remote database

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

```
setsebool -P httpd_can_network_connect_db on
```

Allow access to LDAP server

Use this setting to allow LDAP connections:

```
setsebool -P httpd_can_connect_ldap on
```

Allow access to remote network

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

```
setsebool -P httpd_can_network_connect on
```

Allow access to network memcache

This setting is not required if httpd_can_network_connect is already on:

```
setsebool -P httpd_can_network_memcache on
```

Allow access to SMTP/sendmail

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

```
setsebool -P httpd_can_sendmail on
```

Allow access to CIFS/SMB

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

```
setsebool -P httpd_use_cifs on
```

Allow access to FuseFS

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

```
setsebool -P httpd_use_fusefs on
```

Allow access to GPG for Rainloop

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

```
setsebool -P httpd_use_gpg on
```

Troubleshooting

General 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 ownCloud directory:

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

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.

See this discussion on GitHub to learn more about configuring SELinux correctly for ownCloud.

Redis on RHEL 7 & Derivatives

On RHEL 7 and its derivatives, if you are using Redis for both local server cache and file locking and Redis is configured to listen on a Unix socket instead of a TCP/IP port (*which is recommended if Redis is running on the same system as ownCloud*) you must instruct SELinux to allow daemons to enable cluster mode. You can do this using the following command:

```
setsebool -P daemons_enable_cluster_mode 1
```

Let's Encrypt SSL Certificates

In this section you will find all the details you need to configure ownCloud with Let's Encrypt.

- Using Let's Encrypt SSL Certificates
- Configure Apache with Let's Encrypt

Using Let's Encrypt SSL Certificates

Introduction

This page covers how to configure your web server to use Let's Encrypt as the certificate authority for your ownCloud server. Note that Let's Encrypt is *not officially supported*, and this page is *community-maintained*. Thank you, contributors!

- For ease of handling, SSL-specific directives have been moved into a separately included file. This can help for first-time certificate issuance as well as for reusing configurations.
- The examples shown are based on Ubuntu 18.04.
- Read the Certbot user guide for details of the commands.
- Let's Encrypt CA issues short-lived certificates valid for 90 days. Make sure you renew the certificates at least once in this period, because expired certificates need reissuing. A certificate is due for renewal earliest 30 days before expiring. Certbot can be forced to renew via options at any time as long the certificate is valid.



Raymii.org provides an excellent introduction to strong SSL security measures with Apache, if you would like to know more.

Requirements & Dependencies

You require a domain name with a valid A-Record pointing back to your servers IP address. In case your server is behind a firewall, take the necessary measures to ensure that your server is accessible, worldwide, from the internet, by adding the required firewall and port forward rules.

Install Let's Encrypt's Certbot Client

The Certbot client can be installed in following ways:

1. Installation via APT install
2. With the Ubuntu PPA repository.

Via APT

To install Certbot via APT, run the following commands.

```
sudo apt-get update
sudo apt-get install certbot
```

Via PPA

To install Certbot via the PPA repository, run the following commands.
These will add the repository, update APT's cache, and install Certbot.



This method is preferred when you're using a version of Ubuntu prior to 18.04.

```
sudo apt-get update
sudo apt-get install software-properties-common
sudo add-apt-repository ppa:certbot/certbot
sudo apt-get install certbot
```

Run Certbot

To run Certbot use the following command:

```
sudo certbot
```



Depending on how you installed Let's Encrypt, certbot may also be named letsencrypt or certbot-auto. However, this guide will refer to it as certbot. Please bear that in mind, and update the examples and scripts used in this guide to reflect your Certbot installation.

Updating Certbot

If you need to update Certbot at a later date, run:

```
sudo apt-get install --only-upgrade certbot
```

Register Your Email Address

First Time Registration

Now that Certbot is installed, register your email address for urgent renewal and security notifications. This command also prepares Certbot's environment if it's not already installed. To do this, run the following command:

```
sudo certbot register --agree-tos --email <your-email-address>
```

When it executes, you'll see a question similar to the following, which you can answer "Yes" or "No":

```
Saving debug log to /var/log/letsencrypt/letsencrypt.log

-------------------------------------------------------------------------------
Would you be willing to share your email address with the Electronic Frontier
Foundation, a founding partner of the Let's Encrypt project and the non-profit
organization that develops Certbot? We'd like to send you email about EFF and
our work to encrypt the web, protect its users and defend digital rights.
-------------------------------------------------------------------------------
(Y)es/(N)o:
```

When that completes, you'll see a message similar to the following:

```
IMPORTANT NOTES:
 1. Your account credentials have been saved in your Certbot
    configuration directory at /etc/letsencrypt. You should make a
    secure backup of this folder now. This configuration directory will
    also contain certificates and private keys obtained by Certbot so
    making regular backups of this folder is ideal.
```

Please, **strongly**, consider following its recommendation.

Update Your Registration

In case you want to update your registered email address use following command:



This will affect all the certificates issued using this account.

```
sudo certbot register --update-registration --email <your-email-address>
```

When that completes, you'll see a message similar to the following:

```
Saving debug log to /var/log/letsencrypt/letsencrypt.log

-------------------------------------------------------------------------------
Would you be willing to share your email address with the Electronic Frontier
Foundation, a founding partner of the Let's Encrypt project and the non-profit
organization that develops Certbot? We'd like to send you email about EFF and
our work to encrypt the web, protect its users and defend digital rights.
-------------------------------------------------------------------------------
(Y)es/(N)o: y

IMPORTANT NOTES:
 - Your e-mail address was updated to <your-email-address>
```

Create Let's Encrypt's Config Files

- Create following files in the Let's Encrypt directory. They will help to maintain your certificates.
- Replace the path to Certbot and the Certbot script name based on your installation. You can find it by running which certbot.
- Rename <your-domain-name>.sh with the name of the domain(s) you want to issue a certificate for.
  As an example, the script could be renamed to your-domain-name.com.sh.
- Make all files executable except cli.ini by running sudo chmod +x <script-name>.



All scripts have to be executed with sudo.

```
cd /etc/letsencrypt
touch cli.ini list.sh renew.sh renew-cron.sh delete.sh <your-domain-name>.sh
```

cli.ini

This file defines some default settings used by Certbot. Use the email address you registered with. Comment / un-comment the post-hook parameter according which web server you use.

```
rsa-key-size = 4096
email = <your-email-address>
agree-tos = True
authenticator = webroot
# post-hook = service nginx reload
# post-hook = service apache2 reload
```

list.sh

This script lists all your issued certificates.

```
#!/bin/bash

LE_PATH="/usr/bin"
LE_CB="certbot"

"$LE_PATH/$LE_CB" certificates
```

renew.sh

This script:

- Renews all your issued certificates.
- In case you have enabled the post hook for your webserver in cli.ini, it will reload the web server configuration automatically if a certificate has been renewed.

```
#!/bin/bash

LE_PATH="/usr/bin"
LE_CB="certbot"

"$LE_PATH/$LE_CB" renew
```

renew-cron.sh

This script:

- Renews all your issued certificates but does not upgrade Certbot.
- In case you have enabled the post hook for your webserver in cli.ini, it will reload the web server configuration automatically if a certificate has been renewed.



This script is intended for use via Cron.

```
#!/bin/bash

LE_PATH="/usr/bin"
LE_CB="certbot"

"$LE_PATH/$LE_CB" renew --no-self-upgrade --noninteractive
```

delete.sh

This script deletes an issued certificate.
Use the list.sh script to list issued certificates.

```
#!/bin/bash

LE_PATH="/usr/bin"
LE_CB="certbot"

##
## Retrieve and print a list of the installed Let's Encrypt SSL certificates.
##
function get_certificate_names()
{
  "$LE_PATH/$LE_CB" certificates | grep -iE "certificate name" | awk -F: '{gsub(/\s+/,
"", $2); printf("- %s\n", $2)}'
}

echo "Available Certificates:"

get_certificate_names
echo

read -p "Which certificate do you want to delete: " -r -e answer
if [ -n "$answer" ]; then
  "$LE_PATH/$LE_CB" delete --cert-name "$answer"
fi
```

<your-domain-name>.sh

As an example, this script creates a certificate for following domain / sub-domains. You can add or remove sub-domains as necessary. Use your domain / sub-domain names. The first (sub)domain name used in the script is taken for naming the directories created by Certbot.



You can create different certificates for different sub-domains, such as example.com, www.example.com, and subdomain.example.com, by creating different scripts.

```
#!/bin/bash
# export makes the variable available for all subprocesses

LE_PATH="/usr/bin"
LE_CB="certbot"

# Assumes that example.com www.example.com and subomain.example.com are
the domains
# that you want a certificate for
export DOMAINS="-d example.com -d www.example.com -d
subdomain.example.com"

"$LE_PATH/$LE_CB" certonly --config /etc/letsencrypt/cli.ini "$DOMAINS" # --dry-run
```



You can enable the --dry-run option which does a test run of the client only.

Create an SSL Certificate

With all the scripts created, to create an SSL certificate, run the following command:

```
sudo /etc/letsencrypt/<your-domain-name>.sh
```

After you run the script, you will see output similar to the following:

```
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Obtaining a new certificate
Performing the following challenges:
http-01 challenge for your-domain-name.com
Using the webroot path /var/www/html for all unmatched domains.
Waiting for verification...
Cleaning up challenges
Running post-hook command: service apache2 reload

IMPORTANT NOTES:
 1. Congratulations! Your certificate and chain have been saved at:
    /etc/letsencrypt/live/your-domain-name.com/fullchain.pem
    Your key file has been saved at:
    /etc/letsencrypt/live/your-domain-name.com/privkey.pem
    Your cert will expire on 2018-06-18. To obtain a new or tweaked
    version of this certificate in the future, simply run certbot
    again. To non-interactively renew *all* of your certificates, run
    "certbot renew"
 2. If you like Certbot, please consider supporting our work by:

    Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
    Donating to EFF:                    https://eff.org/donate-le
```

You can see that the SSL certificate's been successfully created, and that it will expire on 2018-06-18.

Listing Existing Certificates

If you want to list (view) existing SSL certificates, use list.sh, which can be run as follows:

```
sudo /etc/letsencrypt/list.sh
```

Depending on the number of certificates, you can expect to see output similar to the following:

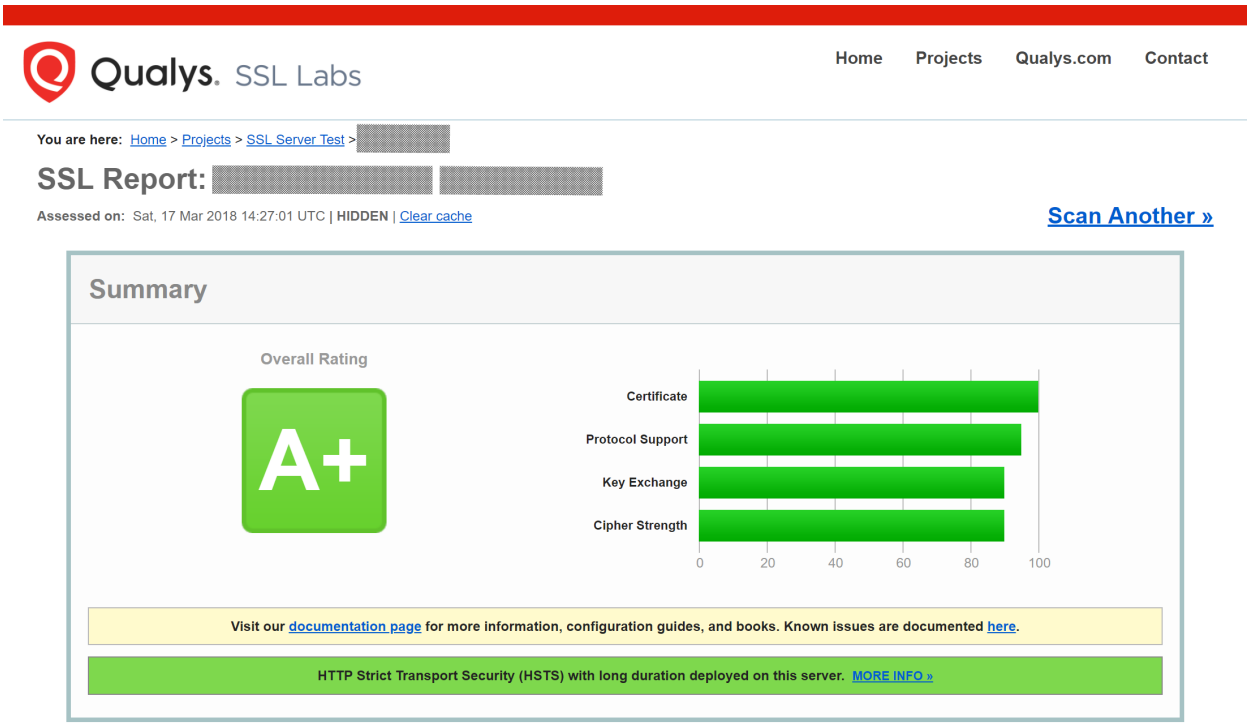
```
-------------------------------------------------------------------------------
Found the following certs:
  Certificate Name: your-domain-name.com
    Domains: your-domain-name.com
    Expiry Date: 2018-06-18 10:57:18+00:00 (VALID: 82 days)
    Certificate Path: /etc/letsencrypt/live/your-domain-name.com/fullchain.pem
    Private Key Path: /etc/letsencrypt/live/your-domain-name.com/privkey.pem
-------------------------------------------------------------------------------
```

Web Server Setup

Refer to the Apache setup guide, to set up your web server and issue a certificate.

Test the Setup

After you have setup and configured the web server and installed the SSL certificate using Certbot, you should now test the security of your new configuration. To do so, you can use the free service of SSL Labs. See an example screenshot of a test run below.



Renewing SSL Certificates

As Let's Encrypts certificates expire every 90 days, you should ensure you renew them before that time.
There are two ways to do so: manually and automatically.

Manual Renewal

If you have provided your email address, you will receive reminder notifications.

```
sudo /etc/letsencrypt/renew.sh
```

If the certificate is not yet due for renewal, you can expect to see output similar to that below:

```
-------------------------------------------------------------------------------
Processing /etc/letsencrypt/renewal/your-domain-name.com.conf
-------------------------------------------------------------------------------
Cert not yet due for renewal

The following certs are not due for renewal yet:
  /etc/letsencrypt/live/your-domain-name.com/fullchain.pem (skipped)
No renewals were attempted.
No hooks were run.
```

Automatic Renewal via Crontab

Certificates are only renewed if they are due, so you can schedule Cron jobs to renew your SSL certificates on a more frequent basis. However, a weekly check is sufficient.

To add a new Cron job to auto-renew your certificates, firstly run the following command to edit the job list.

```
sudo crontab -e
```



It is essential to use sudo to derive proper permissions.

Then, add the following at the end of the existing configuration:

```
30 03 * * 6 /etc/letsencrypt/renew-cron.sh
```

After you save and exit the file, the new job will have been added to the Cron job scheduler.



If you want to use own values, you can check them eg. at crontab.guru or modify the script for other options.

Add Extra Domains to the Certificate

If you want to add an extra domain, like subdomain.example.com, to your certificate, add the domain in the domain shell script above, re-run it and reload the web server config. This can be useful when migrating from a sub-directory to sub-domain access.



This also implies that you need to comment the include directive (please refer to the relevant web server setup) and follow the steps afterwards.

Deleting SSL Certificates

If you want to delete an SSL certificate, use the delete.sh script, running it as follows:

```
sudo /etc/letsencrypt/delete.sh
```

It will start off, as below, by displaying a list of the currently available SSL certificate domain names, and then prompt you to supply the certificate that you want to delete.

```
Available Certificates:

1. your-domain-name.com

Which certificate do you want to delete:
```

Provide the SSL certificate name that you want to delete and click btn:[enter], and the certificate and all of its related files will be deleted. After that you should expect to see a confirmation, as in the example output below.

```
--------------------------------------------------------------------------------
Deleted all files relating to certificate your-domain-name.com.
--------------------------------------------------------------------------------
```

Configure Apache with Let's Encrypt

Introduction

This guide shows how to configure Apache with Let's Encrypt.

Dependencies

To follow this guide, your server needs to have the following dependencies installed:

- Apache 2.4.8 or later
- OpenSSL 1.0.2 or later
- Let's Encrypt

Assumptions

This guide assumes two things:

1. That you are using Ubuntu Linux 18.04. If you are not using Ubuntu 18.04, please adjust the instructions to suit your distribution or operating system.
2. That your ownCloud installation is configured using a VirtualHost (vhost) configuration instead of being configured in the main Apache configuration, and
3. That the vhost configuration file is stored under /etc/apache2/sites-available/. Not all distributions use this location, however. Please refer to your distribution's Apache documentation, to know where to store yours.

Create and Configure a Diffie-Hellman Params File

When using Apache 2.4.8 or later and OpenSSL 1.0.2 or later you can generate and specify a Diffie-Hellman (DH) params file. If not already present in your VirtualHost (vhost) file, add an SSLOpenSSLConfCmd directive and a new certificate with stronger keys, which improves forward secrecy.



The OpenSSL command may take a quite a while to complete, so please be patient.

You can place the generated SSL certificate into any directory of your choice, by running the following command, and changing the value supplied to the -out option. We recommend storing it in /etc/apache2/ in this guide, solely for sakes of simplicity.

```
sudo openssl dhparam -out /etc/apache2/dh4096.pem 4096
```

Once the command completes, add the following directive to your common SSL configuration:

```
SSLOpenSSLConfCmd DHParameters /etc/apache2/dh4096.pem
```

Let's Encrypt ACME-Challenge

After that, add an Alias directive for the /.well-known/acme-challenge location in your HTTP VirtualHost configuration, as in line four in the following example.

```
<virtualHost *.80>
  ServerName mydom.tld

  Alias /.well-known/acme-challenge/ /var/www/letsencrypt/.well-known/acme-
challenge/
  <Directory "/var/www/letsencrypt/.well-known/acme-challenge/">
     Options None
     AllowOverride None
     ForceType text/plain
     RedirectMatch 404 "^(?!/\.well-known/acme-challenge/[\w-]{43}$)"
  </Directory>

  # ... remaining configuration
</virtualHost>
```

Create an SSL VirtualHost Configuration

We recommend creating a separate file for storing the SSL directives. If these directives already exist in this Virtual Host, delete them and include the file instead. This is because, when the certificate has been created, you can use this file in any SSL-enabled VirtualHost configuration for which the certificate is valid, without reissuing the SSL certificate.

```
cd /etc/apache2/
sudo mkdir ssl_rules
touch ssl_rules/ssl_mydom.tld
```

Listing 7. /etc/apache2/ssl_rules/ssl_mydom.tld

```
# Eases letsencrypt initial cert issuing

SSLEngine on
SSLCertificateChainFile  /etc/letsencrypt/live/mydom.tld/fullchain.pem
SSLCertificateKeyFile    /etc/letsencrypt/live/mydom.tld/privkey.pem
SSLCertificateFile       /etc/letsencrypt/live/mydom.tld/cert.pem
```



To improve SSL performance, we recommend that you use the SSLUseStapling and SSLStaplingCache directives. Here's an example configuration:

```
SSLUseStapling on
SSLStaplingCache        shmcb:/tmp/stapling_cache(2097152)
```

With the files created and filled-out, update your HTTPS VirtualHost configuration:

```
<virtualHost *:443>
 ServerName mydom.tld

 # ssl letsencrypt
 # Include /etc/apache2/ssl_rules/ssl_mydom.tld

 #...
</virtualHost>
```



For the moment, comment out the Include directive, as the certificate files do not, currently, exist.

Test and Enable the Apache Configuration

With the configuration created, test it by running one of the following two commands:

```
sudo apache2ctl configtest
sudo apache2ctl -t
```

It should not display any errors. If it doesn't, load your new Apache configuration by running the following command:

```
sudo apache2ctl graceful
```

Create the SSL Certificates

To create the SSL certificates, run the following command:

```
sudo /etc/letsencrypt/<your-domain-name>.sh
```

Next, double check that the certificates have been issued by running the list.sh script.

```
sudo /etc/letsencrypt/list.sh
```

If successful, you will see output similar to that below when the command completes:

```
Saving debug log to /var/log/letsencrypt/letsencrypt.log

-------------------------------------------------------------------------------
Found the following certs:
  Certificate Name: mydom.tld
    Domains: mydom.tld
    Expiry Date: 2018-06-18 10:57:18+00:00 (VALID: 89 days)
    Certificate Path: /etc/letsencrypt/live/mydom.tld/fullchain.pem
    Private Key Path: /etc/letsencrypt/live/mydom.tld/privkey.pem
-------------------------------------------------------------------------------
```

As the certificate files exist, you can uncomment the Include directive in your HTTPS VirtualHost configuration to use them.

```
<virtualHost *:443>
  ServerName mydom.tld

  # ssl letsencrypt
  Include /etc/apache2/ssl_rules/ssl_mydom.tld

  #...
</virtualHost>
```

Reload the Apache Configuration

Finally, reload (or restart) Apache.

It is now ready to serve HTTPS request for the given domain using the issued certificates.

```
sudo service apache2 reload
```

Configuration

In this section, you will find all the information you need for configuring ownCloud.

Database

In this section, you can find out about

- Converting your Database Type
- Database Configuration on Linux

Converting Database Type

Introduction

SQLite is good for testing ownCloud, as well as small, single-user, ownCloud servers. But, **it does not scale** for large, multi-user sites. If you have an existing ownCloud installation which uses SQLite, and you want to convert to a better performing database, such as *MySQL*, *MariaDB* or *PostgreSQL*, you can use the ownCloud command line tool: occ.



ownCloud Enterprise edition does not support SQLite.

Preparation

Add the following to your ownCloud config/config.php:

```
'mysql.utf8mb4' => true,
```

Add, or adjust, the following in /etc/mysql/mariadb.conf.d/50-server.cnf:



You can do the same for MySQL by replacing mariadb.conf.d/50-server.cnf with mysql.conf.d/mysqld.cnf.

```
key_buffer_size        = 32M
table_cache            = 400
query_cache_size       = 128M
#in InnoDB:
innodb_flush_method=O_DIRECT
innodb_flush_log_at_trx_commit=1
innodb_log_file_size=256M
innodb_log_buffer_size = 128M
innodb_buffer_pool_size=2048M
innodb_buffer_pool_instances=3
innodb_read_io_threads=4
innodb_write_io_threads=4
innodb_io_capacity = 500
innodb_thread_concurrency=2
innodb_file_format=Barracuda
innodb_file_per_table=ON
innodb_large_prefix = 1
character-set-server  = utf8mb4
collation-server      = utf8mb4_general_ci
```

Restart the Database Server

When you have changed the database parameters, restart your database by running following command:

```
sudo service mysql restart
```

Run the conversion

After you have restarted the database, run the following occ command in your ownCloud root folder, to convert the database to the new format:

```
{occ-command-example-prefix} db:convert-type [options] type username
hostname database
```



The converter searches for apps in your configured app folders and uses the schema definitions in the apps to create the new table. As a result, tables of removed apps will not be converted — even with option --all-apps For example:

```
{occ-command-example-prefix} db:convert-type --all-apps mysql
oc_mysql_user 127.0.0.1 new_db_name
```

To successfully proceed with the conversion, you must type yes when prompted with the question Continue with the conversion? On success the converter will automatically configure the new database in your ownCloud config config.php.

Unconvertible Tables

If you updated your ownCloud installation then the old tables, which are not used anymore, might still exist. The converter will tell you which ones.

```
The following tables will not be converted:
 oc_permissions
```

You can ignore these tables. Here is a list of known old tables:

- oc_calendar_calendars
- oc_calendar_objects
- oc_calendar_share_calendar
- oc_calendar_share_event
- oc_fscache
- oc_log
- oc_media_albums
- oc_media_artists
- oc_media_sessions
- oc_media_songs
- oc_media_users

- oc_permissions
- oc_queuedtasks
- oc_sharing

Database Configuration on Linux

Introduction

ownCloud requires a database in which administrative data is stored. The following databases are currently supported:

- MySQL / MariaDB
- PostgreSQL
- Oracle (*ownCloud Enterprise edition only*)



The MySQL or MariaDB databases are the recommended database engines.

Requirements

Choosing to use MySQL / MariaDB, PostgreSQL, or Oracle as your database requires that you install and set up the server software first.



Oracle users, see the Oracle Database Configuration guide.

The steps for configuring a third party database are beyond the scope of this document. Please refer to the documentation below, for your database vendor.

- The MariaDB Knowledge Base
- The MySQL documentation
- The Oracle Database documentation
- The PostgreSQL documentation

MySQL / MariaDB

Enabling Binary Logging

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

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

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

The other is to change the BINLOG_FORMAT = STATEMENT in your database

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

Set READ COMMITED as the Transaction Isolation Level

As discussed above, ownCloud 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.

Configuring the Storage Engine

Since ownCloud 7, only InnoDB is supported as a storage engine. Some shared hosts do not support InnoDB and only MyISAM. Running ownCloud in such an environment is not supported.

Parameters

For setting up ownCloud to use any database, use the instructions in the Installation Wizard. You should not have to edit the respective values in the config/config.php. However, in exceptional cases (for example, if you want to connect your ownCloud instance to a database created by a previous installation of ownCloud), some modification might be required.

MySQL

If you decide to use a MySQL or MariaDB database, ensure the following:

- That you have installed and enabled the pdo_mysql extension in PHP.
- That the mysql.default_socket points to the correct socket (if the database runs on the same server as ownCloud).

MariaDB is backward compatible with MySQL. All instructions work for both, so you will not need to replace or revise any existing MySQL client commands. The PHP configuration in /etc/php/{recommended-php-version}/apache2/conf.d/20-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 ownCloud when you log in for the first time. To start the MySQL command-line mode use:

```
mysql -uroot -p;
```

Then a mysql> or MariaDB [root]> prompt will appear. Now enter the following lines and confirm them with the enter key:

```
CREATE DATABASE IF NOT EXISTS owncloud;
GRANT ALL PRIVILEGES ON owncloud.*
  TO 'username'@'localhost'
  IDENTIFIED BY 'password';
```

You can quit the prompt by entering:

```
quit;
```

An ownCloud 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"        => "owncloud",
"dbuser"        => "username",
"dbpassword"    => "password",
"dbhost"        => "localhost",
"dbtableprefix" => "oc_",
```

Configure MySQL for 4-byte Unicode Support

For supporting such features as emoji, both MySQL (or MariaDB) **and** ownCloud need to be configured to use 4-byte Unicode support instead of the default 3-byte. If you are setting up a new ownCloud installation, using version 10.0 or above, **and** you're using a minimum MySQL version of 5.7, then you don't need to do anything, as support is checked during setup and used if available.

However, if you have an existing ownCloud installation that you need to convert to use 4-byte Unicode support or you are working with MySQL earlier than version 5.7, then you need to do two things:

1. In your MySQL configuration, add the configuration settings below. If you already have them configured, update them to reflect the values specified:

   ```
   [mysqld]
   innodb_large_prefix=ON
   innodb_file_format=Barracuda
   innodb_file_per_table=ON
   ```

2. Run the following occ command:

   ```
   ./occ db:convert-mysql-charset
   ```

   When this is done, tables will be created with:

   - A utf8mb4 character set.
   - A utf8mb4_bin collation.
   - row_format set to compressed.



For more information, please either refer to config.sample.php, or have a read through the following links:

- https://dev.mysql.com/doc/refman/5.7/en/innodb-parameters.html#sysvar_innodb_large_prefix
- https://mariadb.com/kb/en/library/innodb-system-variables/#innodb_large_prefix
- http://www.tocker.ca/benchmarking-innodb-page-compression-performance.html
- http://dev.mysql.com/doc/refman/5.7/en/charset-unicode-utf8mb4.html
- https://dev.mysql.com/doc/refman/5.7/en/innodb-file-format.html
- https://dev.mysql.com/doc/refman/5.7/en/innodb-multiple-tablespaces.html
- https://dev.mysql.com/doc/refman/5.7/en/innodb-parameters.html#sysvar_innodb_large_prefix

PostgreSQL

If you decide to use a PostgreSQL database, make sure that you have installed and enabled the PostgreSQL extension and the PostgreSQL PDO extension in PHP. The PHP configuration in /etc/php/{recommended-php-version}/apache2/conf.d/20-pgsql.ini could look like this:

```
# configuration for PHP PostgreSQL module
extension=pdo_pgsql.so
extension=pgsql.so

[PostgresSQL]
pgsql.allow_persistent = On
pgsql.auto_reset_persistent = Off
pgsql.max_persistent = -1
pgsql.max_links = -1
pgsql.ignore_notice = 0
pgsql.log_notice = 0
```



The default configuration for PostgreSQL (at least in Ubuntu 14.04) is to use the peer authentication method. Check /etc/postgresql/9.3/main/pg_hba.conf to find out which authentication method is used in your setup.

To start the PostgreSQL command-line mode use:

```
sudo -u postgres psql -d template1
```

Then a template1=\# prompt will appear. Now enter the following lines - *changing <username> to something appropriate* - and confirm them with the enter key:

```
CREATE USER <username> CREATEDB;
CREATE DATABASE owncloud OWNER <username>;
```

You can quit the prompt by entering:

```
\q
```

An ownCloud instance configured with PostgreSQL contains:

- The path to the socket on which the database is running as the hostname
- The system username the PHP process is using
- An empty password to access it
- 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"        => "owncloud",
"dbuser"        => "username",
"dbpassword"    => "",
"dbhost"        => "/var/run/postgresql",
"dbtableprefix" => "oc_",
```

The host 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. First, 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 ownCloud when you log in for the first time. To start the PostgreSQL command-line mode use:

```
psql -hlocalhost -Upostgres
```

Then a **postgres=\#** prompt will appear. Now enter the following lines and confirm them with the enter key:

```
CREATE USER <username> WITH PASSWORD 'password';
CREATE DATABASE owncloud TEMPLATE template0 ENCODING 'UNICODE';
ALTER DATABASE owncloud OWNER TO <username>;
GRANT ALL PRIVILEGES ON DATABASE owncloud TO <username>;
```

You can quit the prompt by entering:

```
\q
```

An ownCloud 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 contain entries like this:

```
<?php

"dbtype"        => "pgsql",
"dbname"        => "owncloud",
"dbuser"        => "username",
"dbpassword"    => "password",
"dbhost"        => "localhost",
"dbtableprefix" => "oc_",
```

Troubleshooting

How to Workaround General Error: 2006 MySQL Server Has Gone Away

The database request takes too long, and therefore the MySQL server times out. 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 hosts are not allowing access to these config options. For such systems, ownCloud is providing a dbdriveroptions configuration option within your config/config.php where you can pass such options to the database driver. Please refer to the sample PHP 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 hostname (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 can be accessed is by starting the command-line interface:

MySQL

Assuming the database server is installed on the same system you're running the command from, use:

```
mysql -uUSERNAME -p
```

To access a MySQL installation on a different machine, add the -h option with the respective hostname:

```
mysql -uUSERNAME -p -h HOSTNAME
```

```
mysql> SHOW VARIABLES LIKE "version";
+---------------+--------+
| Variable_name | Value  |
+---------------+--------+
| version       | 5.1.67 |
+---------------+--------+
1 row in set (0.00 sec)
mysql> quit
```

PostgreSQL

Assuming the database server is installed on the same system you're running the command from, use:

```
psql -Uusername -downcloud
```

To access a PostgreSQL installation on a different machine, add the -h option with the applicable hostname:

```
psql -Uusername -downcloud -h HOSTNAME
```

```
postgres=# SELECT version();
PostgreSQL 8.4.12 on i686-pc-linux-gnu, compiled by GCC gcc (GCC) 4.1.3
20080704 (prerelease), 32-bit
(1 row)
postgres=# \q
```

Useful SQL Commands

Show Database Users

MySQL	PostgreSQL
SELECT User,Host FROM mysql.user;	SELECT * FROM pg_user;

Show Available Databases

MySQL	PostgreSQL
SHOW DATABASES;	\l

Show ownCloud Tables in Database

MySQL	PostgreSQL
USE owncloud; SHOW TABLES;	\c owncloud; \d

Quit Database

MySQL	PostgreSQL
quit;	\q

How to Solve Deadlock Errors

SQLSTATE[40001]: Serialization failure: 1213 Deadlock found when trying to get lock; try restarting transaction

Explanation

This error occurs when two transactions write and commit to the same rows in separate cluster nodes. Only one of them can successfully commit. The failing one will be aborted. For cluster level aborts, Galera Cluster returns a deadlock error.

Solution

The solution, for Galera Cluster, would be to send all write requests to a single DB node, instead of all of them. Here is a useful guide, when using HAProxy.

The same concept applies when MaxScale is used as a DB proxy. It needs to be configured to send all write requests to a single DB node instead all of them and balance read statements across the rest of the nodes. Here is a useful guide on how to configure MaxScale with Read/Write splitting.

Enabling Causality Checks

Additionally, to solve this issue, when using Galera Cluster, customers should try to set wsrep_sync_wait=1. When enabled, the node triggers causality checks in response to certain types of queries. This is disabled by default.

Encryption

In this section you will find all the details you need to configure encryption in ownCloud.

Enable the Encryption App

Before you can use encryption you must enable the encryption app. You can do this either from the command-line or from the Web-UI.

Enable Encryption From the Command-Line

To enable the encryption app, run the following command:

```
{occ-command-example-prefix} app:enable encryption
```

If the encryption app successfully enables, then you should see the following confirmation:

```
encryption enabled
```



This should never happen, but the encryption app may not be packaged with your ownCloud installation. If so, you will see the following output when you attempt to enable it:

```
encryption not found
```

If that happens, then you need to install it manually. You can do this by cloning the encryption app, using the following command:

```
git clone https://github.com/owncloud/encryption.git apps/encryption
```

Enable Encryption From the Web-UI

To enable encryption from the Web-UI:

1. Go to menu:Settings[Admin > Apps] and click on kbd:[Show disabled apps]
2. When the disabled apps are rendered click btn:[Enable] under "*Default encryption module*".
3. After that go to menu:Settings[Admin > Encryption], and enable btn:[Enable server-side encryption].
4. Then, under "*Default encryption module*", select the desired encryption type, whether "*Master Key*" (recommended) or "*User-key*".
5. Now you must log out and then log back in to initialize your encryption keys.

Disable Encryption

Matthew Setter <matthew@matthewsetter.com> :keywords: encryption, occ :description: This guide will show you how to disable encryption in ownCloud.

You may disable encryption only with occ. Make sure you have backups of all the encryption keys, including those for all users. When you do, put your ownCloud server into single-user mode, and then disable your encryption module with this command:

```
{occ-command-example-prefix} maintenance:singleuser --on
{occ-command-example-prefix} encryption:disable
```



Encryption cannot be disabled without the user's password or file recovery key. If you don't have access to at least one of these then there is no way to decrypt all files.

When decryption has finished, disable single-user mode, using the following command.

```
{occ-command-example-prefix} maintenance:singleuser --off
```

It is possible to disable encryption with the file recovery key, *if* every user uses them. If so, decrypt-all will use it to decrypt all files.

LDAP and Other External User Backends

If you use an external user back-end, such as an LDAP or Samba server, and you change a user's password on that back-end, the user will be prompted to change their ownCloud login to match on their next ownCloud login. The user will need both their old and new passwords to do this.

If you have enabled the recovery key, then — *as an ownCloud admin* — you can change a user's password in the ownCloud Users panel to match their back-end password, and then — of course — notify the user and give them their new password.

Encryption Configuration

Background Information

The primary purpose of the ownCloud server-side encryption is to protect users' files when they're located on remote storages, such as Dropbox and Google Drive, and to do it smoothly and seamlessly from within ownCloud.

From ownCloud 9.0, server-side encryption for local and remote storages can operate independently of each other. By doing so, you can encrypt a remote storage *without* also having to encrypt your home storage on your ownCloud server.



Starting with ownCloud 9.0 we support Authenticated Encryption for all newly encrypted files. See https://hackerone.com/reports/108082 for more technical information about the impact.

For maximum security make sure to configure external storage with "*Check for changes: Never*". This will let ownCloud ignore new files not added via ownCloud. By doing so, a malicious external storage administrator cannot add new files to the storage without your knowledge. However, this is not wise *if* your external storage is subject to legitimate external changes.

ownCloud's server-side encryption encrypts files stored on the ownCloud server and files on remote storages that are connected to your ownCloud server. Encryption and decryption are performed on the ownCloud server. All files sent to remote storage will be encrypted by the ownCloud server and decrypted before serving them to you or anyone whom you have shared them with.



Encrypting files increases their size by roughly 35%. Remember to take this into account when you are both provisioning storage and setting storage quotas. Secondly, user quotas are based on the *unencrypted* file size — **not** the encrypted size.

When files on an external storage are encrypted in ownCloud, you cannot share them directly from the external storage services, only through ownCloud sharing. This is

because the key to decrypt the data **never** leaves the ownCloud server.

ownCloud's server-side encryption generates a strong encryption key, which is unlocked by users' passwords. As a result, your users don't need to track an extra password. All they need to do is log in as they normally would. ownCloud, transparently, encrypts only the contents of files, and not filenames and directory structures.



You should regularly backup all encryption keys to prevent permanent data loss.

The encryption keys are stored in the following directories:

Directory	Description
data/<user>/files_encryption	Users' private keys and all other keys necessary to decrypt the users' files.
data/files_encryption	Private keys and all other keys necessary to decrypt the files stored on a system wide external storage.



You can move the keys to a different location. To do so, refer to the Move Key Location section of the documentation.

When encryption is enabled, all files are encrypted and decrypted by the ownCloud application, and stored encrypted on your remote storage. This protects your data on externally hosted storage. The ownCloud admin and the storage admin will see only encrypted files when browsing backend storage.

Encryption keys are stored only on the ownCloud server, eliminating exposure of your data to third-party storage providers. The encryption application does **not** protect your data if your ownCloud server is compromised, and it does not prevent ownCloud administrators from reading users' files.

This would require client-side encryption, which this application does not provide. If your ownCloud server is not connected to any external storage services, it is better to use other encryption tools, such as file-level or whole-disk encryption.



SSL terminates at or before the webserver on the ownCloud server. Consequently, all files are in an unencrypted state between the SSL connection termination and the ownCloud code that encrypts and decrypts them. This is, potentially, exploitable by anyone with administrator access to your server. For more information, read: How ownCloud uses encryption to protect your data.

Encryption Types

ownCloud provides two encryption types:

User-Key:	Every user has their own private/public key pairs, and the private key is protected by the user's password.
Master Key:	There is only one key (or key pair) and all files are encrypted using that key pair.



These encryption types are **not compatible**.

Before Enabling Encryption

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

You have more options via the occ command's encryption options.



You can't manage encryption without access to the command line. If your ownCloud installation is on a hosted environment and you don't have access to the command line, you won't be able to run occ commands. In this case, **don't enable encryption**!

Enabling Master Key Based Encryption from the Command-Line

To enable encryption via the command-line, involves several commands. Firstly, enable the default encryption module app, using the following command:

```
{occ-command-example-prefix} app:enable encryption
```

Then enable encryption, using the following command:

```
{occ-command-example-prefix} encryption:enable
```

After that, enable the master key, using the following command:

```
{occ-command-example-prefix} encryption:select-encryption-type masterkey
```



The master key mode has to be set up in a newly created instance.

Finally, encrypt all data, using the following command:

```
{occ-command-example-prefix} encryption:encrypt-all --yes
```



This command is not typically required, as the master key is often enabled at install time. As a result, when enabling it, there should be no data to encrypt. But, in case it's being enabled after install, and the installation does have files which are unencrypted, encrypt-all can be used to encrypt them.

View Current Encryption Status

Get the current encryption status and the loaded encryption module:

```
{occ-command-example-prefix} encryption:status
```

This is equivalent to checking Enable server-side encryption on your Admin page:

```
{occ-command-example-prefix} encryption:enable
Encryption enabled

Default module: OC_DEFAULT_MODULE
```

Recreating an Existing Master Key

If the master key needs replacing, for example, because it has been compromised, an occ command is available. The command is encryption:recreate-master-key. It replaces existing master key with new one and encrypts the files with the new key.

Decrypt Master-Key Encryption

You must first put your ownCloud server into single-user mode to prevent any user activity until encryption is completed.

```
{occ-command-example-prefix} maintenance:singleuser --on
Single user mode is currently enabled
```

Decrypt all user data files, or optionally a single user:

```
{occ-command-example-prefix} encryption:decrypt-all [username]
```

Disabling Encryption

To disable encryption, put your ownCloud server into single-user mode, and then disable your encryption module with these commands:

```
{occ-command-example-prefix} maintenance:singleuser --on
{occ-command-example-prefix} encryption:disable
```

Take it out of single-user mode when you are finished, by using the following command:

```
{occ-command-example-prefix} maintenance:singleuser --off
```



You may only disable encryption by using the occ Encryption Commands. Make sure you have backups of all encryption keys, including those for all your users.

Enabling User-Key Based Encryption From the Command-line

Limitations of User-Key Based Encryption

- Users added to groups cannot decrypt files on existing shares.
- OnlyOffice will not work.
- Impersonate will not work.
- OAuth2 does will not work.
- Elasticsearch will not work.

- Users getting access to an external storage which already contains existing encrypted files cannot get access to said files for reasons such as the group case above.
- When having data shared with a group and group membership changes after the share is established, subsequently added users will not be able to open the shared data unless the owner will share it again.

To enable User-Key based encryption:

To be safe, put your server in single user mode, to avoid any issues on a running instance, using the following command:

```
{occ-command-example-prefix} maintenance:singleuser --on
```

Then, enable the default encryption module app, using the following command:

```
{occ-command-example-prefix} app:enable encryption
```

After that, enable encryption, using the following command:

```
{occ-command-example-prefix} encryption:enable
```

Then, enable the user-key, using the following command:

```
{occ-command-example-prefix} encryption:select-encryption-type user-keys
```

Finally, encrypt all data, using the following command:

```
{occ-command-example-prefix} encryption:encrypt-all --yes
```

Now you can turn off the single user mode:

```
{occ-command-example-prefix} maintenance:singleuser --off
```

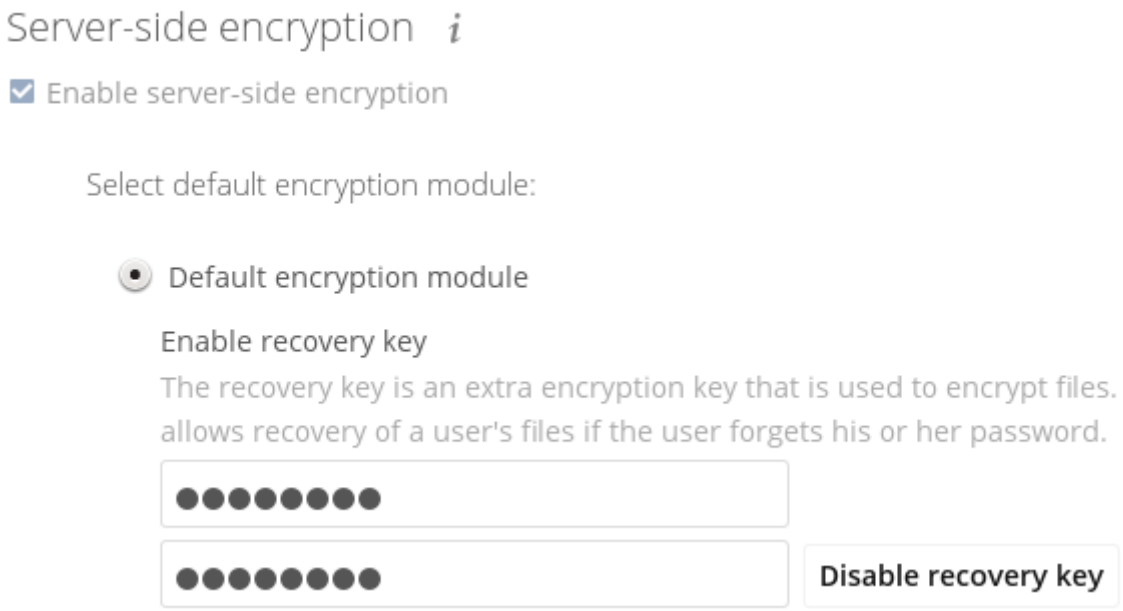
How To Enable Users File Recovery Keys

Once a user has encrypted their files, if they lose their ownCloud password, then they lose access to their encrypted files, as their files will be unrecoverable. It is not possible, when user files are encrypted, to reset a user's password using the standard reset process.

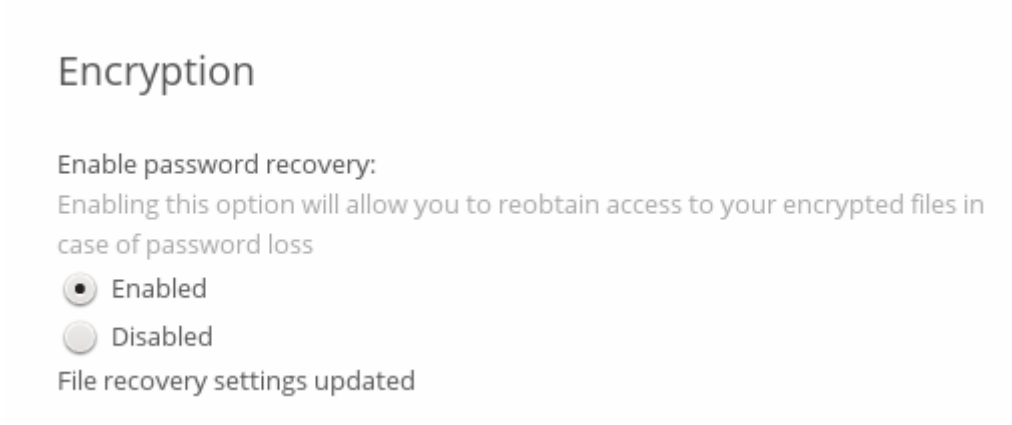
If so, you'll see a yellow banner warning:

"" Please provide an admin recovery password; otherwise, all user data will be lost. ""

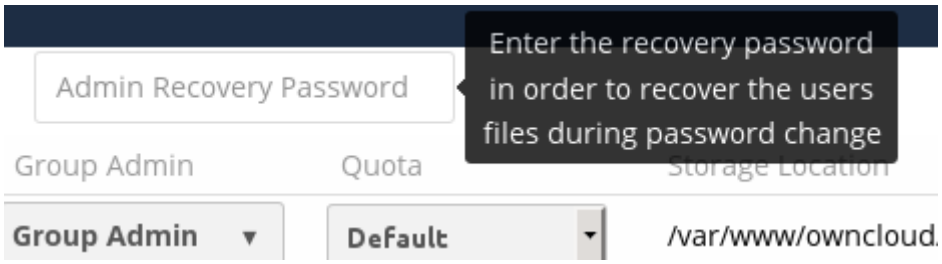
To avoid all this, create a Recovery Key. To do so, go to the Encryption section of your Admin page and set a recovery key password.



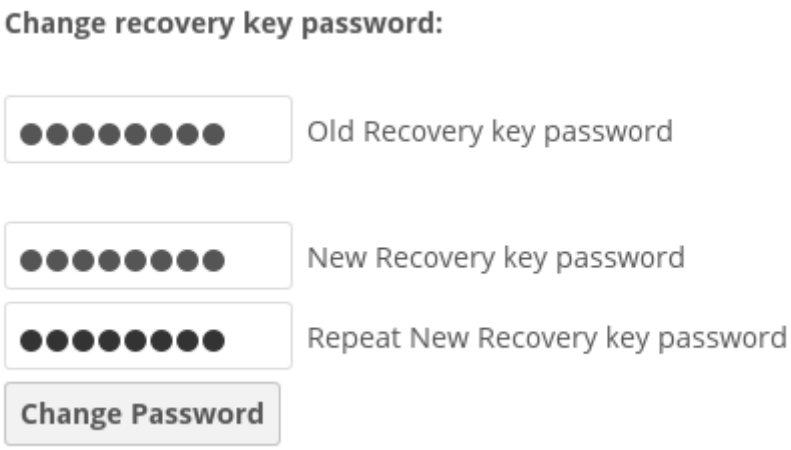
You then need to ask your users to opt-in to the Recovery Key. For the users to do this, they need to go to the **Personal** page and enable the recovery key. This signals that they are OK that the admin might have a way to decrypt their data for recovery reasons. If they do *not* do this, then the Recovery Key won't work for them.



For users who have enabled password recovery, give them a new password and recover access to their encrypted files, by supplying the Recovery Key on the Users page.



You may change your recovery key password.



Sharing a recovery key with a user group is **not** supported. This is only supported with the master key.

Changing The Recovery Key Password

If you have misplaced your recovery key password and need to replace it, here's what you need to do:

1. Delete the recovery key from both data/owncloud_private_keys and data/public-keys
2. Edit your database table oc_appconfig and remove the rows with the config keys recoveryKeyId and recoveryAdminEnabled for the appid files_encryption
3. Login as admin and activate the recovery key again with a new password. This will generate a new key pair
4. All users who used the original recovery key will need to disable it and enable it again. This deletes the old recovery share keys from their files and encrypts their files with the new recovery key

You can only change the recovery key password if you know the original. This is by design, as only admins who know the recovery key password should be able to change it. If not, admins could hijack the recovery key from each other

Replacing the recovery key will mean that all users will lose the possibility to recover their files until they have applied the new recovery key.

Decrypt User-Key Encryption

You must first put your ownCloud server into single-user mode, to prevent any user activity until encryption is completed.

```
{occ-command-example-prefix} maintenance:singleuser --on
Single user mode is currently enabled
```

Disabling Encryption

You may disable encryption only with occ. Make sure you have backups of all the encryption keys, including those for all users. When you do, put your ownCloud server into single-user mode, and then disable your encryption module with this command:

```
{occ-command-example-prefix} maintenance:singleuser --on
{occ-command-example-prefix} encryption:disable
```



Encryption cannot be disabled without the user's password or file recovery key. If you don't have access to at least one of these then there is no way to decrypt all files.

Then, take it out of single-user mode when you are finished with this command:

```
{occ-command-example-prefix} maintenance:singleuser --off
```

It is possible to disable encryption with the file recovery key, *if* every user uses them. If so, "decrypt all" will use it to decrypt all files.



It is **not** planned to move this to the next user login or a background job. If that was done, then login passwords would need to be stored in the database, which could be a security issue.

Move Key Location

View current location of keys:

```
{occ-command-example-prefix} encryption:show-key-storage-root
Current key storage root:  default storage location (data/)
```

You can move the keys to another folder inside your data directory. Moving your keys outside of your data folder is not supported. The folder must already exist, be owned by root and your HTTP group, and be restricted to root and your HTTP group. This example is for Ubuntu Linux. Note that the new folder is relative to your occ directory:

```
mkdir /var/www/owncloud/data/new_keys
chown -R root:www-data /var/www/owncloud/data/new_keys
chmod -R 0770 /var/www/owncloud/data/new_keys
{occ-command-example-prefix} encryption:change-key-storage-root new_keys
Change key storage root from default storage location to new_keys
Start to move keys:
  4 [============================]
Key storage root successfully changed to new_keys
```

Which Files Are Never Encrypted

Only the **data** in the files in data/<user>/files, and external storages (*if enabled*), is encrypted, *not* the filenames or folder structures. The following files are **never** encrypted:

- Existing files in the trash bin & Versions. Only new and changed files after encryption is enabled are encrypted.



You can post encrypt existing files via an occ encryption command.

- Existing files in Version
- Image thumbnails
- Previews from the Files app.
- The search index from the full text search app.
- Third-party app data

There may be other files that are not encrypted.



Only files that are exposed to third-party storage providers are guaranteed to be encrypted.

LDAP and Other External User Back-ends

If you use an external user back-end, such as an LDAP or Samba server, and you change a user's password on that back-end, the user will be prompted to change their ownCloud login to match on their next ownCloud login. The user will need both their old and new passwords to do this.

If you have enabled the recovery key, then you can change a user's password in the ownCloud Users panel to match their back-end password, and then — of course — notify the user and give them their new password.

Encrypting External Mountpoints

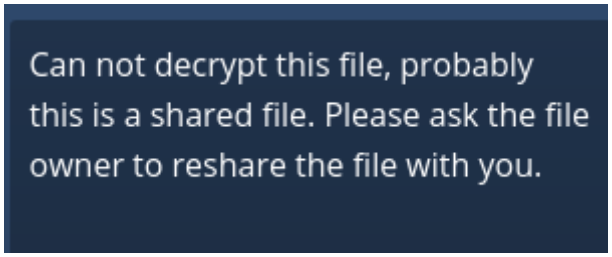
You and your users can encrypt individual external mount points. You must have external storage enabled on your Admin page, and enabled for your users. Encryption settings can be configured in the mount options for an external storage mount; see Mount Options.

Sharing Encrypted Files

After encryption is enabled, your users must also log out and log back in to generate their personal encryption keys. They will see a yellow warning banner that says "Encryption App is enabled, but your keys are not initialized. Please log-out and log-in again."

Also, share owners may need to re-share files after encryption is enabled. Users who are trying to access the share will see a message advising them to ask the share owner to re-share the file with them.

For individual shares, un-share and re-share the file. For group shares, share with any individuals who can't access the share. This updates the encryption, and then the share owner can remove the individual shares.



How To Enable Encryption From the Web-UI

1. First, you must enable the encrypton app, and then select an encryption type. Go to the Apps section of your Admin page, click on btn:[Show disabled Apps] and enable "**Default encryption module**".
2. After that go to the encryption section of your Admin page, and check the checkbox

btn:[Enable server-side encryption].

3. Then select an encryption Type. Masterkey and User-key are the options. Masterkey is recommended.
4. Now you must log out and then log back in to initialize your encryption keys.

Encryption Configuration Quick Guide

Encryption Types

ownCloud provides two encryption types:

User-Key:	Every user has their own private/public key pairs, and the private key is protected by the user's password.
Master Key:	There is only one key (or key pair) and all files are encrypted using that key pair.



These encryption types are **not compatible**. Please see User-Key Limitations for more details

Enable the Encryption App

Before you can use encryption you must enable the encryption app. You can do this either from the command-line or from the Web-UI.

Enable Encryption From the Command-Line

To enable the encryption app, run the following command:

```
{occ-command-example-prefix} app:enable encryption
```

If the encryption app successfully enables, then you should see the following confirmation:

```
encryption enabled
```



This should never happen, but the encryption app may not be packaged with your ownCloud installation. If so, you will see the following output when you attempt to enable it:

```
encryption not found
```

If that happens, then you need to install it manually. You can do this by cloning the encryption app, using the following command:

```
git clone https://github.com/owncloud/encryption.git apps/encryption
```

Enable Encryption From the Web-UI

To enable encryption from the Web-UI:

1. Go to menu:Settings[Admin > Apps] and click on kbd:[Show disabled apps]
2. When the disabled apps are rendered click btn:[Enable] under "*Default encryption module*".
3. After that go to menu:Settings[Admin > Encryption], and enable btn:[Enable server-side encryption].
4. Then, under "*Default encryption module*", select the desired encryption type, whether "*Master Key*" (recommended) or "*User-key*".
5. Now you must log out and then log back in to initialize your encryption keys.

Master Key Encryption

Overview

- The **recommended** type of encryption.
- Best to activate on new instances with no data.
- If you have existing data, use **encrypt all** command. Depending on the amount of existing data, this operation can take a long time.

Activate Master Key-Based Encryption

```
{occ-command-example-prefix} maintenance:singleuser --on
{occ-command-example-prefix} app:enable encryption
{occ-command-example-prefix} encryption:enable
{occ-command-example-prefix} encryption:select-encryption-type masterkey -y
{occ-command-example-prefix} encryption:encrypt-all --yes
{occ-command-example-prefix} maintenance:singleuser --off
```

View the Encryption Status

```
{occ-command-example-prefix} encryption:status
```

Decrypt Encrypted Files

Depending on the amount of existing data, this operation can take a long time.

```
{occ-command-example-prefix} maintenance:singleuser --on
{occ-command-example-prefix} encryption:decrypt-all
{occ-command-example-prefix} maintenance:singleuser --off
```

Deactivate Master Key-based Encryption

```
{occ-command-example-prefix} encryption:disable
# ignore the "already disabled" message
{occ-command-example-prefix} app:disable encryption
```

If the master key has been compromised or exposed, you can recreate it. You will need the current master key for it.

```
{occ-command-example-prefix} encryption:recreate-master-key
```

User-Specific Key-based Encryption

Activate User-Specific Key-based Encryption

```
{occ-command-example-prefix} maintenance:singleuser --on
{occ-command-example-prefix} app:enable encryption
{occ-command-example-prefix} encryption:enable
{occ-command-example-prefix} encryption:select-encryption-type user-keys
{occ-command-example-prefix} encryption:encrypt-all --yes
{occ-command-example-prefix} maintenance:singleuser --off
```

After User-specific encryption is enabled, users must log out and log back in to trigger the automatic personal encryption key generation process.

Set a Recovery Key

- Go to the "*Encryption*" section of your Admin page.
- Set a recovery key password.
- Ask the users to opt-in to the recovery key.

If a user decides not to opt-in to the recovery key and forgets or loses their password, **the user's data cannot be decrypted**. This leads to **permanent data loss**.

They need to:

- Go to menu:Settings[Personal > Encryption]
- Enable the Recovery Key

View the Encryption Status

```
{occ-command-example-prefix} encryption:status
```

Decrypt Encrypted Files

If you have an instance with a few users, you can use this example to decrypt the files. Note that you have to enter the password for each user manually. The ownCloud admin must be certain all users already have enabled the recovery password option in their personal settings page.

```
{occ-command-example-prefix} maintenance:singleuser --on
{occ-command-example-prefix} encryption:decrypt-all
#Choose the "Recovery key" Option
#Enter **Recovery Key** for **each user**

# Recovery Key is a password set by the admin
{occ-command-example-prefix} maintenance:singleuser --off
```

If you have a large instance with many users, use this to decrypt the files:

- Set the variable as export OC_RECOVERY_PASSWORD=1111, then run this set of commands: (Replace "1111" with your actual Recovery Key)

```
export OC_RECOVERY_PASSWORD=1111
{occ-command-example-prefix} maintenance:singleuser --on
sudo -E -u www-data php occ encryption:decrypt-all -m recovery -c yes
{occ-command-example-prefix} maintenance:singleuser --off
```

Deactivate User-Specific Key-based Encryption

```
{occ-command-example-prefix} encryption:disable

# ignore the "already disabled" message
{occ-command-example-prefix} app:disable encryption
```

Cleanup your database

Access your ownCloud database and remove the remaining entries that haven't been automatically removed with this command:

```
DELETE FROM oc_appconfig WHERE appid='encryption';
```

Cleanup your storage

Lastly you have to delete all encryption keys on storage by running this command:

(Modify the path to your data directory according to your installation)

```
find /var/www/html/owncloud/data -type d -name "files_encryption" -exec rm -R {}
+
```

At this point, keys are deleted from storage.

Master Key Based Encryption

Matthew Setter <matthew@matthewsetter.com> :toc: right :toclevels: 1 :keywords: master key based encryption, encryption :description: This guide will show you how to work with Master Key encryption in ownCloud.

Introduction

With Master Key based encryption, there is only one key (or key pair) and all files are encrypted using that key pair. This is highly recommended for new instances to avoid restrictions in functionality of user key encryption.

Enabling Master Key Based Encryption

There are two steps to be made to enable Master Key based encryption.



We strongly encourage you to put your server in single user mode before setting up encryption.

To do so, run the following command:

```
{occ-command-example-prefix} maintenance:singleuser --on
```

Enable the Encryption App

To enable the encryption app, run the following command:

```
{occ-command-example-prefix} app:enable encryption
```

If the encryption app successfully enables, then you should see the following confirmation:

```
encryption enabled
```



This should never happen, but the encryption app may not be packaged with your ownCloud installation. If so, you will see the following output when you attempt to enable it:

```
encryption not found
```

If that happens, then you need to install it manually. You can do this by cloning the encryption app, using the following command:

```
git clone https://github.com/owncloud/encryption.git apps/encryption
```

Enable and Configure Master Key Based Encryption

To enable and configure Master Key based encryption via the command-line, involves several commands.

1. Enable encryption.
2. Enable the master key.



Master Key Mode has to be setup in a newly created instance.

3. Encrypt all data.

The following example shows how to carry out these steps.

```
{occ-command-example-prefix} encryption:enable
{occ-command-example-prefix} encryption:select-encryption-type masterkey
{occ-command-example-prefix} encryption:encrypt-all --yes
```



This command is not typically required, as the master key is often enabled at install time. As a result, when enabling it, there *should* be no data to encrypt. However, if you have enabled master key encryption post-installation, existing files will not be automatically encrypted; **only** newly created files. To encrypt existing files use the encrypt-all command as described above. Before doing so, set the instance into single user mode for that task.

View Current Encryption Status

Retrieves the current encryption status and the name of the loaded encryption module.

```
{occ-command-example-prefix} encryption:status
```

This is equivalent to checking "Enable server-side encryption" on your Admin page:

```
{occ-command-example-prefix} encryption:enable
Encryption enabled

Default module: OC_DEFAULT_MODULE
```

Recreate an Existing Master Key

If the master key needs replacing, for example, because it has been compromised, an occ command is available. The command is encryption:recreate-master-key. It replaces existing master key with new one and encrypts the files with the new key.

Decrypt Master-Key Based Encryption

You must first put your ownCloud server into single-user mode to prevent any user activity until encryption is completed.

```
{occ-command-example-prefix} maintenance:singleuser --on
Single user mode is currently enabled
```

Decrypt all user data files, or optionally a single user:

```
{occ-command-example-prefix} encryption:decrypt-all [username]
```

Disable Encryption

To disable encryption, put your ownCloud server into single-user mode, and then disable your encryption module with these commands:

```
{occ-command-example-prefix} maintenance:singleuser --on
{occ-command-example-prefix} encryption:disable
```

Take it out of single-user mode when you are finished, by using the following command:

```
{occ-command-example-prefix} maintenance:singleuser --off
```



You may only disable encryption by using the occ Encryption Commands. Make sure you have backups of all encryption keys, including those for all your users.

Sharing Encrypted Files

After encryption is enabled, your users must also log out and log back in to (automatically) generate their personal encryption keys. They will see a yellow warning banner that says

"" Encryption App is enabled, but your keys are not initialized. Please log-out and log-in again. ""

Also, share owners may need to re-share files after encryption is enabled. Users who are trying to access the share will see a message advising them to ask the share owner to re-share the file with them.

For individual shares, un-share and re-share the file. For group shares, share with any individuals who can't access the share. This updates the encryption, and then the share owner can remove the individual shares.

User-Key Based Encryption

Limitations

- Users added to groups cannot decrypt files on existing shares.
- OnlyOffice will not work.
- Impersonate will not work.
- OAuth2 does will not work.
- Elasticsearch will not work.
- Users getting access to an external storage which already contains existing encrypted files cannot get access to said files for reasons such as the group case above.
- When having data shared with a group and group membership changes after the share is established, subsequently added users will not be able to open the shared data unless the owner will share it again.

Enable User-Key Based Encryption

To enable User-Key based encryption requires two steps:

1. Enable the encryption app
2. Enable and configure User-Key based Encryption



We strongly encourage you to put your server in single user mode before setting up encryption.

To do so, run the following command:

```
{occ-command-example-prefix} maintenance:singleuser --on
```

Enable the Encryption App

To enable the encryption app, run the following command:

```
{occ-command-example-prefix} app:enable encryption
```

If the encryption app successfully enables, then you should see the following confirmation:

```
encryption enabled
```



This should never happen, but the encryption app may not be packaged with your ownCloud installation. If so, you will see the following output when you attempt to enable it:

```
encryption not found
```

If that happens, then you need to install it manually. You can do this by cloning the encryption app, using the following command:

```
git clone https://github.com/owncloud/encryption.git apps/encryption
```

Enable and Configure User-Key Based Encryption

To enable and configure User-Key based encryption, you need to:

1. Enable the default encryption module app
2. Enable encryption
3. Enable the user-key, using the following command:
4. Encrypt all data
5. Turn off the single user mode

The following example shows how to carry out these steps.

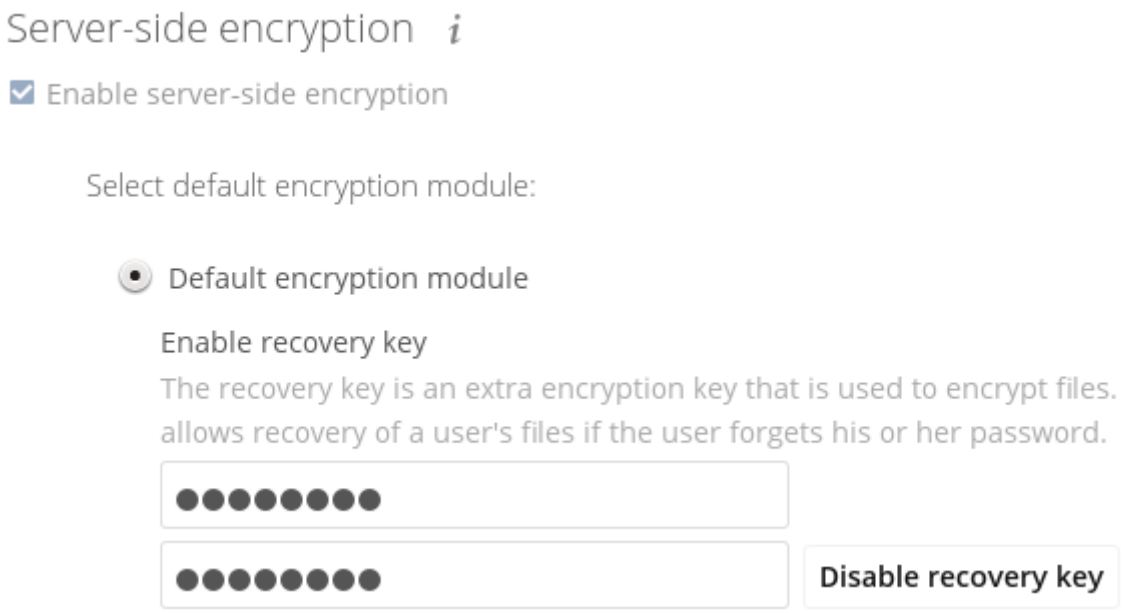
```
{occ-command-example-prefix} encryption:enable
{occ-command-example-prefix} encryption:select-encryption-type user-keys
{occ-command-example-prefix} encryption:encrypt-all --yes
{occ-command-example-prefix} maintenance:singleuser --off
```

How To Enable Users File Recovery Keys

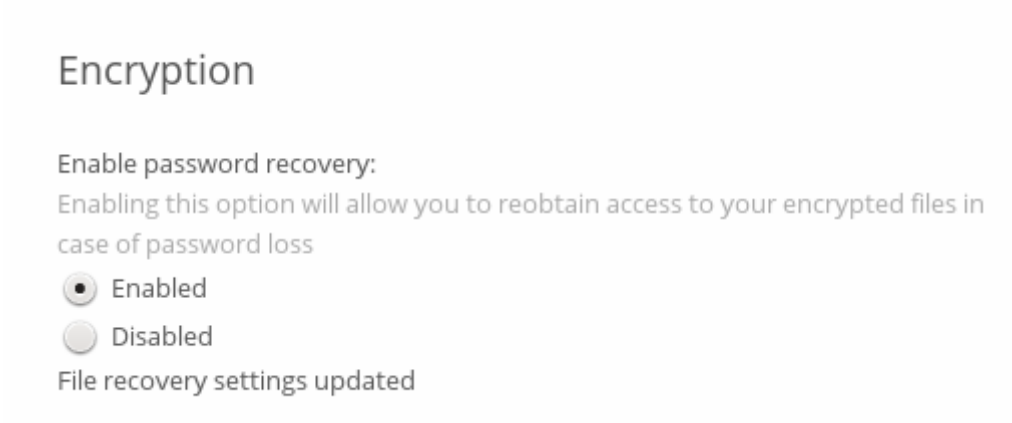
Once a user has encrypted their files, if they lose their ownCloud password, then they lose access to their encrypted files, as their files will be unrecoverable. It is not possible, when user files are encrypted, to reset a user's password using the standard reset process. If so, you'll see a yellow banner warning:

"" Please provide an admin recovery password; otherwise, all user data will be lost. ""

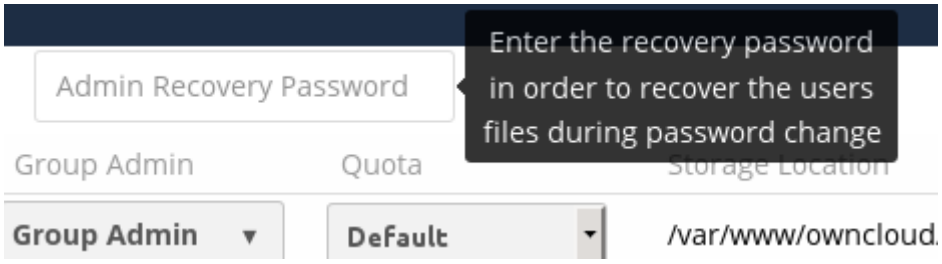
To avoid all this, create a recovery key. To do so, go to menu:Settings[Admin > encryption] and set a recovery key password.



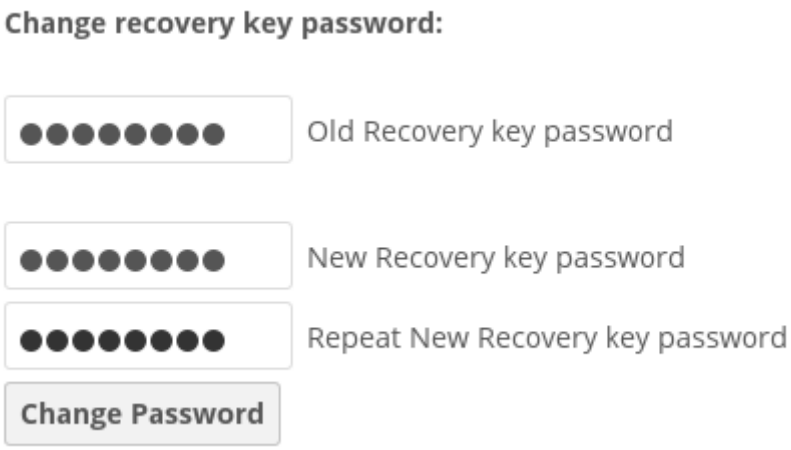
You then need to ask your users to opt-in to the Recovery Key. For the users to do this, they need to go to the ''**Personal**'' page and enable the recovery key. This signals that they accept that the admin might have a way to decrypt their data for recovery reasons. If they do *not* do this, then the recovery key won't work for them.



For users who have enabled password recovery, give them a new password and recover access to their encrypted files, by supplying the Recovery Key on the Users page.



Because the recovery key is password protected, you may change its password now.



Sharing a recovery key with a user group is **not** supported. This is only supported with the master key.

Changing The Recovery Key Password

If you have misplaced your recovery key password and need to replace it, here's what you need to do:

1. Delete the recovery key from both data/owncloud_private_keys and data/public-keys
2. Edit your database table oc_appconfig and remove the rows with the config keys recoveryKeyId and recoveryAdminEnabled for the appid files_encryption
3. Login as admin and activate the recovery key again with a new password. This will generate a new key pair
4. All users who used the original recovery key will need to disable it and enable it again. This deletes the old recovery share keys from their files and encrypts their files with the new recovery key

You can only change the recovery key password if you know the original. This is by design, as only admins who know the recovery key password should be able to change it. If not, admins could hijack the recovery key from each other

Replacing the recovery key will mean that all users will lose the possibility to recover their files until they have applied the new recovery key.

Decrypt User-Key Encryption

You must first put your ownCloud server into single-user mode, to prevent any user activity until encryption is completed.

```
{occ-command-example-prefix} maintenance:singleuser --on
Single user mode is currently enabled
```

Sharing Encrypted Files

After encryption is enabled, your users must also log out and log back in to (automatically) generate their personal encryption keys. They will see a yellow warning banner that says

"" Encryption App is enabled, but your keys are not initialized. Please log-out and log-in again. ""

Also, share owners may need to re-share files after encryption is enabled. Users who are trying to access the share will see a message advising them to ask the share owner to re-share the file with them.

For individual shares, un-share and re-share the file. For group shares, share with any individuals who can't access the share. This updates the encryption, and then the share owner can remove the individual shares.

Moving Encryption Key Location

View current location of keys:

```
{occ-command-example-prefix} encryption:show-key-storage-root
Current key storage root:  default storage location (data/)
```

You can move the keys to another folder inside your data directory. Moving your keys outside of your data folder is not supported. The folder must already exist, be owned by root and your HTTP group, and be restricted to root and your HTTP group.

This example is for *Ubuntu Linux*. Note that the new folder is relative to your occ directory (which in the example below is assumed that owncloud is installed at /var/www/owncloud):

```
mkdir /var/www/owncloud/data/new_keys
chown -R root:www-data /var/www/owncloud/data/new_keys
chmod -R 0770 /var/www/owncloud/data/new_keys
{occ-command-example-prefix} encryption:change-key-storage-root new_keys
Change key storage root from default storage location to new_keys
Start to move keys:
  4 [============================]
Key storage root successfully changed to new_keys
```

Sharing Encrypted Files

After encryption is enabled, your users must also log out and log back in to (automatically) generate their personal encryption keys. They will see a yellow warning banner that says

"" Encryption App is enabled, but your keys are not initialized. Please log-out and log-in again. ""

Also, share owners may need to re-share files after encryption is enabled. Users who are trying to access the share will see a message advising them to ask the share owner to re-share the file with them.

For individual shares, un-share and re-share the file. For group shares, share with any individuals who can't access the share. This updates the encryption, and then the share owner can remove the individual shares.

External Storage

In this section you will find all the details you need to configure external storage in ownCloud.

External Storage Authentication Mechanisms

Introduction

ownCloud storage backends accept one or more authentication schemes such as passwords, OAuth, or token-based, to name a few examples. Each authentication scheme may be implemented by multiple authentication mechanisms. Different mechanisms require different configuration parameters, depending on their behaviour.

Special Mechanisms

The **None** authentication mechanism requires no configuration parameters, and is used when a backend requires no authentication.

The **Built-in** authentication mechanism itself requires no configuration parameters, but is used as a placeholder for legacy storages that have not been migrated to the new system and do not take advantage of generic authentication mechanisms. The authentication parameters are provided directly by the backend.

Password-based Mechanisms

The **Username and password** mechanism requires a manually-defined username and password. These get passed directly to the backend.

The **Log-in credentials, save in session** mechanism uses the ownCloud login credentials of the user to connect to the storage. These are not stored anywhere on the server, but rather in the user session, giving increased security. The drawbacks are that sharing is disabled when this mechanism is in use, as ownCloud has no access to the storage credentials, and background file scanning does not work.



here is a workaround that allows background file scanning when using **Log-in credentials, save in session**, and that is using Ajax cron mode. Be aware that the Ajax cron mode is triggered by browsing the ownCloud Web GUI.

Known Limitations

Please be aware that any operations must be performed by the logged-in mount owner, as credentials are not stored anywhere. As a result, there are three known limitations, for both admin and personal mounts where both have the "*log-in credentials, save in session*" option.

These are:

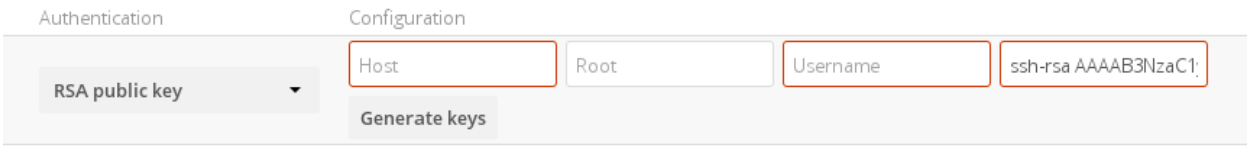
1. Directly sharing the storage or any of its sub-folders will go through, but the recipient will not see the share mounted. This is because the mount cannot be set up due to missing credentials. Federated sharing is also affected, because it works on a "*public link share token*" basis, which itself doesn't contain the user's storage password. As a result, the storage cannot be mounted in this case either.
2. Any background task operating on the storage, such as background scanning.
3. Any occ command that operates on the storage, such as occ files:scan, will have no effect.



Enterprise Users Only
The enterprise version has a mode called "**Save in DB**" where the credentials are saved, in encrypted form, in the database (via the WND app). In this mode, all of the above operations work.

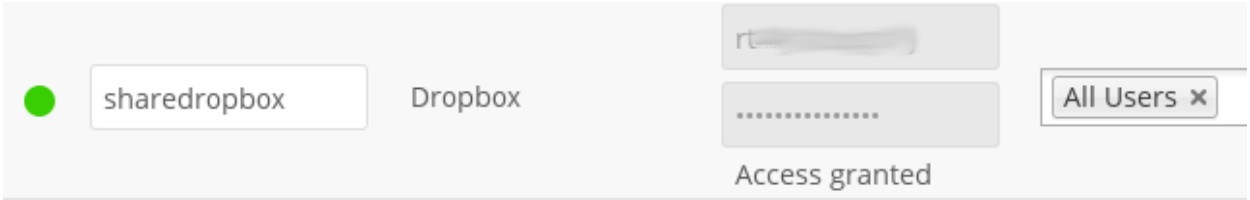
Public-key Mechanisms

Currently only the RSA mechanism is implemented, where a public/private keypair is generated by ownCloud and the public half shown in the GUI. The keys are generated in the SSH format, and are currently 1024 bits in length. Keys can be regenerated with a button in the GUI.



OAuth

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



If ownCloud client's are unable to connect to your ownCloud server, check that the bearer authorization header is not being stripped out.

Amazon S3

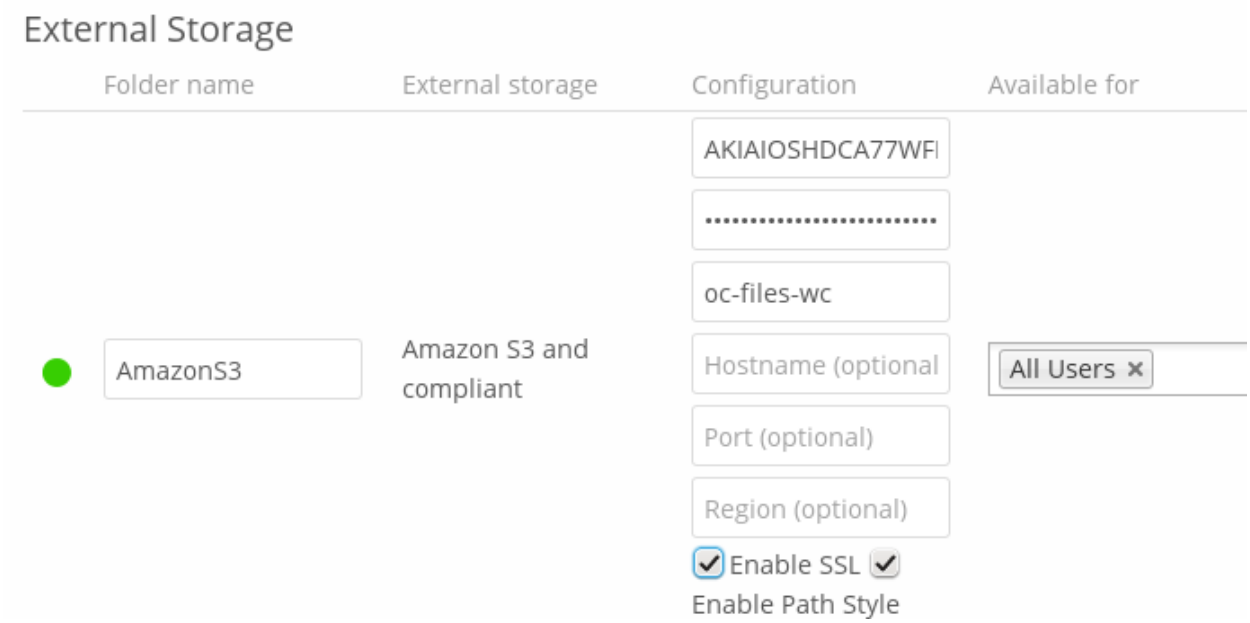
To connect your Amazon S3 buckets to ownCloud, you will need:

- S3 access key
- S3 secret key
- Bucket name

In the **Folder name** field enter a local folder name for your S3 mountpoint. If this does not exist it will be created.

In the **Available for** field enter the users or groups who have permission to access your S3 mount.

The Enable SSL checkbox enables HTTPS connections; using HTTPS is always highly-recommended.



Optionally, you can override the hostname, port and region of your S3 server, which is required for non-Amazon servers such as Ceph Object Gateway.

Enable path style is usually not required (and is, in fact, incompatible with newer Amazon datacenters), but can be used with non-Amazon servers where the DNS infrastructure cannot be controlled. Ordinarily, requests will be made with http://bucket.hostname.domain/, but with path style enabled, requests are made with http://hostname.domain/bucket instead.

See External Storage Configuration for additional mount options and information.

See auth_mechanisms for more information on authentication schemes.

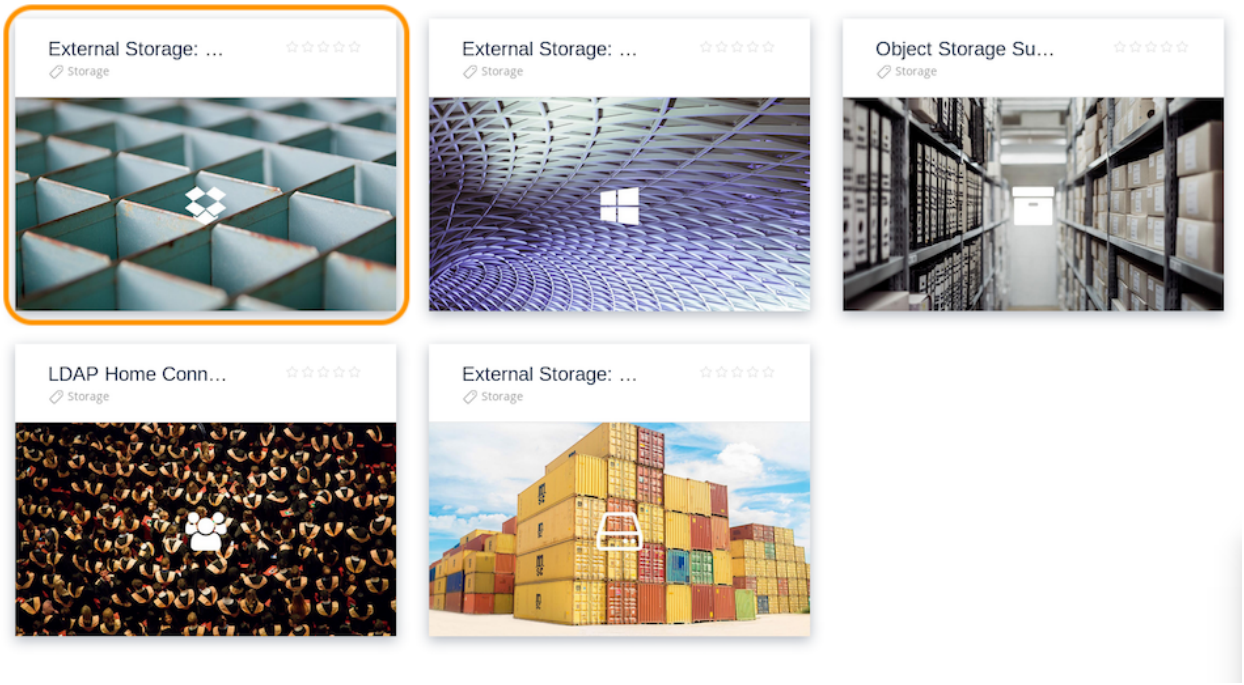
Dropbox

Introduction

To connect Dropbox to your ownCloud installation requires four steps to be completed.

Install ownClouds Dropbox app

Install the External Storage Dropbox app from the ownCloud Marketplace



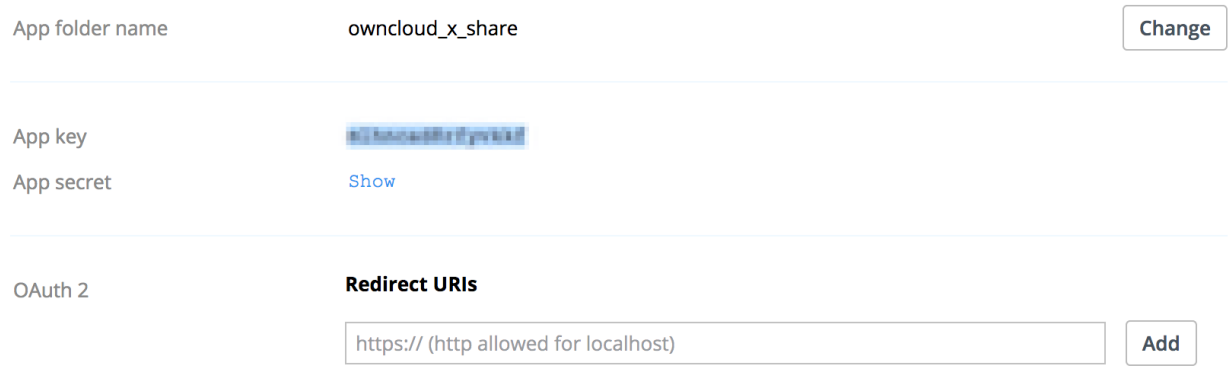
1. Click btn:[Market] in the ownCloud web UI drop-down menu on the left side
2. Go to the **Storage** category
3. Select **External Storage: Dropbox** App
4. Click btn:[INSTALL]

Create a Dropbox app

Next, you need to create a Dropbox app. To do that, open the new app creation form, where you see three questions:

1. "Choose an API" –> "Dropbox API"
2. "Choose the type of access" –> "App folder"
3. "Name your app"

With all of the required details filled out, click the blue btn:[Create app] button, in the bottom, right-hand corner. After you do that, the settings page for the application loads.



Redirect URI: Here you must enter the exact URL of the page where you configure the storage.

Examples:

When configuring as an **admin**:

```
http(s)://<<Server_Address>>/index.php/settings/admin?sectionid=storage
```

When configuring as a **user**:

```
http(s)://<<Server_Address>>/index.php/settings/personal?sectionid=storage
```

Create a Dropbox Share

To create a Dropbox share, under menu:Admin[Settings > Storage], check the btn:[Enable external storage] checkbox, if it's not already checked. Then, in the drop-down list under menu:External storage[], click the btn:[Dropbox] option.

There are two Dropbox options in the drop-down list, as Dropbox functionality is currently part of ownCloud's core. However, the internal Dropbox functionality should be removed in ownCloud 10.0.4.

Then, you need to provide a name for the folder in the "Folder name" field, and a "client key" and "client secret", located in the "Configuration" column. The client key and client secret values are the "App key" and "App secret" fields which you saw earlier in your Dropbox app's configuration settings page.

After you have added these three settings, click btn:[Grant access]. ownCloud then interacts with Dropbox's API to set up the new shared folder. If the process is successful, a green circle icon appears, at the far left-hand side of the row, next to the folder's name.



Other Options

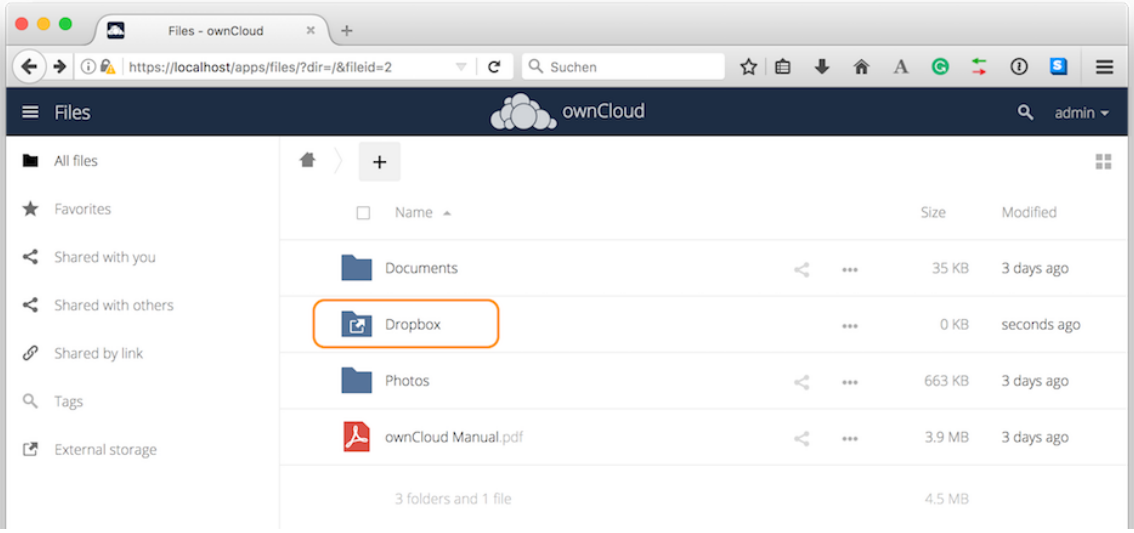
If you want to restrict access to the share to a select list of users and groups, you can add them to the field in the "Available for" column.

Using the Dropbox Share

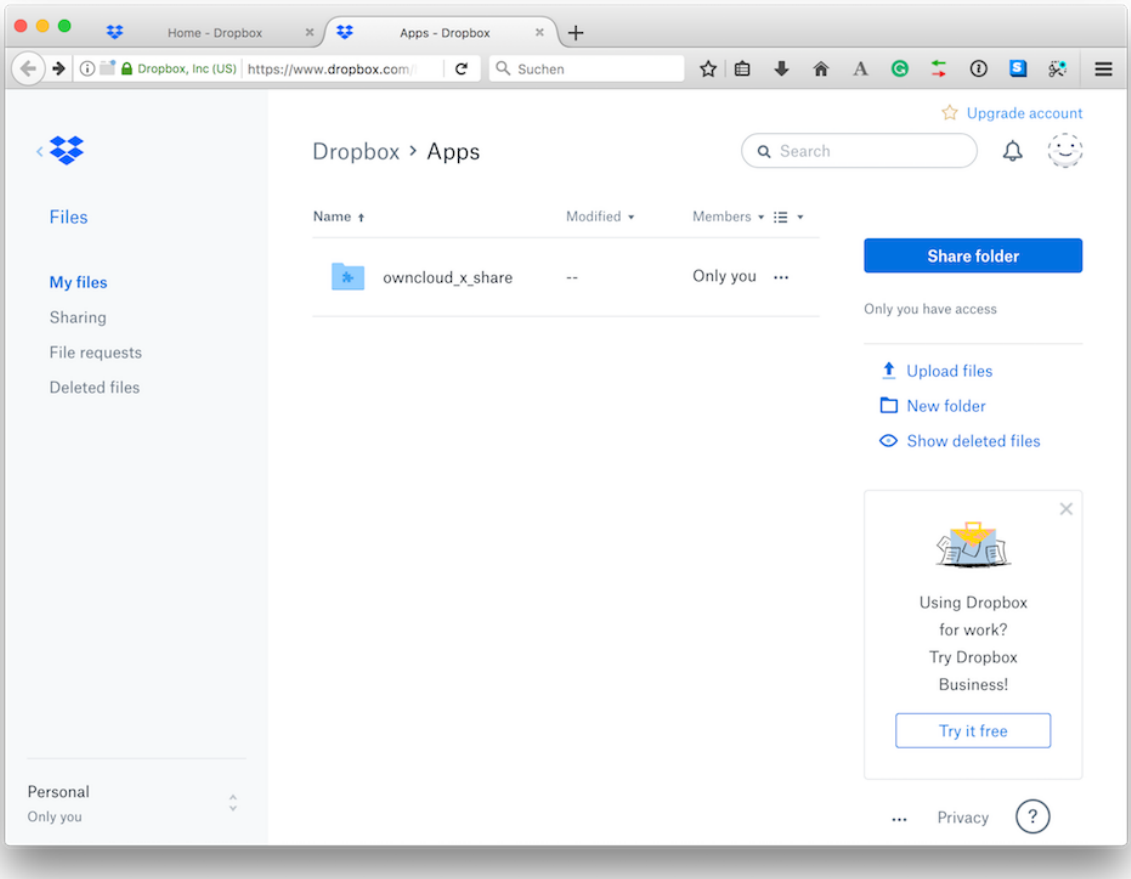
After a Dropbox-backed share is created, a new folder is available under "All Files". It has the name that you gave it when you created the share, and it is represented by an external share folder icon, as in the image below.



This links to a new folder in your Dropbox account, under "Dropbox > Apps", with the name of the Dropbox app that you created.

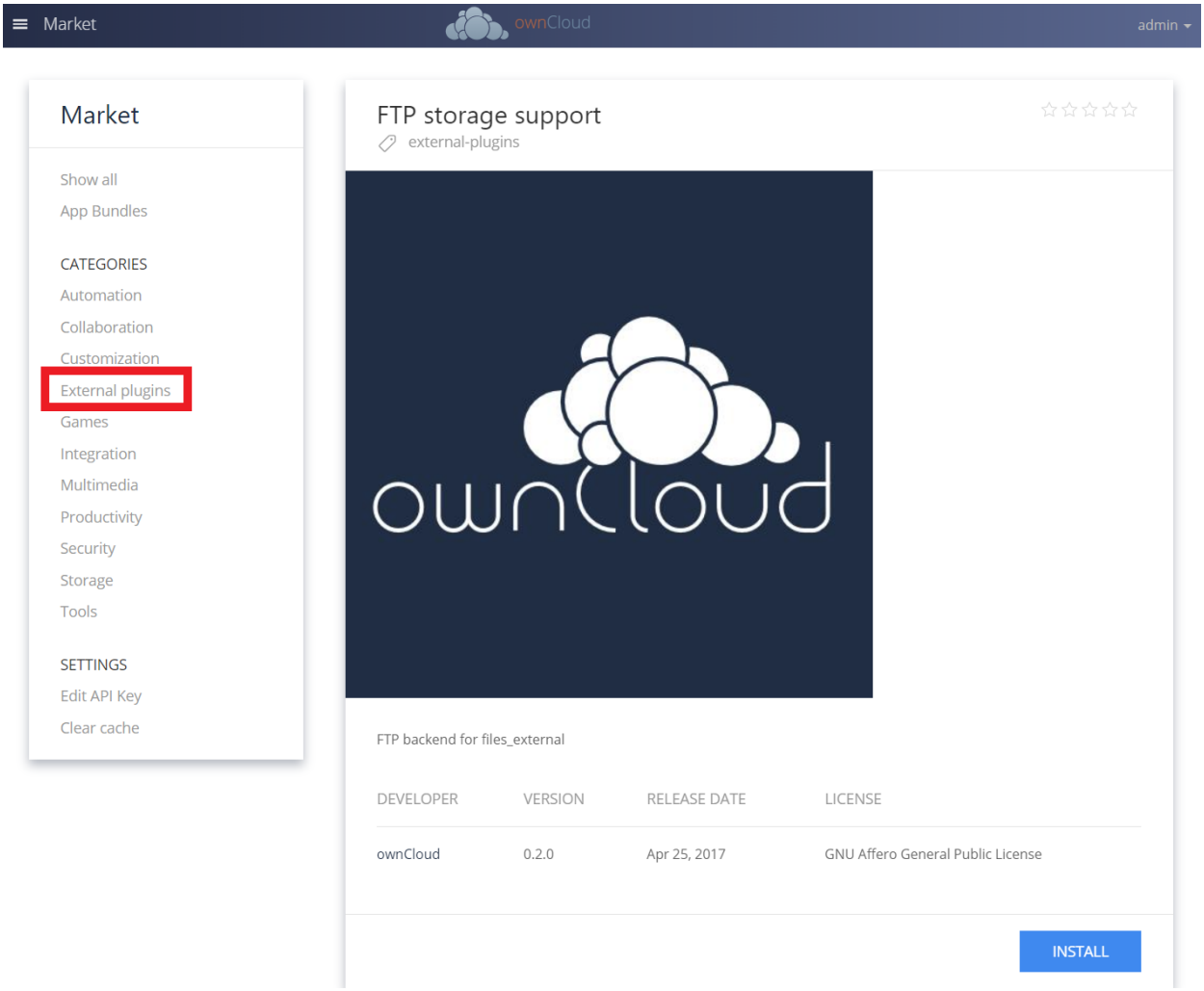


Now, if you add files and folders in either the new Dropbox folder or the new ownCloud folder, after being synced, they will be visible inside the other.



FTP/FTPS

If you want to mount an FTP Storage, please install the FTP Storage Support app from the ownCloud Marketplace.



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)
- Username and password to access the resource
- Remote Subfolder, the FTP directory to mount in ownCloud. ownCloud 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, ownCloud 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. See Importing System-wide and Personal SSL Certificates for more information.



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 Configuration GUI for additional mount options and information.

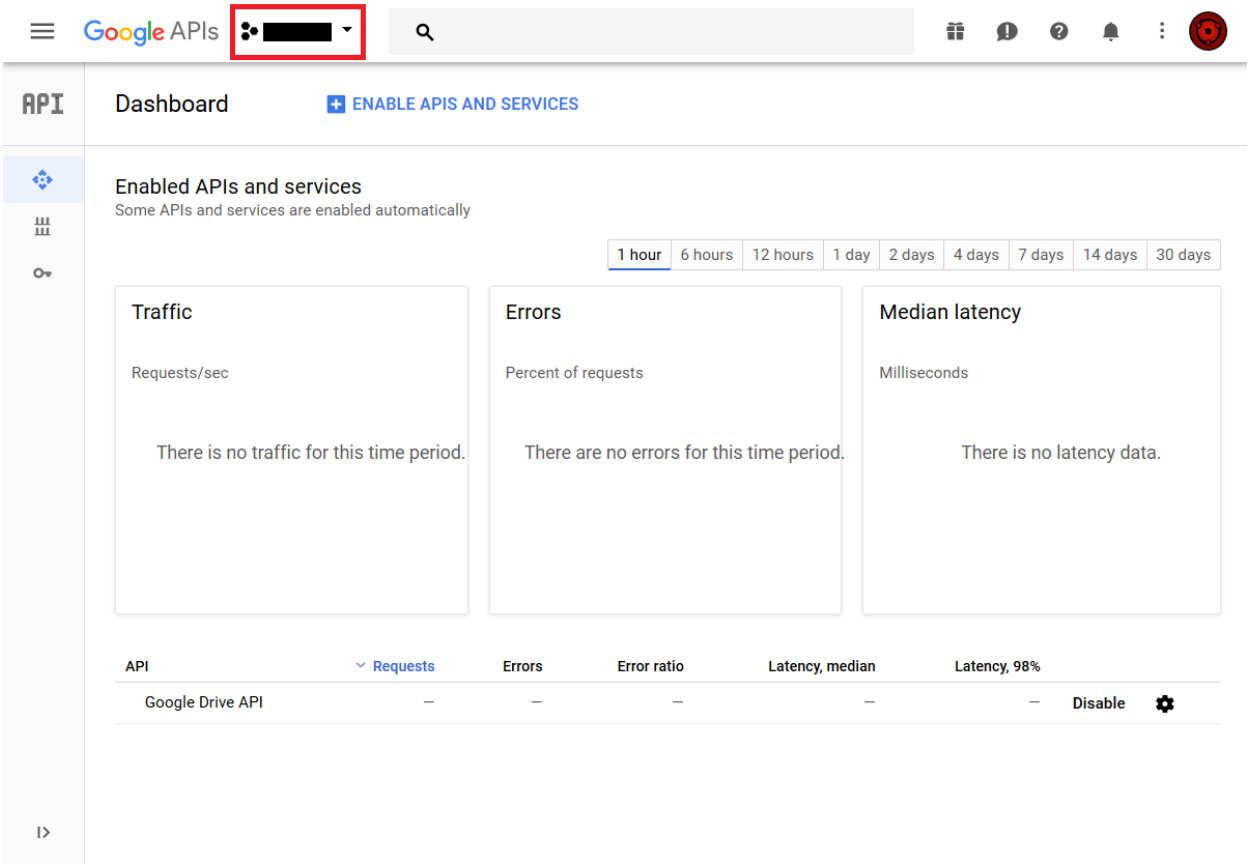
FTP uses the password authentication scheme; see External Storage Authentication mechanisms for more information on authentication schemes.

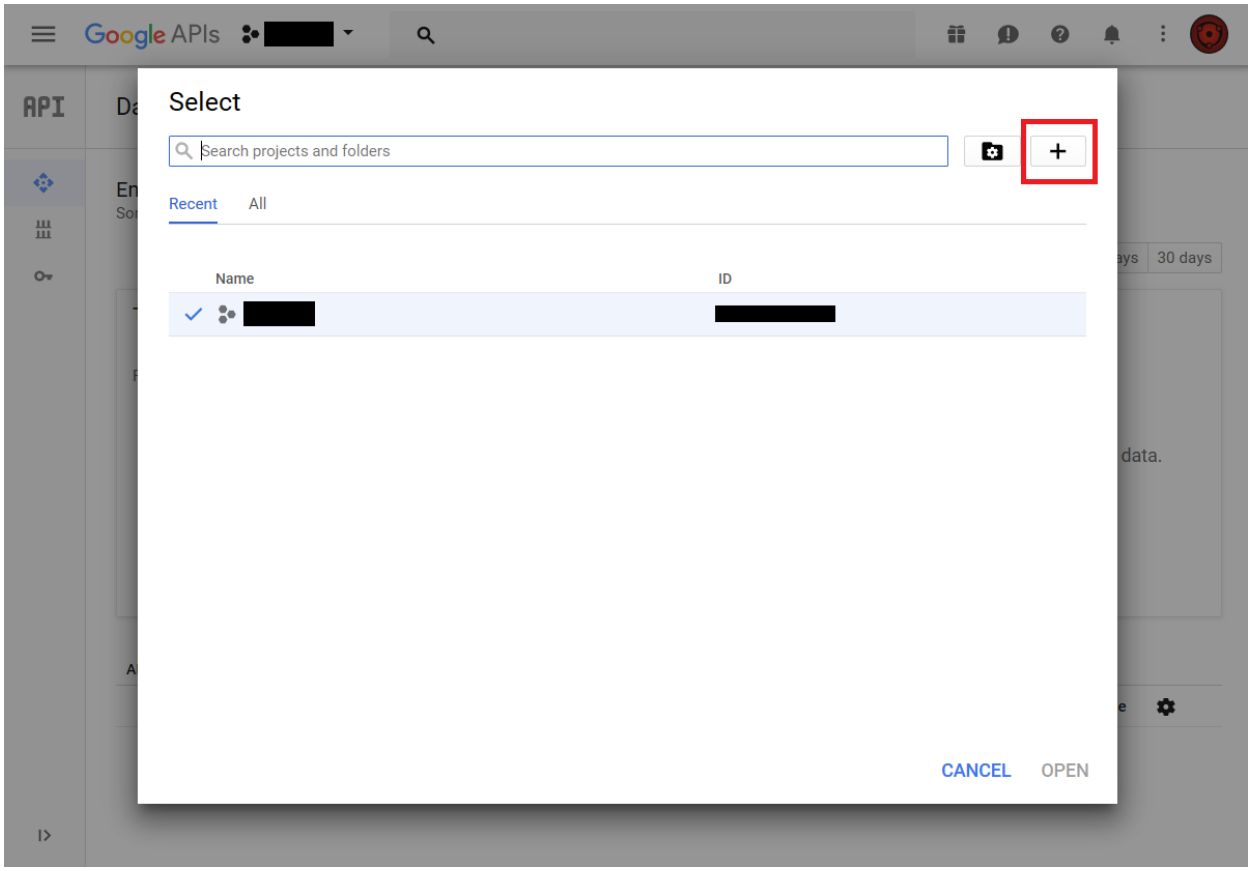
Google Drive

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

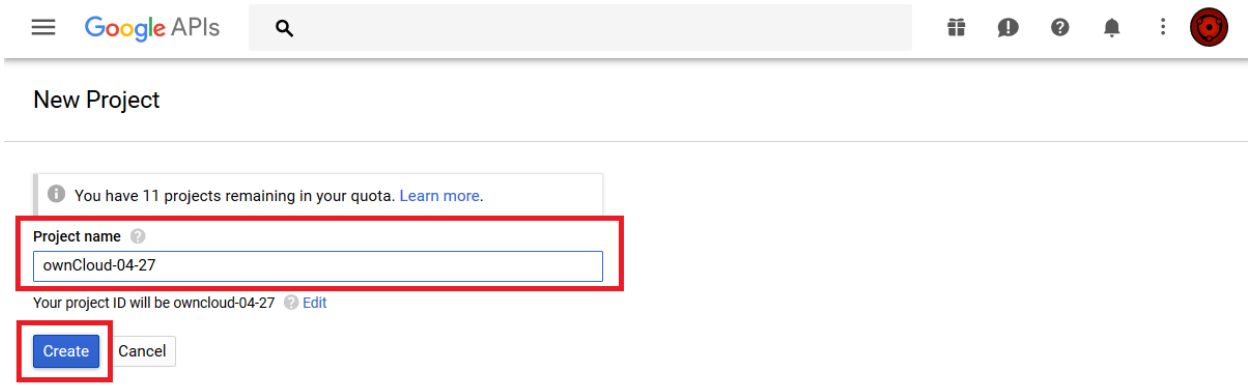
All applications that access a Google API must be registered through the Google Cloud Console. Follow along carefully because the Google interface is a bit of a maze and it's easy to get lost.

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

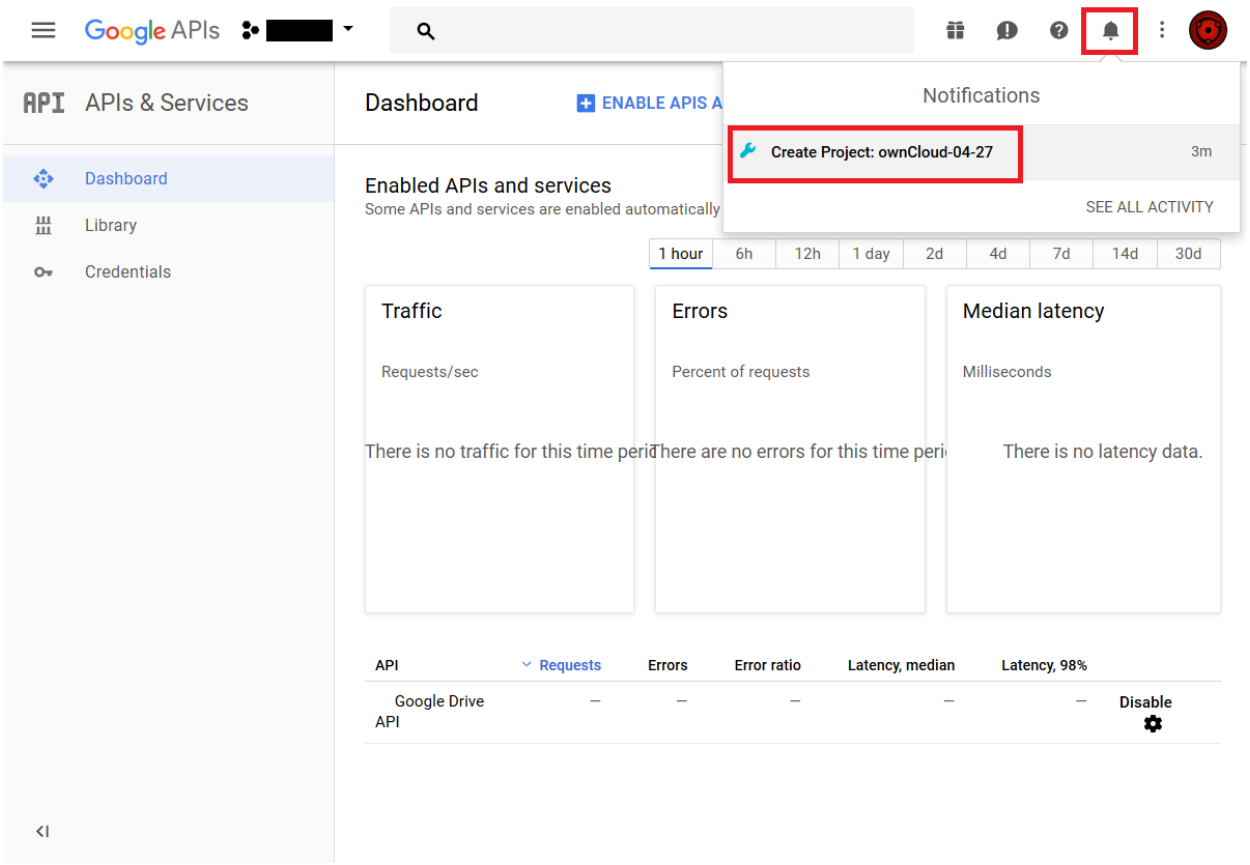




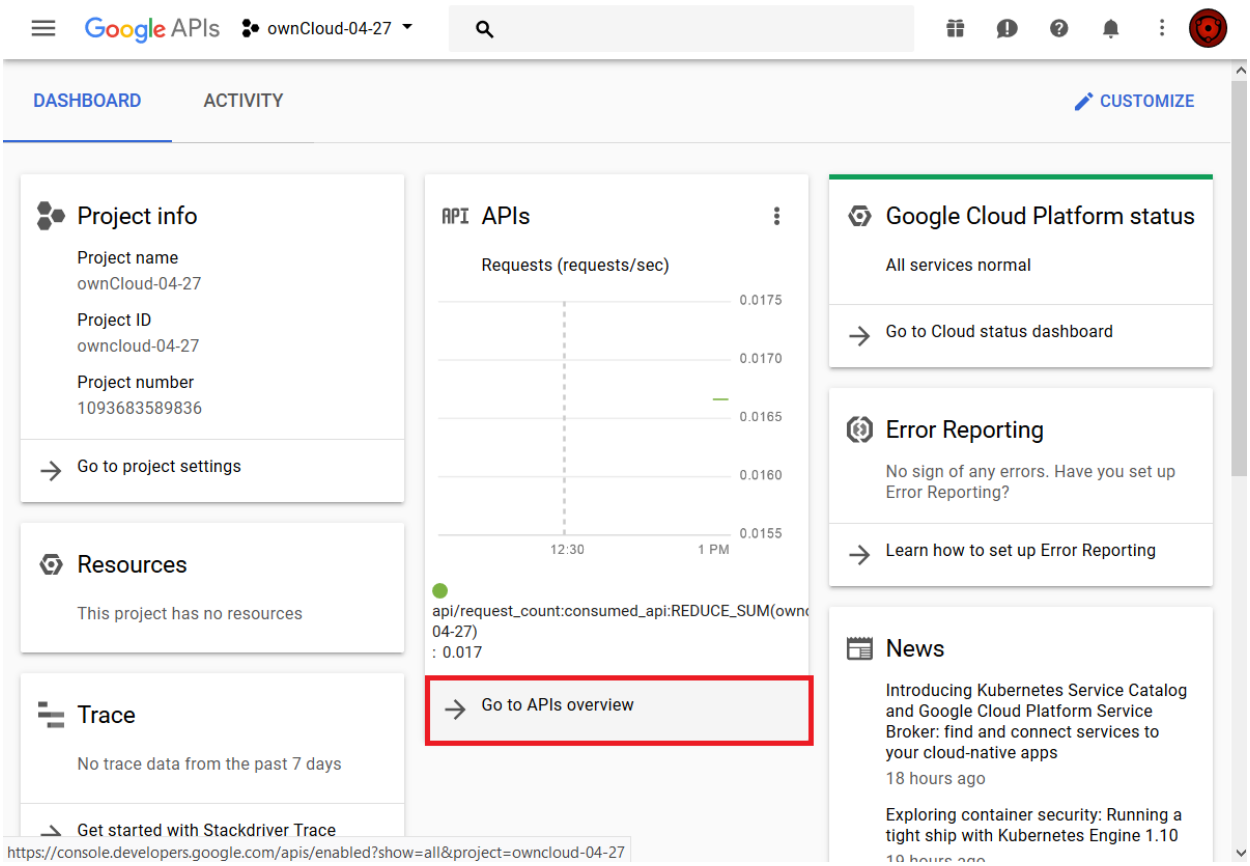
Give your project a name, and either accept the default **Project ID** or create your own, then click the btn:[Create] button. For this example a random name was chosen, "owncloud-04-27". However, feel free to choose your own name.

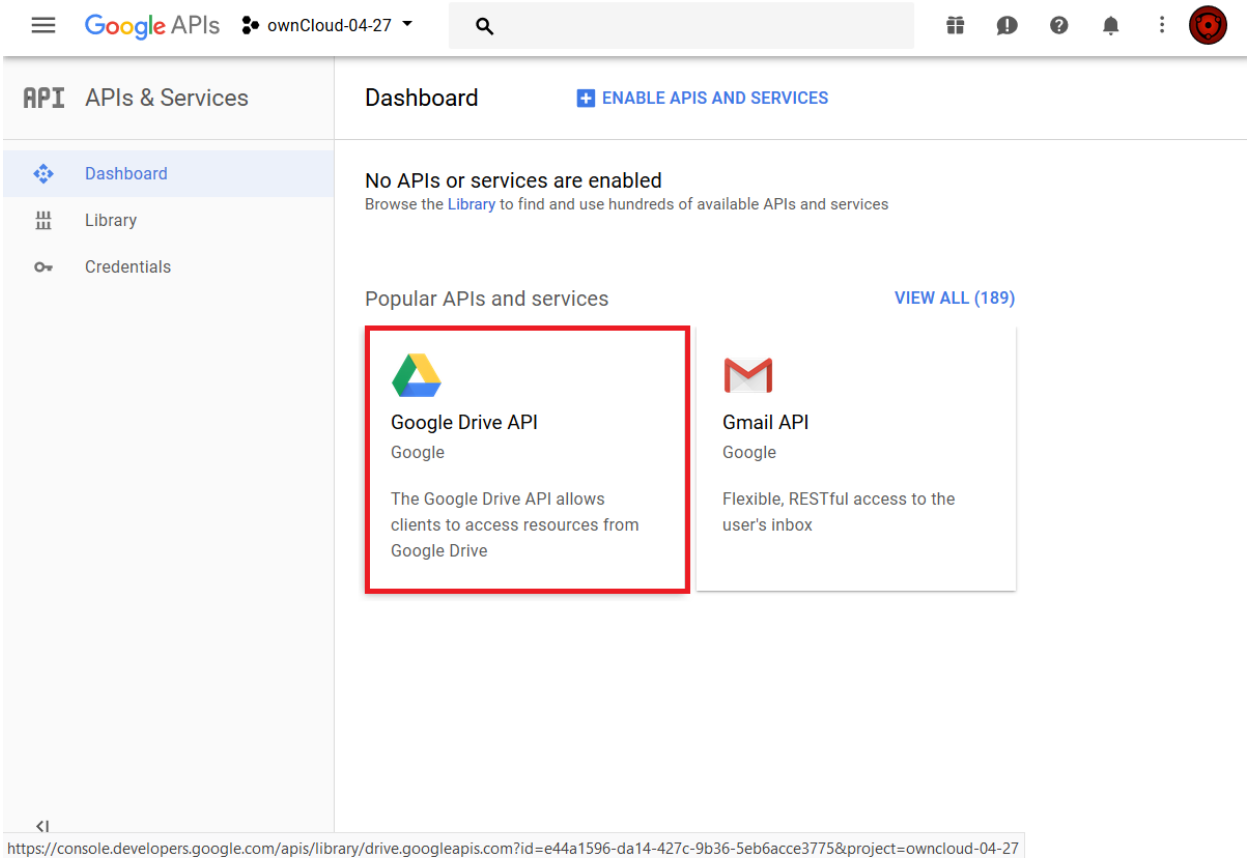


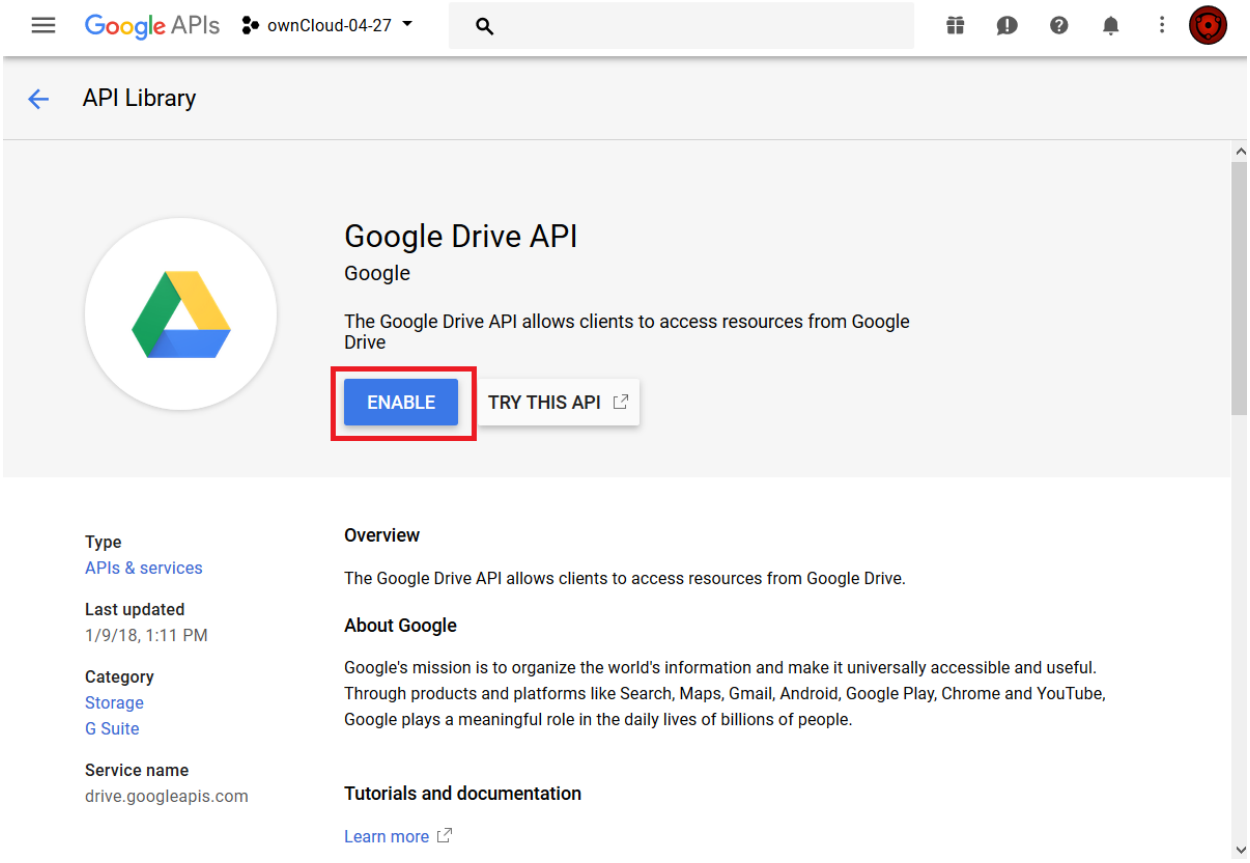
After your project is created, click on the btn:[notifications bell] and select your project.



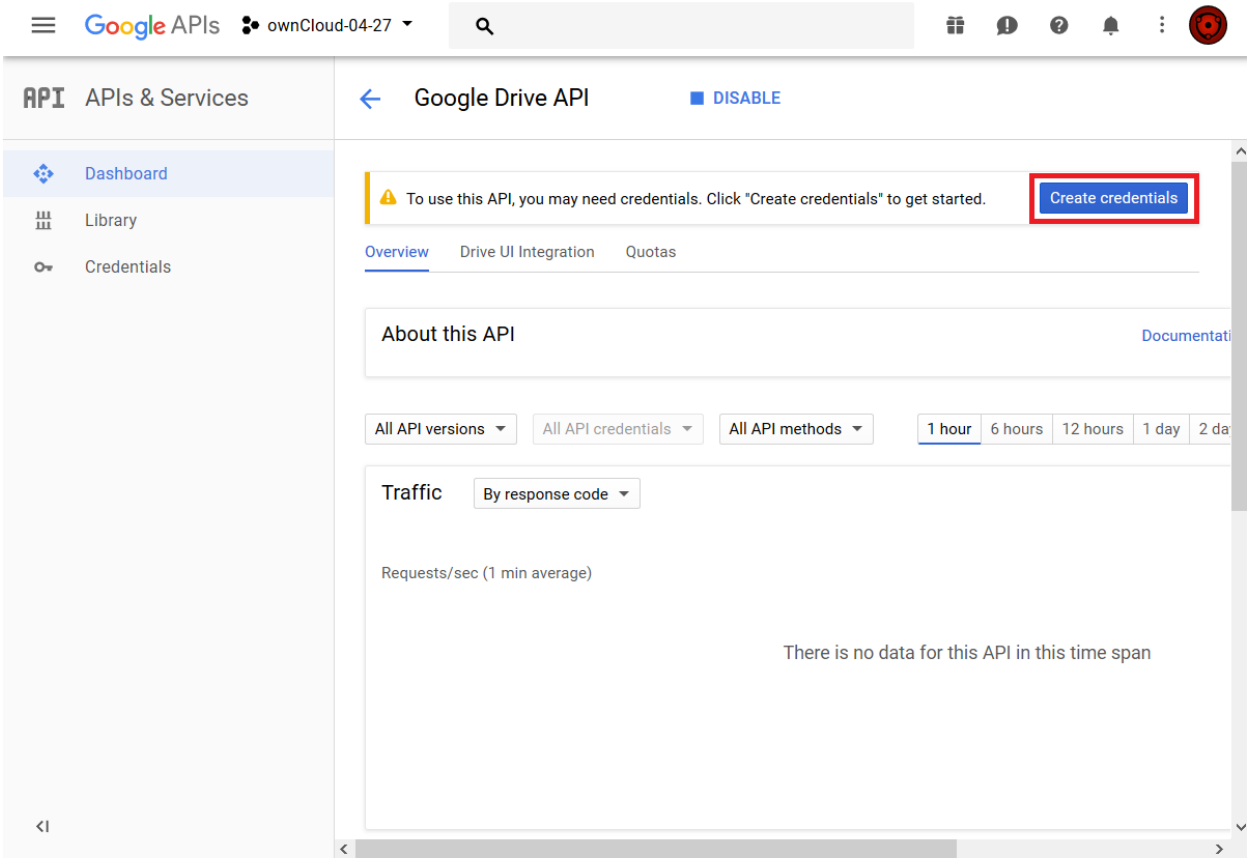
Go to Api overview to select google's API.



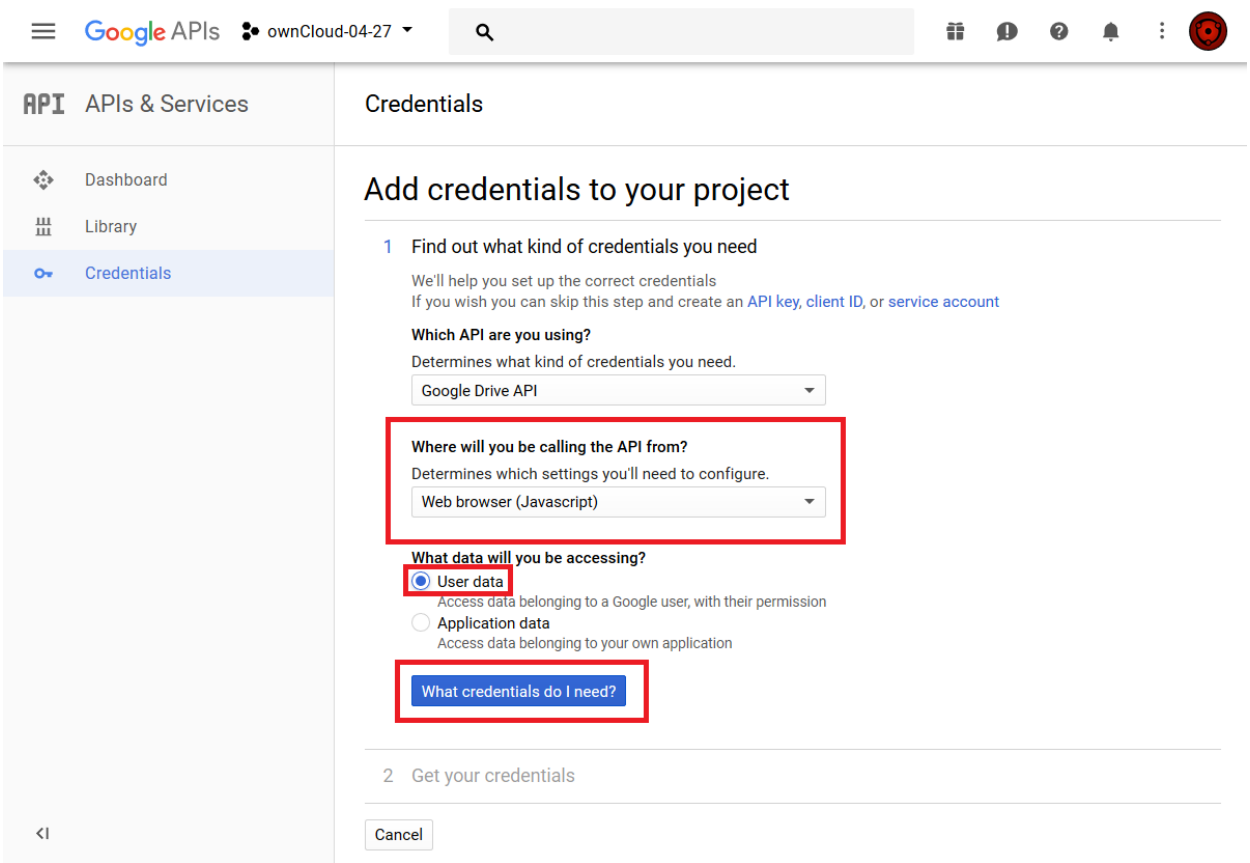




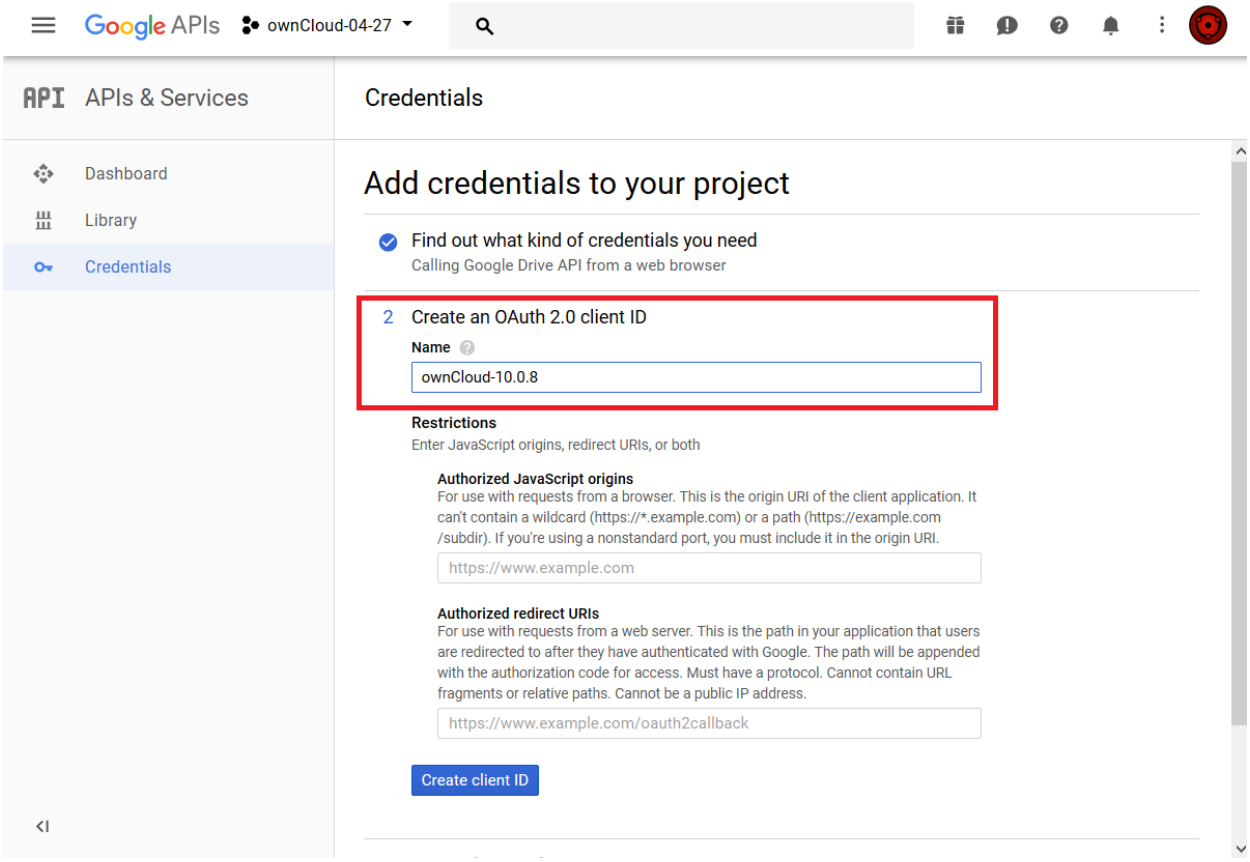
Now you must create your credentials.



First, select btn:[Web Browser] and btn:[User data].



The next screen that opens is **Create OAuth 2.0 Client ID**. Enter your app name.



Authorized JavaScript Origins is your root domain, for example https://example.com, without a trailing slash. Examples:

```
https://example.com
http://example.com
IP/owncloud
```

You need to configure **Authorized Redirect URIs**, and they must be in this form:

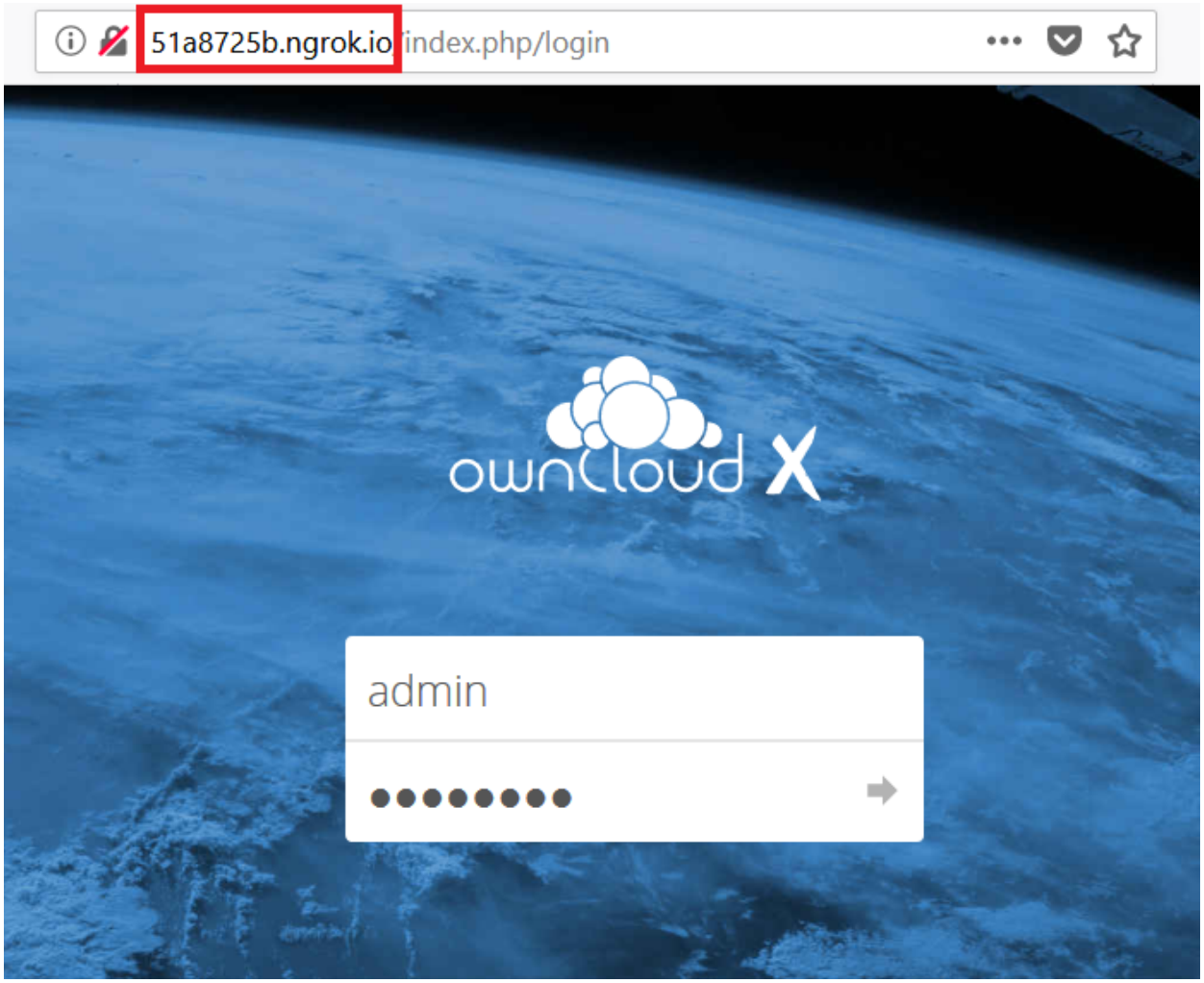
```
https://example.com/owncloud/index.php/settings/admin?sectionid=storage
https://example.com/owncloud/index.php/settings/personal?sectionid=storage
```

If you are configuring storage as an Administrator - choose the admin URI, if you are a user and configuring a storage - pick the personal URI.

If you are not sure what your exact URIs are - here is a quick guide to figure it out.

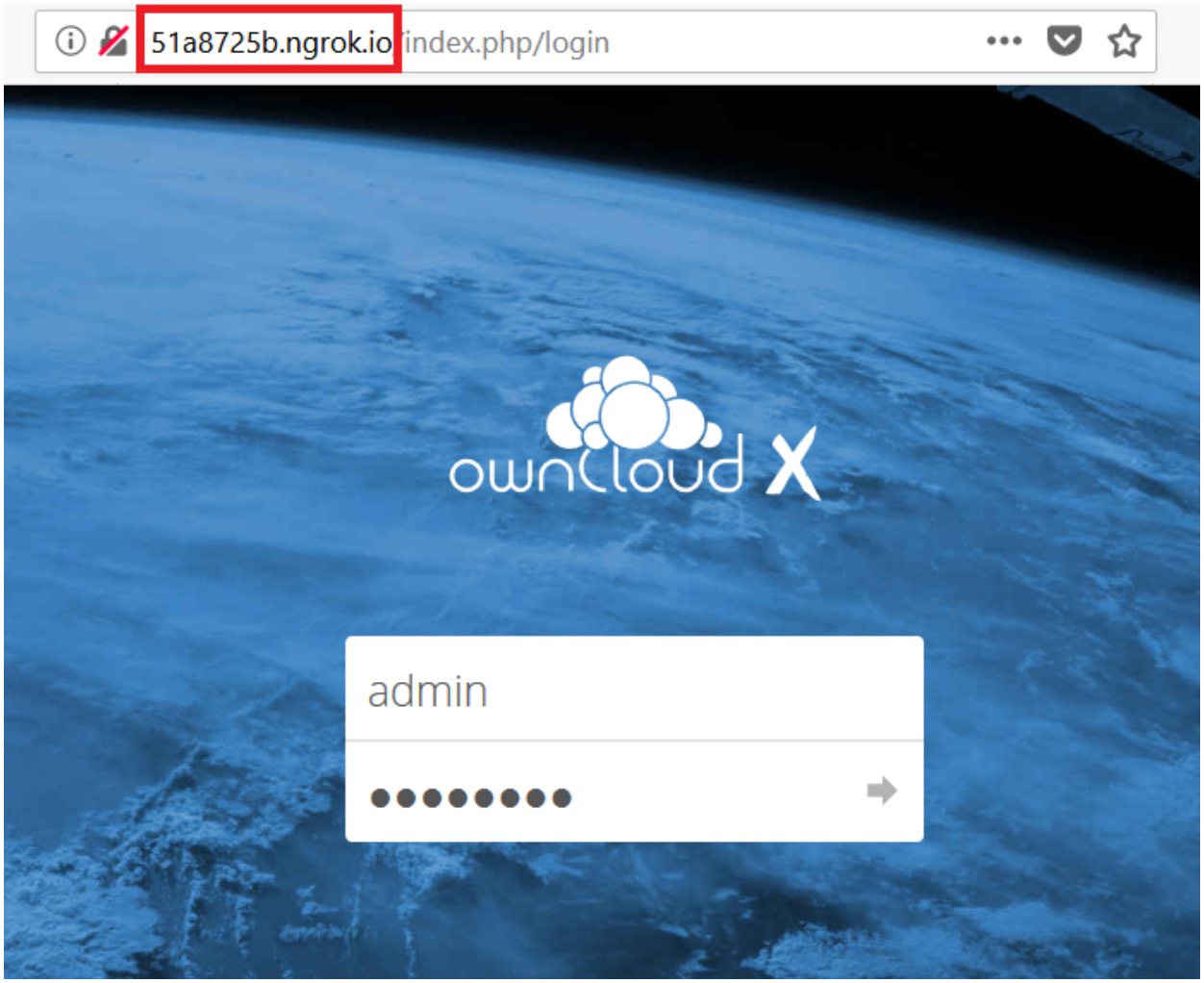
Authorized JavaScript Origins

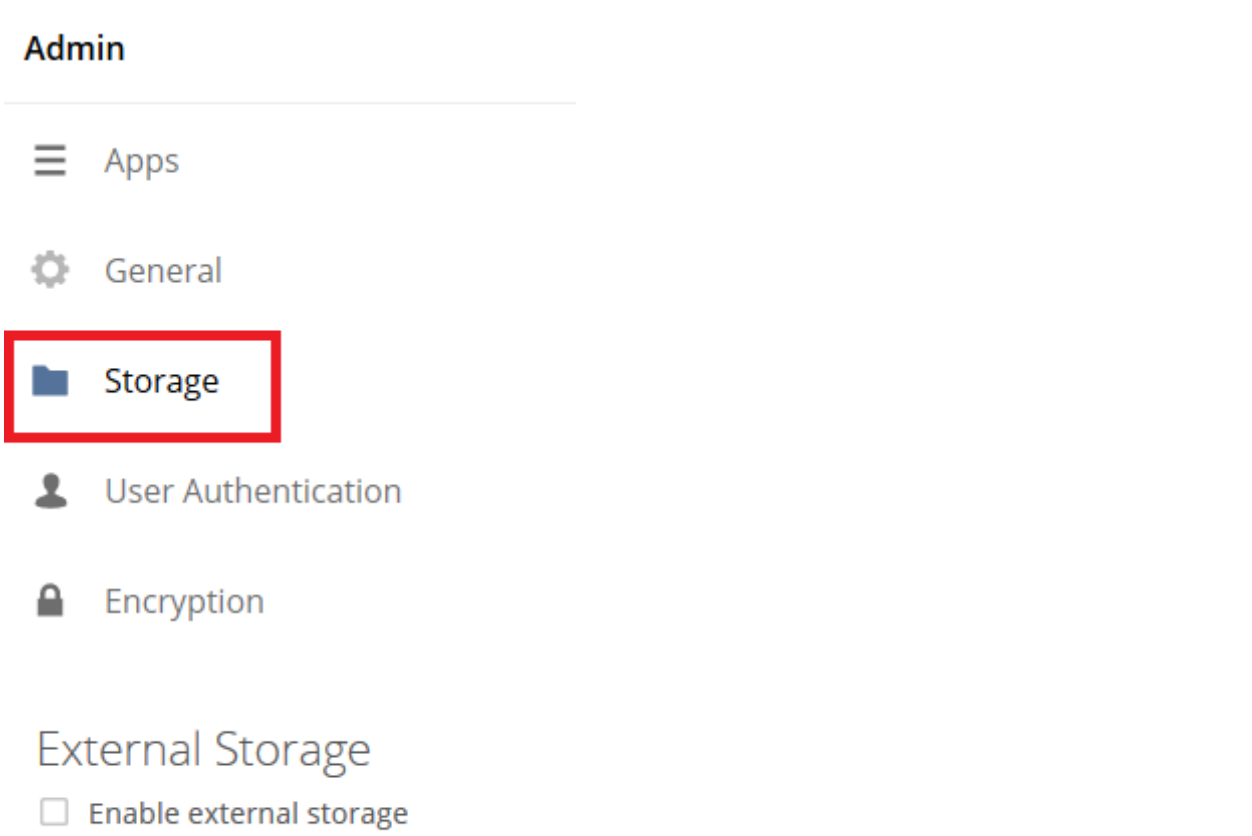
This is just the address you access your ownCloud server at, where you see the login screen.

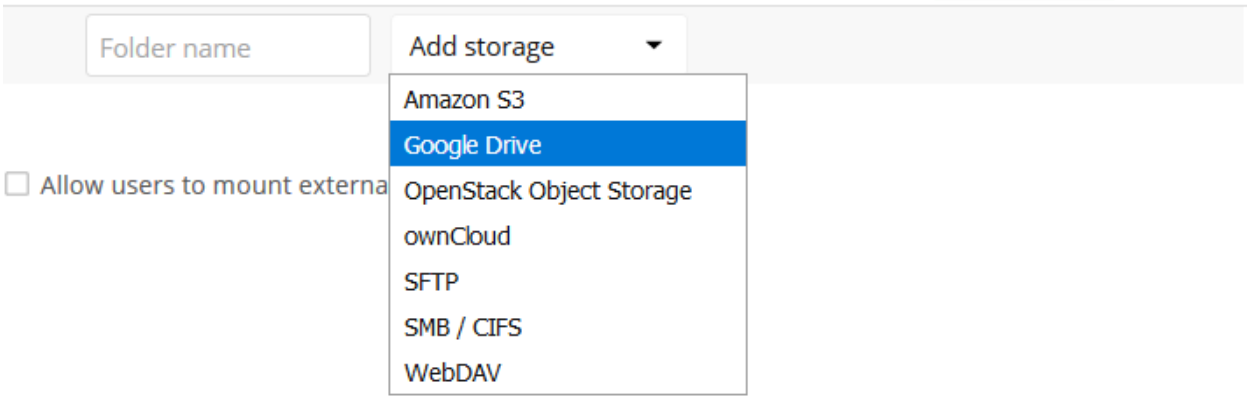


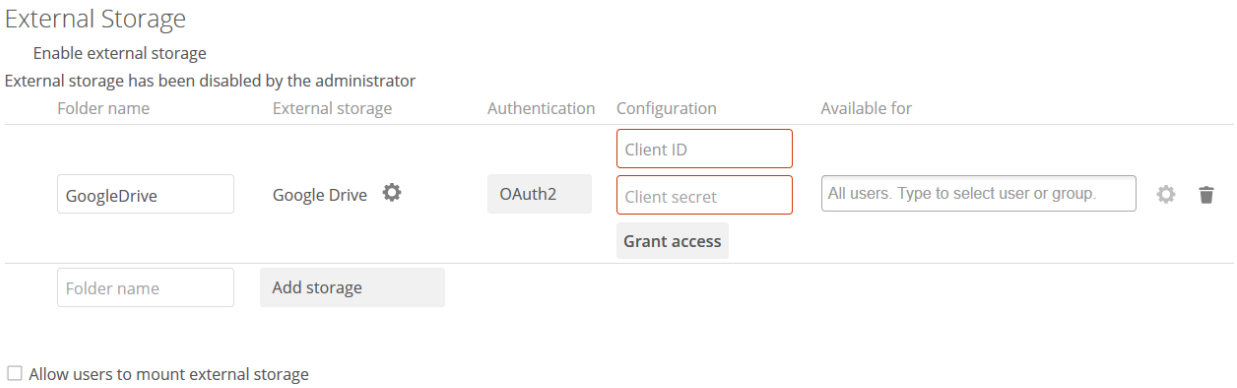
Authorized Redirect URIs

If you have not already enabled the Google Drive storage, here is how you do it:


51a8725b.ngrok.io/index.php/login
ownCloud X
admin

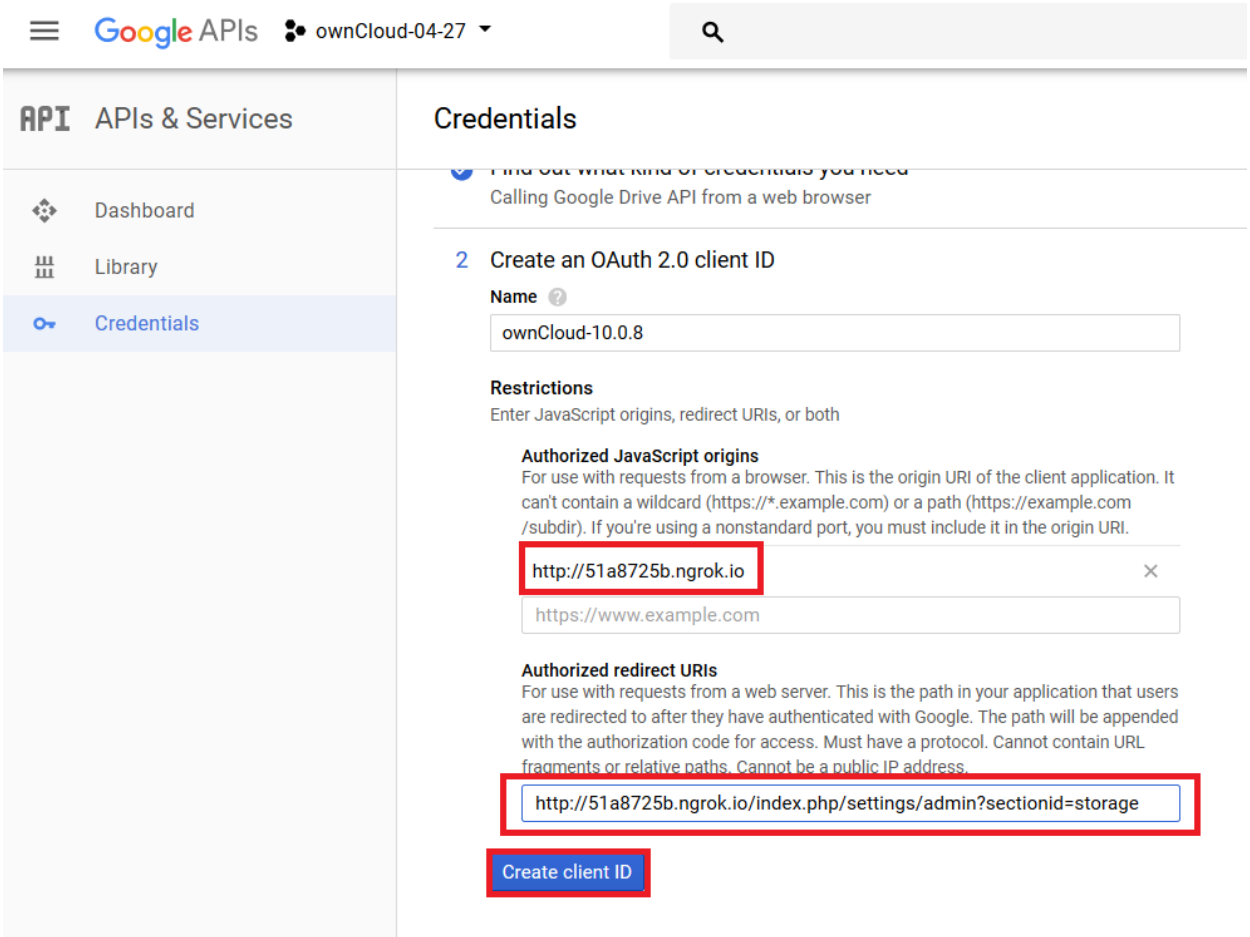

Admin
Apps
General
Storage
User Authentication
Encryption
External Storage
Enable external storage



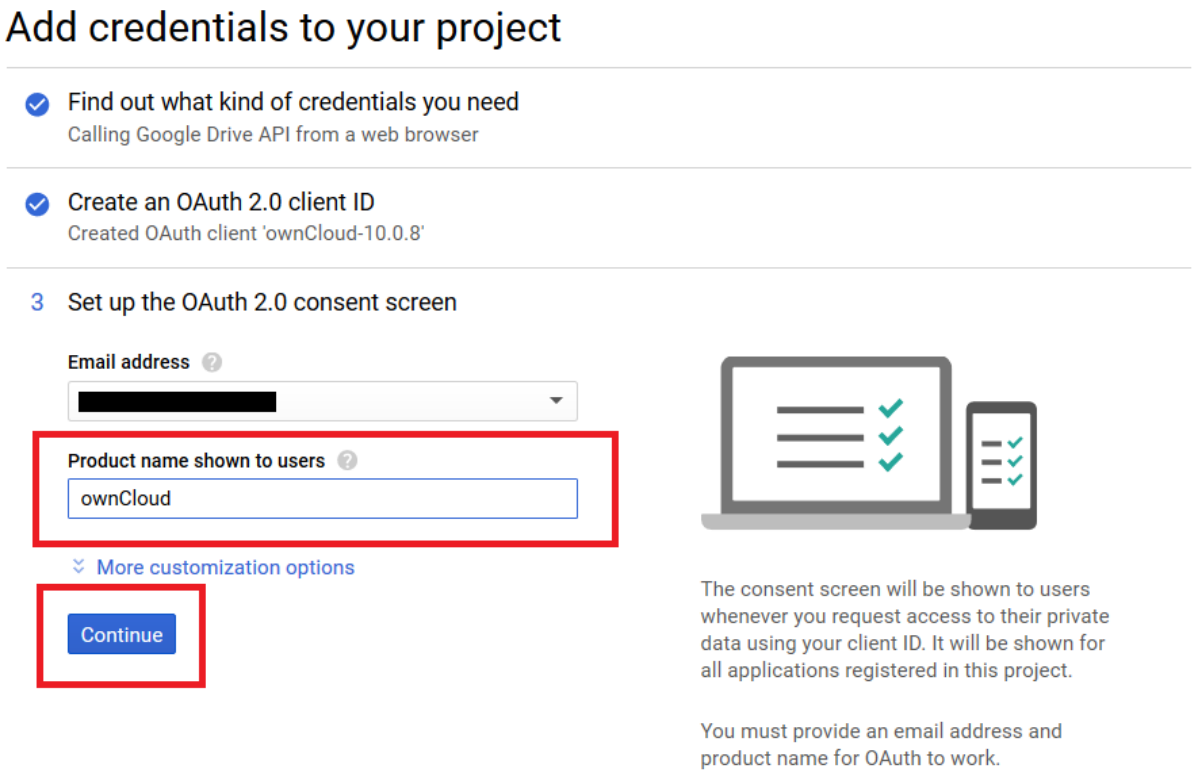


51a8725b.ngrok.io/index.php/settings/admin?sectionid=storage

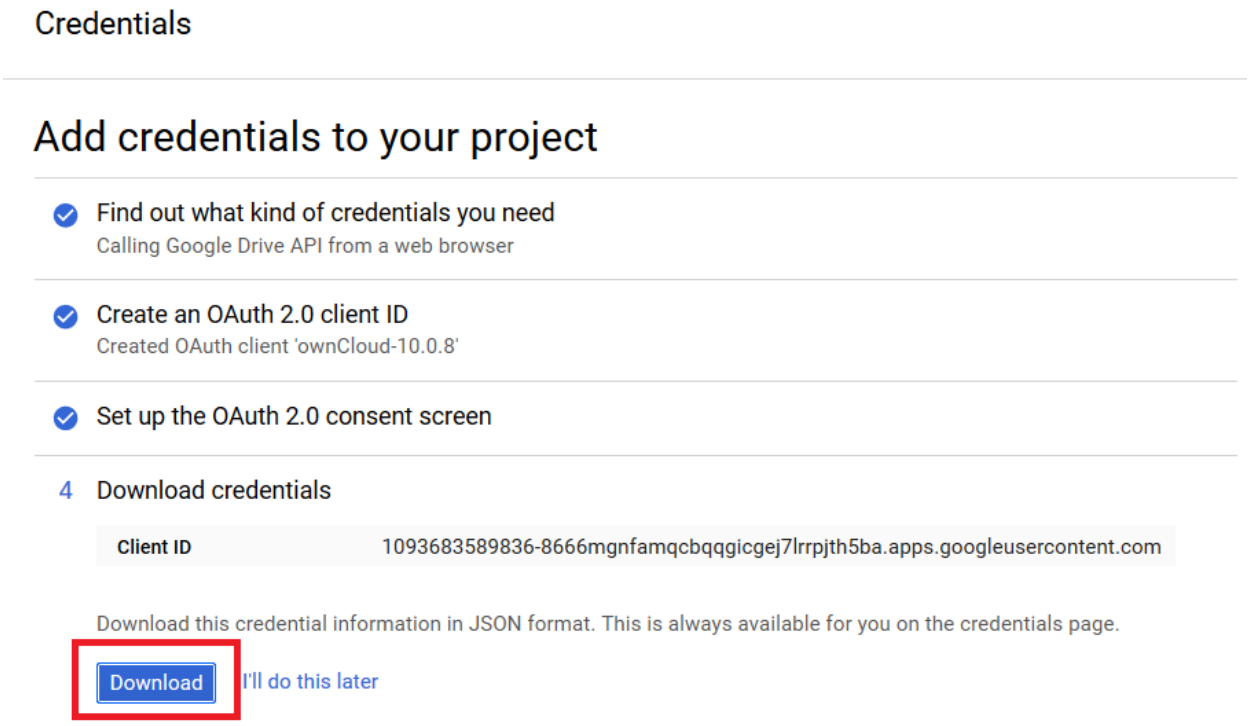
Here is the correct result:



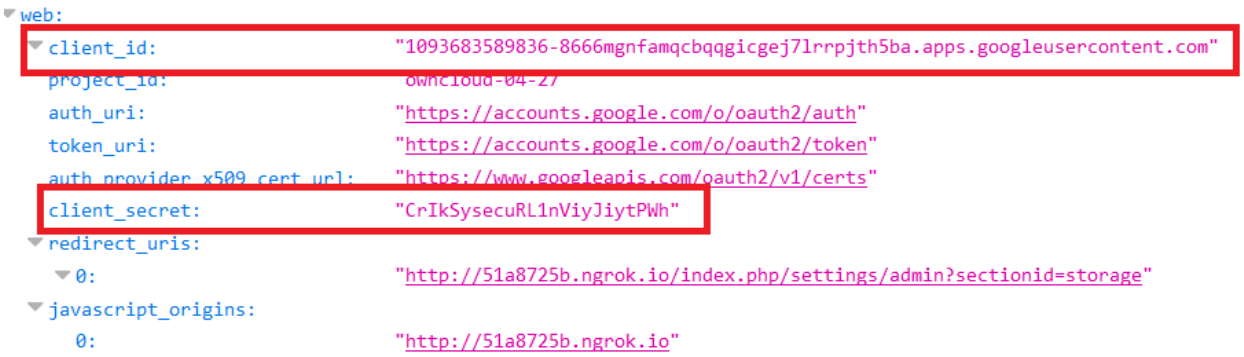
Now we have to create a consent screen. This is the information in the screen Google shows you when you connect your new Google app to ownCloud the first time.



Now you can download the credentials as a JSON file.



You can see either open this file with the editor of your choice (SublimeText for example), or you can put in in your web browser. This is when you do the later:

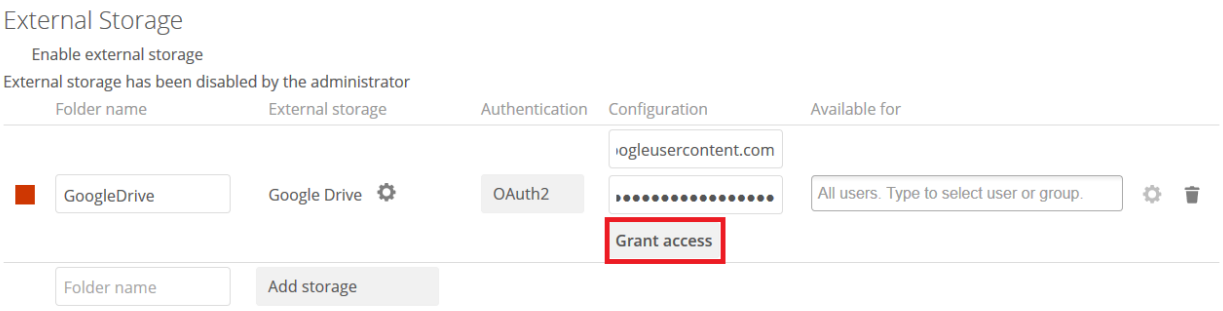


Enter the Client ID and Client Secret in the app and click btn:[Grant Access].

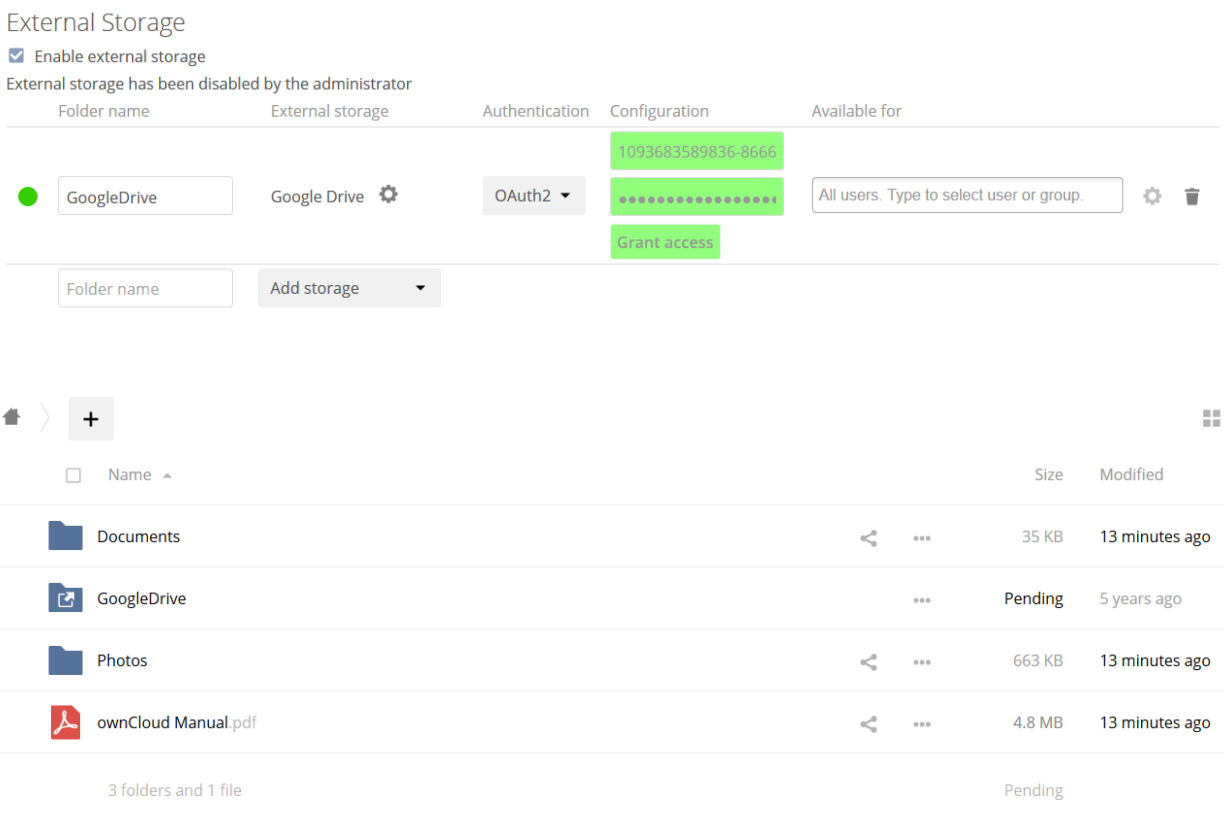
Now you have everything you need to mount your Google Drive in ownCloud.

Your consent page appears when ownCloud makes a successful connection.

Click btn:[Allow].



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



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

Local

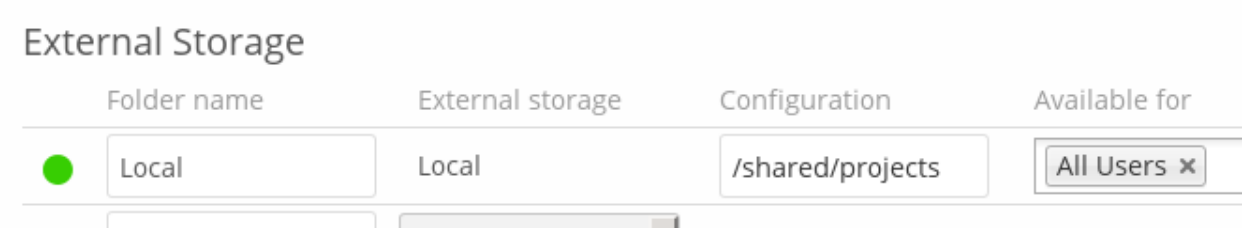
Local storage provides the ability to mount any directory on your ownCloud server that is:

- Outside of your ownCloud data/ directory
- Both readable and writable by your HTTP server user

Since this is a significant security risk, Local storage is only configurable via the ownCloud admin settings. Non-admin users cannot create Local storage mounts.

See Set Strong Directory Permissions for information on correct file permissions, and find your HTTP user PHP Version and Information.

To manage Local storage, navigate to admin, and then to Storage. You can see an example in the screenshot below.



In the **Folder name** field enter the folder name that you want to appear on your ownCloud Files page. In the **Configuration** field enter the full file path 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.

In addition to these steps, you have to ensure that Local storage is enabled in your ownCloud installation's config/config.php file. It should have the following configuration:

```
'files_external_allow_create_new_local' => 'true',
```

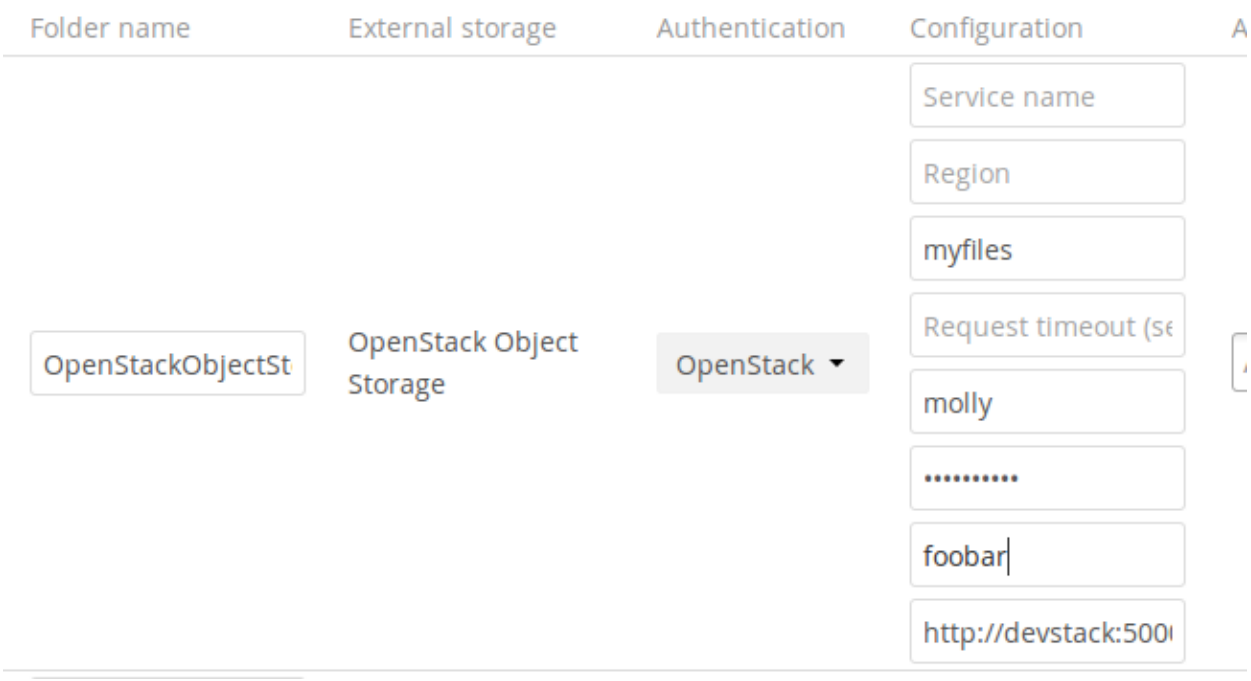
See Configuring External Storage (GUI) for additional mount options and information, and External Storage Authentication mechanisms for more information on authentication schemes.

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 ownCloud configuration needs:

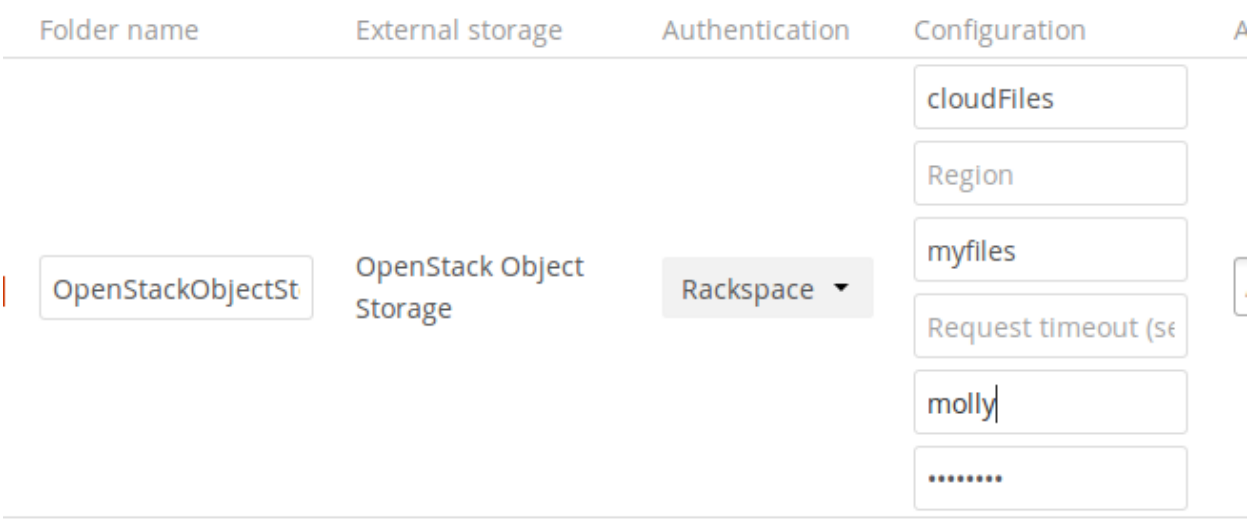
- **Bucket**. This is user-defined; think of it as a subdirectory of your total storage. The bucket will be created if it does not exist.
- **Username** of your account.
- **Password** of your account.
- **Tenant name** of your account. (A tenant is similar to a user group.)
- **Identity Endpoint URL**, the URL to log in to your OpenStack account.



The Rackspace authentication mechanism requires:

- **Bucket**
- **Username**
- **API key**.

You must also enter the term **cloudFiles** in the **Service name** field.



It may be necessary to specify a **Region**. Your region should be named in your account information, and you can read about Rackspace regions at About Regions.

The timeout of HTTP requests is set in the **Request timeout** field, in seconds.

See Configuring External Storage (GUI) for additional mount options and information, and External Storage Authentication mechanisms for more information on authentication schemes.

ownCloud

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

When filling in the **URL** field, use the path to the root of the ownCloud installation, rather than the path to the WebDAV endpoint. So, for a server at https://example.com/owncloud, use https://example.com/owncloud and not https://example.com/owncloud/remote.php/dav.

See ../external_storage_configuration_gui for additional mount options and information.

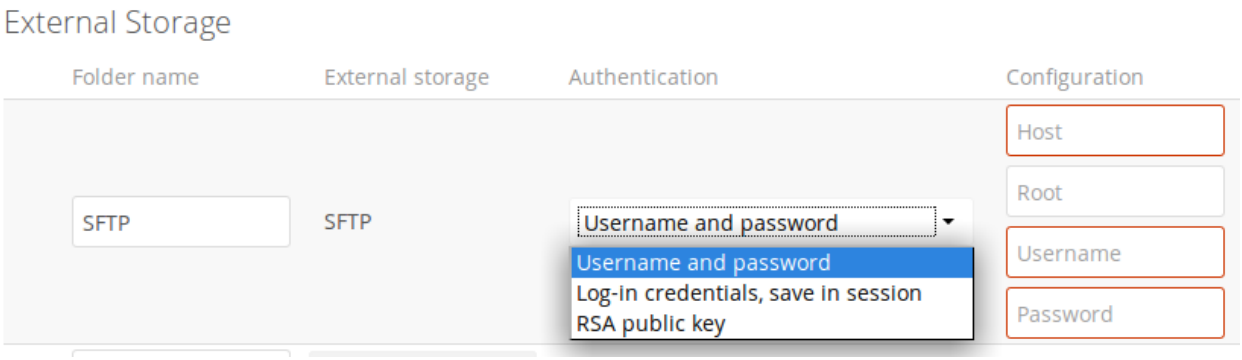
See auth_mechanisms for more information on authentication schemes.

SFTP

ownCloud's SFTP (FTP over an SSH tunnel) backend supports both password and public key authentication.

The **Host** field is required; a port can be specified as part of the **Host** field in the following format: hostname.domain:port. The default port is 22 (SSH).

For public key authentication, you can generate a public/private key pair from your **SFTP with secret key login** configuration.



After generating your keys, you need to copy your new public key to the destination server to .ssh/authorized_keys. ownCloud 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_configuration_gui for additional mount options and information.

See auth_mechanisms for more information on authentication schemes.

Samba File Server Configuration (SMB/CIFS)

ownCloud can connect to Windows file servers, and other SMB-compatible servers (e.g., Samba), by using the SMB/CIFS backend.

Dependencies

There are two dependencies to connect to your ownCloud installation to Windows file servers,

1. libsmbclient-php or smbclient
2. The Samba Client

libsmbclient-php or smbclient

ownCloud's SMB/CIFS backend requires either the libsmbclient-php module (version 0.8.0+) or the smbclient command (and its dependencies) to be installed on the ownCloud server. We highly recommend libsmbclient-php, but it isn't required. If it is installed smbclient won't be needed. Most Linux distributions provide libsmbclient-php and typically name it php-smbclient.

The Samba Client

The Samba client must be installed on your Linux system. It is included in all Linux distributions; on *Debian*, *Ubuntu*, and other Debian derivatives this is smbclient. On *SUSE*, *Red Hat*, *CentOS*, and other Red Hat derivatives it is samba-client. You also need which and stdbuf, which should be included in most Linux distributions.

Configuration

When configuring it, you will need the following information:

- The folder name, which will be your local mount point.
- The URL of the Samba server.
- The username or domain/username used to login to the Samba server.
- The password to login to the Samba server.
- The share name to mount on the remote Samba server.
- The remote subfolder inside the remote Samba share to mount. This is optional, as it defaults to /.

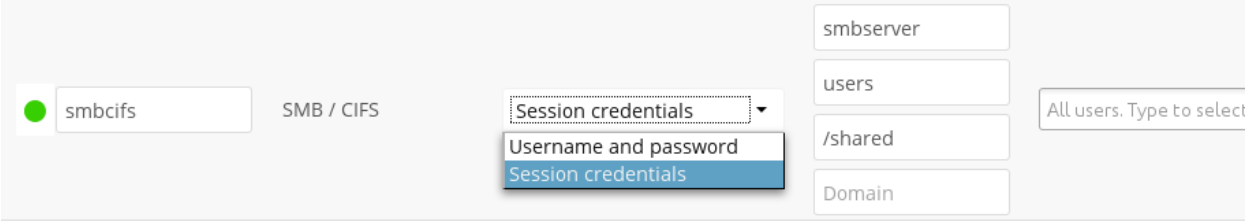


To assign the ownCloud logon username automatically to the subfolder, use $user instead of a subfolder name.

- The ownCloud 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 Active Directory (AD), or when using session credentials where the username cannot be modified. This is concatenated with the username, so the backend gets domain\username



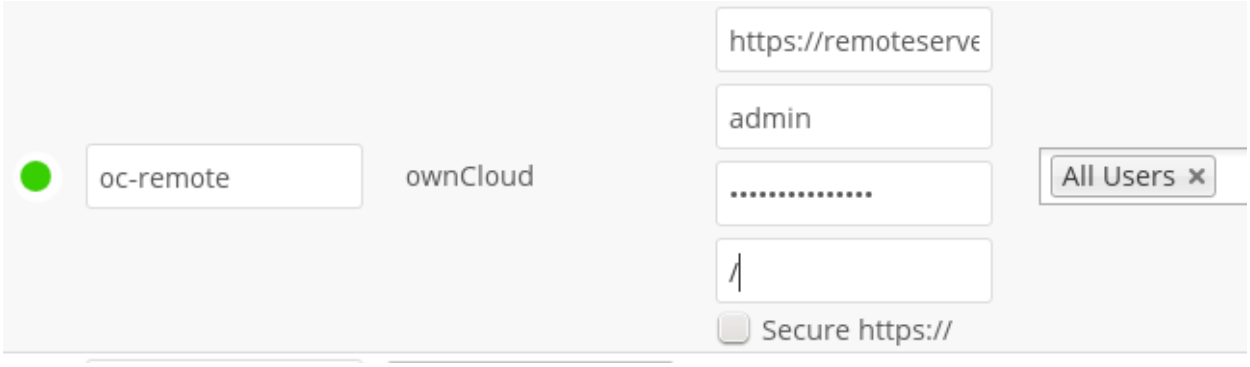
Further Information

- The External Storage Configuration GUI for additional mount options and information.
- External Storage Authentication Mechanisms for more information on authentication schemes.

WebDAV

Use this backend to mount a directory from any WebDAV server, or another ownCloud

server.



You need the following information:

- The name of your local mountpoint. Optionally, a Remote Subfolder can be specified to change the destination directory. The default is to use the whole root.
- The URL of the WebDAV or ownCloud server.
- The username and password for the remote server.

	We always recommend https:// for security reasons, so encourage you to enable btn:[Secure https://].

	CPanel users should install Web Disk to enable WebDAV functionality.

Further Reading

- See Configuring External Storage (GUI) for additional mount options and information.
- See External Storage Authentication Mechanisms for more information on authentication schemes.

Files

This section contains all of the file-related configuration documentation. It includes such topics as:

- Default Files Configuration
- Federated Cloud Sharing Configuration.

Big File Upload Configuration

System Configuration

- Make sure that the latest version of PHP, supported by ownCloud, 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. For example, if the average upload file size is **4GB** and the average number of users uploading at the same time is **25**, then you'll need 200GB of temp space, as the formula below shows.

  ```
  2 x 4 GB x 25 users = 200 GB required temp space
  ```

  Twice as much space is required because the file chunks will be put together into a new file before it is finally moved into the user's folder.



In Centos and RHEL, Apache has a few more default configurations within systemd. You will have to set the temp directory in two places:

1. In php.ini, e.g., sys_temp_dir = "/scratch/tmp"
2. In /usr/lib/systemd/system/httpd.service:

```
PrivateTmp=false
```

Configuring Your Web Server



ownCloud comes with its own owncloud/.htaccess file. Because php-fpm can't read PHP settings in .htaccess these settings must be set in the owncloud/.user.ini file.

Set the following two parameters inside the corresponding php.ini file (see the **Loaded Configuration File** section of PHP Version and Information to find your relevant php.ini files):

```
php_value upload_max_filesize = 16G
php_value post_max_size = 16G
```

Adjust these values for your needs. If you see PHP timeouts in your logfiles, increase the timeout values, which are in seconds, as in the example below:

```
php_value max_input_time 3600
php_value max_execution_time 3600
```

mod_reqtimeout

1707-update-big-file-upload-docs 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.

Disable mod_reqtimeout On Ubuntu

On Ubuntu, you can disable the module by running the following command:

```
a2dismod reqtimeout
```

Disable mod_reqtimeout On CentOS

On CentOS, comment out the following line in /etc/httpd/conf/httpd.conf:

```
LoadModule reqtimeout_module modules/mod_reqtimeout.so
```

When you have done run asdismod or updated /etc/httpd/conf/httpd.conf, restart Apache.



There are also several other configuration options in your web server config which could prevent the upload of larger files. Please see your web server's manual, for how to configure those values correctly:

Apache

- LimitRequestBody
- SSLRenegBufferSize

Apache with mod_fcgid

- FcgidMaxRequestInMem
- FcgidMaxRequestLen



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 usual may no longer be necessary, once bug #51747 is fixed.

Configuring PHP

If you don't want to use the ownCloud .htaccess or .user.ini file, you may configure PHP instead. Make sure to comment out any lines .htaccess about upload size, if you entered any.



If you are running ownCloud 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, as in the following example:

```
upload_max_filesize = 16G
post_max_size = 16G
```

Tell PHP which temp file you want it to use:

```
upload_tmp_dir = /var/big_temp_file/
```

Output Buffering must be turned off in .htaccess or .user.ini or php.ini, or PHP will return memory-related errors:

```
output_buffering = 0
```

Configuring ownCloud

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.

If you have configured the session_lifetime setting in your config.php. See Sample Config PHP Parameters, to 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 entirely from your configuration to reset it to the default shown in the config.sample.php.

General Upload Issues

Various environmental factors could cause a restriction of the upload size. Examples are:

- The LVE Manager of CloudLinux which sets an I/O limit.
- Some services like Cloudflare are also known to cause uploading issues.
- Upload limits enforced by proxies used by your clients.
- Other webserver modules like described in General Troubleshooting.

Long-Running Uploads

For very long-running uploads (those lasting longer than 1 hr) to public folders, *when chunking is not in effect*, 'filelocking.ttl' should be set to a significantly large value. If not, large file uploads will fail with a file locking error, because the Redis garbage collection will delete the initially acquired file lock after 1 hour by default.

To estimate a good value, use the following formula:

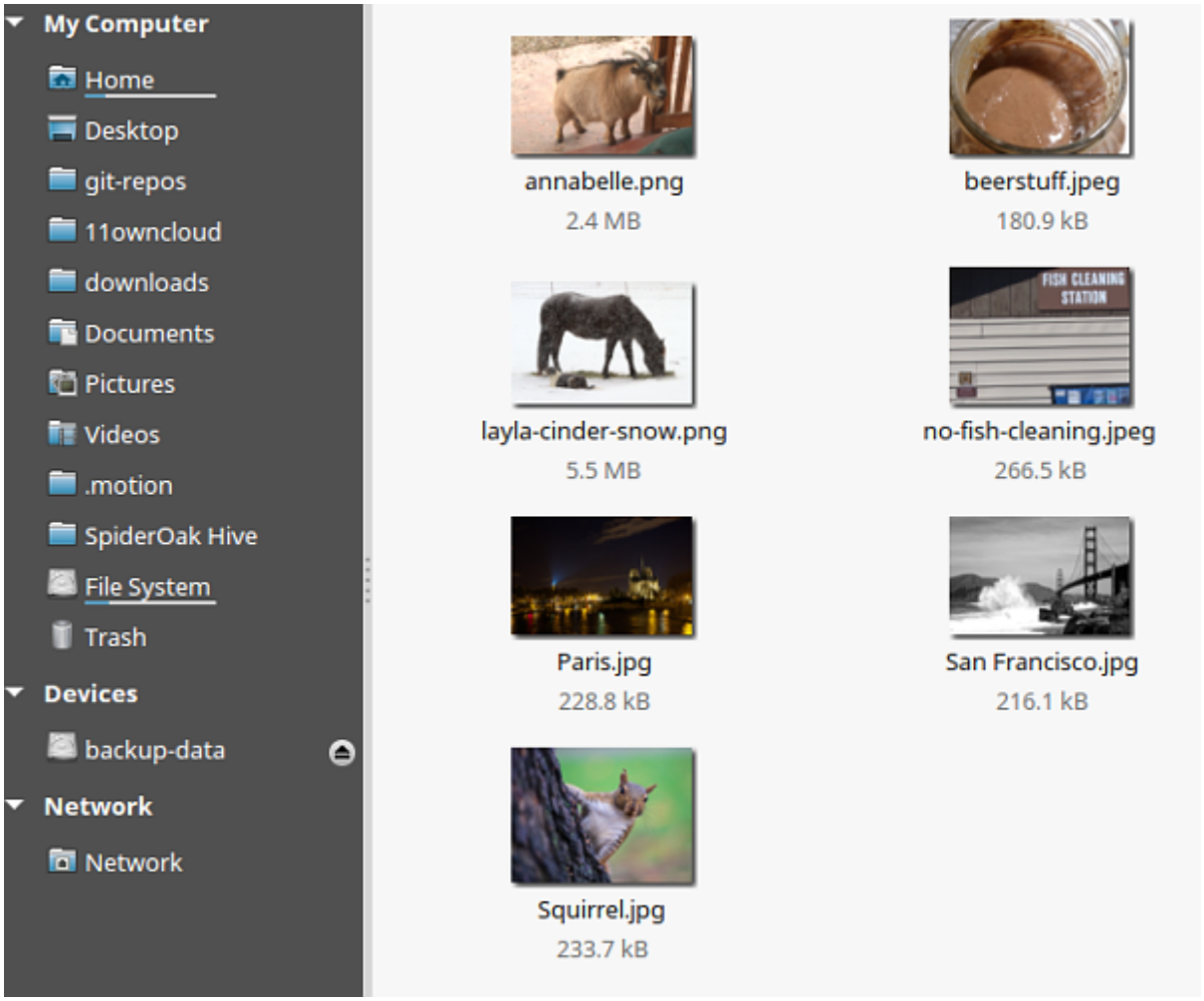
```
time in seconds = (maximum upload file size / slowest assumed upload connection).
```

For the value of "*slowest assumed upload connection*", take the **upload** speed of the user with the slowest connection and divide it by two. For example, let's assume that the user with the slowest connection has an 8MBit/s DSL connection; which usually indicates the download speed. This type of connection would, usually, have 1MBit/s upload speed (but confirm with the ISP). Divide this value in half, to have a buffer when there is network congestion, to arrive at 512KBit/s as the final value.

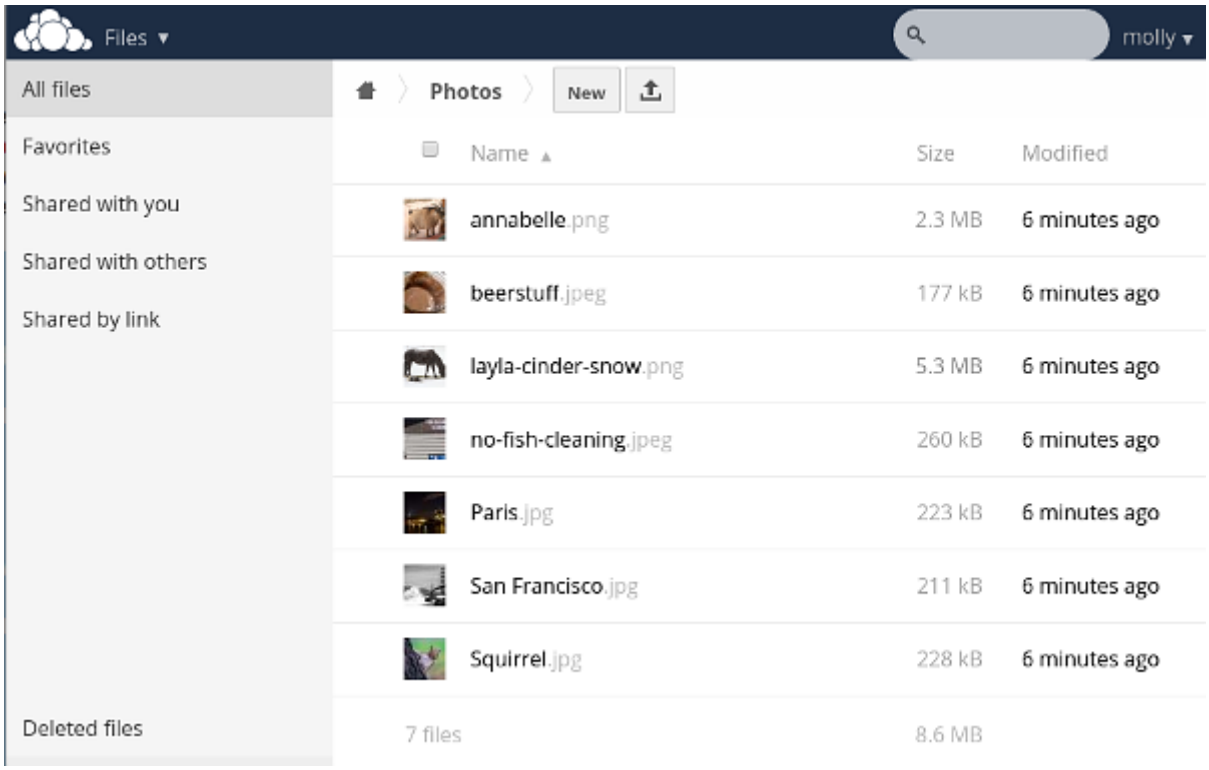
Providing Default Files

You may distribute a set of default files and folders to all users by placing them in the owncloud/core/skeleton directory on your ownCloud server. These files appear only to new users after their initial login, and existing users will not see files that are added to this directory after their first login. The files in the skeleton directory are copied into the users' data directories, so they may change and delete the files without affecting the originals.

This screenshot shows a set of photos in the skeleton directory.



They appear on the user's ownCloud Files page just like any other files.



Additional Configuration

The configuration option skeletondirectory available in your config.php allows you to configure the directory where the skeleton files are located.

These files will be copied to the data directory of new users.

Leave this directory empty if you do not want to copy any skeleton files.

The value of the skeletondirectory key **must not be empty** if you decide to use it in your config.php.



See Sample Config PHP Parameters for more the complete list of config.php options.

Configuring External Storage (GUI)

Introduction

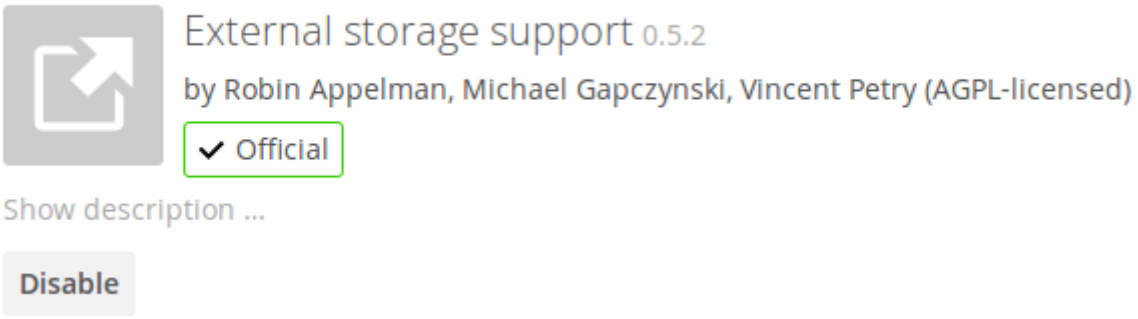
The External Storage Support application enables you to mount external storage services and devices as secondary ownCloud storage devices. You may also allow users to mount their own external storage services.

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

Also new in 9.0 is an option for the ownCloud admin to enable or disable sharing on individual external mountpoints. Sharing on such mountpoints is disabled by default.

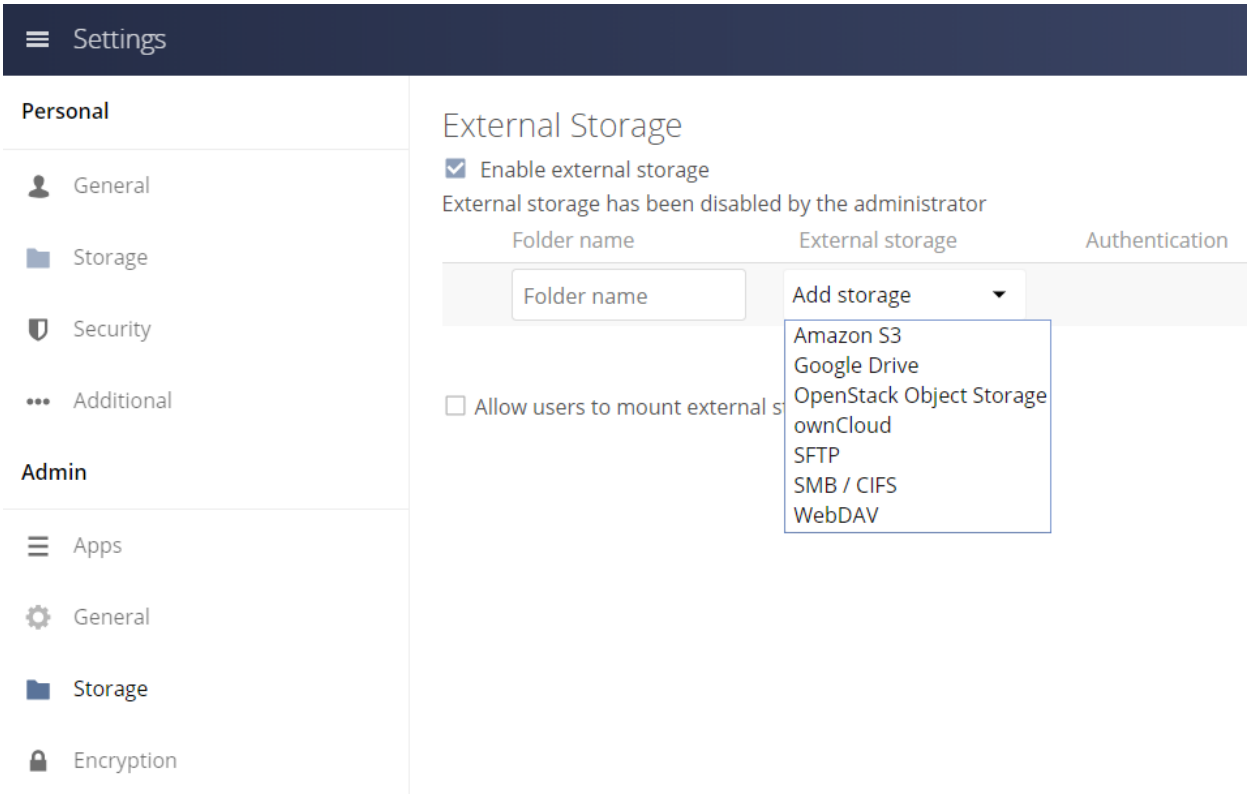
Enabling External Storage Support

Tick the checkbox under Settings > Storage > "Enable External Storage".



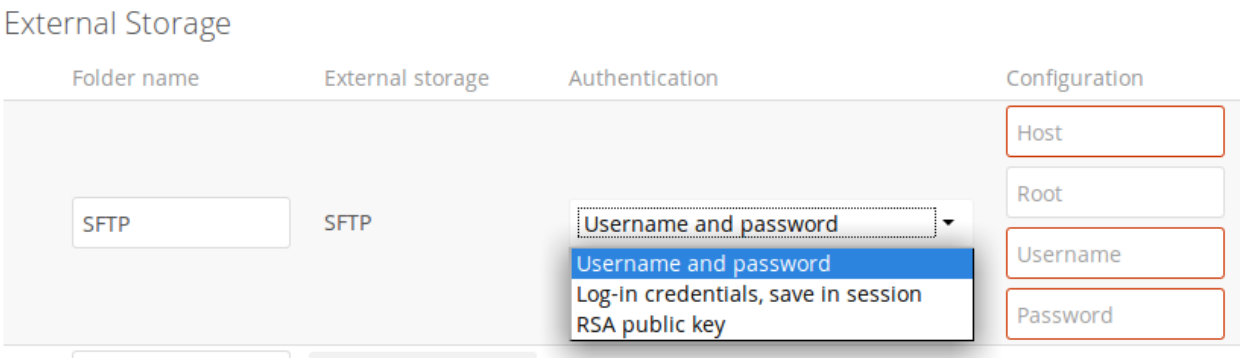
Storage Configuration

To create a new external storage mount, select an available backend from the dropdown **Add storage**. Each backend has different required options, which are configured in the configuration fields.



Each backend may also accept multiple authentication methods. These are selected with the dropdown under **Authentication**. Different backends support different authentication mechanisms; some specific to the backend, others are more generic. See external_storage/auth_mechanisms for more detailed information.

When you select an authentication mechanism, the configuration fields change as appropriate for the mechanism. The SFTP backend, for one example, supports **username and password**, **Log-in credentials, save in session**, and **RSA public key**.

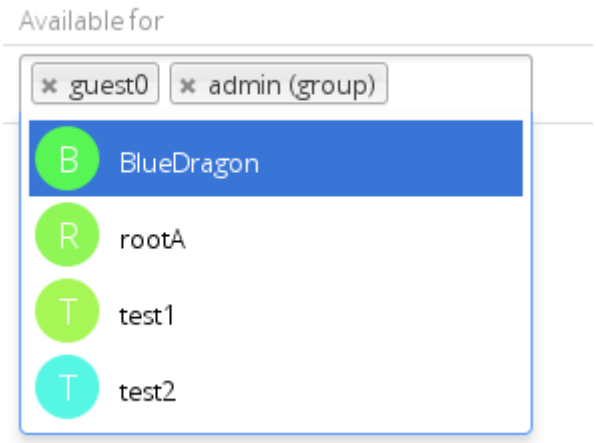


Required fields are marked with a red border. When all required fields are filled, the storage is automatically saved. A green dot next to the storage row indicates the storage is ready for use. A red or yellow icon indicates that ownCloud could not connect to the external storage, so you need to re-check 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 btn:[colored icon] or reload your Admin page.

User and Group Permissions

A storage configured in a user's Personal settings is available only to the user that created it. A storage configured in the Admin settings is available to all users by default, and it can be restricted to specific users and groups in the **Available for** field.



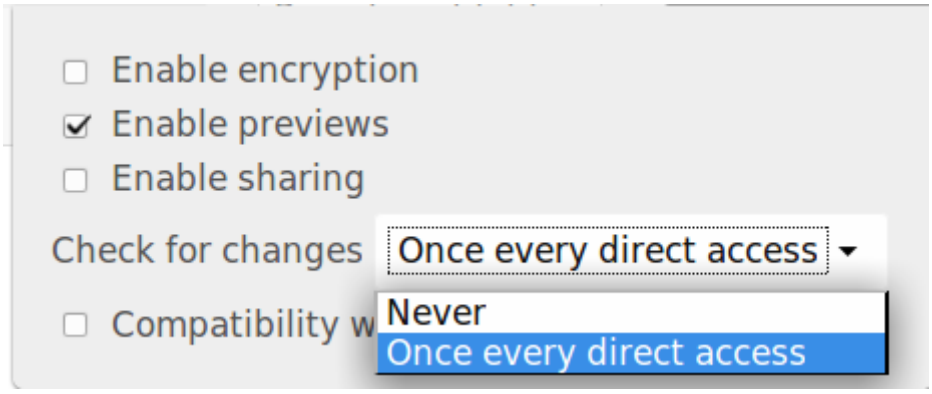
Mount Options

Hover your cursor to the right of any storage configuration to expose the settings button and trashcan. Click the trashcan to delete the mountpoint. The settings button allows you to configure each storage mount individually with the following options:

- Encryption
- Previews
- Enable Sharing
- Filesystem check frequency (Never, Once per direct access)

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

Enable Sharing allows the ownCloud admin to enable or disable sharing on individual mountpoints. When sharing is disabled the shares are retained internally, so that you can re-enable sharing and the previous shares become available again. Sharing is disabled by default.



Using Self-Signed Certificates

When using self-signed certificates for external storage mounts the certificate must be imported into ownCloud.



Please refer to Importing System-wide and Personal SSL Certificates for more information.

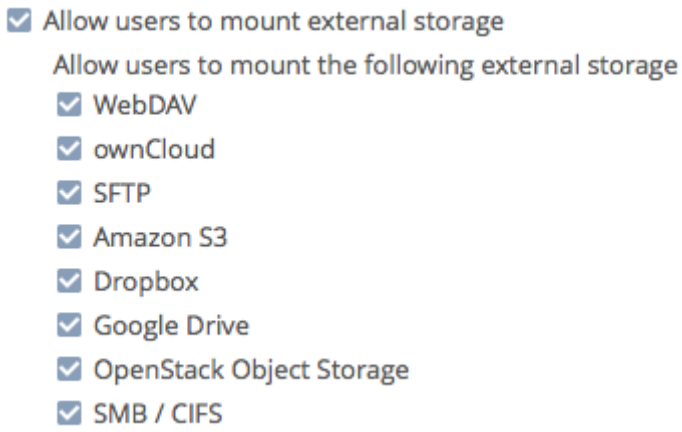
Available storage backends

The following backends are provided by the external storages app. Other apps may provide their own backends, which are not listed here.

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

Allow Users to Mount External Storage

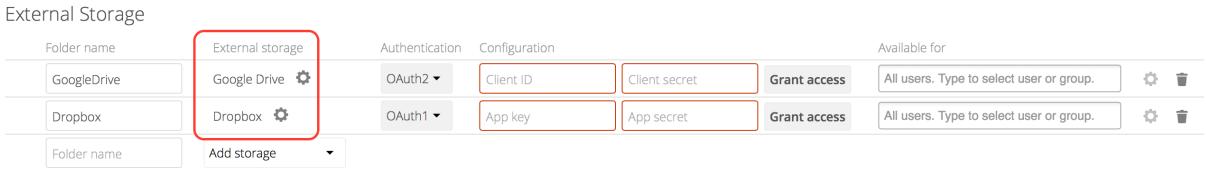
Check "*Allow users to mount external storage*" to allow your users to mount storages on external services. Then enable the backends you want to allow.



Be careful with the choices that you enable, as it allows a user to make potentially arbitrary connections to other services on your network!

Setting Up Google Drive and Dropbox Connections

When an external storage is created which uses either Google Drive or Dropbox, a link to the respective configuration page is available, next to the service name.



In the screenshot above, you can see that two external storage connections have been created, but not configured. One goes to Google Drive, the other to Dropbox. If you click the btn:[cog icon] next to the name of either, the respective app configuration page will open in a new tab, or a new window. From there, you can manage the configuration and obtain the respective credentials needed for configuring the connection.

Detecting Files Added to External Storages

We recommend configuring the background job Webcron or Cron to enable ownCloud to automatically detect files added to your external storages.

You cannot scan/detect changed files on external storage mounts when you select the **Log-in credentials, save in session** authentication mechanism. However, there is a workaround, and that is to use Ajax cron mode. See Password-based Mechanisms for more information.

ownCloud may not always be able to find out what has been changed remotely (files changed without going through ownCloud), especially when it's very deep in the folder hierarchy of the external storage.

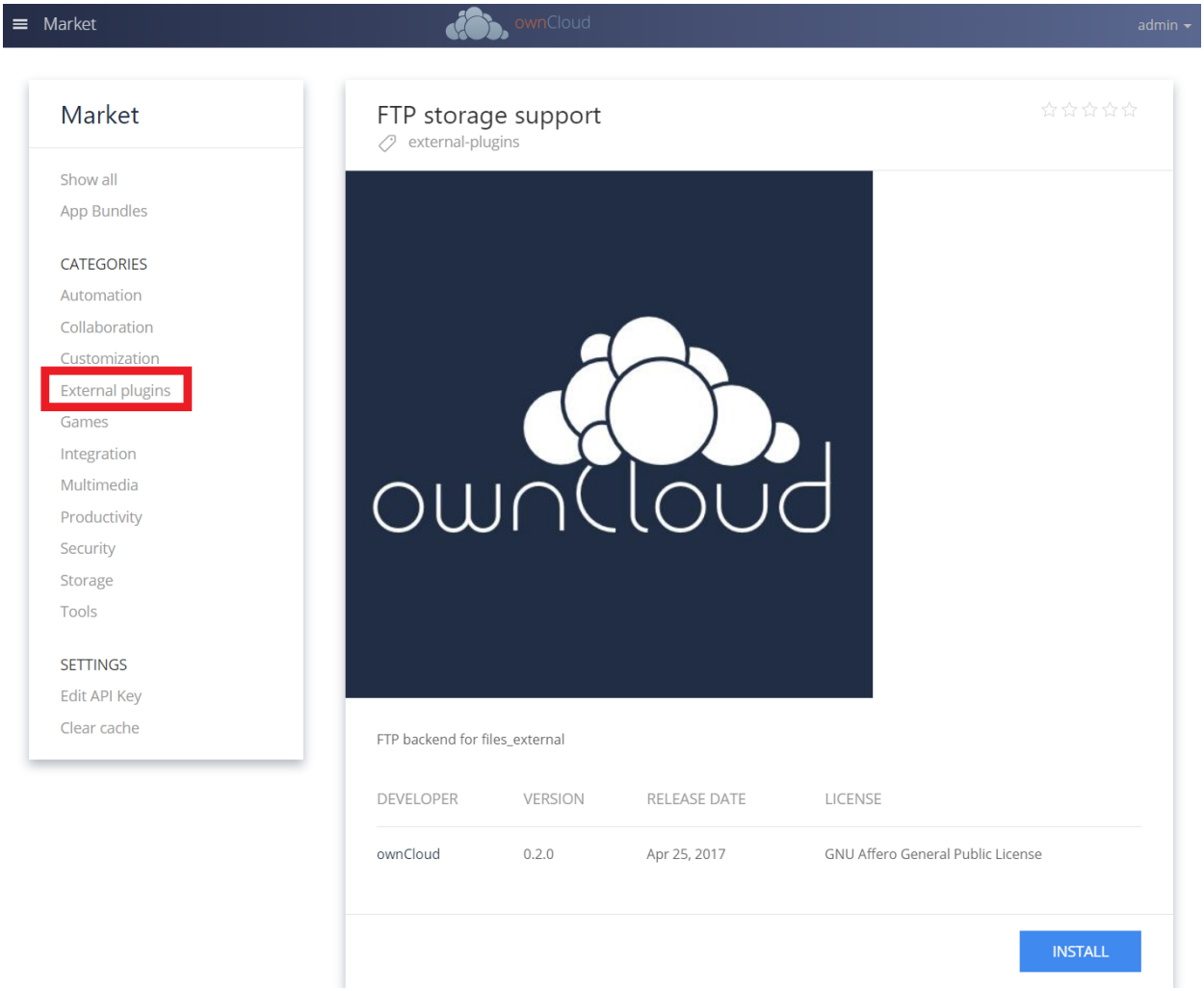
You might need to setup a cron job that runs {occ-command-example-prefix} files:scan --all. Alternatively, replace –all with the user name to trigger a rescan of the user's files periodically, for example every 15 minutes, which includes the mounted external storage.



See the occ's file operations for more information.

FTP

If you want to mount an FTP Storage, please install `the FTP Storage Support app`_ from the ownCloud market.



Configuring Federation Sharing

Introduction

Federated Cloud Sharing is managed by the Federation app. When you enable the Federation app you can easily and securely link file shares between ownCloud servers, in effect creating a "cloud" of ownCloud installations.



For security reasons federated sharing **strictly requires HTTPS (SSL/TLS)**.



We strongly recommend using HTTP for development and testing purposes. However, to do so, you have to set 'sharing.federation.allowHttpFallback' ⇒ true, in **config/config.php**.

Configuration

Follow these steps to establish a trusted connection between two servers.

1. Verify that both servers have SSL certificates. If you open the server URL in your browser and see a lock icon on the left-hand side of the address bar, the certificate is valid.

   Lock icon in the address bars in Firefox, Google Chrome, and Safari.

2. Verify that the 'overwrite.cli.url' ⇒ 'https://<SERVER_URL>' setting is configured to the correct URL, instead of `localhost, in **config.php**.
3. Reset the federation job in your oc_jobs table. This job is required to get the verification token from the other server to establish a federation connection between two servers. The resetting ensures that it will be executed when we run system:cron later.

   ```
   mysql -u root -e "update oc_jobs set last_run=0 where
   class='OCA\\Federation\\SyncJob';" owncloud;
   mysql -u root -e "update oc_jobs set last_checked=0 where
   class='OCA\\Federation\\SyncJob';" owncloud;
   ```

4. Navigate to **admin settings → sharing → Federation**
5. Add **server 1** to the trusted servers on **server 2**.
6. Add **server 2** to the trusted servers on **server 1**.
7. Now run the cron job in your ownCloud directory (for example /var/www/owncloud/).

   ```
   {occ-command-example-prefix} system cron
   ```

8. Now the check should be green
9. Sync now your users with

   ```
   occ dav:sync-system-addressbook
   occ federation:sync-addressbook
   ```

10. Configure automatic acceptance of new federated shares.

    ```
    occ config:app:set federation auto_accept_trusted --value '0'
    occ config:app:set federatedfilesharing auto_accept_trusted --value 'yes'
    ```

Working With Proxies

There are ownCloud instances that are not connected to the internet. They have no possibility to reach the public network. Therefore Federation will not work without a proxy.

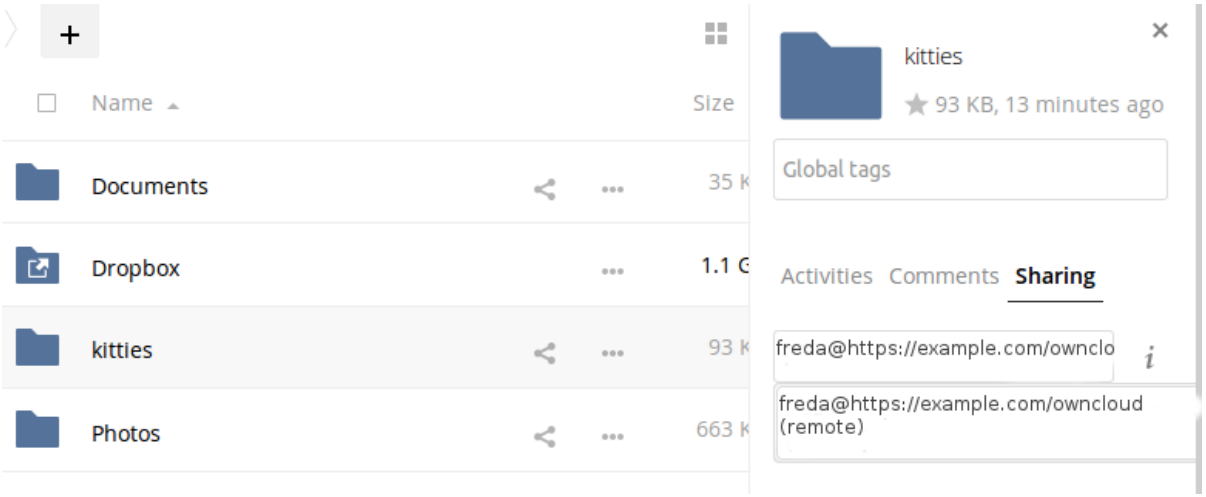
To set the proxy and proxyuserpwd configuration variables, in config/config.php. proxy sets the proxy's hostname, and proxyuserpwd sets the username and password

credentials, in username:password format.

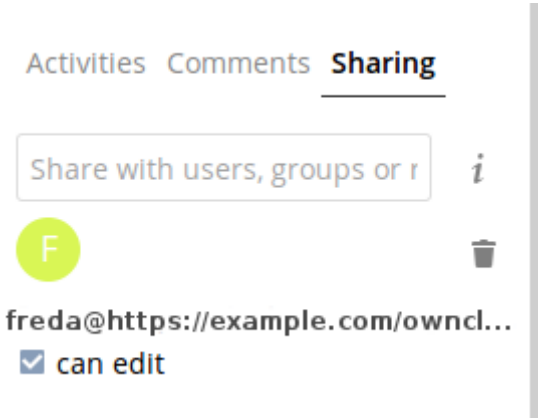
Creating a New Federation Share

Follow these steps to create a new Federation share between two ownCloud 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. Then, create a federated share by entering username@serveraddress in the sharing dialog (for example freda@https://example.com/owncloud). When ownCloud verifies the link, it displays it with the **(remote)** label. Click on this label to establish the link.



3. 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 btn:[trash can] icon.

Known Issues

Persistent Locks Are Not Guaranteed

There is a known bug propagated persistent locks to federated instances. If a user creates an exclusive lock on a share, no other users should be able to modify it, nor its contents, and all users should see a lock icon on the share.

However, this isn't the case. The following functionality has been recorded:

- The user who created the lock sees the lock icon throughout the share.
- The top-level of the share for receivers shows the lock icon.

- Sub-items of the share **do not show the lock icon**.
- The share and its contents **can still be modified by all users**; specifically:
  - Sub-items **can be deleted**.
  - Sub-items **can be created**.

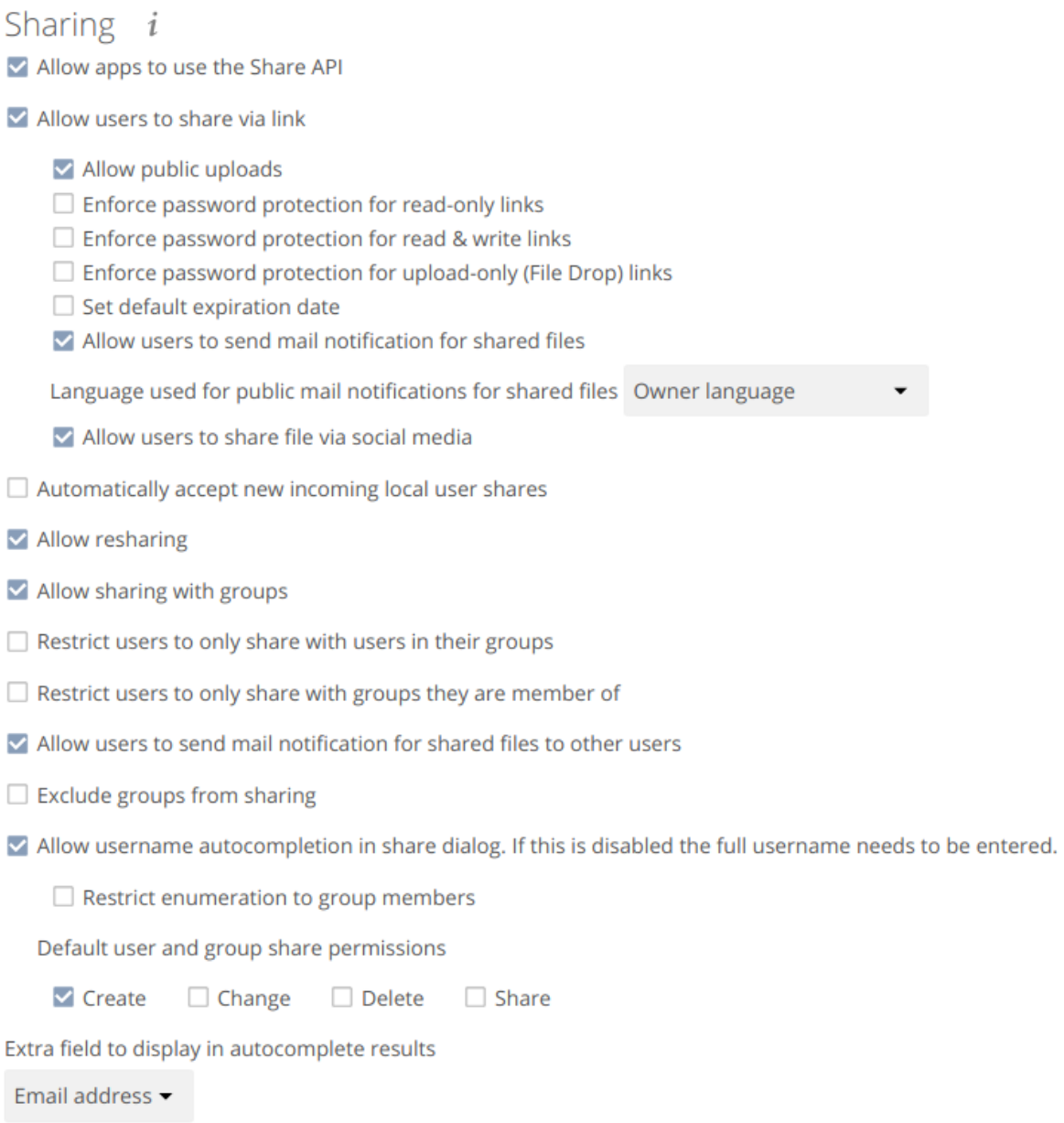
File Sharing

Introduction

The sharing policy is configured on the Admin page in the **"Sharing"** section.



If you don’t see the sharing section, try disabling your AdBlock browser plugin.
It might also be related to another installed ad blocker in your browser.
If so, please disable the plugin and see if that resolves the situation.



From this section, ownCloud users can:

- Share files with their ownCloud groups and other users on the same ownCloud server

- Share files with ownCloud users on other ownCloud servers, for more details see Federated Cloud Sharing Configuration.
- Create public link shares for people who are not ownCloud users.

You have control of a number of user permissions on file shares:

- Allow users to share files
- Allow users to create public link shares
  - Allow public uploads to public link shares
  - Enforce password protection on public link shares
  - Set default expiration date on public link shares
  - Allow users to send mail notification for shared files
  - Set the language used for public mail notification for shared files
  - Allow users to share file via social media
- Automatically accept new incoming local user shares
- Allow resharing
- Default user and group share permissions
  - Restrict users to only share with users in their groups
  - Restrict users to only share with groups they are a member of
- Allow email notifications of new public link shares
- Exclude groups from creating shares
- Allow username autocompletion in share dialog
  - Restrict enumeration to group members
  - Default user and group share permissions
- Extra field to display in autocomplete results



ownCloud includes a Share Link Password Policy app.

Settings Explained

Allow apps to use the Share API

Check this option to enable users to share files. If this is not checked, no users can create file shares.

Allow users to share via link

Check this option to enable creating public link shares for people who are not ownCloud users via hyperlink.

Allow public uploads

Check this option to allow anyone to upload files to public link shares.

Enforce password protection

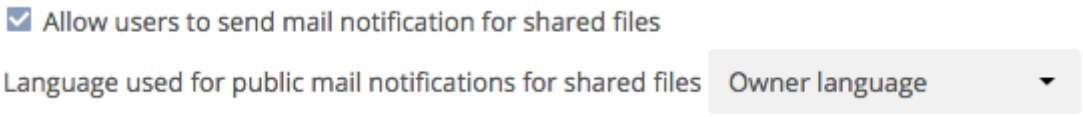
Check this option to force users to set a password on all public link shares. This does not apply to local user and group shares.

Set default expiration date

Check this option to set a default expiration date on public link shares.

Allow users to send mail notification for shared files

Check this option to enable sending notifications from ownCloud. When clicked, the administrator can choose the language for public mail notifications for shared files.



What this means is that email notifications will be sent in the language of the user that shared an item. By default the language is the share owner's language.

However, it can be changed to any of the currently available languages. It is also possible to change this setting on the command-line by using the occ config:app:set command, as in this example:

```
{occ-command-example-prefix} \
  config:app:set core shareapi_public_notification_lang \
  --value '<language code>'
```

	In the above example <language code> is an ISO 3166-1 alpha-2 two-letter country code, such as **ru**, **gb**, **us**, and **au**.

	To use this functionality, your ownCloud server must be configured to send mail.

Allow users to share file via social media

Check this option to enable displaying of a set of links that allow for quickly sharing files and share links via **Twitter**, **Facebook**, **Google+**, **Diaspora**, and email.



Automatically accept new incoming local user shares

Disabling this option activates the "Pending Shares" feature. Users will be notified and have to accept new incoming user shares before they appear in the file list and are available for access giving them more control over their account. More information about pending shares can be found in the release notes.

Allow resharing

Check this option to enable users to re-share files shared with them.

Default user and group share permissions

Administrators can define the permissions for user/group shares that are set by

default when users create new shares. As shares are created instantly after choosing the recipient, administrators can set the default to e.g. read-only to avoid creating shares with too many permissions unintentionally.

Restrict users to only share with users in their groups

Check this option to confine sharing within group memberships.



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.

Restrict users to only share with groups they are a member of

When this option is enabled, users can only share with groups they are a member of. They can still share with all users of the instance but not with groups they are not a member of. To restrict sharing to users in groups the sharer is a member of the option "Restrict users to only share with users in their groups" can be used. More information about more granular sharing restrictions can be found in the release notes.

Allow users to send mail notification for shared files

Check this option to enable users to send an email notification to every ownCloud user that the file is shared with.

Exclude groups from sharing

Check this option to prevent members of specific groups from creating any file shares. When you check this, you'll get a dropdown list of all your groups to choose from. Members of excluded groups can still receive shares, but not create any.

Allow username autocompletion in share dialog

Check this option to enable auto-completion of ownCloud usernames.

Restrict enumeration to group members

Check this option to restrict auto-completion of ownCloud usernames to only those users who are members of the same group(s) that the user is in.

Extra field to display in autocomplete results

The autocomplete dropdowns in ownCloud usually show the display name of other users when it is set. If it's not set, they show the user ID / login name, as display names are not unique you can run into situations where you can't distinguish the proposed users. This option enables to add mail addresses or user ID's to make them distinguishable.

Blacklist Groups From Receiving Shares

Sometimes it's necessary or desirable to block groups from receiving shares. For example, if a group has a significant number of users (> 5,000) or if it's a system group, then it can be advisable to block it from receiving shares. In these cases, ownCloud administrators can blacklist one or more groups, so that they do not receive shares.

To blacklist one or more groups, via the Web UI, under **"Admin → Settings → Sharing"**, add one or more groups to the "*Files Sharing*" list. As you type the group's name, if it exists, it will appear in the drop down list, where you can select it.



Transferring Files to Another User

You may transfer files from one user to another with occ. The command transfers either all or a limited set of files from one user to another. It also transfers the shares and metadata info associated with those files (*shares*, *tags*, and *comments*, etc). This is useful when you have to transfer a user's files to another user before you delete them.

Trashbin contents are not transferred.

Here is an example of how to transfer all files from one user to another.

```
occ files:transfer-ownership <source-user> <destination-user>
```

Here is an example of how to transfer *a limited group* a single folder from one user to another. In it, folder/to/move, and any file and folder inside it will be moved to <destination-user>.

```
{occ-command-example-prefix} files:transfer-ownership --path="folder/to/move"
<source-user> <destination-user>
```

When using this command keep two things in mind:

1. The directory provided to the --path switch **must** exist inside data/<source-user>/files.
2. The directory (and its contents) won't be moved as is between the users. It'll be moved inside the destination user's files directory, and placed in a directory which follows the format: transferred from <source-user> on <timestamp>. Using the example above, it will be stored under: data/<destination-user>/files/transferred from <source-user> on 20170426_124510/



See the occ command, for a complete occ command reference.)

Creating Persistent File Shares

When a user is deleted, their files are also deleted. As you can imagine, this is a problem if they created file shares that need to be preserved, because these disappear as well. In ownCloud 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.

Create Shares Programmatically

If you need to create new shares using command-line scripts, there are two available option.

- occ files_external:create
- occ files_external:import

occ files_external:create

This command provides for the creation of both personal (for a specific user) and general shares. The command's configuration options can be provided either as individual arguments or collectively, as a JSON object. For more information about the command, refer to the the occ files-external documentation.

Personal Share

```
{occ-command-example-prefix} files_external:create /my_share_name
windows_network_drive \
   password::logincredentials \
   --config={host=127.0.0.1, share='home', root='$user', domain='
owncloud.local'} \
   --user someuser
```

```
{occ-command-example-prefix} files_external:create /my_share_name
windows_network_drive \
   password::logincredentials \
   --config host=127.0.0.1 \
   --config share='home' \
   --config root='$user' \
   --config domain='somedomain.local' \
   --user someuser
```

General Share

```
{occ-command-example-prefix} files_external:create /my_share_name
windows_network_drive \
   password::logincredentials \
   --config={host=127.0.0.1, share='home', root='$user', domain='
owncloud.local'}
```

```
{occ-command-example-prefix} files_external:create /my_share_name
windows_network_drive \
   password::logincredentials \
   --config host=127.0.0.1 \
   --config share='home' \
   --config root='$user' \
   --config domain='somedomain.local'
```

occ files_external:import

You can create general and personal shares passing the configuration details via JSON files, using the occ files_external:import command.

General Share

```
{occ-command-example-prefix} files_external:import /import.json
```

Personal Share

```
{occ-command-example-prefix} files_external:import /import.json --user someuser
```

In the two examples above, here is a sample JSON file, showing all of the available configuration options that the command supports.

```
{
   "mount_point": "\/my_share_name",
   "storage": "OCA\\windows_network_drive\\lib\\WND",
   "authentication_type": "password::logincredentials",
   "configuration": {
      "host": "127.0.0.1",
      "share": "home",
      "root": "$user",
      "domain": "owncloud.local"
   },
   "options": {
      "enable_sharing": false
   },
   "applicable_users": [],
   "applicable_groups": []
}
```

Share Permissions

Permissions Masks

READ	1
UPDATE	2 ("can update" in web UI)
CREATE	4 ("can create" in web UI)
DELETE	8 ("can delete" in web UI)
SHARE	16 ("can reshare" in web UI)

File Operations Shorthand for the Later Table

Operation	Description
download	download/read/get a file or display a folder contents
upload	a new file can be uploaded/created (file target does not exist)
upload_overwrite	a file can overwrite an existing one

Operation	Description
rename	rename file to new name, all within the shared folder
move_in	move a file from outside the shared folder into the shared folder
move_in_overwrite	move a file from outside the shared folder and overwrite a file inside the shared folder.  SabreDAV automatically deletes the target file first before moving, so requires DELETE permission too.
move_in_subdir	move a file already in the shared folder into a subdir within the shared folder
move_in_subdir_overwrite	move a file already in the shared folder into a subdir within the shared folder and overwrite an existing file there
move_out	move a file to outside of the shared folder
move_out_subdir	move a file out of a subdir of the shared folder into the shared folder
copy_in	copy a file from outside the shared folder into the shared folder
copy_in_overwrite	copy a file from outside the shared folder and overwrite a file inside the shared folder  SabreDAV automatically deletes the target file first before copying, so requires DELETE permission too.
delete	delete a file inside the shared folder
mkdir	create folder inside the shared folder
rmdir	delete folder inside the shared folder

The following lists what operations are allowed for the different permission combinations (share permission is omitted as it is not relevant to file operations):

Operation(s)	Permission Combinations
READ (aka read-only)	• download

Operation(s)	Permission Combinations
READ + CREATE	• download • upload • move_in • copy_in • mkdir
READ + UPDATE	• download • upload_overwrite • rename
READ + DELETE	• download • move_out • delete • rmdir
READ + CREATE + UPDATE	• download • upload • upload_overwrite • rename • move_in • copy_in • mkdir
READ + CREATE + DELETE	• download • upload • move_in • move_in_overwrite • move_in_subdir • move_in_subdir_overwrite • move_out • move_out_subdir • copy_in • copy_in_overwrite • delete • mkdir • rmdir

Operation(s)	Permission Combinations
READ + UPDATE + DELETE	• download • upload_overwrite • rename • move_out • delete • rmdir
READ + CREATE + UPDATE + DELETE (all permissions)	• download • upload • upload_overwrite • rename • move_in • move_in_overwrite • move_in_subdir • move_in_subdir_overwrite • move_out • move_out_subdir • copy_in • copy_in_overwrite • delete • mkdir • rmdir

Controlling File Versions

How Versions are Created

Every time when a file gets rewritten to the storage, the versions app (files_versions) creates a new backup copy of the file. These backups are stored inside a folder files_versions which is inside the users root folder. The app will add the suffix .v followed by the unix timestamp of the creation date of the backup copy.

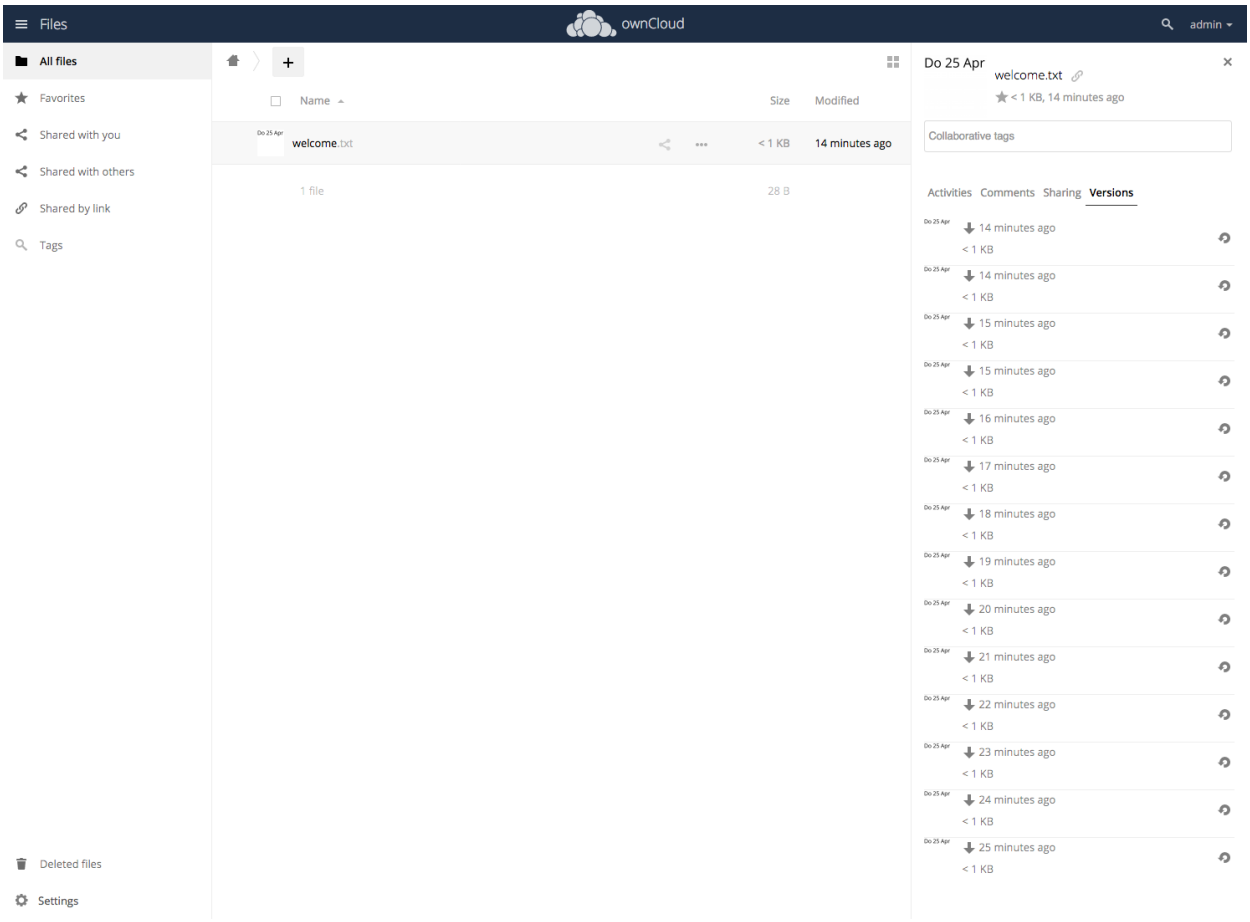
```
.
├── files
│   └── welcome.txt
└── files_versions
    ├── welcome.txt.v1556203470
    ├── welcome.txt.v1556203501
    └── welcome.txt.v1556203567
```



File versioning only gets triggered if the change is made via the ownCloud ecosystem. It does not get triggered if the change is made at a mounted filesystem directly.

Versions are displayed in the WebUI in the details view in the right sidebar if you click

on the file row in the file listing. You can restore the current file to one of the earlier backup copies in the list, by clicking on the btn:[restore] icon of the specific version.



How Versions are Deleted

The versions app deletes old file versions automatically to ensure that users do not exceed their storage quotas. This is done by automatic background jobs which clean up the versions following a specific pattern. This pattern defines the expiration date for each backup version.

Default Versions Delete Patterns

This is the default pattern used to delete old versions:

- For the last second we keep one version
- For the last 10 seconds ownCloud keeps one version every 2 seconds
- For the last minute ownCloud keeps one version every 10 seconds
- For the last hour ownCloud keeps one version every minute
- For the last 24 hours ownCloud keeps one version every hour
- For the last 30 days ownCloud keeps one version every day
- If the versions are older than 30 days ownCloud keeps one version every week

The versions are adjusted along this pattern every time a new version is created and the background job was executed.

Example

Time Period before last Expiration	Maximum Number of Versions:
1 second	1

Time Period before last Expiration	Maximum Number of Versions:
10 seconds	5
1 minute	6
1 hour	59
1 day	23
30 days	30



The versions app never uses more that 50% of the user's storage quota. If the stored versions exceed this limit, ownCloud deletes the oldest file versions until it meets the disk space limit again.



Adjust the 'versions_retention_obligation' setting in config.php to avoid filling up the user's quota.

Change the Expiration Settings

You may alter the default pattern in config.php. The default setting is auto, which sets the default pattern:

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

Possible Config Values

D, auto	Keep versions at least for D days, apply expiration rules to all versions that are older than D days
auto, D	Delete all versions that are older than D days automatically, delete other versions according to expiration rules
D1, D2	Keep versions for at least D1 days and delete when they exceed D2 days.
disabled	Disable Versions; no files will be deleted.

Example 1:

Keep all versions for at least 10 days, apply expiration rules to all versions that are older than 10 days. This will keep a lot more versions during the last 10 days compared to the default pattern.

```
'versions_retention_obligation' => '10, auto',
```

Example 2:

Apply expiration rules to all versions that are created during the last 30 days and do not keep any versions older than 30 days.

```
'versions_retention_obligation' => 'auto, 30',
```

Example 3:

Do not apply any expiration rules. Delete all versions after 30 days.

```
'versions_retention_obligation' => '30, 30',
```

Enterprise File Retention

Enterprise customers have additional tools for managing file retention policies; see Advanced File Tagging With the Workflow App.

Transactional File Locking

ownCloud'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

Transactional File locking will not prevent multiple users from editing the same document, nor 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 is in ownCloud core, and replaces the old File Locking app. The File Locking app was removed from ownCloud in version 8.2.1. If your ownCloud server still has the File Locking app, you **must** visit your Apps page to verify that it is disabled; the File Locking app and Transactional File Locking cannot both operate at the same time.

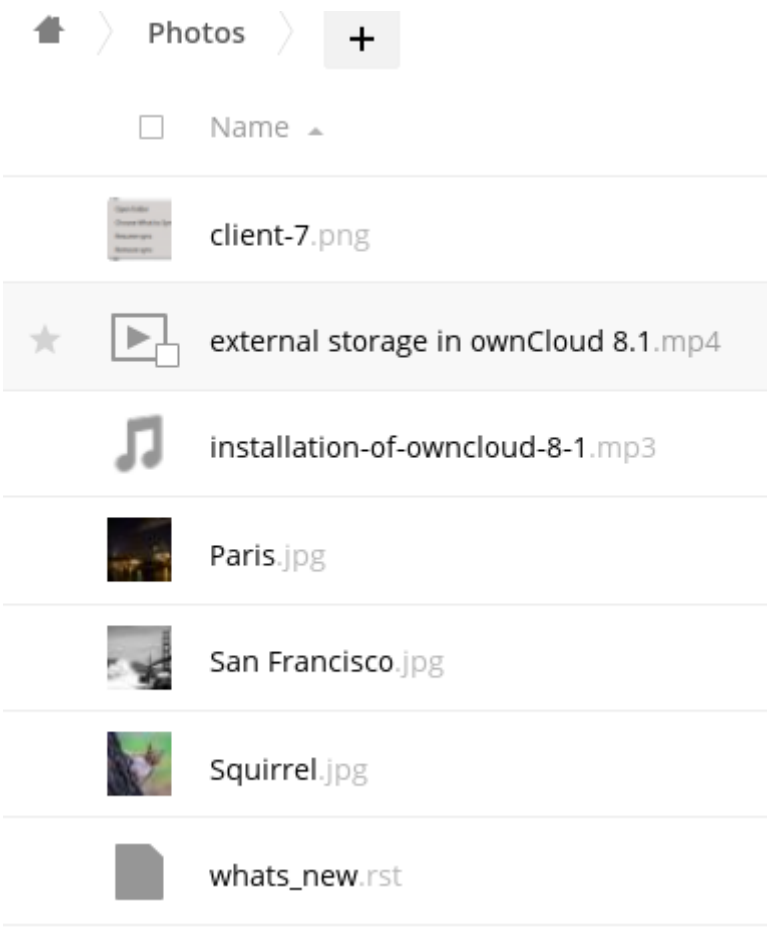
File locking is enabled by default, using the database locking backend. This places a significant load on your database. Using memcache.locking relieves the database load and improves performance. Admins of ownCloud servers with heavy workloads should install a memory cache .

Previews Configuration

Introduction

The ownCloud thumbnail system generates previews of files for all ownCloud apps that display files, such as Files.

The following image shows some examples of previews of various file types.



By default, ownCloud can generate previews for the following filetypes:

- Images files
- Cover of MP3 files
- Text documents

Older versions of ownCloud also supported the preview generation of other file types such as PDF, SVG or various office documents. Due to security concerns those providers have been disabled by default and are considered unsupported. While those providers are still available, we discourage enabling them, and they are not documented.

Parameters

Please notice that the ownCloud preview system comes already with sensible defaults, and therefore it is usually unnecessary to adjust those configuration values. If you want to configure previews, add or change the following parameters in config/config.php.

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 and will display generic icons instead of thumbnails.

Set the configuration option enable_previews to false:

```
'enable_previews' => false,
```

Maximum preview size

There are two configuration options for setting the maximum size (in pixels) of a preview. These are preview_max_x which represents the x-axis and preview_max_y which represents the y-axis. The default value you can reference in config/config.sample.php is set to 2048.

The following example would limit previews to a maximum size of 100 px × 100 px:

```
'preview_max_x' => 100,
'preview_max_y' => 100,
```



If you want no limit applied for one or both of these values then set them to null.

Maximum scale factor

If a lot of small pictures are stored on the ownCloud 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:

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

If you want to disable scaling at all, you can set the config value to `1':

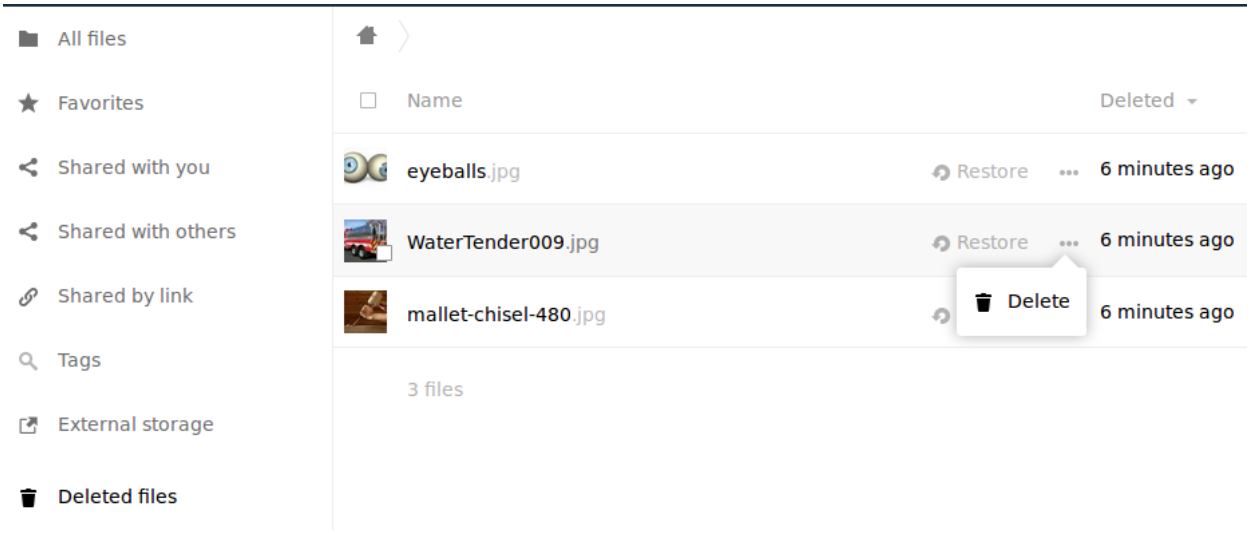
```
'preview_max_scale_factor' => 1,
```

If you want to disable the maximum scaling factor, you can set the config value to null:

```
'preview_max_scale_factor' => null,
```

Managing the Trash Bin

The ownCloud Trashbin (files_trashbin) permanently deletes files according to users' storage quotas and file ages. When a user deletes a file it is not immediately removed from your ownCloud server, but goes into the Trashbin. Then the user has the options to un-delete the file, or to delete it permanently.



As the ownCloud server administrator, you have two occ commands for permanently deleting files from the Trashbin manually, without waiting for the normal aging-out process:

```
trashbin
 trashbin:cleanup   Remove deleted files
 trashbin:expire    Expires the users trashbin
```

The trashbin:cleanup command removes the deleted files of all users, or you may specify certain users in a space-delimited list. This example removes all the deleted files of all users:

```
{occ-command-example-prefix} trashbin:cleanup
Remove all deleted files
Remove deleted files for users on backend Database
 user1
 user2
 user3
 user4
```

This example removes the deleted files of user2 and user4:

```
{occ-command-example-prefix} trashbin:cleanup user2 user4
 Remove deleted files of user2
 Remove deleted files of user4
```

trashbin:expire deletes only expired files according to the trashbin_retention_obligation setting in config.php. The default setting is auto, which keeps files in the Trashbin for 30 days, then deletes the oldest files as space is needed to keep users within their storage quotas. Files may not be deleted if the space is not needed.

The default is to delete expired files for all users, or you may list users in a space-delimited list:

```
{occ-command-example-prefix} trashbin:cleanup user1 user2
 Remove deleted files of user1
 Remove deleted files of user2
```

See the **Deleted Files** section in Sample PHP Configuration Parameters, and the Trash Bin section of the occ commands.

General Topics

In this section you will find information about:

- Code Signing
- General Troubleshooting
- Impersonating Users

Code Signing

Introduction

ownCloud supports code signing for the core releases, and for ownCloud 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 ownCloud.

FAQ

Why Did ownCloud 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 ownCloud?

The ownCloud project is open source and always will be. We do not want to make it more difficult for our users to run ownCloud. Any code signing errors on upgrades will not prevent ownCloud 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 ownCloud 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 ownCloud is enabled when the ownCloud release version branch has been set to stable.

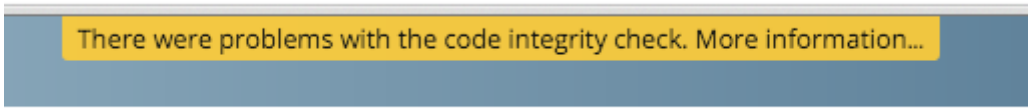
For custom distributions of ownCloud it is recommended to change the release version branch in version.php to something else than "stable".

Is Code Signing Mandatory For Apps?

Code signing is optional for all third-party applications.

Fixing Invalid Code Integrity Messages

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

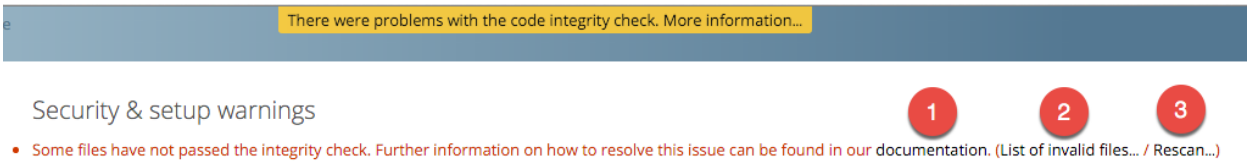




The yellow banner is only shown for admin users.

Clicking on this link will take you to your ownCloud admin page, which provides the following options:

1. Link to this documentation entry.
2. Show a list of invalid files.
3. Trigger a rescan.



To debug issues caused by the code integrity check click on btn:[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.
- tasks
  - EXCEPTION
     - OC\IntegrityCheck\Exceptions\InvalidSignatureException
     - Certificate has been revoked.

Raw output
==========
Array
(
   [core] => Array
      (
         [INVALID_HASH] => Array
            (
               [/index.php] => Array
                  (
                     [expected] =>
                     f1c5e2630d784bc9cb02d5a28f55d6f24d06dae2a0fee685f3
                     c2521b050955d9d452769f61454c9ddfa9c308146ade10546c
                     fa829794448eaffbc9a04a29d216
                     [current] =>
                     ce08bf30bcbb879a18b49239a9bec6b8702f52452f88a9d321
                     42cad8d2494d5735e6bfa0d8642b2762c62ca5be49f9bf4ec2
                     31d4a230559d4f3e2c471d3ea094
```

```
            )

        [/version.php] => Array
            (
                [expected] =>
                c5a03bacae8dedf8b239997901ba1fffd2fe51271d13a00cc4
                b34b09cca5176397a89fc27381cbb1f72855fa18b69b6f87d7
                d5685c3b45aee373b09be54742ea
                [current] =>
                88a3a92c11db91dec1ac3be0e1c87f862c95ba6ffaaaa3f2c3
                b8f682187c66f07af3a3b557a868342ef4a271218fe1c1e300
                c478e6c156c5955ed53c40d06585
            )

    )

    [EXTRA_FILE] => Array
        (
            [/test.php] => Array
                (
                    [expected] =>
                    [current] =>
                    09563164f9904a837f9ca0b5f626db56c838e5098e0ccc1d8b
                    935f68fa03a25c5ec6f6b2d9e44a868e8b85764dafd1605522
                    b4af8db0ae269d73432e9a01e63a
                )

        )

)

[calendar] => Array
(
    [EXCEPTION] => Array
        (
            [class] => OC\IntegrityCheck\Exceptions\InvalidSignature
            Exception
            [message] => Signature data not found.
        )

)
[tasks] => Array
(
    [EXCEPTION] => Array
        (
            [class] => OC\IntegrityCheck\Exceptions\InvalidSignatureException
            [message] => Certificate has been revoked.
        )

)
```

```
)
```

In above error output it can be seen that:

1. In the ownCloud core (that is, the ownCloud server itself) the files index.php and version.php do have the wrong version.
2. In the ownCloud core the unrequired extra file /test.php has been found.
3. It was not possible to verify the signature of the calendar application.
4. The certificate of the task application was revoked.

You have to do the following steps to solve this:

1. Upload the correct index.php and version.php files from e.g. the archive of your ownCloud version.
2. Delete the test.php file.
3. Contact the developer of the application. A new version of the app containing a valid signature file needs to be released.
4. Contact the developer of the application. A new version of the app signed with a valid signature needs to be released.

For other means on how to receive support please take a look at https://owncloud.org/support/. After fixing these problems verify by clicking btn:[Rescan].



When using a FTP client to upload those files make sure it is using the Binary transfer mode instead of the ASCII transfer mode.

Rescans

Rescans are triggered at installation, and by updates. You may run scans manually with the occ command. The first command scans the ownCloud core 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 the occ command to learn more about using occ.

Errors

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 ownCloud 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 ownCloud 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.
    - Certificate has been revoked.
      - The certificate which was used to sign the application was revoked.

General Troubleshooting

Introduction

If you have trouble installing, configuring or maintaining ownCloud, please refer to our community support channels:

- The ownCloud Forum



The ownCloud forum have a FAQ category where each topic corresponds to typical errors or frequently occurring issues.

- The ownCloud User mailing list

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 ownCloud healthy and sustainable!

If you are using ownCloud in a business or otherwise large scale deployment, note that ownCloud Inc. offers the Enterprise Edition with commercial support options.

Bugs

If you think you have found a bug in ownCloud, 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.

Check the ownCloud System Requirements, especially supported browser versions. When you see warnings about code integrity, refer to Code Signing.

Disable Third-Party Apps

Third-party apps may cause problems during upgrades. To avoid this happening, we strongly encourage administrators to always disable third-party apps before upgrades, and for troubleshooting purposes.

ownCloud Logfiles

In a standard ownCloud installation the log level is set to Normal. To find any issues you need to raise the log level to All in your config.php file, or to **Everything** on your ownCloud Admin page. Please see Logging Configuration for more information on these log levels.

Some logging - for example JavaScript console logging - needs debugging enabled. Edit config/config.php and change 'debug' ⇒ false, to 'debug' ⇒ true, Be sure to change it back when you are finished.

For JavaScript issues you will also need to view the javascript console. All major browsers have developer tools for viewing the console, and you usually access them by pressing F12. For Firefox we recommend to installing the Firebug extension.



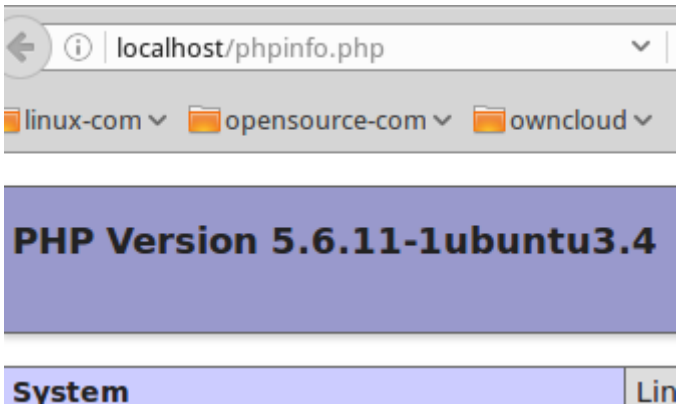
The logfile of ownCloud is located in the data directory owncloud/data/owncloud.log.

PHP Version and Information

You will need to know your PHP version and configurations. To do this, create a plain-text file named **phpinfo.php** and place it in your Web root, for example /var/www/html/phpinfo.php. (Your Web root may be in a different location; your Linux distribution documentation will tell you where.) This file contains just this line:

```
<?php phpinfo(); ?>
```

Open this file in a Web browser by pointing your browser to localhost/phpinfo.php:



Your PHP version is at the top, and the rest of the page contains abundant system information such as active modules, active .ini files, and much more. When you are finished reviewing your information you must delete phpinfo.php, or move it outside of your Web directory, because it is a security risk to expose such sensitive data.

Debugging Sync Issues



The data directory on the server is exclusive to ownCloud and must not be modified manually.

Disregarding this can lead to unwanted behaviours like:

- Problems with sync clients
- Undetected changes due to caching in the database

If you need to directly upload files from the same server please use a WebDAV command line client like cadaver to upload files to the WebDAV interface at:

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

Common problems / error messages

Some common problems / error messages found in your logfiles as described above:

- SQLSTATE[HY000] [1040] Too many connections → You need to increase the connection limit of your database, please refer to the manual of your database for more information.
- SQLSTATE[HY000]: General error: 5 database is locked → You're using SQLite which can't handle a lot of parallel requests. Please consider converting to another database like described in converting Database Type.
- SQLSTATE[HY000]: General error: 2006 MySQL server has gone away → Please refer to Troubleshooting for more information.
- SQLSTATE[HY000] [2002] No such file or directory → There is a problem accessing your SQLite database file in your data directory (data/owncloud.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 or expected filesize 4734206 got 458752 → 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. On Apache with mod_php using a multi-processing module other than prefork could be another reason. Further information is available in the forums.
- No basic authentication headers were found → This error is shown in your data/owncloud.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 ownCloud via WebDAV, CalDAV and CardDAV clients is failing. More information on how to correctly configure your environment can be found at the forums.

OAuth2

ownCloud clients cannot connect to the ownCloud server

If ownCloud clients cannot connect to your ownCloud server, check to see if PROPFIND requests receive HTTP/1.1 401 Unauthorized responses. If this is happening, more than likely your webserver configuration is stripping out the bearer authorization header.

If you're using the Apache web server, add the following SetEnvIf directive to your Apache configuration, whether in the general Apache config, in a configuration include file, or in ownCloud's .htaccess file.

```
SetEnvIf Authorization "(.*)" HTTP_AUTHORIZATION=$1
```

Missing Data Directory

During the normal course of operations, the ownCloud data directory may be temporarily unavailable for a variety of reasons. These can include network timeouts on mounted network disks, unintentional unmounting of the partition on which the directory sits, or a corruption of the RAID setup. If you have experienced this, here's how ownCloud works and what you can expect.

During normal operation, ownCloud's data directory contains a hidden file, named .ocdata. The purpose of this file is for setups where the data folder is mounted (such as via NFS) and for some reason the mount disappeared. If the directory isn't available, the data folder would, in effect, be completely empty and the .ocdata would be missing. When this happens, ownCloud will return a 503 Service not available error, to prevent clients believing that the files are gone.

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 ownCloud itself.



In the following the paths to the logfiles of a default Debian installation running Apache2 with mod_php is assumed. On other Web servers, Linux distros or operating systems they can differ.

- The logfile of Apache2 is located in /var/log/apache2/error.log.
- The logfile of PHP can be configured in your /etc/php5/apache2/php.ini. You need to set the directive log_errors to On and choose the path to store the logfile in the error_log directive. After those changes you need to restart your Web server.
- The logfile of ownCloud is located in the data directory /var/www/owncloud/data/owncloud.log.

Web Server and PHP Modules



Lighttpd is not supported with ownCloud — and some ownCloud features may not work *at all* on Lighttpd.

There are some Web server or PHP modules which are known to cause various problems like broken up-/downloads. The following shows a draft overview of these modules:

Apache

- libapache2-mod-php5filter (use libapache2-mod-php5 instead)
- mod_dav
- mod_deflate
- mod_evasive
- mod_pagespeed
- mod_proxy_html (can cause broken PDF downloads)
- mod_reqtimeout

- mod_security
- mod_spdy together with libapache2-mod-php5 / mod_php (use fcgi or php-fpm instead)
- mod_xsendfile / X-Sendfile (causing broken downloads if not configured correctly)

PHP

- eAccelerator

Troubleshooting WebDAV

General troubleshooting

ownCloud 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.

Error 0x80070043 The network name cannot be found. while adding a network drive

The windows native WebDAV client might fail with the following error message:

```
Error 0x80070043 "The network name cannot be found." while adding a network drive
```

A known workaround for this issue is to update your web server configuration.

Apache

You need to add the following rule set to your main web server or virtual host configuration, or the .htaccess file in your document root.

```
# Fixes Windows WebDav client error 0x80070043 "The network name cannot be found."
RewriteEngine On
RewriteCond %{HTTP_USER_AGENT} ^(DavClnt)$
RewriteCond %{REQUEST_METHOD} ^(OPTIONS)$
RewriteRule .* - [R=401,L]
```

Service Discovery

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

If you want to use CalDAV or CardDAV clients together with ownCloud it is important to have a correct working setup of the following URLs:

> https://example.com/.well-known/carddav
> https://example.com/.well-known/caldav

Those need to be redirecting your clients to the correct DAV endpoints. If running ownCloud at the document root of your Web server the correct URL is:

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

and if running in a subfolder like owncloud:

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

For the first case the .htaccess file shipped with ownCloud should do this work for your when running Apache. You only need to make sure that your Web server is using this file.

If your ownCloud instance is installed in a subfolder called owncloud and you're running Apache create or edit the .htaccess file within the document root of your Web server and add the following lines:

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

Now change the URL in the client settings to just use:

https://example.com

instead of e.g.

https://example.com/owncloud/remote.php/dav/principals/username.

There are also several techniques to remedy this, which are described extensively at the Sabre DAV website.

Unable to update Contacts or Events

If you get an error like:

PATCH https://example.com/remote.php/dav HTTP/1.0 501 Not Implemented

it is likely caused by one of the following reasons:

Using Pound reverse-proxy/load balancer
: As of writing this Pound doesn't support the HTTP/1.1 verb. Pound is easily patched to support HTTP/1.1.

Misconfigured Web server
: Your Web server is misconfigured and blocks the needed DAV methods. Please refer to Troubleshooting WebDAV above for troubleshooting steps.

Client Sync Stalls

One known reason is stray locks. These should expire automatically after an hour. If stray locks don't expire (identified by e.g. repeated file.txt is locked and/or Exception\\\\FileLocked messages in your data/owncloud.log), make sure that you are running system cron and not Ajax cron (See Background Jobs). See https://github.com/owncloud/core/issues/22116 and https://central.owncloud.org/t/file-is-locked-how-to-unlock/985 for some discussion and additional info of this issue.

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.

Impersonating Users

Introduction

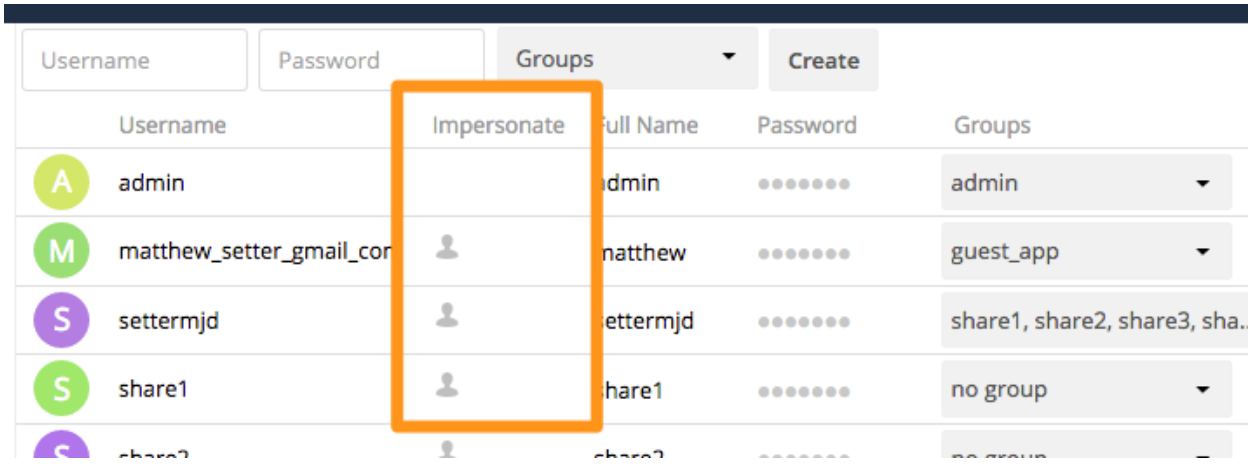
Sometimes you may need to use your ownCloud installation as another user, whether to help users debug an issue or to get a better understanding of what they see when they use their ownCloud account. The ability to do so is a feature delivered via an ownCloud app called Impersonate.



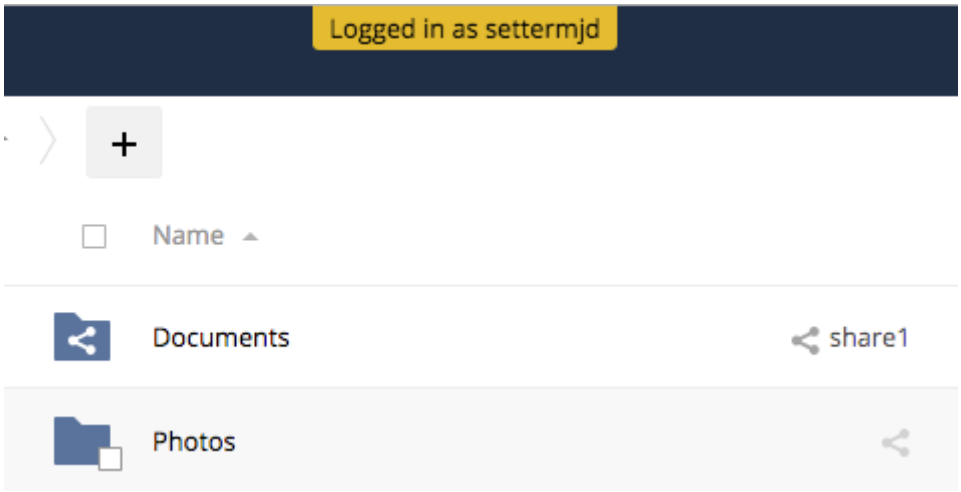
This functionality is available only to administrators.

Impersonating a User

When installed, you can then impersonate users; in effect, you will be logged in as said user. To do so, go to the Users list, where you will now see a new column available called "**Impersonate**", as in the screenshot below.



Click the gray head icon next to the user that you want to impersonate. Doing so will log you in as that user, temporarily pausing your current session. You will see a notification at the top of the page that confirms you're now logged in as (or impersonating) that user.



Anything that you see until you log out will be what that user would see.

Ending an Impersonation

When you're ready to stop impersonating the user, log out and you will return to your normal user session.

Allow Some or All Group Administrators To Impersonate Users

As a security measure, the application lets ownCloud administrators restrict the ability to impersonate users to:

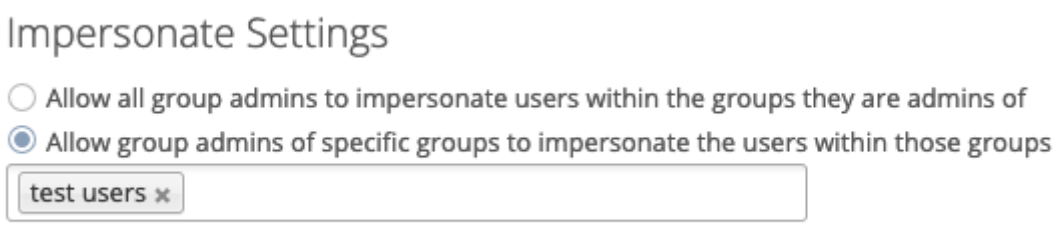
- All group administrators.
- Specific group administrators.



By default, when the Impersonate app is installed, only the ownCloud administrator will be allowed to impersonate users. When the app is installed and configured, ownCloud administrators retain the ability to impersonate all users of an ownCloud instance.

When enabled and configured, only a group's administrator can impersonate members of their group. For example, if an ownCloud administrator restricts user impersonation only to the group: group1, then only group1's administrators can impersonate users belonging to `group1. No other users can impersonate other users.

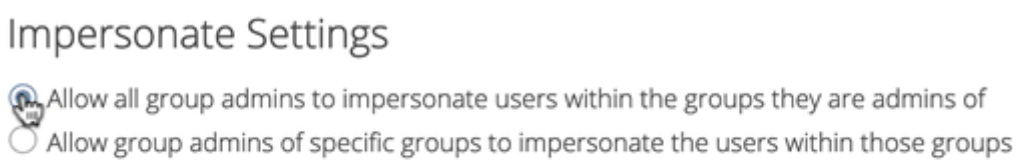
To configure it, in the administrator settings panel, which you can find under menu:administrator[Settings > Admin > User Authentication], you'll see a section titled: "**Impersonate Settings**" (which you can see below).



If you want to allow group admins to impersonate users within groups which they administer, click btn:[Allow all group admins to impersonate users within the groups they are admins of].

If you want to limit impersonation to specific group admins, first click btn:[Allow group

admins of specific groups to impersonate the users within those groups]. With the option checked, click into the textbox underneath it. You will see a list of the matching groups on your ownCloud installation appear, which will change, based on what you type in the textbox.



Choose one or more groups from the list, and they will be added to the textbox, restricting this functionality to only those groups.

Full Text Search

Introduction

The search_elastic app integrates full text search into ownCloud, powered by Elasticsearch.

Prerequisites

There is only one pre-requisite for the search_elastic app, and that's a fully functioning Elasticsearch server with the ingest-attachment processor to be present. The ingest-attachment processor lets Elasticsearch extract file attachments in common formats, such as PPT, XLS, and PDF.

To install the processor, from your Elasticsearch installation directory, run the following command:

```
bin/elasticsearch-plugin install ingest-attachment
service elasticsearch restart
```



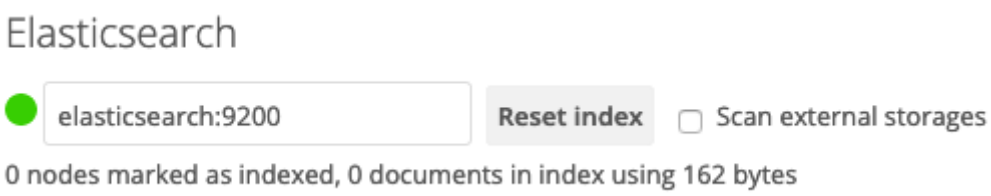
Only Elasticsearch version 5.6 is fully supported. Versions 6 and 7 are yet to be supported.

Install and Configure the search_elastic App

To install the search_elastic app:

1. Download and extract the tarball of the latest release of the search_elastic app to the apps directory (or custom apps directory) of your ownCloud instance.
2. Use the occ app:enable command to enable the search_elastic application.
3. Go to menu:Settings[Search (admin)] and set the hostname (or IP address) and port of the Elasticsearch server and click btn:["Reset Index"].

Configuring Elasticsearch in ownCloud



The index can also be managed from the command-line, via the occ search:index commands. The commands let administrators *create*, *rebuild*, *reset*, and *update* the search index.

Known Limitations

Currently, there are several known limitations to the app; these are:

- Files are shown twice in search results when searching by filename.
- If a shared file is renamed by the sharee (share receiver), the sharee cannot find the file using the new filename.
- If the search query doesn't match the *start* of at least one word in an available file's name, no results are returned.
- Search results are not updated when a text file is rolled back to an earlier version.
- The app does not return results for federated share files.
- The app only works with the default encryption module "*Master Key*".

How To Install and Configure an LDAP Proxy-Cache Server

Background

To reduce network traffic overhead and avoid problems either logging in or performing user searches while sharing, it's an excellent idea to implement an LDAP proxy cache. An LDAP proxy cache server, similar to other kinds of caching servers, is a special type of LDAP replica. It can cache a range of LDAP records, often resulting in improved LDAP server performance.

Specifically, the records which need to be cached for improved ownCloud performance are:

- Users that are allowed to log in
- Groups (limited to the allowed users)
- Search fields (e.g., sAMAccountName, CN, SN, givenName, and displayName)

How To Set Up the Server

To set up the LDAP Proxy-Cache server work through the following five steps:

1. Install OpenLDAP
2. Configure the Server
3. Enable the Configuration File
4. Check the Log
5. Perform a Test Search
6. Configure the ownCloud LDAP app

Install OpenLDAP

There are a number of LDAP server implementations available. The one used in this guide is OpenLDAP.



While OpenLDAP does work on most of the common operating systems, for the purposes of this guide we'll be using a Debian-based Linux distribution.

First, update your system to ensure that you are using the latest packages. Then, run the following command:

```
sudo apt-get update && apt-get upgrade -y
```

Next, install OpenLDAP and its associated packages by running the following command:

```
sudo apt-get install slapd ldap-utils -y
```

Configure the Server

With OpenLDAP installed and running, you now need to configure it. One way of doing so is to create a configuration file. Create /etc/ldap/slapd.conf with your text editor of choice, and add the following configuration to it.

```
# This an example of a config file:

# See slapd.conf(5)

# Global Directives:

# Schema and objectClass definitions
include         /etc/ldap/schema/core.schema
include         /etc/ldap/schema/cosine.schema
include         /etc/ldap/schema/nis.schema
include         /etc/ldap/schema/inetorgperson.schema

# Where the pid file is put. The init.d script
# will not stop the server if you change this.
pidfile         /var/run/slapd/slapd.pid

# List of arguments that were passed to the server
argsfile        /var/run/slapd/slapd.args

# Read slapd.conf(5) for possible values
# Change loglevel to "any" if you want to see everything.
loglevel        none

# Where the dynamically loaded modules are stored
modulepath      /usr/lib/ldap
```

```
# Here are the recommended modules:

# module for the target ldap-server
moduleload  back_ldap.la

# module for your local database
moduleload  back_hdb.la

# module for rewriting attributes
moduleload  rwm

# caching module
moduleload  pcache.la

# module to enable memberof in LDAP
moduleload memberof.la

# The maximum number of entries that is returned for a search operation
sizelimit 500

# The tool-threads parameter sets the actual amount of cpu's that is used
# for indexing.
tool-threads 1

# Type of backend, for example "ldap"
backend         ldap

# Type of database
database        ldap

# If you only have read access, set this to "yes"
readonly        yes

# Set which protocol to use, we suggest "3"
protocol-version  3

# remember bind credentials
rebind-as-user

# If you want to save time and don't want to list all the refferals, set to "yes"
norefs  yes

# Same as above
chase-referrals no

# Specify the URL of your ldap server and the port.
# For unencrypted access use the port 389, for encrypted 636
# If you have to use 636, you will also probably have to import
# the certificate of your target server. restart your webserver after you do.
uri "ldap://192.168.178.2:389"
```

```
# The base of your directory in database, for example "dc=ldap01,dc=com"
suffix          "dc=ldap01,dc=com"

# rootdn directive for specifying a superuser on the database.
# If you don't have access to the admin user, use the one you have.
rootdn          "cn=admin,dc=ldap01,dc=com"

# Now we start initialising the modules
# First the rewrite module
overlay         rwm

# Now we rewrite the attributes
rwm-map         attribute uid sAMAccountName
rwm-map         attribute dn distinguishedName

# Next one is optional, if you want memberof, for the groups,
# you have to load it.
overlay         memberof

# Now we load the caching module
overlay pcache

# The directive enables proxy caching
# See slapo-pcache

# pcache <database> <max_entries> <numattrsets> <entry_limit> <cc_period>
# Parameters:
#
# <database> for cached entries.
# <max_entries> when reached - cache replacement is invoked
# <numattrsets> = pcacheAttrset
# <entry_limit> limit to the number of entries returned
# <cc_period> Consistency check time to wait
pcache hdb 100000 3 1000 100

# pcachePersist { TRUE | FALSE }
# Write cached results into the database
# Results remain in database after restart
pcachePersist TRUE

# Where the database file are physically stored for database #1
directory       "/var/lib/ldap"

# Caching templates for general search

# pcacheAttrset <index> <attrs...>
# First set the index number
# Then set the attribute to cache
pcacheAttrset   0 1.1
```

```
# pcacheTemplate <template_string> <attrset_index> <ttl>
# First define the querry sting to cache
# Then reference the Attrset
# Last set the time-to-live
pcacheTemplate  (&(|(objectClass=))) 0 3600
pcacheTemplate (objectClass=*) 0 3600

# User Name Field (Advanced Tab)
pcacheAttrset   1 displayname
pcacheTemplate (objectClass=*) 1 3600

# Group Field
pcacheAttrset   2 memberOf
pcacheTemplate (objectClass=*) 2 3600


# This an example of a config file:

# See slapd.conf(5)

# Global Directives:

# Schema and objectClass definitions
include         /etc/ldap/schema/core.schema
include         /etc/ldap/schema/cosine.schema
include         /etc/ldap/schema/nis.schema
include         /etc/ldap/schema/inetorgperson.schema

# Where the pid file is put. The init.d script
# will not stop the server if you change this.
pidfile         /var/run/slapd/slapd.pid

# List of arguments that were passed to the server
argsfile        /var/run/slapd/slapd.args

# Read slapd.conf(5) for possible values
# Change loglevel to "any" if you want to see everything.
loglevel        none

# Where the dynamically loaded modules are stored
modulepath      /usr/lib/ldap

# Here are the recommended modules:

# module for the target ldap-server
moduleload  back_ldap.la

# module for your local database
moduleload  back_hdb.la
```

```
# module for rewriting attributes
moduleload  rwm

# caching module
moduleload  pcache.la

# module to enable memberof in LDAP
moduleload memberof.la


# The maximum number of entries that is returned for a search operation
sizelimit 500

# The tool-threads parameter sets the actual amount of cpu's that is used
# for indexing.
tool-threads 1

# Type of backend, for example "ldap"
backend         ldap

# If you only have read access, set this to "yes"
readonly        yes

# Set which protocol to use, we suggest "3"
protocol-version  3

# remember bind credentials
rebind-as-user

# If you want to save time and don't want to list all the refferals, set to "yes"
norefs  yes

# Same as above
chase-referrals no

# Specify the URL of your ldap server and the port.
# For unencrypted access use the port 389, for encrypted 636
# If you have to use 636, you will also probably have to import
# the certificate of your target server.
# Restart your webserver after you do.
uri "ldap://192.168.178.2:389"

# The base of your directory in database, for example "dc=ldap01,dc=com"
suffix          "dc=ldap01,dc=com"

# rootdn directive for specifying a superuser on the database.
# If you don't have access to the admin user, use the one you have.
rootdn          "cn=admin,dc=ldap01,dc=com"
```

```
# Now we start initialising the modules
# First the rewrite module
overlay        rwm

# Now we rewrite the attributes
rwm-map        attribute uid sAMAccountName
rwm-map        attribute dn distinguishedName

# Next one is optional, if you want memberof, for the groups,
# you have to load it.
overlay        memberof

# Now we load the caching module
overlay pcache

# The directive enables proxy caching
# See slapd-pcache

# pcache <database> <max_entries> <numattrsets> <entry_limit> <cc_period>
# Parameters:
#
# <database> for cached entries.
# <max_entries> when reached - cache replacement is invoked
# <numattrsets> = pcacheAttrset
# <entry_limit> limit to the number of entries returned
# <cc_period> Consistency check time to wait
pcache hdb 100000 3 1000 100

# pcachePersist { TRUE | FALSE }
# Write cached results into the database
# Results remain in database after restart
pcachePersist TRUE

# Where the database file are physically stored for database #1
directory      "/var/lib/ldap"

# Caching templates for general search

# pcacheAttrset <index> <attrs...>
# First set the index number
# Then set the attribute to cache
pcacheAttrset   0 1.1

# pcacheTemplate <template_string> <attrset_index> <ttl>
# First define the query string to cache
# Then reference the Attrset
# Last set the time-to-live
pcacheTemplate  (&(|(objectClass=))) 0 3600
pcacheTemplate (objectClass=*) 0 3600
```

```
# User Name Field (Advanced Tab)
pcacheAttrset   1 displayname
pcacheTemplate (objectClass=*) 1 3600

# Group Field
pcacheAttrset   2 memberOf
pcacheTemplate (objectClass=*) 2 3600
```



This configuration only caches queries from a single Active Directory server. To cache queries from multiple Active Directory servers, a configuration is available below.

After you've done that, save the file and test that there are no errors in the configuration by running:

```
sudo slaptest -f /etc/ldap/slapd.conf
```



If you see warnings in the console output, they are not crucial.

Enable the Configuration File

Next, we need to tell OpenLDAP to use our configuration. To do so, open /etc/default/slapd and add the following line to it:

```
SLAPD_CONF=/etc/ldap/slapd.conf
```

With that done, restart OpenLDAP by running the following command:

```
sudo service slapd restart
```

Open the Log

With OpenLDAP running, review the system log output with the following command:

```
tail -f /var/log/syslog | grep QUERY
```

If there is no such file, you need to install a Syslog daemon. We recommend using Rsyslog. To install it, run the following command:

```
sudo apt install rsyslog
```

Perform a Test Search

Now that the server's installed, configured, and running, we next need to perform a test search. This will check that records are being correctly cached. To do so, run one of the following commands below, after updating it with values from your Active Directory server configuration.

```
sudo ldapsearch -h localhost -x -LLL \
   -D "cn=admin,cn=users,dc=example,dc=com" \
   -b "cn=users,dc=example,dc=com" \
   -w "Password" "(cn=Administrator)" name
```

```
sudo ldapsearch -H ldaps://localhost:636 -x -LLL \
   -D "cn=admin,cn=users,dc=example,dc=com" \
   -b "cn=users,dc=example,dc=com" \
   -W "(cn=Administrator)" name
```

Table 1. Description of Options

Option	Description
-h	Host address (Example: localhost or 192.168.1.1)
-H	Host address (Example: ldap:// or ldaps:// hostname or ip and port :389 or :636)
-x	Simple authentication
-b	Search Base, (Example: cn=Users,dc=example,dc=com)
-D	User with permissions (Example: cn=Admin,dc=example,dc=com)
-LLL	Show only results, no extra information
-w	Password ("Password")
-W	Password, will ask for password and hide your input
(cn=Administrator)	Filter the search
name	Show only these attributes

If the results include: "Query cachable" and "Query answered (x) times", then the setup works.

Configure ownCloud LDAP App

Configuring the ownCloud LDAP application involves several step; these are:

- Enable the LDAP application
- Configure the LDAP application

Enable the LDAP Application

First, login to your ownCloud server as an ownCloud admin; then:

- Click on the menu:Settings[] dropdown menu in the top-right corner.
  - Then, click on menu:Admin[Apps].
  - Click on btn:[Show Disabled Apps] and enable the "*LDAP Integration*" app, and reload the page.
- Click on the menu in the left-hand side, menu:Admin[User Authentication].
  - Select LDAP, if available and not already selected.

Configure the LDAP Application

- Select the menu:Server[] tab.
  - In the first field, enter the server address (either the IP address or hostname). TIP: You can click on the button to detect your server's port or enter it manually.
  - In the next two fields, enter the user DN of the user you want to log in with, and the password.
  - Click on btn:[Detect Base DN], or enter the base DN manually.
  - Click on btn:[Test Base DN].
- If you fulfill all the requirements, you should get a green light and see the message: Configuration OK.
- Select menu:Users[] tab.
  - Select the objectclass for the users, for example user.
  - Click btn:[Verify settings and count users] near the bottom of the form. You will then see the number of users found.
- Select menu:Login Attributes[] tab.
  - A configuration appears; adjust it to your users configuration.
  - If required, adjust the login parameters additional login attributes.
  - You can check users with any of the allowed login options. You can adjust them or leave them the way they are.
- Select menu:Groups[] tab.
  - Select all the objectclasses for your groups, for example group. Then, verify your settings
- Select menu:Advanced[] tab.
  - Under "Configuration Settings":
- Configuration Active should be selected.
- Adjust the Cache TTL (time to live) value as required. However, ownCloud usually auto-selects the best settings for each AD configuration.
  - Under "Directory Settings"
- Check if the Group-Member association is Member (AD). This is important for the users being shown in their respective groups.
- Select Nested groups, if you have them.
- Select menu:Expert[] tab.
  - In the "Internal Username Attribute" field, we need to set CN for the users being shown with their unique name. If you leave that field empty, each user will get a unique UID as a string of numbers and letters.
  - At the bottom of the form, click both btn:[Clear Username-LDAP User Mapping] and btn:[Clear Groupname-LDAP Group Mapping], and then test your configuration by clicking btn:[Test Configuration].
- Navigate to our ownCloud User administration page and check if all your users are listed properly, and shown in the right groups.
- Go to the homepage of your ownCloud server and try to share something with one of your users

If everything is set up correctly, you now have an LDAP proxy server to your active directory that will reduce the network traffic by caching the searches your perform.

Cache Multiple Active Directory Servers

If you have more than one Active Directory Server that you want to cache, in /etc/ldap/slapd.conf add the following configuration instead, adjusting it as necessary. The ownCloud LDAP app settings are the same as in section 6.

```
# This an example of a config file:

# See slapd.conf(5)

# Global Directives:

# Schema and objectClass definitions
include         /etc/ldap/schema/core.schema
include         /etc/ldap/schema/cosine.schema
include         /etc/ldap/schema/nis.schema
include         /etc/ldap/schema/inetorgperson.schema
include     /etc/ldap/schema/misc.schema

# Where the pid file is put. The init.d script
# will not stop the server if you change this.
pidfile         /var/run/slapd/slapd.pid

# List of arguments that were passed to the server
argsfile        /var/run/slapd/slapd.args

# Read slapd.conf(5) for possible values
# Change loglevel to "any" if you want to see everything.
loglevel        none

# Where the dynamically loaded modules are stored
modulepath      /usr/lib/ldap

# Here are the recommended modules:

# module for meta-database
moduleload back_meta.la

# module for the target ldap-server
moduleload  back_ldap.la

# module for your local database
moduleload  back_hdb.la

# module for rewriting attributes
moduleload  rwm

# caching module
moduleload  pcache.la

# module to enable memberof in ldap
```

```
moduleload memberof.la

# The maximum number of entries that is returned for a search operation
sizelimit 500

# The tool-threads parameter sets the actual amount of cpu's that is used
# for indexing.
tool-threads 1

# If you want to save time and don't want to list all the refferals, set "yes"
norefs  yes

# Same as above
chase-referrals no

# See slapd-meta

# database type, for multiple ADS "meta" is required
database    meta

# now we create a local ldap tree
# in our tree we put the multiple ADS on different branches
# we need a suffix, an admin, and a password
suffix          "dc=owncloud,dc=com"
rootdn      "cn=Administrator,cn=Users,dc=example,dc=com"
rootpw "Password"

# now we specify our ADs
# First-AD
# uri <protocol>://[<host>]/<naming context>
uri             "ldap://first.ad.com:389/
cn=users,dc=first,dc=example,dc=com"

# here we need to set the virtual name to the real name
# the virtual name is a branch in our new created ldap tree
# suffixmassage <virtual naming context> <real naming context>
suffixmassage   "cn=users,dc=first,dc=example,dc=com"
"cn=users,dc=first,dc=ad,dc=com"

# authentication parameters
idassert-bind   bindmethod=simple
       binddn="cn=user01,cn=users,dc=first,dc=owncloud,dc=com"
credentials="Password01"

# Second-AD
uri             "ldaps://second.ad.com:636/cn=users,dc=second,dc=example,dc=com"
suffixmassage   "cn=users,dc=second,dc=example,dc=com"
"cn=users,dc=second,dc=ad,dc=com"
idassert-bind   bindmethod=simple
              binddn="cn=user02,cn=users,dc=second,dc=owncloud,dc=com"
```

```
        credentials="Password02"

# Now we start initialising the modules
# First the rewrite module
overlay       rwm

# Now we rewrite the attributes
rwm-map       attribute uid sAMAccountName
rwm-map       attribute dn distinguishedName

# Next one is optional, if you want memberof, for the groups,
# you have to load it.
overlay       memberof

# Now we load the caching module
overlay pcache

# The directive enables proxy caching
# See slapo-pcache

# pcache <database> <max_entries> <numattrsets> <entry_limit> <cc_period>
# Parameters:
#
# <database> for cached entries.
# <max_entries> when reached - cache replacement is invoked
# <numattrsets> = pcacheAttrset
# <entry_limit> limit to the number of entries returned
# <cc_period> Consistency check time to wait
pcache hdb 100000 3 1000 100

# pcachePersist { TRUE | FALSE }
# Write cached results into the database
# Results remain in database after restart
pcachePersist TRUE

# Where the database files are physically stored for database #1
directory      "/var/lib/ldap"

# Caching templates for general search

# pcacheAttrset <index> <attrs...>
# First set the index number
# Then set the attribute to cache
pcacheAttrset   0 1.1

# pcacheTemplate <template_string> <attrset_index> <ttl>
# First define the query sting to cache
# Then reference the Attrset
# Last set the time-to-live
pcacheTemplate  (&(|(objectClass=))) 0 3600
```

```
pcacheTemplate (objectClass=*) 0 3600

# User Name Field (Advanced Tab)
pcacheAttrset   1 displayname
pcacheTemplate (objectClass=*) 1 3600

# Group Field
pcacheAttrset   2 memberOf
pcacheTemplate (objectClass=*) 2 3600
```

Mimetypes Management

Introduction

ownCloud allows you to create aliases for mimetypes and map file extensions to a mimetype. These allow administrators the ability to change the existing icons that ownCloud uses to represent certain file types and folders, as well as to use custom icons for mimetypes and file extensions which ownCloud doesn't natively support. This is handy in a variety of situations, such as when you might want a custom audio icon for audio mimetypes, instead of the default file icon.

Mimetype Aliases

ownCloud's default mimetype configuration is defined in owncloud/resources/config/mimetypealiases.dist.json, which you can see a snippet of below. The mimetype's on the left, and the icon used to represent that mimetype is on the right.

```
{
 "application/coreldraw": "image",
 "application/font-sfnt": "image",
 "application/font-woff": "image",
 "application/illustrator": "image",
 "application/epub+zip": "text",
 "application/javascript": "text/code",
}
```

Stepping through that file, you can see that:

- the image icon is used to represent Corel Draw, SFNT and WOFF font files, and Adobe Illustrator files.
- ePub files are represented by the text file icon.
- JavaScript files are represented by the text/code icon.

Changing Existing Icons and Using Custom Icons

If you want to change one or more of the existing icons which ownCloud uses, or if you want to expand the available list, here's how to do so.

First, create a copy of resources/config/mimetypealiases.dist.json, naming it mimetypealiases.json and storing it in config/. This is required for two reasons:

1. It will take precedence over the default file.

2. The original file will get replaced on each ownCloud upgrade.

Then, either override one or more existing definitions or add new, custom, aliases as required.



Please refer to the ownCloud theming documentation for where to put the new image files.

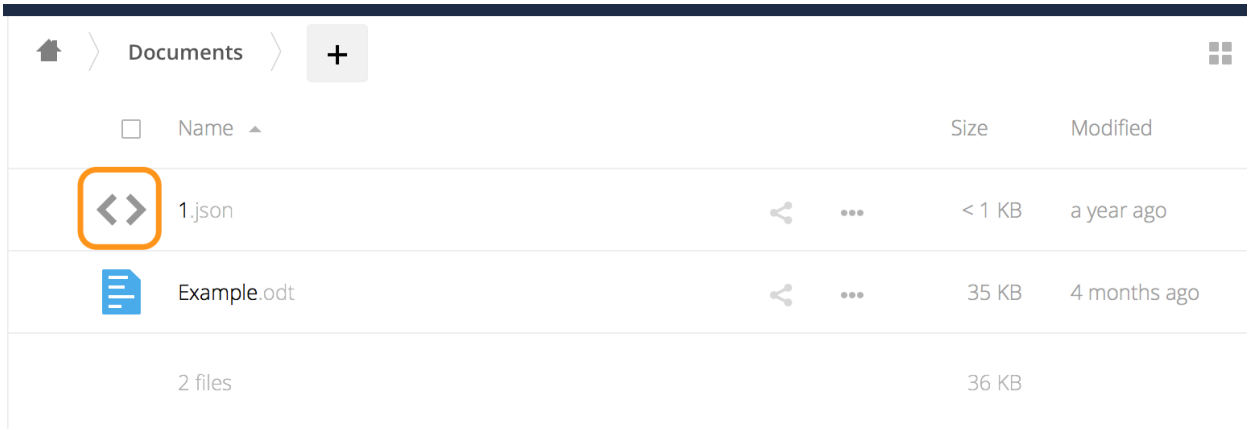
Some common mimetypes that may be useful in creating aliases are:

Mimetype	Description
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

Once you have made changes to config/mimetypealiases.json, use the occ command to propagate the changes throughout your ownCloud installation. Here is an example for Ubuntu Linux:

```
$ {occ-command-example-prefix} maintenance:mimetype:update-js
```

Example - Changing the JSON File Icon



Let's step through an example, from start to finish, of changing the icon that ownCloud uses to represent JSON files, which you can see above.

1. From the root directory of your ownCloud installation, copy resources/config/mimetypealiases.dist.json to /config/mimetypealiases.json.
2. Update the alias for application/json, which you should find on line 8, to match the following, and save the file:

```
"application/json": "text/json",
```

1. Copy a new SVG icon to represent JSON files to core/img/filetypes, calling it text-json.svg.

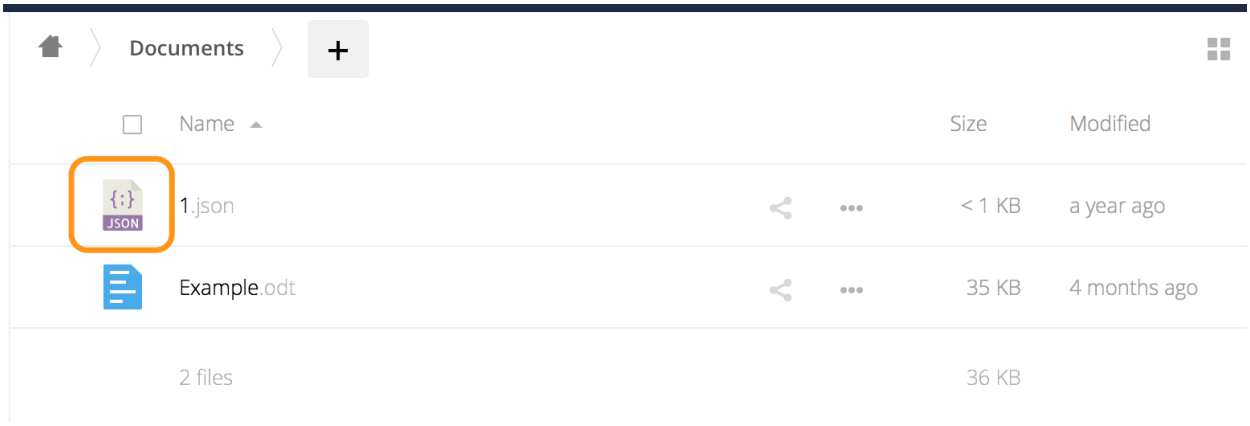


The name and location of the file are important. The location is because the core/img/filetypes directory stores the mimetype file icons. The name is important as it's a rough mapping between the alias name and the icon's file name, i.e., text/json becomes text-json.

1. Run the following command to update the mimetype alias database.

```
$ {occ-command-example-prefix} maintenance:mimetype:update-js
```

After doing so, whenever you view a folder that contains JSON files or upload one, your new icon file will be used to represent the file, as in the image below.



Mimetype Mapping

ownCloud allows administrators to map a file extension to a mimetype, e.g., such as mapping files ending in mp3 to audio/mpeg. Which then, in turn, allows ownCloud to show the audio icon.

The default file extension to mimetype mapping configuration is stored in resources/config/mimetypemapping.dist.json. This is similar to resources/config/mimetypealiases.dist.json, and also returns a basic JSON array.

```
{
 "3gp": ["video/3gpp"],
 "7z": ["application/x-7z-compressed"],
 "accdb": ["application/msaccess"],
 "ai": ["application/illustrator"],
 "apk": ["application/vnd.android.package-archive"],
 "arw": ["image/x-dcraw"],
 "avi": ["video/x-msvideo"],
 "bash": ["text/x-shellscript"],
 "json": ["application/json", "text/plain"],
}
```

In the example above, you can see nine mimetypes mapped to file extensions. Each of them, except the last (json), maps a file extension to a mimetype. Now take a look at the JSON example.

In this case, ownCloud will first check if a mimetype alias is defined for application/json, in mimetypealiases.json. If it is, it will use that icon. If not, then ownCloud will fall back to using the icon for text/plain.

If you want to update or extend the existing mapping, as with updating the mimetype aliases, create a copy of resources/config/mimetypemapping.dist.json and name it mimetypemapping.json and storing it in config/. Then, in this new file, make any changes required.



Please refer to the ownCloud theming documentation for where to put the new image files.

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 specialized mimetypes to fallback to generic icons when the relevant icons are unavailable.

Server Configuration

In this section you will find all the details you need to configure ownCloud.

Configuring the Activity App

Introduction

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

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

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

Enabling the Activity App

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

Configuring your ownCloud for the Activity App



A working e-mail configuration is required to configure your ownCloud to send out e-mail notifications. Furthermore, it is recommended to configure the background job Webcron or Cron.

Email notifications for shared files can be enabled/disabled by administrators with the "*Allow users to send mail notifications for shared files to other users*", available in menu:Settings[Admin > Sharing]. There is also a configuration option activity_expire_days available in your config.php which allows you to clean-up older activities from the database.

Background Jobs

Introduction

A system like ownCloud sometimes requires tasks to be done on a regular basis without requiring user interaction or hindering ownCloud's performance. For that reason, as a system administrator, you can configure background jobs (for example, database clean-ups) to be executed without any user interaction.

These jobs are typically referred to as Cron Jobs. Cron jobs are commands or shell-based scripts that are scheduled to periodically run at fixed times, dates, or intervals. To run Cron jobs with ownCloud, we recommend that you can use the occ system:cron command.

System

```
system
 system:cron          Execute background jobs as cron
```

```
{occ-command-example-prefix} -h system:cron
-Usage:
 system:cron [options]
```

Options

-p, --progress	Shows a progress bar - for use in manual execution. Do not use when executing from crontab

To execute background jobs using cron, you can use the system:cron command, as in the following example:

```
{occ-command-example-prefix} system:cron
```

If the --progress or -p argument is specified, then progress output will be displayed in the console, as in the example below.

```
Executing: 12 - OCA\UpdateNotification\Notification\BackgroundJob
 13 [------------>--------------]
```

If neither of these arguments is provided, no output will be displayed by the command.



Displaying progress information is useful when you want visual confirmation that background jobs have been executed. However, in a non-interactive environment, such as crontab, it should not be used.

Updating an Existing System Cron Configuration

If you have already automated background jobs via Cron, you must update the relevant crontab entry using the example below as a guide.

```
# Instead of the following configuration
/usr/bin/php -f /path/to/your/owncloud/cron.php

# Use the following one instead
{occ-command-example-prefix} system:cron
```

This command does not work if:

- Maintenance or Admin-only (single user) modes are enabled
- Background jobs are disabled

Cron Jobs

You can schedule Cron jobs in three ways: Cron, Webcron, or AJAX. These can all be configured in the admin settings menu. However, the recommended method is to use Cron. The following sections describe the differences between each method.

There are a number of things to keep in mind when choosing an automation option:

1. While the default method is AJAX, though the preferred way is to use Cron. The reason for this distinction is that AJAX is easier to get up and running. As a result, it makes sense (often times) to accept it in the interests of expediency. However, doing so is known to cause issues, such as backlogs and potentially not running every job on a heavily-loaded system. What's more, an increasing amount of ownCloud automation has been migrated from AJAX to Cron in recent versions. For this reason, we encourage you to not use it for too long — especially if your site is rapidly growing.
2. While Webcron is better than AJAX, it too has limitations. For example, running Webcron will only remove a single item from the job queue, not all of them. Cron, however, will clear the entire queue.

It's for this reason that we encourage you to use Cron — if at all possible.

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 which the web server might have.

For example, to run a Cron job on a *nix system every minute, under the default web server user (often, www-data or wwwrun) you must set up the following Cron job to call the occ system:cron command:

```
# crontab -u www-data -e
* * * * * /usr/bin/php /path/to/your/owncloud/occ system:cron
```

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

```
# crontab -u www-data -l
* * * * * /usr/bin/php /path/to/your/owncloud/occ system:cron
```



You have to make sure that PHP is found by Cron; hence why we've deliberately added the full path.

Please refer to the crontab man page for the exact command syntax if you don't want to have it run every minute.



There are other methods to invoke programs by the system regularly, e.g., systemd timers

Webcron

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

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

AJAX

The AJAX scheduling method is the default option. However, it is also the *least* reliable. Each time a user visits the ownCloud page, a single background job is executed. The advantage of this mechanism, however, 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.



Especially when using the Activity App or external storages, where new files are added, updated, or deleted one of the other methods should be used.

Parallel Task Execution

Regardless of the approach which you take, since ownCloud 9.1, Cron jobs can be run in parallel. This is done by running system:cron multiple times. Depending on the process which you're automating, this may not be necessary. However, for longer-running tasks, such as those which are LDAP related, it may be very beneficial.

There is no way to do so via the ownCloud UI. But, the most direct way to do so, is by opening three console tabs and in each one run /usr/bin/php /path/to/your/owncloud/occ system:cron. Each of these processes would acquire their own list of jobs to process without overlapping any other.

Available Background Jobs

A number of existing background jobs are available to be run just for specific tasks.



These jobs are generally only needed on large instances and can be run as background jobs. If the number of users in your installation ranges between 1,000 and 3,000, or if you're using LDAP and it becomes a bottleneck, then admins can delete several entries in the oc_jobs table and replace them with the corresponding occ command, which you can see here:

- OCA\\DAV\\CardDAV\\SyncJob → occ dav:sync-system-addressbook
- OCA\\Federation\\SyncJob → occ federation:sync-addressbooks
- OCA\\Files_Trashbin\\BackgroundJob\\ExpireTrash → occ trashbin:expire
- OCA\\Files_Versions\\BackgroundJob\\ExpireVersions → occ versions:expire

If used, these should be scheduled to run on a daily basis.

While not exhaustive, these include:

CleanupChunks

The CleanupChunks command, occ dav:cleanup-chunks, will clean up outdated chunks (uploaded files) more than a certain number of days old and needs to be added to your crontab.



There is no matching background job to delete from the oc_jobs table.

ExpireTrash

The ExpireTrash job, contained in OCA\Files_Trashbin\BackgroundJob\ExpireTrash, will remove any file in the ownCloud trash bin which is older than the specified maximum file retention time. It can be run, as follows, using the OCC trashbin command:

```
occ trashbin:expire
```

ExpireVersions

The ExpireVersions job, contained in OCA\Files_Versions\BackgroundJob\ExpireVersions, will expire versions of files which are older than the specified maximum version retention time. It can be run, as follows, using the OCC versions command:

```
occ versions:expire
```



Please take care when adding ExpireTrash and ExpireVersions as Cron jobs. Make sure that they're not started in parallel on multiple machines. Running in parallel on a single machine is fine. But, currently, there isn't sufficient locking in place to prevent them from conflicting with each other if running in parallel across multiple machines.

SyncJob (CardDAV)

The CardDAV SyncJob, contained in OCA\DAV\CardDAV\SyncJob, syncs the local system address book, updating any existing contacts, and deleting any expired contacts. It can be run, as follows, using the OCC dav command:

```
occ dav:sync-system-addressbook
```

SyncJob (Federation)

OCAFederationSyncJob

It can be run, as follows, using the OCC federation sync command:

```
occ federation:sync-addressbooks
```

Troubleshooting

Forbidden error for Scanner.php

If you find a **Forbidden** error message in your log files, with a reference to the Scanner.php file, then you should:

- Check if you have any shares with the status pending.
- Configure conditional logging for cron to see more output.

Memory Caching

Introduction

You can significantly improve ownCloud server performance by using memory caching. This is the process of storing frequently-requested objects in-memory for faster retrieval later. There are two types of memory caching available:

A PHP opcode Cache (OPcache): An opcode cache stores compiled PHP scripts so they don't need to be re-compiled every time they are called. These compiled PHP scripts are stored in-memory, on the server on which they're compiled.

A Data Cache: A data cache stores copies of *data*, *templates*, and other types of *information-based files*. Depending on the cache implementation, it can be either *local*, or specific, to one server, or *distributed* across multiple servers. This cache type is ideal when you have a scale-out installation.

Supported Caching Backends

The caching backends supported by ownCloud are:

- APCu: APCu 4.0.6 and up is required. Alternatively you can use the Zend OPCache. However, **it is not a data cache**, only an opcode cache.
- Redis: This is a distributed cache for multi-server ownCloud installations. Version 2.2.6 or higher of the PHP Redis extension is required.
- Memcached: This is a distributed cache for multi-server ownCloud installations.



You may use *both* a local and a distributed cache. The recommended ownCloud caches are APCu and Redis. If you do not install and enable a local memory cache you will see a warning on your ownCloud admin page. If you enable only a distributed cache in your config.php (memcache.distributed) and not a local cache (memcache.local) you will still see the cache warning.

Cache Directory Location

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

Quick Configuration Example

Install APCu and Redis:

```
apt install php-apcu php-redis redis-server
```

Enable them by adding this to your config.php:

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

Restart your web server and you are ready to go.

Cache Types

APCu

We recommend that you use APCu, because it is both a data cache *and* available in most Linux distributions.

Installing APCu

```
# On Ubuntu/Debian/Mint systems
apt install php-apcu
```

With that done, assuming that you don’t encounter any errors, restart Apache and the extension is ready to use.

Redis

Redis is an excellent modern memory cache to use for both distributed caching and as a local cache for transactional file locking, because it guarantees that cached objects are available for as long as they are needed.

The Redis PHP module must be at least version 2.2.6 or higher. 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 Installing Redis on other distributions.

Installing Redis

If you have Ubuntu 16.04 or higher:

```
apt install redis-server php-redis
```

The installer will automatically launch Redis and configure it to launch at startup.

After that, assuming that you don't encounter any errors, restart Apache and the extension is ready to use.

Additional notes for Redis vs. APCu on Memory Caching

APCu is faster at local caching than Redis. If you have enough memory, use APCu for memory caching and Redis for file locking. If you are low on memory, use Redis for both.

Clearing the Redis Cache

The Redis cache can be flushed from the command-line using the redis-cli tool, as in the following example:

```
sudo redis-cli
SELECT <dbIndex>
FLUSHDB
```

<dbIndex> is the number of Redis database where the cache is stored. It is zero by default at ownCloud. To check what yours is currently set to, check the dbindex value in config/config.php. Here's an example of what to look for:

```
'redis' => [
   'host' => 'localhost',      // For a Unix domain socket, use
'/var/run/redis/redis.sock'
   'port' => {std-port-redis},  // Set to 0 when using a Unix socket
   'timeout' => 0,
   'password' => '',           // Optional, if not defined no password will be used.
   'dbindex' => 0              // Optional, if undefined SELECT will not run and will
                          // use Redis Server's default DB Index.
],
```

Further Reading

- https://redis.io/commands/select
- https://redis.io/commands/flushdb

Memcached

Memcached is a reliable old-timer for shared caching on distributed servers. It performs well with ownCloud with one exception: it is not suitable to use with Transactional File Locking. This is because it does not store locks, and data can disappear from the cache at any time. Given that, Redis is the best memory cache to use.



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

Installing Memcached

On Ubuntu/Debian//Mint

On Ubuntu/Debian/Mint run the following command:

```
apt-get install memcached php5-memcached
```



The installer will automatically start memcached and configure it to launch at startup.

Configuration File Paths

PHP Version	Filename
{minimum-php-version}	/etc/opt/rh/rh-php{minimum-php-version-short-code}/php.d/25-memcached.ini

After that, assuming that you don't encounter any errors:

1. Restart your Web server
2. Add the appropriate entries to config.php (which you can find an example of below)
3. Refresh your ownCloud admin page

Clearing the Memcached Cache

The Memcached cache can be flushed from the command-line using a range of common Linux/Unix tools, including netcat and telnet. The following example uses telnet to login, run the flush_all command, and logout:

```
telnet localhost {std-port-memcache}
flush_all
quit
```

For more information see:

- https://github.com/memcached/memcached/wiki/Commands#flushall

Configuring Memory Caching

Memory caches must be explicitly configured in ownCloud by:

1. Installing and enabling your desired cache (whether that be the PHP extension and/or the caching server).
2. Adding the appropriate entry to ownCloud's config.php.

See config_sample_php_parameters for an overview of all possible config parameters. After installing and enabling your chosen memory cache, verify that it is active by viewing the PHP configuration details.

APCu Configuration

To use APCu, add this line to config.php:

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

With that done, refresh your ownCloud admin page, and the cache warning should disappear.

Redis Configuration

Regardless of whether you have setup Redis to use TCP or a Unix socket, we recommend adding the following for best performance:

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

Redis Configuration Using TCP

The following example config.php configuration connects to a Redis cache via TCP:

```
'memcache.local' => '\OC\Memcache\Redis',
'redis' => [
   'host' => 'localhost',
   'port' => {std-port-redis},
],
```

Redis Configuration Using Unix Sockets

If Redis is running on the same server as ownCloud, it is recommended to configure it to use Unix sockets. Then, configure ownCloud to communicate with Redis, as in the following example.

```
# Change the host value, based on the socket's location in your distribution
'memcache.local' => '\OC\Memcache\Redis',
'redis' => [
    'host' => '/var/run/redis/redis.sock',
    'port' => 0,
],
```

If setting up Redis to be accessed via a Unix socket from a webserver user, then consider the following:

1. Make the webserver user www-data member of the group redis in /etc/group, e.g., redis:x:110:www-data
2. In your Redis configuration (/etc/redis/redis.conf) set unixsocketperm to 770

To see a benchmark comparison, run:

```
sudo redis-benchmark -q -n 100000
sudo redis-benchmark -s /var/run/redis/redis-server.sock -q -n 100000
```

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

Memcached Configuration

Redis is very configurable; This example uses APCu for the local cache, Memcached as the distributed memory cache, and lists all the servers in the shared cache pool with their port numbers:

```
'memcache.local' => '\OC\Memcache\APCu',
'memcache.distributed' => '\OC\Memcache\Memcached',
'memcached_servers' => [
    ['localhost', {std-port-memcache}],
    ['server1.example.com', {std-port-memcache}],
    ['server2.example.com', {std-port-memcache}],
 ],
```

Configuration Recommendations Based on Type of Deployment

Small/Private Home Server

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

Small Organization, Single-server Setup

Use APCu for local caching, Redis for file locking

```
'memcache.local' => '\OC\Memcache\APCu',
'memcache.locking' => '\OC\Memcache\Redis',
'redis' => [
   'host' => 'localhost',
   'port' => {std-port-redis},
],
```

Large Organization, Clustered Setup

Use Redis for everything except a local memory cache. Use the server's IP address or hostname so that it is accessible to other hosts:

```
'memcache.distributed' => '\OC\Memcache\Redis',
'memcache.locking' => '\OC\Memcache\Redis',
'memcache.local' => '\OC\Memcache\APCu',
'redis' => [
   'host' => 'server1',      // hostname example
   'host' => '12.34.56.78',  // IP address example
   'port' => {std-port-redis},
],
```

Configuring Transactional File Locking

Transactional File Locking prevents simultaneous file saving. It is enabled by default and uses the database to store the locking data. This places a significant load on your

database. It is recommended to use a cache backend instead. You have to configure it in config.php as in the following example, which uses Redis as the cache backend:

```
'filelocking.enabled' => true,
'memcache.locking' => '\OC\Memcache\Redis',
'redis' => [
    'host' => 'localhost',
    'port' => {std-port-redis},
    'timeout' => 0.0,
    'password' => '', // Optional, if not defined no password will be used.
 ],
```



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

Caching Exceptions

If ownCloud is configured to use either Memcached or Redis as a memory cache, please be aware that you may encounter issues with functionality. When these occur, it is usually a result of PHP being incorrectly configured, or the relevant PHP extension not being available.

In the table below, you can see all of the known reasons for reduced or broken functionality related to caching.

Setup/Configuration	Result
If file locking is enabled, but the locking cache class is missing, then an exception will appear in the web UI	The application will not be usable
If file locking is enabled and the locking cache is configured, but the PHP module missing.	There will be a white page/exception in web UI. It will be a full page issue, and the application will not be usable
All enabled, but the Redis server is not running	The application will be usable. But any file operation will return a "500 Redis went away" exception
If Memcache is configured for local and distributed, but the class is missing	There will be a white page and an exception written to the logs, This is because autoloading needs the missing class. So there is no way to show a page

Config.php Parameters

Introduction

ownCloud uses the config/config.php file to control server operations. config/config.sample.php lists all the configurable parameters within ownCloud, along with example or default values. This document provides a more detailed reference. Most options are configurable on your Admin page, so it is usually not necessary to edit config/config.php.



The installer creates a configuration containing the essential parameters.
Only manually add configuration parameters to config/config.php if you need to use a special value for a parameter. **Do not copy everything from** config/config.sample.php **. Only enter the parameters you wish to modify!**

ownCloud supports loading configuration parameters from multiple files. You can add arbitrary files ending with .config.php in the config/ directory, for example you could place your email server configuration in email.config.php. This allows you to easily create and manage custom configurations, or to divide a large complex configuration file into a set of smaller files. These custom files are not overwritten by ownCloud, and the values in these files take precedence over config.php.

Default Parameters

These parameters are configured by the ownCloud installer and are required for your ownCloud server to operate.

Unique identifier for your ownCloud installation

This unique identifier is 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 ownCloud. Needs to start with a letter.

'instanceid' ⇒ 'd3c944a9a',

Code Sample

```
'instanceid' => '',
```

Auto-generated salt used to hash all passwords

The salt used to hash all passwords and is auto-generated by the ownCloud 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.

Code Sample

```
'passwordsalt' => '',
```

Define 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. Please consider that for backend processes like background jobs or occ commands, the URL parameter in key overwrite.cli.url is used. For more details, please see that key.

Code Sample

```
'trusted_domains' => [
   'demo.example.org',
   'otherdomain.example.org',
 ],
```

Define global list of CORS domains

All users can use tools running CORS (Cross-Origin Resource Sharing) requests from the listed domains.

Code Sample

```
'cors.allowed-domains' => [
   'https://foo.example.org',
 ],
```

Define the directory where user files are stored

This defaults to data/ in the ownCloud directory.

The SQLite database is also stored here, when you use SQLite. (SQLite is not available in ownCloud Enterprise Edition)

Code Sample

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

Define the directory where the crash logs will be stored

By default, this will be the same as the one configured as "datadirectory".

The directory MUST EXIST and be WRITABLE by the web server. Note that crashes are extremely rare (although they can come in burst due to multiple requests), so the default location is usually fine. Also note that the log can contain sensitive information, but it should be useful to pinpoint where is the problem.

Code Sample

```
'crashdirectory' => '/var/www/owncloud/data',
```

Current version number of your ownCloud installation

This is set up during installation and update, so you shouldn’t need to change it.

Code Sample

```
'version' => '',
```

Show or hide the ownCloud version information in status.php

This hardens an ownCloud instance by hiding the version information in status.php.

This can be a legitimate step. Please consult the documentation before enabling this.

Code Sample

```
'version.hide' => false,
```

Show or hide the server hostname in status.php

Optional config option, defaults to hidden.

Code Sample

```
'show_server_hostname' => false,
```

Show the short hostname in status.php

Optional config option, defaults to use the gethostname() return value.

Code Sample

```
'use_relative_domain_name' => false,
```

Identify the database used with this installation

See also config option supportedDatabases

Available: - sqlite (SQLite3 - Not in Enterprise Edition) - mysql (MySQL/MariaDB) - pgsql (PostgreSQL) - oci (Oracle - Enterprise Edition Only)

Code Sample

```
'dbtype' => 'mysql',
```

Define the database server host name

For example localhost, hostname, hostname.example.com, or the IP address.

To specify a port use: hostname:##; To specify a Unix socket use: localhost:/path/to/socket.

Code Sample

```
'dbhost' => '',
```

Define the ownCloud database name

The name of the ownCloud database which is set during installation.

You should not need to change this.

Code Sample

```
'dbname' => 'owncloud',
```

Define the ownCloud database user

This must be unique across ownCloud instances using the same SQL database.

This is setup during installation, so you shouldn't need to change it.

Code Sample

```
'dbuser' => '',
```

Define the password for the database user

This is set up during installation, so you shouldn't need to change it.

Code Sample

```
'dbpassword' => '',
```

Define the prefix for the ownCloud tables in the database

Code Sample

```
'dbtableprefix' => '',
```

Indicate whether the ownCloud instance was installed successfully

true indicates a successful installation, false indicates an unsuccessful installation.

Code Sample

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

User Experience

These optional parameters control some aspects of the user interface. Default values, where present, are shown.

Define the default language of your ownCloud instance

Using ISO_639-1 language codes such as en for English, de for German, and fr for French.

Overrides automatic language detection on public pages like login or shared items. User's language preferences configured under personal → language override this setting after they have logged in.

Code Sample

```
'default_language' => 'en_GB',
```

Define the default app to open on user login

Use the app names as they appear in the URL after clicking them in the Apps menu, such as files, documents or calendar etc. You can use a comma-separated list of app names, so if the first app is not enabled for a user then ownCloud will try the second one, and so on. If no enabled apps are found it defaults to the Files app.

Code Sample

```
'defaultapp' => 'files',
```

Enable or disable avatars or user profile photos

true enables avatars, or user profile photos, false disables them.

These appear on the User page, on user's Personal pages and are used by some apps (contacts, mail, etc).

Code Sample

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

Allow or disallow users to change their display names

true allows users to change their display names (on their Personal pages), false prevents them from changing their display names.

Code Sample

```
'allow_user_to_change_display_name' => true,
```

Define the lifetime of the remember login cookie

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

Code Sample

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

Define the lifetime of a session after inactivity

The default is 20 minutes, expressed in seconds.

Code Sample

```
'session_lifetime' => 60 * 20,
```

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.

Code Sample

```
'session_keepalive' => true,
```

Enforce token only authentication for apps and clients connecting to ownCloud

If enabled, all access requests using the users password are blocked for enhanced security.

Users have to generate special app-passwords (tokens) for their apps or clients in their personal settings which are further used for app or client authentication. Browser logon is not affected.

Code Sample

```
'token_auth_enforced' => false,
```

Define additional login buttons on the logon screen

Provides the ability to create additional login buttons on the logon screen, for e.g., SSO integration 'login.alternatives' ⇒ [['href' ⇒ 'https://www.testshib.org/Shibboleth.sso/ProtectNetwork?target=https%3A%2F%2Fmy.owncloud.tld%2Flogin%2Fsso-saml%2F', 'name' ⇒ 'ProtectNetwork', 'img' ⇒ '/img/PN_sign-in.gif'], ['href' ⇒ 'https://www.testshib.org/Shibboleth.sso/OpenIdP.org?target=https%3A%2F%2Fmy.owncloud.tld%2Flogin%2Fsso-saml%2F', 'name' ⇒ 'OpenIdP.org', 'img' ⇒ '/img/openidp.png'],]

Code Sample

```
'login.alternatives' => [],
```

Enable or disable ownCloud's built-in CSRF protection mechanism

In some specific setups CSRF protection is handled in the environment, e.g., running F5 ASM. In these cases the built-in mechanism is not needed and can be disabled. Generally speaking, however, this config switch should be left unchanged.



leave this as is if you're not sure what it does.

Code Sample

```
'csrf.disabled' => false,
```

Define the directory where the skeleton files are located

These files will be copied to the data directory of new users.

Leave this directory empty if you do not want to copy any skeleton files. A valid path must be given for this key otherwise errors will be generated in owncloud.log.

Code Sample

```
'skeletondirectory' => '/path/to/owncloud/core/skeleton',
```

Define the user_backends app

Those need to be enabled first and allow you to configure alternate authentication backends.

Supported backends are: IMAP (OC_User_IMAP), SMB (OC_User_SMB), and FTP (OC_User_FTP).

Code Sample

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

Define a custom link to reset passwords

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.

If you do not want to provide any link, replace the URL with 'disabled'.

Code Sample

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

Allow medial search on user account properties

These account properties can be display name, user id, email, and other search terms.

Allows finding 'Alice' when searching for 'lic'. May slow down user search. Disable this if you encounter slow username search in the sharing dialog.

Code Sample

```
'accounts.enable_medial_search' => true,
```

Allow medial search on the group id

Allows finding 'test' in groups when searching for 'es'.

This is only used in the DB group backend (local groups). This won't be used against LDAP, Shibboleth or any other group backend.

Code Sample

```
'groups.enable_medial_search' => true,
```

Define minimum characters entered before a search returns results

Defines the minimum characters entered before a search returns results for users or groups in the share autocomplete form. Lower values increase search time especially for large backends.

Any exact matches to a user or group will be returned, even though less than the minimum characters have been entered. The search is case insensitive. For example, entering "tom" will always return "Tom" if there is an exact match.

Code Sample

```
'user.search_min_length' => 2,
```

Mail Parameters

These configure the email settings for ownCloud notifications and password resets.

Define the email RETURN address

The return address that you want to appear on emails sent by the ownCloud server.

Example: oc-admin@example.com, substituting your own domain, of course.

Code Sample

```
'mail_domain' => 'example.com',
```

Define the email FROM address

FROM address that overrides the built-in sharing-noreply and lostpassword-noreply FROM addresses.

Code Sample

```
'mail_from_address' => 'owncloud',
```

Enable or disable SMTP class debugging

Code Sample

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

Define the mode for sending an email

Modes to use for sending mail: sendmail, smtp, qmail or php.

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

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

For the sendmail option you need an installed and working email system on the server, with /usr/sbin/sendmail installed on your Unix system.

For qmail the binary is /var/qmail/bin/sendmail, and it must be installed on your Unix system.

Code Sample

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

Define the IP address of your mail server host

Depends on mail_smtpmode. May contain multiple hosts separated by a semi-colon.

If you need to specify the port number, append it to the IP address separated by a colon, like this: 127.0.0.1:24.

Code Sample

```
'mail_smtphost' => '127.0.0.1',
```

Define the port for sending an email

Depends on mail_smtpmode.

Code Sample

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

Define the SMTP server timeout

Depends on mail_smtpmode. Sets the SMTP server timeout in seconds.

You may need to increase this if you are running an anti-malware or spam scanner.

Code Sample

```
'mail_smtptimeout' => 10,
```

Define the SMTP security style

Depends on mail_smtpmode. Specify when you are using ssl or tls.

Leave empty for no encryption.

Code Sample

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

Define the SMTP authentication

Depends on mail_smtpmode. Change this to true if your mail server requires authentication.

Code Sample

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

Define the SMTP authentication type

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

Code Sample

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

Define the SMTP authentication username

Depends on mail_smtpauth. Specify the username for authenticating to the SMTP server.

Code Sample

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

Define the SMTP authentication password

Depends on mail_smtpauth. Specify the password for authenticating to the SMTP server.

Code Sample

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

Proxy Configurations

Override automatic proxy detection

The automatic hostname detection of ownCloud 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.

Code Sample

```
'overwritehost' => '',
```

Override protocol (http/https) usage

When generating URLs, ownCloud attempts to detect whether the server is accessed via https or http. However, if ownCloud is behind a proxy and the proxy handles the https calls, ownCloud would not know that ssl is in use, which would result in incorrect URLs being generated.

Valid values are http and https.

Code Sample

```
'overwriteprotocol' => '',
```

Override ownClouds webroot

ownCloud attempts to detect the webroot for generating URLs automatically.

For example, if www.example.com/owncloud is the URL pointing to the ownCloud instance, the webroot is /owncloud. When proxies are in use, it may be difficult for ownCloud to detect this parameter, resulting in invalid URLs.

Code Sample

```
'overwritewebroot' => '',
```

Override condition for the remote IP address with a regular expression

This option allows you to define a manual override condition as a regular expression for the remote IP address. The keys overwritewebroot, overwriteprotocol, and overwritehost are subject to this condition.

For example, defining a range of IP addresses starting with 10.0.0. and ending with 1 to 3: * ^10\.0\.0\.[1-3]$

Code Sample

```
'overwritecondaddr' => '',
```

Override cli URL

Use this configuration parameter to specify the base URL for any URLs which are generated within ownCloud using any kind of command line tools (cron or occ).

The value should contain the full base URL: https://www.example.com/owncloud As an example, alerts shown in the browser to upgrade an app are triggered by a cron background process and therefore uses the url of this key, even if the user has logged on via a different domain defined in key trusted_domains. When the user clicks an alert like this, he will be redirected to that URL and must logon again.

Code Sample

```
'overwrite.cli.url' => '',
```

Define the Phoenix base URL

If phoenix.baseUrl is set, public and private links will be redirected to this url.

Phoenix will handle these links accordingly.

As an example, in case 'phoenix.baseUrl' is set to 'http://phoenix.example.com', the shared link 'http://ocx.example.com/index.php/s/THoQjwYYMJvXMdW' will be redirected by ownCloud to 'http://phoenix.example.com/index.html#/s/THoQjwYYMJvXMdW'.

Code Sample

```
'phoenix.baseUrl' => '',
```

Define clean URLs without /index.php

This parameter will be written as RewriteBase on update and installation of ownCloud to your .htaccess file. While this value is often simply the URL path of the ownCloud 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 ownCloud is accessible at. So if ownCloud is accessible via https://mycloud.org/owncloud the correct value would most likely be /owncloud. If ownCloud 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-command-example-prefix} maintenance:update:htaccess. Now, when the following conditions are met ownCloud URLs won't contain index.php:

- mod_rewrite is installed
- mod_env is installed

Code Sample

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

Define the URL of your proxy server

Example: proxy.example.com:8081.

Code Sample

```
'proxy' => '',
```

Define proxy authentication

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

The format is: username:password.

The username and the password need to be urlencoded to avoid breaking the delimiter syntax "username:password@hostname:port

Example: usern@me needs to be encoded as usern%40ame.

Code Sample

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

Deleted Items (trash bin)

These parameters control the Deleted files app.

Define the trashbin retention obligation

If the trash bin app is enabled (default), this setting defines the policy for when files and folders in the trash bin will be permanently deleted.

The app allows for two settings, a minimum time for trash bin retention, and a maximum time for trash bin retention. Minimum time is the number of days a file will be kept, after which it may be deleted. Maximum time is the number of days at which it is guaranteed to be deleted. Both minimum and maximum times can be set together to explicitly define file and folder deletion. For migration purposes, this setting is installed initially set to auto, which is equivalent to the default setting in ownCloud 8.1 and before.

Available values:

- auto default setting. Keeps files and folders in the deleted files for up to 30 days, automatically deleting them (at any time) if space is needed. Note: files may not be removed if space is not required.
- D, auto keeps files and folders in the trash bin for D+ days, delete anytime if space needed (Note: files may not be deleted if space is not needed)
- auto, D delete all files in the trash bin that are older than D days automatically, delete other files anytime if space needed
- D1, D2 keep files and folders in the trash bin for at least D1 days and delete when exceeds D2 days
- disabled trash bin auto clean disabled, files and folders will be kept forever

Code Sample

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

Define the trashbin purge limit

This setting defines the percentage of free space occupied by deleted files that triggers auto purging of deleted files for this user

Code Sample

```
'trashbin_purge_limit' => 50,
```

File versions

These parameters control the Versions app.

Define the files versions retention obligation

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 ownCloud 8.1 and before.

Available values:

- auto default setting. Automatically expire versions according to expire rules. Please refer to https://doc.owncloud.org/server/latestadmin_manual/configuration/files/file_versioning.html for more information.
- D, auto keep versions at least for D days, apply expire rules to all versions that are older than D days
- auto, D delete all versions that are older than D days automatically, delete other versions according to expire rules
- D1, D2 keep versions for at least D1 days and delete when exceeds D2 days
- disabled versions auto clean disabled, versions will be kept forever

Code Sample

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

ownCloud Verifications

ownCloud performs several verification checks. There are two options, true and false.

Enable or disable updatechecker

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

This option is only applicable to ownCloud core. It is not applicable to app updates.

Code Sample

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

Define the updatechecker URL

URL that ownCloud should use to look for updates

Code Sample

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

Check for an internet connection

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

Code Sample

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

Check for a .well-known setup

Allows ownCloud to verify a working .well-known URL redirect.

This is done by attempting to make a request from JS to https://your-domain.com/.well-known/caldav/

Code Sample

```
'check_for_working_wellknown_setup' => true,
```

Define if config.php is read only

In certain environments it is desired to have a read-only configuration file.

When this switch is set to true ownCloud will not verify whether the configuration is writable. However, it will not be possible to configure all options via the Web interface. Furthermore, when updating ownCloud it is required to make the configuration file writable again for the update process.

Code Sample

```
'config_is_read_only' => false,
```

Define ownCloud operation modes

This defines the mode of operations. The default value is 'single-instance' which means that ownCloud is running on a single node, which might be the most common operations mode. The only other possible value for now is 'clustered-instance' which means that ownCloud is running on at least 2 nodes. The mode of operations has various impact on the behavior of ownCloud.

Code Sample

```
'operation.mode' => 'single-instance',
```

Logging

These parameters configure the logging options. For additional information or advanced configuration, please see the logging section in the documentation.

Define the log type

By default the ownCloud logs are sent to the owncloud.log file in the default ownCloud data directory.

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

Code Sample

```
'log_type' => 'owncloud',
```

Define the log path

Log file path for the ownCloud logging type.

Defaults to [datadirectory]/owncloud.log

Code Sample

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

Define the log level

Loglevel to start logging at. Valid values are: 0 = Debug, 1 = Info, 2 = Warning, 3 = Error, and 4 = Fatal. The default value is Warning.

Code Sample

```
'loglevel' => 2,
```

Define the syslog tag

If you maintain different instances and aggregate the logs, you may want to distinguish between them. syslog_tag can be set per instance with a unique id. Only available if log_type is set to syslog.

The default value is ownCloud.

Code Sample

```
'syslog_tag' => 'ownCloud',
```

Define the syslog format

The syslog format can be changed to remove or add information.

In addition to the %replacements% below %level% can be used, but it is used as a dedicated parameter to the syslog logging facility anyway.

Code Sample

```
'log.syslog.format' =>
'[%reqId%][%remoteAddr%][%user%][%app%][%method%][%url%] %message%',
```

Define log conditions

Log condition for log level increase based on conditions. Once one of these conditions is met, the required log level is set to debug. This allows to debug specific requests, users or apps

Supported conditions: - shared_secret: If a request parameter with the name log_secret is set to this value the condition is met - users: If the current request is done by one of the specified users, this condition is met - apps: If the log message is invoked by one of the specified apps, this condition is met - logfile: The log message invoked by the specified apps get redirected to this logfile, this condition is met Note: Not applicable when using syslog

Defaults to an empty array

Code Sample

```
'log.conditions' => [
   [
      'shared_secret' => '57b58edb6637fe3059b3595cf9c41b9',
      'users' => ['user1'],
      'apps' => ['files_texteditor'],
      'logfile' => '/tmp/test.log'
   ],
   [
      'shared_secret' => '57b58edb6637fe3059b3595cf9c41b9',
      'users' => ['user1'],
      'apps' => ['files_mediaviewer'],
      'logfile' => '/tmp/mediaviewer.log'
   ],
 ],
```

Define the log date format

This uses PHP.date formatting; see http://php.net/manual/en/function.date.php

Code Sample

```
'logdateformat' => 'F d, Y H:i:s',
```

Define the log timezone

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

Code Sample

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

Define logging if Cron ran successfully

Log successful cron runs.

Code Sample

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

Define the maximum log rotation file size

Enables log rotation and limits the total size of the logfiles.

The default is 0 or false which disables log 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 the defined limit. If a rotated log file is already present, it will be overwritten. If enabled, only the active log file and one rotated file are stored.

Code Sample

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

Alternate Code Locations

Some of the ownCloud code may be stored in alternate locations.

Define alternative app directories

If you want to store apps in a custom directory instead of ownCloud's default /apps, you need to modify the apps_paths key. There, you need to add a new associative array that contains three elements. These are:

- path The absolute file system path to the custom app folder.
- url The request path to that folder relative to the ownCloud web root, prefixed with /.
- writable Whether users can install apps in that folder. After the configuration is added, new apps will only install in a directory where writable is set to true.

The configuration example shows how to add a second directory, called /apps-external. Here, new apps and updates are only written to the /apps-external directory. This eases upgrade procedures of owncloud where shipped apps are delivered to apps/ by default. OC::$SERVERROOT points to the web root of your instance. Please see the Apps Management description on how to move custom apps properly.

Code Sample

```
'apps_paths' => [
  0 =>
    [
      'path' => OC::$SERVERROOT.'/apps',
      'url' => '/apps',
      'writable' => false,
    ],
  1 =>
    [
      'path' => OC::$SERVERROOT.'/apps-external',
      'url' => '/apps-external',
      'writable' => true,
    ],
],
```

Previews

ownCloud supports previews of image files, the covers of MP3 files, and text files. These options control enabling and disabling previews, and thumbnail size.

Enable preview generation

By default, ownCloud can generate previews for the following filetypes:

- Image files
- Covers of MP3 files
- Text documents

Valid values are true, to enable previews, or false, to disable previews

Code Sample

```
'enable_previews' => true,
```

Define the preview path

Location of the thumbnails folder, defaults to data/$user/thumbnails where $user is the current user. When specified, the format will change to $previews_path/$user where $previews_path is the configured previews base directory and $user will be substituted with the user id automatically.

For example if previews_path is /var/cache/owncloud/thumbnails then for a logged in user user1 the thumbnail path will be /var/cache/owncloud/thumbnails/user1.

Code Sample

```
'previews_path' => '',
```

Define the maximum x-axis width for previews

The maximum width, in pixels, of a preview.

A value of null means there is no limit.

Code Sample

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

Define the maximum y-axis width for previews

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

Code Sample

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

Define the maximum preview scale factor

If a lot of small pictures are stored on the ownCloud instance and the preview system generates blurry previews, you might want to consider setting a maximum scale factor. By default, pictures are upscaled to 10 times the original size. A value of 1 or null disables scaling.

Code Sample

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

Define the maximum preview filesize limit

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

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

Code Sample

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

Define the custom path for the LibreOffice / OpenOffice binary

Code Sample

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

Define additional arguments for LibreOffice / OpenOffice

Use this setting if LibreOffice/OpenOffice requires additional arguments.

Code Sample

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

Define preview providers

Only register providers that have been explicitly enabled

The following providers are enabled by default:

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

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

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

The following providers are not available in Microsoft Windows:

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

Code Sample

```
'enabledPreviewProviders' => [
   'OC\Preview\PNG',
   'OC\Preview\JPEG',
   'OC\Preview\GIF',
   'OC\Preview\BMP',
   'OC\Preview\XBitmap',
   'OC\Preview\MP3',
   'OC\Preview\TXT',
   'OC\Preview\MarkDown'
 ],
```

Comments

Global settings for the Comments infrastructure

Define an alternative Comments Manager

Replaces the default Comments Manager Factory. This can be utilized if an own or 3rdParty CommentsManager should be used that – for instance – uses the filesystem instead of the database to keep the comments.

Code Sample

```
'comments.managerFactory' => '\OC\Comments\ManagerFactory',
```

Define an alternative System Tags Manager

Replaces the default System Tags Manager Factory. This can be utilized if an own or 3rdParty SystemTagsManager should be used that – for instance – uses the filesystem instead of the database to keep the tags.

Code Sample

```
'systemtags.managerFactory' => '\OC\SystemTag\ManagerFactory',
```

Maintenance

These options are for halting user activity when you are performing server maintenance.

Enable maintenance mode to disable ownCloud

If you want to prevent users from logging in to ownCloud 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 ownCloud instantly.

Code Sample

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

Enable or disable single user mode

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

Code Sample

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

SSL

Extra SSL options to be used for configuration

Code Sample

```
'openssl' => [
   'config' => '/absolute/location/of/openssl.cnf',
 ],
```

Allow the configuration of system wide trusted certificates

Code Sample

```
'enable_certificate_management' => false,
```

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.

Memory caching backend for locally stored data

- Used for host-specific data, e.g. file paths

Code Sample

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

Memory caching backend for distributed data

- Used for installation-specific data, e.g. database caching
- If unset, defaults to the value of memcache.local

Code Sample

```
'memcache.distributed' => '\OC\Memcache\Memcached',
```

Define Redis connection details

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.

Code Sample

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

Define Redis Cluster connection details

Only for use with Redis Clustering, for Sentinel-based setups use the single server configuration above, and perform HA on the hostname.

Redis Cluster support requires the php module phpredis in version 3.0.0 or higher.

Available failover modes: - \RedisCluster::FAILOVER_NONE - only send commands to master nodes (default) - \RedisCluster::FAILOVER_ERROR - failover to slaves for read commands if master is unavailable - \RedisCluster::FAILOVER_DISTRIBUTE - randomly distribute read commands across master and slaves

Code Sample

```
'redis.cluster' => [
   'seeds' => [ // provide some/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_DISTRIBUTE
  ],
```

Define server details for memcached servers to use for memory caching

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

Code Sample

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

Define connection options for memcached

For more details please see http://apprize.info/php/scaling/15.html

Code Sample

```
'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,
  ],
```

Define the location of the cache folder

The 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.

Code Sample

```
'cache_path' => '',
```

Define the TTL for garbage collection

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 ownCloud Client as upload isn't completed within one day.

Code Sample

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

Define the DAV chunk base directory

Location of the chunk folder, defaults to data/$user/uploads where $user is the current user. When specified, the format will change to $dav.chunk_base_dir/$user where $dav.chunk_base_dir is the configured cache directory and $user is the user.

Code Sample

```
'dav.chunk_base_dir' => '',
```

Sharing

Global settings for Sharing

Define an alternative Share Provider

Replaces the default Share Provider Factory. This can be utilized if own or 3rdParty Share Providers are used that – for instance – use the filesystem instead of the database to keep the share information.

Code Sample

```
'sharing.managerFactory' => '\OC\Share20\ProviderFactory',
```

Allow schema fallback for federated sharing servers

When talking with federated sharing server, allow falling back to HTTP instead of hard forcing HTTPS

Code Sample

```
'sharing.federation.allowHttpFallback' => false,
```

All other configuration options

Define additional database driver options

Additional driver options for the database connection, eg. to enable SSL encryption in MySQL or specify a custom wait timeout on a cheap hoster.

Code Sample

```
'dbdriveroptions' => [
    PDO::MYSQL_ATTR_SSL_CA => '/file/path/to/ca_cert.pem',
    PDO::MYSQL_ATTR_INIT_COMMAND => 'SET wait_timeout = 28800'
  ],
```

Define sqlite3 journal mode

sqlite3 journal mode can be specified using this configuration parameter - can be 'WAL' or 'DELETE' see for more details https://www.sqlite.org/wal.html

Code Sample

```
'sqlite.journal_mode' => 'DELETE',
```

Define MySQL 3/4 byte character handling

During setup, if requirements are met (see below), this setting is set to true and MySQL can handle 4 byte characters instead of 3 byte characters.

If you want to convert an existing 3-byte setup into a 4-byte setup please set the parameters in MySQL as mentioned below and run the migration command: {occ-command-example-prefix} db:convert-mysql-charset The config setting will be set automatically after a successful run.

Consult the documentation for more details.

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

```
[mysqld]
innodb_large_prefix=ON
innodb_file_format=Barracuda
innodb_file_per_table=ON
```

Tables will be created with * character set: utf8mb4 * collation: utf8mb4_bin * row_format: compressed

See: https://dev.mysql.com/doc/refman/5.7/en/charset-unicode-utf8mb4.html https://dev.mysql.com/doc/refman/5.7/en/innodb-parameters.html#sysvar_innodb_large_prefix https://mariadb.com/kb/en/mariadb/xtradbinnodb-server-system-variables/#innodb_large_prefix http://www.tocker.ca/benchmarking-innodb-page-compression-performance.html http://mechanics.flite.com/blog/2014/07/29/using-innodb-large-prefix-to-avoid-error-1071/

Code Sample

```
'mysql.utf8mb4' => false,
```

Define supported database types

Database types that are supported for installation.

Available: - sqlite (SQLite3 - Not in Enterprise Edition) - mysql (MySQL) - pgsql

(PostgreSQL) - oci (Oracle - Enterprise Edition Only)

Code Sample

```
'supportedDatabases' => [
    'sqlite',
    'mysql',
    'pgsql',
    'oci',
  ],
```

Define the location for temporary files

Override where ownCloud stores temporary files. Useful in situations where the system temporary directory is on a limited space ramdisk or is otherwise restricted, or if external storages which do not support streaming are in use.

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

Code Sample

```
'tempdirectory' => '/tmp/owncloudtemp',
```

Define the hashing cost

The hashing cost used by hashes generated by ownCloud.

Using a higher value requires more time and CPU power to calculate the hashes. As this number grows, the amount of work (typically CPU time or memory) necessary to compute the hash increases exponentially.

Code Sample

```
'hashingCost' => 10,
```

Define blacklisted files

Blacklist a specific file or files and disallow the upload of files with this name. .htaccess is blocked by default.



USE THIS ONLY IF YOU KNOW WHAT YOU ARE DOING.

Code Sample

```
'blacklisted_files' => [
    '.htaccess'
  ],
```

Define blacklisted files regular expression(s)

Blacklist files that match any of the given regular expressions and disallow the upload of those files. The matching is case-insensitive.



USE THIS ONLY IF YOU KNOW WHAT YOU ARE DOING.

Code Sample

```
'blacklisted_files_regex' => [
    '.*\.ext',
    '^somefilename.*'
  ],
```

Define excluded directories

Exclude specific directory names and disallow scanning, creating and renaming using these names. The matching is case insensitive.

Excluded directory names are queried at any path part like at the beginning, in the middle or at the end and will not be further processed if found. Please see the documentation for details and examples. Use when the storage backend supports, e.g. snapshot directories to be excluded.



USE THIS ONLY IF YOU KNOW WHAT YOU ARE DOING.

Code Sample

```
'excluded_directories' => [
    '.snapshot',
    '~snapshot',
  ],
```

Define excluded directories regular expression(s)

Exclude directory names that match any of the given regular expressions and disallow scanning, creating and renaming using these names. The matching is case insensitive.

Excluded directory names are queried at any path part like at the beginning, in the middle or at the end and will not be further processed if found. Please see the documentation for details and examples. Use when the storage backend supports, e.g. snapshot directories to be excluded.



USE THIS ONLY IF YOU KNOW WHAT YOU ARE DOING.

Code Sample

```
'excluded_directories_regex' => [
    '^backup.*',
    '.*backup$',
  ],
```

Define files that are excluded from integrity checking

Exclude files from the integrity checker command

Code Sample

```
'integrity.excluded.files' => [
    '.DS_Store',
    'Thumbs.db',
    '.directory',
    '.webapp',
    '.htaccess',
    '.user.ini',
  ],
```

Define apps or themes that are excluded from integrity checking

The list of apps that are allowed and must not have a signature.json file present.

Besides ownCloud apps, this is particularly useful when creating ownCloud themes, because themes are treated as apps. The app is identified with it´s app-id. The app-id can be identified by the foldername of the app in your apps directory. The following example allows app-1 and theme-2 to have no signature.json file.

Code Sample

```
'integrity.ignore.missing.app.signature' => [
    'app-id of app-1',
    'app-id of theme-2',
  ],
```

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

Code Sample

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

Define the default cipher for encrypting files

Currently AES-128-CFB and AES-256-CFB are supported.

Code Sample

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

Define the minimum supported ownCloud desktop client version

Define the minimum ownCloud desktop client version that is allowed to sync with this server instance. All connections made from earlier clients will be denied by the server.

As shipped, the value here is the oldest desktop client that is technically compatible with the server. The version number seen here does not imply official support or test coverage on behalf of ownCloud.



Lowering this value may lead to unexpected behaviour, and can include data loss.

Code Sample

```
'minimum.supported.desktop.version' => '2.3.3',
```

Define whether to include external storage in quota calculation

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

Code Sample

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

Define how often filesystem changes are detected

Specifies how often the local filesystem (the ownCloud data/ directory, and NFS mounts in data/) is checked for changes made outside ownCloud. This does not apply to external storages.

→ Never check the filesystem for outside changes, provides a performance increase when it’s certain that no changes are made directly to the filesystem

→ Check each file or folder at most once per request, recommended for general use if outside changes might happen.

Code Sample

```
'filesystem_check_changes' => 0,
```

Define where part files are located

By default ownCloud will store the part files created during upload in the same storage as the upload target. Setting this to false will store the part files in the root of the users folder which might be required to work with certain external storage setups that have limited rename capabilities.

Code Sample

```
'part_file_in_storage' => true,
```

Define the location of mount.json

Defaults to data/mount.json in the ownCloud directory.

Code Sample

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

Prevent cache changes due to changes in the filesystem

When true, prevent ownCloud from changing the cache due to changes in the filesystem for all storage.

Code Sample

```
'filesystem_cache_readonly' => false,
```

Define ownClouds internal secret

Secret used by ownCloud for various purposes, e.g. to encrypt data.

If you lose this string there will be data corruption.

Code Sample

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

Define list of trusted proxy servers

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

Code Sample

```
'trusted_proxies' => [
   '203.0.113.45',
   '198.51.100.128'
 ],
```

Define forwarded_for_headers

Headers that should be trusted as client IP address in combination with trusted_proxies. If the HTTP header looks like 'X-Forwarded-For', then use 'HTTP_X_FORWARDED_FOR' here.

If set incorrectly, a client can spoof their IP address as visible to ownCloud, bypassing access controls and making logs useless!

If not set, defaults to 'HTTP_X_FORWARDED_FOR'.

Code Sample

```
'forwarded_for_headers' => [
   'HTTP_X_FORWARDED',
   'HTTP_FORWARDED_FOR'
 ],
```

Define the maximum filesize for animated GIF´s

Max file size for animating gifs on public-sharing-site.

If the gif is bigger, it'll show a static preview.

Value represents the maximum filesize in megabytes. Default is 10. Set to -1 for no limit.

Code Sample

```
'max_filesize_animated_gifs_public_sharing' => 10,
```

Enable transactional file locking

Transactional file locking is enabled by default.

Prevents concurrent processes from accessing the same files at the same time. Can help prevent side effects that would be caused by concurrent operations. Mainly relevant for very large installations with many users working with shared files.

Code Sample

```
'filelocking.enabled' => true,
```

Define the TTL for file locking

Set the lock's time-to-live in seconds.

Any lock older than this will be automatically cleaned up. If not set this defaults to either 1 hour or the php max_execution_time, whichever is higher.

Code Sample

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

Define the memory caching backend for file locking

Because most memcache backends can clean values without warning, using redis is highly recommended to **avoid data loss**.

Code Sample

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

Disable the web based updater

The web based updater is enabled by default.

Code Sample

```
'upgrade.disable-web' => false,
```

Define whether or not to enable automatic update of market apps

Set to false to disable.

Code Sample

```
'upgrade.automatic-app-update' => true,
```

Place this ownCloud instance into debugging mode

Only enable this for local development and not in production environments This will disable the minifier and outputs some additional debug information

WARNING: Be warned that, if you set this to true, exceptions display stack traces on the web interface, **including passwords**, — **in plain text!**. We strongly encourage you never to use it in production.

Code Sample

```
'debug' => false,
```

Define the data-fingerprint of the current data served

This is a property used by the clients to find out if a backup has been restored on the server. Once a backup is restored run {occ-command-example-prefix} maintenance:data-fingerprint To set this to a new value.

Updating/Deleting this value can make connected clients stall until the user has resolved conflicts.

Code Sample

```
'data-fingerprint' => '',
```

Define if you have copied the sample configuration

This entry is just here to show a warning in case somebody copied the sample configuration.



DO NOT ADD THIS SWITCH TO YOUR CONFIGURATION!

If you, brave person, have read until here be aware that you should not modify **ANY** settings in this file without reading the documentation.

Code Sample

```
'copied_sample_config' => true,
```

Enable or disable the files_external local mount option

Set this property to true if you want to enable the files_external local mount option.

Default: false

Code Sample

```
'files_external_allow_create_new_local' => false,
```

Enable or disable debug logging for SMB access

Set this property to true if you want to enable debug logging for SMB access.

Code Sample

```
'smb.logging.enable' => false,
```

Enable or disable async DAV extensions

Code Sample

```
'dav.enable.async' => false,
```

Apps Config.php Parameters

Introduction

This document describes parameters for apps maintained by ownCloud that are not part of the core system. All keys are only valid if the corresponding app is installed and enabled. You must copy the keys needed to the active config.php file.

Multiple configuration files

ownCloud supports loading configuration parameters from multiple files. You can add arbitrary files ending with .config.php in the config/ directory.

Example:

You could place your email server configuration in email.config.php. This allows you to easily create and manage custom configurations, or to divide a large complex configuration file into a set of smaller files. These custom files are not overwritten by ownCloud, and the values in these files take precedence over config.php.

ownCloud may write configurations into config.php. These configurations may conflict with identical keys already set in additional config files. Be careful when using this capability!

App: Activity

Possible values: activity_expire_days days.

Define the retention for activities of the activity app

Code Sample

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

App: Admin Audit

Possible values: log.conditions ARRAY.

Possible values: admin_audit.groups ARRAY.

Configure the path to the log file

Code Sample

```
'log.conditions' => [
  [
    'apps' => ['admin_audit'],
    // Adjust the path below, to match your setup
    'logfile' => '/var/www/owncloud/data/admin_audit.log'
  ]
],
```

Filter the groups that messages are logged for.

Code Sample

```
'admin_audit.groups' => [
  'group1',
  'group2'
],
```

App: LDAP

Possible values: ldapIgnoreNamingRules doSet or false.

Possible values: user_ldap.enable_medial_search true or false.

Define parameters for the LDAP app

Code Sample

```
'ldapIgnoreNamingRules' => false,
'user_ldap.enable_medial_search' => false,
```

App: Market

Possible values: appstoreurl URL.

Define the download URL for apps

Code Sample

```
'appstoreurl' => 'https://marketplace.owncloud.com',
```

App: Firstrunwizard

Possible values: customclient_desktop URL.

Possible values: customclient_android URL.

Possible values: customclient_ios URL.

Define the download links for ownCloud clients

Configuring the download links for ownCloud clients, as seen in the first-run wizard and on Personal pages

Code Sample

```
'customclient_desktop' =>
   'https://owncloud.org/install/#install-clients',
'customclient_android' =>
   'https://play.google.com/store/apps/details?id=com.owncloud.android',
'customclient_ios' =>
   'https://itunes.apple.com/us/app/owncloud/id543672169?mt=8',
```

App: Richdocuments

Possible values: collabora_group string.

Define the group name for users allowed to use Collabora

Code Sample

```
'collabora_group' => '',
```

OpenID Connect(OIDC) Configuration

App: openidconnect

Configure OpenID Connect

The provider-url, client-id and client-secret variables are to be taken from the OpenID Connect Provider's setup. The loginButtonName variable can be freely chosen, depending on the installation.



The provider-params configuration array only needs to be used if the OpenID Connect Provider does NOT support service discovery.

autoRedirectOnLoginPage
: If true, the login page will automatically be redirected to the OpenID Connect Provider, as when the button is pressed. The default is false.

mode
: This is the attribute in the owncloud accounts table to search for users. The default value is email. An alternative value: userid.

search-attribute
: This is the claim from the OpenID Connect user information which shall be used for searching in the accounts table. The default value is email. For more information about the claim, see https://openid.net/specs/openid-connect-core-1_0.html#Claims.

use-token-introspection-endpoint
: There are tokens which are not JSON WebToken(JWT) and information like the expiry cannot be read from the token itself. In these cases, the OpenID Connect

Provider needs to call on the token introspection endpoint to get this information. The default value is false. See https://tools.ietf.org/html/rfc7662 for more information on token introspection.

Code Sample

```
'openid-connect' => [
   'autoRedirectOnLoginPage' => false,
   'client-id' => '',
   'client-secret' => '',
   'loginButtonName' => 'OpenId Connect',
   'mode' => 'userid',
   // Only required if the OpenID Connect Provider does not support service
discovery
   'provider-params' => [
      'authorization_endpoint' => '',
      'end_session_endpoint' => '',
      'jwks_uri' => '',
      'registration_endpoint' => '',
      'token_endpoint' => '',
      'token_endpoint_auth_methods_supported' => '',
      'userinfo_endpoint' => ''
   ],
   'provider-url' => '',
   'search-attribute' => 'sub',
   'use-token-introspection-endpoint' => true,
 ],
```

Custom Client Download Repositories

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

```
<?php

 "customclient_desktop" => "https://owncloud.org/sync-clients/",
 "customclient_android" =>
"https://play.google.com/store/apps/details?id=com.owncloud.android",
 "customclient_ios"     =>
"https://itunes.apple.com/us/app/owncloud/id543672169?mt=8",
```

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

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

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

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

```
unset OCC_UPDATE_URL
```

Email Configuration

Introduction

ownCloud is capable of sending emails for a range of reasons. These include:

- Password reset emails
- Notifying users of new file shares
- Changes in files
- Activity notifications

To make use of them, users need to configure which notifications they want to receive. They can do this on their Personal pages.



To be able to send emails, a functioning mail server must be available, whether locally in your network, or remotely.

Configuring an SMTP Server

To configure ownCloud to interact with an SMTP server, you can either update config/config.php by hand, or use the graphical Email Configuration Wizard, which updates config/config.php for you.

The Graphical Email Configuration Wizard

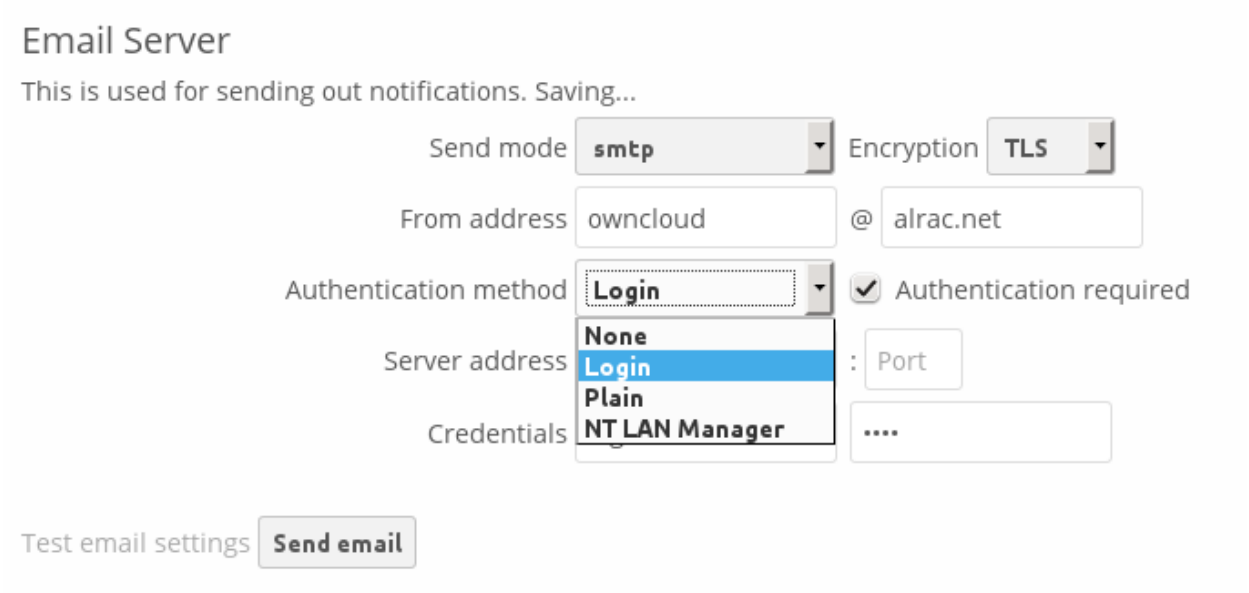
The wizard supports three mail server types: *SMTP*, *PHP*, and *Sendmail*. Use SMTP for a remote email server, and either PHP or Sendmail when your mail server is on the same machine as ownCloud.



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.

You need the following information from your mail server administrator to connect ownCloud to a remote SMTP server:

- Encryption type: None, SSL/TLS or STARTTLS.
- The From address you want your outgoing ownCloud mails to use.
- Whether authentication is required.
- Authentication method: None, Login, Plain, or NT LAN Manager.
- The server's IP address or fully-qualified domain name (FQDN).
- Login credentials, if required.

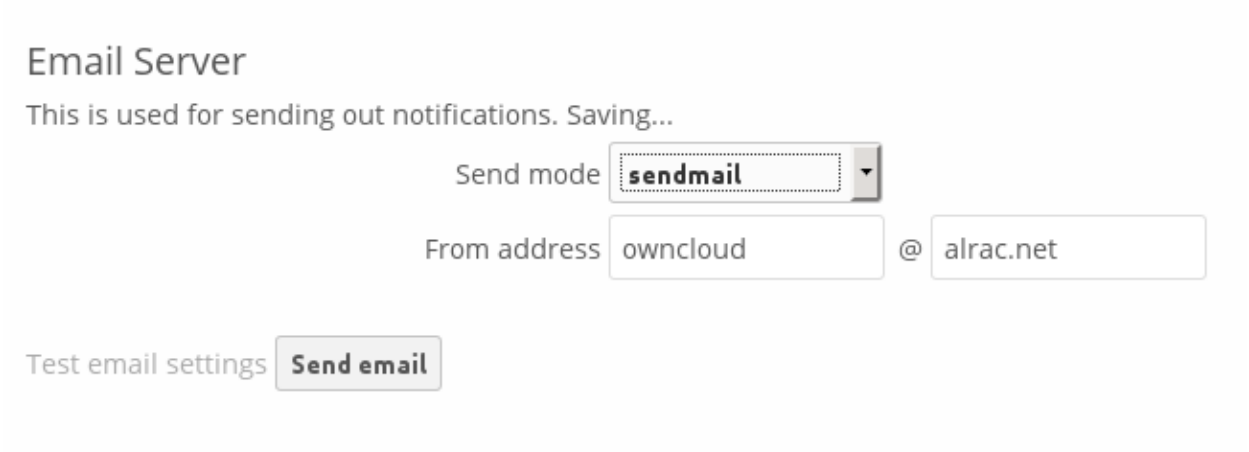


Your changes are saved immediately, and you can click the btn:[Send Email] button to test your configuration. This sends a test message to the email address you configured on your Personal page. The test message says:

> If you received this email, the settings seem to be correct.
>
> --
> ownCloud
> web services under your control

Configuring PHP and Sendmail

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



How do you decide which one to use? PHP mode uses your local sendmail binary. Use this if you want to use php.ini to control some of your mail server functions, such as setting *paths*, *headers*, or passing extra command options to the sendmail binary. These vary according to which server you are using, so consult your server's documentation to see what your options are.

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

Setting Mail Server Parameters in config.php

If you prefer, you may set your mail server parameters in config/config.php. The following examples are for SMTP, PHP, Sendmail, and Qmail.

SMTP

If you want to send email using a local or remote SMTP server it is necessary to enter the name or IP address of the server, optionally followed by a colon separated port number, e.g. **:425**. If this value is not given the default port 25/tcp will be used unless you change that by modifying the **mail_smtpport** parameter. Multiple servers can be entered, separated by semicolons:

```
<?php

"mail_smtpmode"    => "smtp",
"mail_smtphost"    => "smtp-1.server.dom;smtp-2.server.dom:425",
"mail_smtpport"    => 25,
```

Or:

```
<?php

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

If a malware or SPAM scanner is running on the SMTP server it might be necessary that you increase the SMTP timeout to e.g., 30s:

```
<?php

"mail_smtptimeout"  => 30,
```

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

```
<?php

"mail_smtpsecure"   => '',
```

If the SMTP server only accepts secure connections you can choose between the following two variants:

SSL/TLS

A secure connection will be initiated using SSL/TLS via SMTPS on the default port 465/tcp:

```
<?php

"mail_smtphost"    => "smtp.server.dom:465",
"mail_smtpsecure"  => 'ssl',
```

STARTTLS

A secure connection will be initiated using STARTTLS via SMTP on the default port 25/tcp:

```
<?php

"mail_smtphost"    => "smtp.server.dom",
"mail_smtpsecure"  => 'tls',
```

An alternative is the port 587/tcp (recommended):

```
<?php

"mail_smtphost"    => "smtp.server.dom:587",
"mail_smtpsecure"  => 'tls',
```

Authentication

And finally it is necessary to configure if the SMTP server requires authentication, if not, the default values can be taken as is.

```
<?php

"mail_smtpauth"    => false,
"mail_smtpname"    => "",
"mail_smtppassword" => "",
```

If SMTP authentication is required you have to set the required username and password and can optionally choose between the authentication types **LOGIN** (default) or **PLAIN**.

```
<?php

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

PHP Mail

If you want to use PHP mail it is necessary to have an installed and working email system on your server. Which program in detail is used to send email is defined by the

configuration settings in the **php.ini** file. On *nix systems this will most likely be Sendmail. ownCloud should be able to send email out of the box.

```
<?php

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

Sendmail

If you want to use the well known Sendmail program to send email, it is necessary to have an installed and working email system on your *nix server. The Sendmail binary (/usr/sbin/sendmail) is usually part of that system. ownCloud should be able to send email out of the box.

```
<?php

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

Qmail

If you want to use the qmail program to send email, it is necessary to have an installed and working qmail email system on your server. The Sendmail binary (**/var/qmail/bin/sendmail**) will then be used to send email. ownCloud should be able to send email out of the box.

```
<?php

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

Send a Test Email

Regardless of how you have configured ownCloud to interact with an email server, 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.

Using Self-Signed Certificates

When using self-signed certificates on the remote SMTP server the certificate must be imported into ownCloud. Please refer to import_ssl_cert for more information.

Troubleshooting

If you are unable to send email, try turning on debugging. Do this by enabling the mail_smtpdebug parameter in config/config.php.

```
<?php

"mail_smtpdebug" => true;
```



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 ownCloud installation is served. If you have a different mail domain name you can override this behavior by setting the following configuration parameter:

```
<?php

"mail_domain" => "example.com",
```

This setting results in every email sent by ownCloud (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:{std-port-mysql}  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 smtps
- 993/tcp/udp is encrypted imaps
- 995/tcp/udp is encrypted pop3s

How can I determine if the SMTP server supports SMTPS?

A good indication that the SMTP server supports SMTPS is that it is listening on 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.



You must enter the marked lines to obtain the information displayed.

```
telnet smtp.domain.dom 25
```

```
Trying 192.168.1.10...
Connected to smtp.domain.dom.
Escape character is '^]'.
220 smtp.domain.dom ESMTP Exim 4.80.1 Tue, 22 Jan 2013 22:39:55 +0100
EHLO your-server.local.lan                  # <<< enter this command
250-smtp.domain.dom Hello your-server.local.lan [ip-address]
250-SIZE 52428800
250-8BITMIME
250-PIPELINING
250-AUTH PLAIN LOGIN CRAM-MD5               # <<< Supported auth protocols
250-STARTTLS                                # <<< Encryption is supported
250 HELP
QUIT                                        # <<< enter this command
221 smtp.domain.dom closing connection
Connection closed by foreign host.
```

Enabling Debug Mode

If you are unable to send email, it might be useful to activate further debug messages by enabling the mail_smtpdebug parameter:

```
<?php

"mail_smtpdebug" => true,
```



Immediately after pressing the btn:[Send email] button, as described before, several **SMTP → get_lines(): ...** messages appear on the screen. This is expected behavior and can be ignored.

Using Email Templates

Most emails sent from ownCloud are based on editable email templates, which are a mixture of PHP and HTML. The currently available templates are:

Email	Format	Description	File Location
Activity notification mail	plain text	Notification of activities that users have	core/templates/mail.php
		enabled in the Notifications section of	
		their Personal pages.	
Lost password mail		Password reset email for users who lose	core/templates/lostpassword/email.php

Email	Format	Description	File Location
		their passwords.	
New user email	HTML		settings/templates/email.new_user.php
	plain text		settings/templates/email.new_user_plain_text.php
Public link share email	HTML	Notify users of new public link shares.	core/templates/mail.php
	plain text		core/templates/altmail.php
New file share email	HTML	Notify users of new file shares.	core/templates/internalmail.php
	plain text		core/templates/internalaltmail.php

In addition to providing the email templates, this feature enables you to apply any pre-configured themes to the email. To modify an email template to users:

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

The templates are written in PHP and HTML, and are already loaded with the relevant variables such as *username*, *share links*, and *filenames*. You can, if you are careful, edit these — even without knowing PHP or HTML. Don't touch any of the code, but it's OK to edit the text portions of the messages.

For example, this the lost password mail template:

```
<?php

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

You could change the text portion of the template, Use the following link to reset your password: to say something else, such as:

```
Click the following link to reset your password.
If you did not ask for a password reset, ignore this message.
```

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



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

Excluding Directories and Blacklisting Files

Definitions of terms

Blacklisted

Files that may harm the ownCloud environment like a foreign .htaccess file. Blacklisting prevents anyone from uploading blacklisted files to the ownCloud server.

Excluded

Existing directories on your ownCloud server, including directories on external storage mounts, that are excluded from being processed by ownCloud. In effect they are invisible to ownCloud.

Both types are defined in config.php. Blacklisted files and excluded directories are not scanned by ownCloud, not viewed, not synced, and cannot be created, renamed, deleted, or accessed via direct path input from a file manager. Even when a filepath is entered manually via a file explorer, the path cannot be accessed.

For example configurations please see the config.sample.php file.



Many filesystems do not allow the coexistence of a file and folder with exactly the same name on the same directory level. Therefore no differentiation is made in processing files and folders for blacklisting or excluding, as it would just return a deny at a later stage. With the implementation made, you get an immediate error message.

Example: The storage backend has a reserved directory name ".snapshot" which is excluded by configuration. If you try to add a file or folder via the browser or sync a file or folder from the client named ".snapshot", you will get an immediate ownCloud triggered deny.

Impact on System Performance

If you have a filesystem mounted with 200,000 files and directories and 15 snapshots in rotation, you would now scan and process 200,000 elements plus 200,000 x 15 = 3,000,000 elements additionally. These additional 3,000,000 elements, 15 times more than the original quantity, would also be available for viewing and synchronisation. Because this is a big and unnecessary overhead, most times confusing to clients, further processing can be eliminated by using excluded directories.

Blacklisted Files

By default, ownCloud blacklists the file .htaccess to secure the running instance, which is important when using Apache as webserver. A foreign .htaccess file could overwrite rules defined by ownCloud. There is no explicit need to enter the file name .htaccess as parameter to the blacklisted_files array in config.php, but you can add more blacklisted file names if necessary. You can also prevent uploading, e.g., Outlook .pst files, with this mechanism to avoid massive backend space allocation.

Excluded Directories

Reasons for Excluding Directories

1. Enterprise storage systems, or special filesystems like ZFS and BtrFS are capable of snapshots. These snapshots are directories and keep point-in-time views of the data
   a. Snapshot directories are read-only
   b. There is no common naming convention for these directories, and there most likely will never be. For example, NetApp uses .snapshot and ~snapshot, EMC e.g., .ckpt, HDS e.g., .latest and ~latest, and the ZFS filesystem uses .zfs
   c. It does not make sense for these directories to be visible to users as they are used to ease backup, restoration, and cloning
2. Directories which are part of the mounted filesystem, but must not be user accessible/visible
3. Manual managed but user invisible backup directories

Example:

If you have a snapshot-capable storage or filesystem where snapshots are enabled and presented to clients, each directory will contain a "special" visible directory named e.g. .snapshot. Depending on the system, you may find underneath a list of snapshots taken and in the next lower level the complete set of files and directories which were present when the snapshot was created. In most systems, this mechanism is true in all directory levels:

```
/.snapshot
 /nightly.0
    /home
    /dat
    /pictures
    file_1
    file_2
 /nightly.1
    /home
    /dat
    /pictures
    file_1
    file_2
 /nightly.2
    /home
    /dat
    /pictures
    file_1
    file_2
 ...
/home
/dat
/pictures
file_1
file_2
...
```

Example excluded_directories entries in config.php can look like this:

```
'excluded_directories' => [
   '.snapshot',
   '~snapshot',
   'dir1',
   'dir2',
],
```

Note that these are not pathnames, but directory names without any slashes. Excluding dir1 excludes:

```
/home/dir1
/etc/stuff/dir1
```

But not:

```
/home/.dir1
/etc/stuff/mydir1
```

Reasons for Blacklisting Files

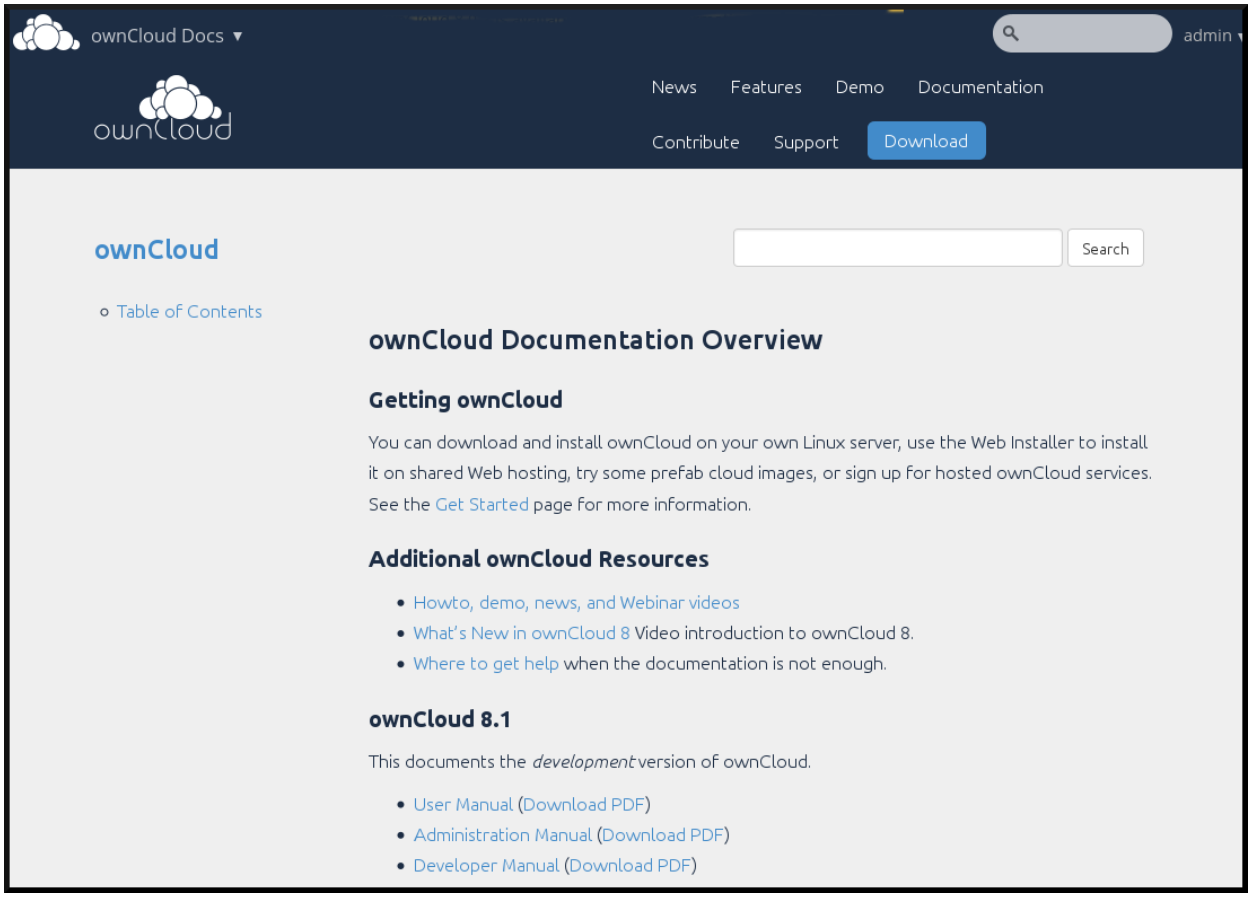
The reason for blacklisting files can be explained based on the example of a .htaccess file. Such a file can lead to a unwanted behaviour of your webserver when visible to ownCloud in a directory.

Example blacklisted_files entries in config.php can look like this:

```
'blacklisted_files' => [
   'hosts',
   'evil_script.sh',
],
```

Linking External Sites

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

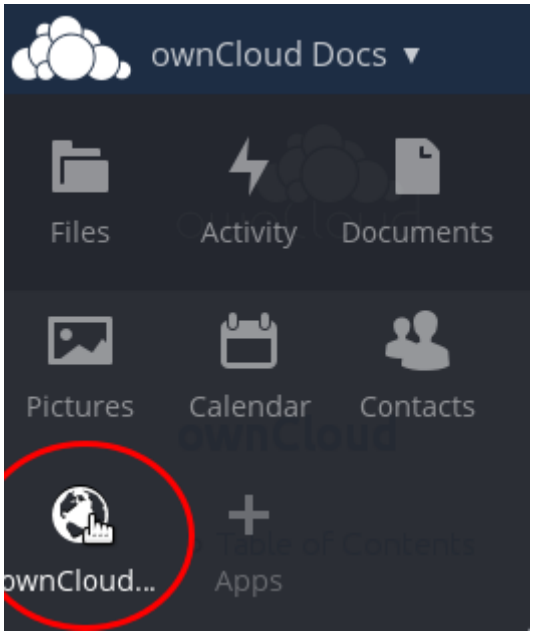


This is useful for quick access to important Web pages such as the ownCloud manuals and informational pages for your company, and for presenting external pages inside your custom ownCloud branding, if you use your own custom themes.

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

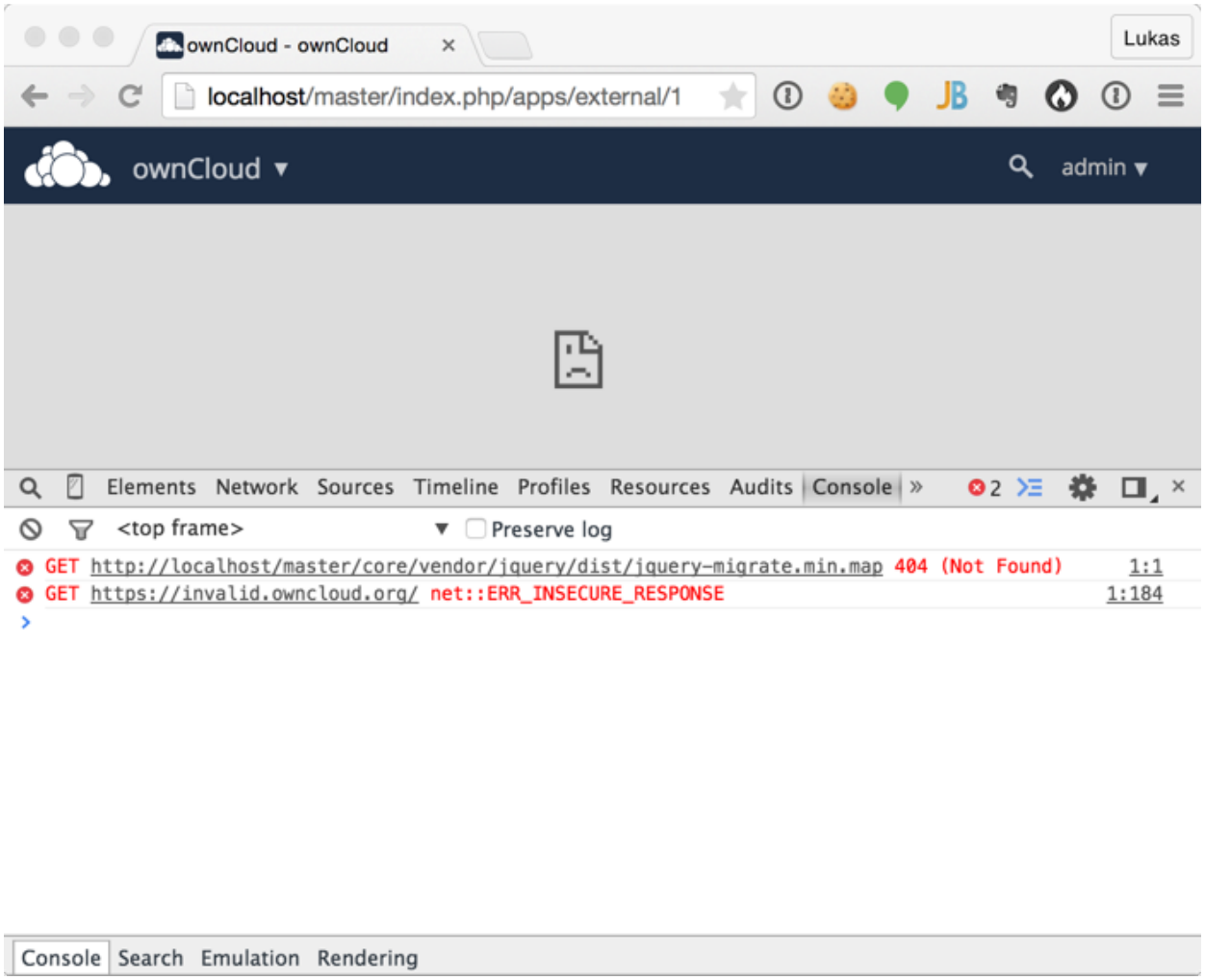


The links appear in the ownCloud dropdown menu on the top left after refreshing your page, and have globe icons.

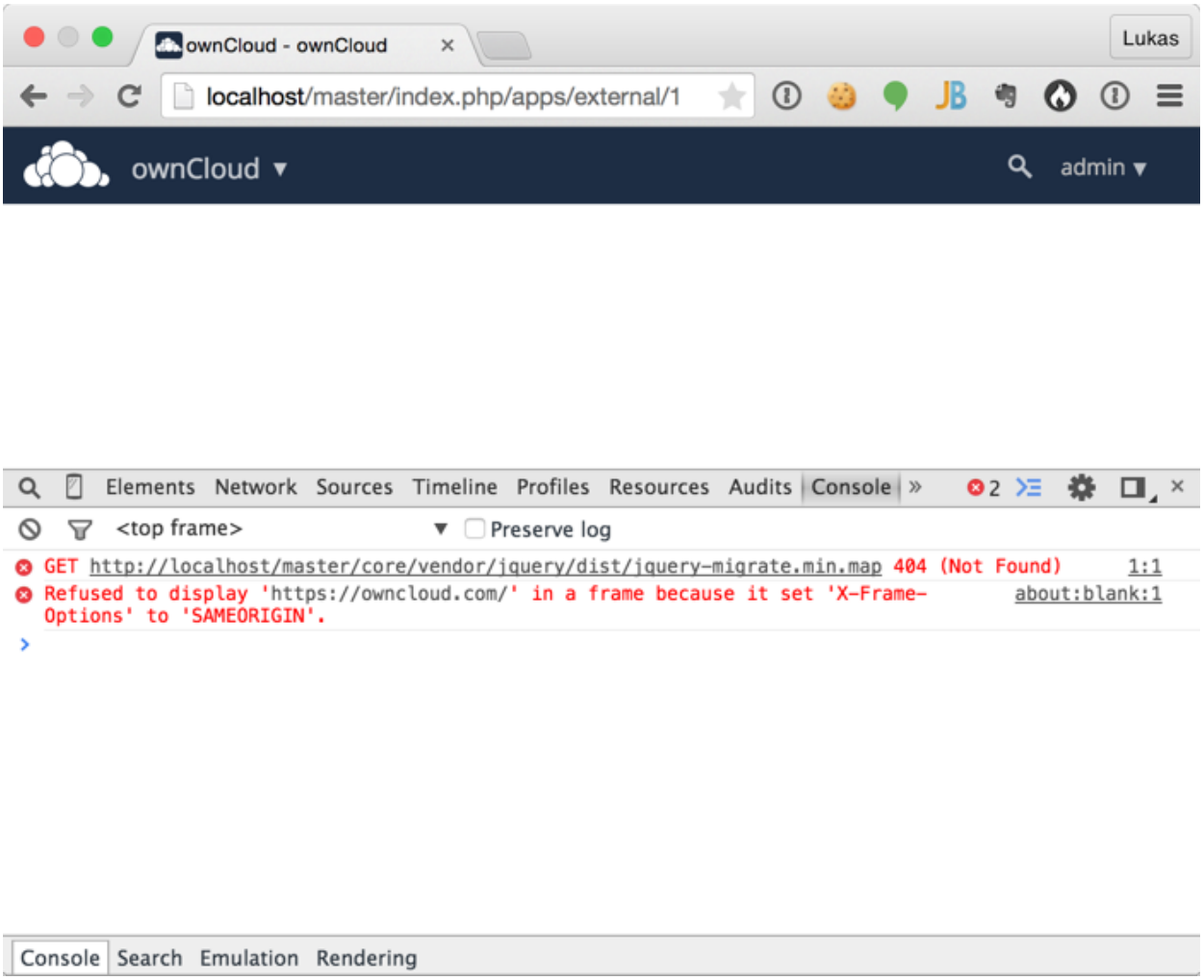


Your links may or may not work correctly due to the various ways that Web browsers and Web sites handle HTTP and HTTPS URLs, and because the External Sites app embeds external links in IFrames. Modern Web browsers try very hard to protect Web surfers from dangerous links, and safety apps like Privacy Badger and ad-blockers may block embedded pages. It is strongly recommended to enforce HTTPS on your ownCloud 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 ownCloud.

Most Web sites that offer login functionalities use the X-Frame-Options or Content-Security-Policy HTTP header which instructs browsers to not allow their pages to be embedded for security reasons (e.g. "Clickjacking"). You can usually verify the reason why embedding the website is not possible by using your browser's console tool. For example, this page has an invalid SSL certificate.



On this page, X-Frame-Options prevents the embedding.



There isn't much you can do about these issues, but if you're curious you can see what is happening.

Hardening and Security Guidance

Introduction

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



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

Limit on Password Length

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

Rate Limiting

Currently ownCloud deliberately does not provide any form of rate-limiting (though it does provide brute-force protection). This is because ownCloud needs to integrate in to a diverse range of environments and infrastructure, which often already provide

specialized rate-limiting solutions, e.g., *Apache*, *HAProxy*, and *F5*.

If you are yet to implement a rate-limiting solution for your ownCloud instance, start by retrieving a list of all active routes. This information is obtained by running occ's security:routes command, as in the following example.

```
{occ-command-example-prefix} security:routes
```

It should print a list of all the routes, as in the following, truncated, example.

```
+-------------------------------------------------------+----------------+
| Path                                        | Methods        |
+-------------------------------------------------------+----------------+
| /apps/encryption/ajax/adminRecovery                   | POST           |
| /apps/encryption/ajax/changeRecoveryPassword          | POST           |
| /apps/encryption/ajax/getStatus                       | GET            |
| /apps/encryption/ajax/setEncryptHomeStorage           | POST           |
| /apps/encryption/ajax/updatePrivateKeyPassword        | POST           |
| /apps/encryption/ajax/userSetRecovery                 | POST           |
| /apps/federatedfilesharing/                           | GET            |
| /apps/federatedfilesharing/notifications              | POST           |
+-------------------------------------------------------+----------------+
```

With that information, you are then able to begin customising a rate-limiting solution specific to your ownCloud installation.

Further Reading

- Rate limiting with Apache
  - mod_cband
  - mod_evasive
  - mod_ratelimit
  - mod_security
  - Rate limiting with Fail2Ban
  - Fail2Ban
  - Fail2Ban Behind A Proxy/Load Balancer
- Rate limiting with HaProxy
- Rate limiting with F5

Operating system

Give PHP read access to /dev/urandom

ownCloud 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 ownCloud 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.



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

Enable hardening modules such as SELinux

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

Deployment

Place data directory outside of the web root

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

Disable preview image generation

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

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

Use HTTPS

Using ownCloud 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 adding a setting such as the following in the Apache VirtualHosts configuration containing the <VirtualHost *:80> entry:

```
Redirect permanent / https://example.com/
```

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 ownCloud 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 containing the <VirtualHost *:443> entry:

```
<IfModule mod_headers.c>
  Header always set Strict-Transport-Security "max-age=15552000;
includeSubDomains"
</IfModule>
```

If you don't have access to your Apache configuration it is also possible to add this to the main .htaccess file shipped with ownCloud. Make sure you're adding it below the line:

```
#### DO NOT CHANGE ANYTHING ABOVE THIS LINE ####
```

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



This requires the mod_headers extension in Apache.

Proper SSL configuration

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

We recommend using the Mozilla SSL Configuration Generator to generate a suitable configuration suited for your environment, and the free Qualys SSL Labs Tests gives good guidance on whether your SSL server is correctly configured.

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

Use a dedicated domain for ownCloud

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

Ensure that your ownCloud instance is installed in a DMZ

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

Serve security Related Headers by the Web server

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

- X-Content-Type-Options: nosniff: Instructs some browsers to not sniff the mimetype of files. This is used for example to prevent browsers from interpreting text files as JavaScript.
- X-XSS-Protection: 1; mode=block: Instructs browsers to enable their browser side Cross-Site-Scripting filter.

- X-Robots-Tag: none: Instructs search machines to not index these pages.
- X-Frame-Options: SAMEORIGIN: Prevents embedding of the ownCloud instance within an iframe from other domains to prevent Clickjacking and other similar attacks.

These headers are hard-coded into the ownCloud 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.

Use Fail2ban

Another approach to hardening the server(s) on which your ownCloud installation rest is using an intrusion detection system. An excellent one is Fail2ban. Fail2ban is designed to protect servers from brute force attacks. It works by monitoring log files (such as those for *ssh*, *web*, *mail*, and *log* servers) for certain patterns, specific to each server, and taking actions should those patterns be found.

Actions include banning the IP from which the detected actions are being made from. This serves to both make the process more difficult as well as to prevent DDOS-style attacks. However, after a predefined time period, the banned IP is normally un-banned again.

This helps if the login attempts were genuine, so the user doesn't lock themselves out permanently. An example of such an action is users attempting to brute force login to a server via ssh. In this case, Fail2ban would look for something similar to the following in /var/log/auth.log.

```
Mar 15 11:17:37 yourhost sshd[10912]: input_userauth_request: invalid user audra
[preauth]
Mar 15 11:17:37 yourhost sshd[10912]: pam_unix(sshd:auth): check pass; user
unknown
Mar 15 11:14:51 yourhost sshd[10835]: PAM 2 more authentication failures;
logname= uid=0 euid=0 tty=ssh ruser= rhost=221.194.44.231  user=root
Mar 15 11:14:57 yourhost sshd[10837]: pam_unix(sshd:auth): authentication
failure; logname= uid=0 euid=0 tty=ssh ruser= rhost=221.194.44.231  user=root
Mar 15 11:14:59 yourhost sshd[10837]: Failed password for root from
221.194.44.231 port 46838 ssh2
Mar 15 11:15:04 yourhost sshd[10837]: message repeated 2 times: [ Failed
password for root from 221.194.44.231 port 46838 ssh2]
Mar 15 11:15:04 yourhost sshd[10837]: Received disconnect from 221.194.44.231:
11:  [preauth]
```



If you're not familiar with what's going on, this snippet highlights a number of failed login attempts being made.

Using Fail2ban to secure an ownCloud login

On Ubuntu, you can install Fail2ban using the following commands:

```
apt update && apt upgrade
apt install fail2ban
```

Fail2ban installs several default filters for *Apache*, and various other services, but none for ownCloud. Given that, we have to define our own filter. To do so, you first need to make sure that ownCloud uses your local timezone for writing log entries; otherwise, fail2ban cannot react appropriately to attacks. To do this, edit your config.php file and add the following line:

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



Adjust the timezone to the one that your server is located in, based on PHP's list of supported timezones.

This change takes effect as soon as you save config.php. You can test the change by:

1. Entering false credentials at your ownCloud login screen
2. Checking the timestamp of the resulting entry in ownCloud's log file.

Next, define a new Fail2ban filter rule for ownCloud. To do so, create a new file called /etc/fail2ban/filter.d/owncloud.conf, and insert the following configuration:

```
[Definition]
failregex={.*Login failed: \'.*\' \(Remote IP: \'<HOST>\'\)"}
ignoreregex =
```

This filter needs to be loaded when Fail2ban starts, so a further configuration entry is required to be added in /etc/fail2ban/jail.d/defaults-debian.conf, which you can see below:

```
[owncloud]
enabled = true
port = 80,443
protocol = tcp
filter = owncloud
maxretry = 3
bantime = 10800
logpath = /var/owncloud_data/owncloud.log
```

This configuration:

1. Enables the filter rules for TCP requests on ports 80 and 443.
2. Bans IPs for 10800 seconds (3 hours).
3. Sets the path to the log file to analyze for malicious logins



The most important part of the configuration is the logpath parameter. If this does not point to the correct log file, Fail2ban will either not work properly or refuse to start.

After saving the file, restart Fail2ban by running the following command:

```
service fail2ban restart
```

To test that the new ownCloud configuration has been loaded, use the following command:

```
fail2ban-client status
```

If "owncloud" is listed in the console output, the filter is both loaded and active. If you want to test the filter, run the following command, adjusting the path to your owncloud.log, if necessary:

```
fail2ban-regex /var/owncloud_data/owncloud.log /etc/fail2ban/filter.d/owncloud.conf
```

The output will look similar to the following, if you had one failed login attempt:

```
fail2ban-regex /var/www/owncloud_data/owncloud.log
/etc/fail2ban/filter.d/owncloud.conf

Running tests
=============

Use   failregex file : /etc/fail2ban/filter.d/owncloud.conf
Use         log file : /var/www/owncloud_data/owncloud.log

Results
=======

Failregex: 1 total
|-  #) [# of hits] regular expression
|   1) [1] {.*Login failed: \'.*\' \(Remote IP: \'<HOST>\'\)"}
`-

Ignoreregex: 0 total

Date template hits:
|- [# of hits] date format
|  [40252] ISO 8601
`-

Lines: 40252 lines, 0 ignored, 1 matched, 40251 missed
```

The Failregex counter increments by 1 for every failed login attempt. To un-ban an IP,

which was locked either during testing or unintentionally, use the following command:

```
fail2ban-client set owncloud unbanip <IP>
```

You can check the status of your ownCloud filter with the following command:

```
fail2ban-client status owncloud
```

This will produce an output similar to this:

```
Status for the jail: owncloud
|- filter
|  |- File list:    /var/www/owncloud_data/owncloud.log
|  |- Currently failed: 1
|  `- Total failed: 7
`- action
   |- Currently banned: 0
   |  `- IP list:
   `- Total banned: 1
```

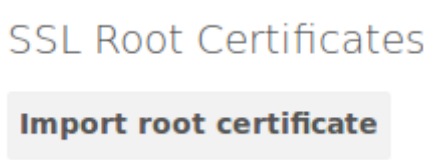
Importing System-wide and Personal SSL Certificates

Introduction

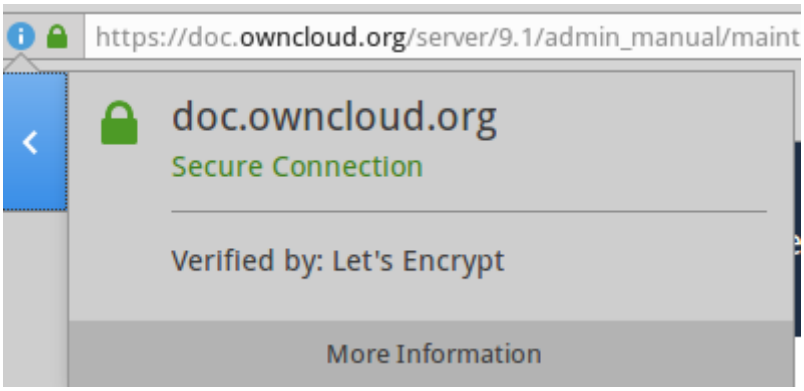
Modern Web browsers try to keep us safe, and so they blast us with scary warnings when sites have the smallest errors in their SSL certificates, or when they use self-signed SSL certificates. ownCloud admins encounter this when creating Federation shares, or setting up external storage mounts. There is no reason against using self-signed certificates on your own networks; they're fast, free, and easy.

Importing Personal SSL Certificates

ownCloud has several methods for importing self-signed certificates so that you don't have to hassle with Web browser warnings. When you allow your users to create their own external storage mounts or Federation shares, they can import SSL certificates for those shares on their Personal pages.



Click the **Import root certificate** button to open a file picker. You can distribute copies of your SSL certificates to your users (via an ownCloud share!), or users can download them from their Web browsers. Click on the little padlock icon and click through until you see a btn:[View Certificate] button, then keep going until you can download it. In Firefox and Chromium there is an btn:[Export] button for downloading your own copy of a site's SSL certificate.

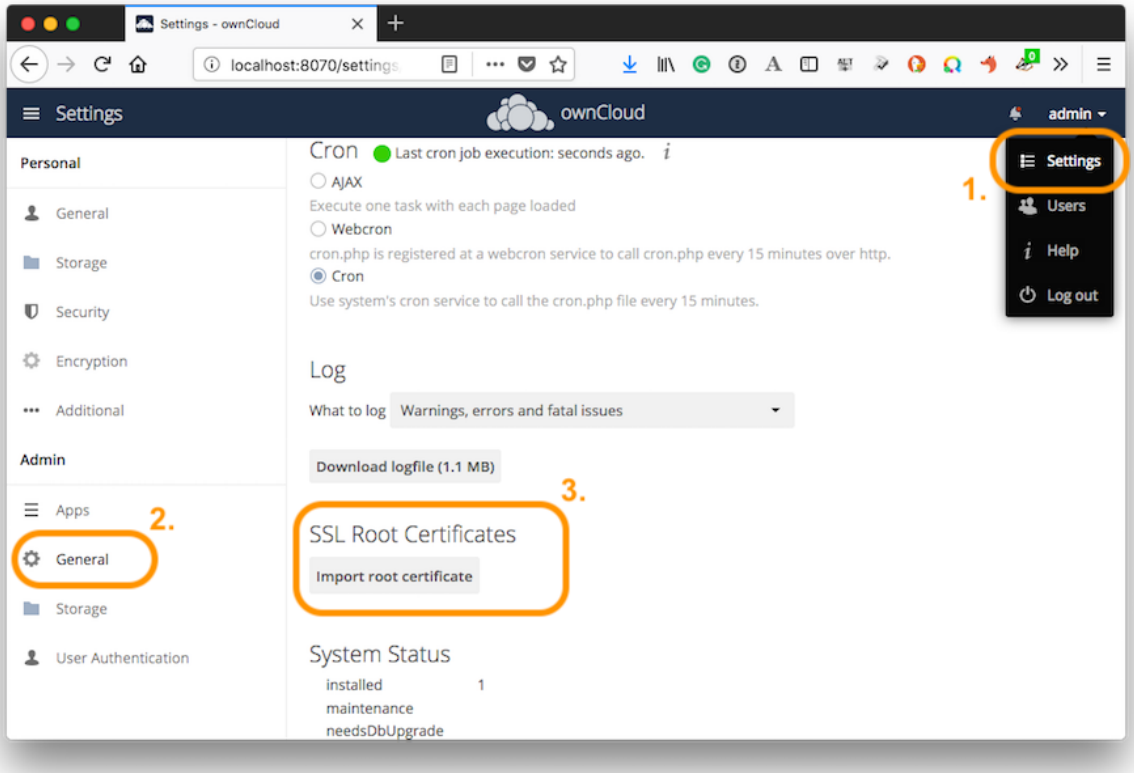


Site-wide SSL Import

The personal imports only work for individual users. You can enable site-wide SSL certificates for all of your users on your ownCloud admin page. To enable this, you must add this line to your config.php file:

```
'enable_certificate_management' => true,
```

Then you'll have an btn:[Import root certificate] button on your admin page, just like the one on your personal page. Navigate to it by clicking menu:Settings[General > SSL Root Certificates] which is located almost at the bottom.



Using OCC to Import and Manage SSL Certificates

The occ command has options for listing and managing your SSL certificates:

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

See Using the occ Command to learn about how to use occ.

Enable index.php-less URLs

Introduction

Since ownCloud 9.0.3 you need to explicitly configure and enable index.php-less URLs (e.g. https://example.com/apps/files/ instead of https://example.com/index.php/apps/files/). The following documentation provides the needed steps to configure this for the Apache Web server.

Prerequisites

Before being able to use index.php-less URLs you need to enable the mod_rewrite and mod_env Apache modules. Furthermore a configured AllowOverride All directive within the vhost of your Web server is needed. Please have a look at the Apache manual for how to enable and configure these.

Furthermore these instructions are only working when using Apache together with the mod_php Apache module for PHP. Other modules like php-fpm or mod_fastcgi are unsupported.

Finally the user running your Web server (e.g. www-data) needs to be able to write into the .htaccess file shipped within the ownCloud root directory (e.g., /var/www/owncloud/.htaccess). If you have applied strong permissions, the user might be unable to write into this file and the needed update will fail. You need to revert this strong permissions temporarily by following the steps described in setting permissions for updating.

Configuration steps

The first step is to configure the overwrite.cli.url and htaccess.RewriteBase config.php options (See config_sample_php_parameters). If you're accessing your ownCloud instance via https://example.com/ the following two options need to be added / configured:

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

If the instance is accessed via https://example.com/owncloud the following configuration is needed:

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

As a second step ownCloud needs to enable index.php-less URLs. This is done:

- during the next update of your ownCloud instance
- by manually running the occ command occ maintenance:update:htaccess (See occ_command)

Afterwards your instance should have index.php-less URLs enabled.

Troubleshooting

If accessing your ownCloud installation fails after following these instructions and you see messages like this in your ownCloud log:

```
The requested uri(\\/login) cannot be processed by the script
'\\/owncloud\\/index.php'
```

make sure that you have configured the two config.php options listed above correctly.

Using the occ Command

ownCloud's occ command (ownCloud console) is ownCloud's command-line interface. You can perform many common server operations with occ, such as installing and upgrading ownCloud, managing users and groups, encryption, passwords, LDAP setting, and more.

occ is in the owncloud/ directory; for example /var/www/owncloud 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 ownCloud files and directories.

Run occ As Your HTTP User

The HTTP user is different on the various Linux distributions.

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



See Setting Strong Permissions to learn how to find your HTTP user.

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-PHP54 installed, the command looks like this:

```
sudo -u apache /opt/rh/php54/root/usr/bin/php /var/www/html/owncloud/occ
```

Example Commands

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

```
{occ-command-example-prefix}
ownCloud version 10.0.8

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 {occ-command-example-prefix} list. Run it with the -h option for syntax help:

```
{occ-command-example-prefix} -h
```

Display your ownCloud version:

```
{occ-command-example-prefix} -V
 ownCloud version 10.0.8
```

Query your ownCloud server status:

```
{occ-command-example-prefix} status
  - installed: true
  - version: 10.0.8.5
  - versionstring: 10.0.8
  - edition: Community
```

occ has *options*, *commands*, and *arguments*. Commands are required. Options are optional. Arguments can be required *or* optional. The, generic, syntax is:

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

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

```
{occ-command-example-prefix} help maintenance:mode
Usage:
 maintenance:mode [options]

Options
    --on             Enable maintenance mode
    --off            Disable maintenance mode
    --output[=OUTPUT] Output format (plain, json or json_pretty, default is plain)
[default: "plain"]
 -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

```
{occ-command-example-prefix} status --output=json
{"installed":true,"version":"9.0.0.19","versionstring":"9.0.0","edition":""}
```

or json_pretty

```
{occ-command-example-prefix} status --output=json_pretty
{
  "installed": true,
  "version": "10.0.8.5",
  "versionstring": "10.0.8",
  "edition": "Community"
}
```

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

Core Commands

This command reference covers the ownCloud core commands.

App Commands

The app commands list, enable, and disable apps.

```
app
 app:check-code   check code to be compliant
 app:disable      disable an app
 app:enable       enable an app
 app:getpath      Get an absolute path to the app directory
 app:list         List all available apps
```

List Available Apps

List all of your installed apps or optionally provide a search pattern to restrict the list of apps to those whose name matches the given regular expression. The output shows whether they are enabled or disabled.

```
{occ-command-example-prefix} app:list [--] [<search-pattern>]
```

Arguments

search-pattern	Show only those apps whose names match the given search pattern (regular expression).

Options

--disabled	Only display disabled apps. If the app was previously enabled, the app version is also displayed. When used, the output will contain the app's version number as well, *if* it was previously enabled.
--enabled	Only display enabled apps. When used, the output will contain the app's version number as well.
--output[=OUTPUT]	The output format to use (plain, json or json_pretty). [default: "plain"]
--shipped=<SHIPPED>	If SHIPPED is set to true, only shipped apps will be listed. If SHIPPED is set to false, only non-shipped apps will be listed.

Enable an App

Enable an app, for example the Market app.

```
{occ-command-example-prefix} app:enable market
market enabled
```

Disable an App

```
{occ-command-example-prefix} app:disable market
market disabled
```



Be aware that the following apps cannot be disabled: *DAV*, *FederatedFileSharing*, *Files* and *Files_External*.

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

```
{occ-command-example-prefix} app:check-code notifications
App is compliant - awesome job!
```

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

```
{occ-command-example-prefix} app:check-code foo_app
Analysing /var/www/owncloud/apps/files/foo_app.php
4 errors
  line   45: OCP\Response - Static method of deprecated class must not be called
  line   46: OCP\Response - Static method of deprecated class must not be called
  line   47: OCP\Response - Static method of deprecated class must not be called
  line   49: OC_Util - Static method of private class must not be called
```

You can get the full file path to an app.

```
{occ-command-example-prefix} app:getpath notifications
/var/www/owncloud/apps/notifications
```

Background Jobs Selector

Use the background command to select which scheduler you want to use for controlling background jobs. This is the same as using the **Cron** section on your ownCloud Admin page.

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

Examples

```
# Set the background scheduler to Ajax
{occ-command-example-prefix} background:ajax

# Set the background scheduler to Cron
{occ-command-example-prefix} background:cron

# Set the background scheduler to Webcron
{occ-command-example-prefix} background:webcron
```



See background jobs configuration to learn more.

Config Commands

The config commands are used to configure the ownCloud 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 configuration settings
 config:list            List all configuration settings
 config:system:delete   Delete a system config value
 config:system:get      Get a system config value
 config:system:set      Set a system config value
```

config:list

The config:list command lists all configuration values, both for your ownCloud setup, along with any apps.

```
{occ-command-example-prefix} config:list [options] [--] [<app>]
```

Arguments

app	Name of the app. You can use "*system*" to get the config.php values, or "*all*" (the default) for all apps and system.

Options

--private	Use this option when you want to include sensitive configs, like passwords and salts.

By default, passwords and other sensitive data are omitted from the report so that the output can be posted publicly (e.g., as part of a bug report). You can see a sample output in the example below.


```
{
    "system": {
        "passwordsalt": "***REMOVED SENSITIVE VALUE***",
        "secret": "***REMOVED SENSITIVE VALUE***",
        "trusted_domains": [
            "localhost",
        ],
        "datadirectory": "\/var\/www\/localhost\/data",
        "overwrite.cli.url": "http:\/\/localhost",
        "dbtype": "mysql",
        "version": "10.3.0.4",
        "dbname": "owncloud",
        "dbhost": "localhost",
```

```
        "dbtableprefix": "oc_",
        "dbuser": "***REMOVED SENSITIVE VALUE***",
        "dbpassword": "***REMOVED SENSITIVE VALUE***",
        "logtimezone": "UTC",
        "shareapi_allow_public_notification": "yes",
        "apps_paths": [
           {
              "path": "\/var\/www\/localhost\/apps",
              "url": "\/apps",
              "writable": false
           },
           {
              "path": "\/var\/www\/localhost\/apps-external",
              "url": "\/apps-external",
              "writable": true
           }
        ],
        "installed": true,
        "instanceid": "ocfp00rezy80",
        "loglevel": 2,
        "maintenance": false
     },
     "apps": {
        "backgroundjob": {
           "lastjob": "13"
        },
        "comments": {
           "enabled": "yes",
           "installed_version": "0.3.0",
           "types": "logging,dav"
        },
        "core": {
           "backgroundjobs_mode": "cron",
           "enable_external_storage": "yes",
           "first_install_version": "10.3.0.2",
           "installedat": "1569845065.1792",
           "lastcron": "1571930489",
           "lastupdateResult": "[]",
           "lastupdatedat": "1572536814",
           "oc.integritycheck.checker": "{\"systemtags\":{\"EXCEPTION\":{\"class\":
\"OC\\\\IntegrityCheck\\\\Exceptions\\\\MissingSignatureException\",\"message\":\"Si
gnature data not found.\"}},\"comments\":{\"EXCEPTION\":{\"class\":\"OC
\\\\IntegrityCheck\\\\Exceptions\\\\MissingSignatureException\",\"message\":\"Signat
ure data not found.\"}}}",
           "public_files": "files_sharing\/public.php",
           "public_webdav": "dav\/appinfo\/v1\/publicwebdav.php",
           "shareapi_allow_mail_notification": "yes",
           "umgmt_set_password": "false",
           "umgmt_show_backend": "true",
           "umgmt_show_email": "true",
```

```
            "umgmt_show_is_enabled": "true",
            "umgmt_show_last_login": "true",
            "umgmt_show_password": "false",
            "umgmt_show_quota": "true",
            "umgmt_show_storage_location": "false",
            "vendor": "owncloud"
        },
        "dav": {
            "enabled": "yes",
            "installed_version": "0.5.0",
            "types": "filesystem"
        },
        "federatedfilesharing": {
            "enabled": "yes",
            "installed_version": "0.5.0",
            "types": "filesystem"
        },
        "federation": {
            "enabled": "yes",
            "installed_version": "0.1.0",
            "types": "authentication"
        },
        "files": {
            "cronjob_scan_files": "500",
            "enabled": "yes",
            "installed_version": "1.5.2",
            "types": "filesystem"
        },
        "files_external": {
            "allow_user_mounting": "yes",
            "enabled": "yes",
            "installed_version": "0.7.1",
            "types": "filesystem",
            "user_mounting_backends": "googledrive,owncloud,sftp,smb,dav,\\OC\\Files
\\Storage\\SFTP_Key,\\OC\\Files\\Storage\\SMB_OC"
        },
        "files_sharing": {
            "enabled": "yes",
            "installed_version": "0.12.0",
            "types": "filesystem"
        },
        "files_trashbin": {
            "enabled": "yes",
            "installed_version": "0.9.1",
            "types": "filesystem"
        },
        "files_versions": {
            "enabled": "yes",
            "installed_version": "1.3.0",
            "types": "filesystem"
```

```
        },
        "provisioning_api": {
            "enabled": "yes",
            "installed_version": "0.5.0",
            "types": "prevent_group_restriction"
        },
        "systemtags": {
            "enabled": "yes",
            "installed_version": "0.3.0",
            "types": "logging"
        },
        "updatenotification": {
            "enabled": "yes",
            "installed_version": "0.2.1",
            "types": ""
        }
    }
}
```


Displaying Sensitive Information

To generate a full report which includes sensitive values, such as passwords and salts, use the --private option, as in the following example.

```
{occ-command-example-prefix} config:list --private
```

Filtering Information Reported

The output can be filtered to just the core information, core and apps, or one specific app. In the example below, you can see how to filter in each of these ways.

```
# List only system configuration details
{occ-command-example-prefix} config:list -- system

# List system and app configuration details
# This is the default, so doesn't need to be explicitly specified
{occ-command-example-prefix} config:list -- all

# List configuration details of the dav app
{occ-command-example-prefix} config:list -- dav
```

Below is an example of listing the config details for a single app.

```
{
  "apps": {
    "files_versions": {
      "enabled": "yes",
      "installed_version": "1.3.0",
      "types": "filesystem"
    }
  }
}
```


config:import

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.

```
{occ-command-example-prefix} config:import filename.json
```

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

```
{occ-command-example-prefix} config:import < local-backup.json
```



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

Getting a Single Configuration Value

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

config:system:get

```
{occ-command-example-prefix} config:system:get [options] [--] <name> (
<name>)...
```

Arguments

name	Name of the config to get. Specify multiple for array parameter.

Options

--default-value[=DEFAULT-VALUE]	If no default value is set and the config does not exist, the command will exit with 1.
--output=[OUTPUT]	The output format to use (plain, json or json_pretty, default is plain).

config:app:get

```
{occ-command-example-prefix} config:app:set [options] [--] <app> <name>
```

Arguments

app	Name of the app.
name	Name of the config to get.

Options

--default-value[=DEFAULT-VALUE]	If no default value is set and the config does not exist, the command will exit with 1.
--output=[OUTPUT]	The output format to use (plain, json or json_pretty, default is plain).

Examples

```
{occ-command-example-prefix} config:system:get version
10.0.8.5

{occ-command-example-prefix} 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.

config:system:set

```
{occ-command-example-prefix} config:system:set [options] [--] <name> (
<name>)...
```

Arguments

name	Name of the config parameter, specify multiple for array parameter.

Options

--type=[TYPE]	Value type to use (string, integer, double, boolean, json, default is string). Note: you must use json to write multi array values.
--value=[VALUE]	The new value of the config.
--update-only	Only updates the value, if it is not set before, it is not being added.
--output=[OUTPUT]	The output format to use (plain, json or json_pretty, default is plain).

Examples

Adding Redis to the configuration:

```
{occ-command-example-prefix} config:system:set \
  redis \
  --value '{"host": "{oc-examples-server-ip}", "port": "6379"}' \
  --type json

System config value redis set to json {"host": "{oc-examples-server-ip}", "port":
"6379"}
```

config:app:set

```
{occ-command-example-prefix} config:app:set [options] [--] <app> <name>
```

Arguments

app	Name of the app.
name	Name of the config to set.

Options

--value=[VALUE]	The new value of the config.
--update-only	Only updates the value, if it is not set before, it is not being added.
--output=[OUTPUT]	The output format to use (plain, json or json_pretty, default is plain).

Examples

```
{occ-command-example-prefix} config:system:set \
  logtimezone \
  --value="Europe/Berlin"
System config value logtimezone set to Europe/Berlin
```

```
{occ-command-example-prefix} config:app:set \
  files_sharing \
  incoming_server2server_share_enabled \
  --value=true \
  --type=boolean
Config value incoming_server2server_share_enabled for app files_sharing set to yes
```

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

```
{occ-command-example-prefix} config:system:set \
  doesnotexist \
  --value=true \
  --type=boolean \
  --update-only
Value not updated, as it has not been set before.
```



In order to write a boolean, float, JSON, 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. Please see table above for available types.

Examples

Disable the maintenance mode:

```
{occ-command-example-prefix} config:system:set maintenance \
  --value=false \
  --type=boolean

ownCloud is in maintenance mode - no app have been loaded
System config value maintenance set to boolean false
```

Create the app_paths config setting (using a JSON payload because of multi array values):

```
{occ-command-example-prefix} config:system:set apps_paths \
    --type=json \
    --value='[
      {
         "path":"/var/www/owncloud/apps",
         "url":"/apps",
         "writable": false
      },
      {
         "path":"/var/www/owncloud/apps-external",
         "url":"/apps-external",
         "writable": true
      }
   ]'
```

Setting an Array of Configuration Values

Some configurations (e.g., the trusted domain setting) are an array of data. The array starts counting with 0. In order to set (and also get) the value of one key, you can specify multiple config names separated by spaces:

```
{occ-command-example-prefix} config:system:get trusted_domains
localhost
owncloud.local
sample.tld
```

To replace sample.tld with example.com trusted_domains ⇒ 2 needs to be set:

```
{occ-command-example-prefix} config:system:set trusted_domains 2
--value=example.com
System config value trusted_domains => 2 set to string example.com

{occ-command-example-prefix} config:system:get trusted_domains
localhost
owncloud.local
example.com
```

Deleting a Single Configuration Value

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

config:system:delete

```
{occ-command-example-prefix} config:system:delete [options] [--] <name>
(<name>)...
```

Arguments

name	Name of the config to delete, specify multiple for array parameter.

Options

--error-if-not-exists	Checks whether the config exists before deleting it.
--output=[OUTPUT]	The output format to use (plain, json or json_pretty, default is plain).

config:app:delete

```
{occ-command-example-prefix} config:app:delete [options] [--] <app> <name>
```

Arguments

app	Name of the app.
name	Name of the config to delete.

Options

--error-if-not-exists	Checks whether the config exists before deleting it.
--output=[OUTPUT]	The output format to use (plain, json or json_pretty, default is plain).

Examples:

```
{occ-command-example-prefix} config:system:delete maintenance:mode
System config value maintenance:mode deleted

{occ-command-example-prefix} config:app:delete myappname provisioning_api
Config value provisioning_api of app myappname 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.

```
{occ-command-example-prefix} config:system:delete doesnotexist --error-if-not
-exists
Config provisioning_api of app appname could not be deleted because it did not
exist
```

Config Reports

If you're working with ownCloud support and need to send them a configuration summary, you can generate it using the configreport:generate command. This command generates the same JSON-based report as the Admin Config Report, which you can access under admin → Settings → Admin → General → Generate Config Report → Download ownCloud config report.

From the command-line in the root directory of your ownCloud installation, run it as your webserver user as follows, (assuming your webserver user is www-data):

```
{occ-command-example-prefix} configreport:generate
```

This will generate the report and send it to STDOUT. You can optionally pipe the output to a file and then attach it to an email to ownCloud support, by running the following command:

```
{occ-command-example-prefix} configreport:generate > generated-config-
report.txt
```

Alternatively, you could generate the report and email it all in one command, by running:

```
{occ-command-example-prefix} configreport:generate | mail \
  -s "configuration report" \
  -r <the email address to send from> \
  support@owncloud.com
```



These commands are not available in single-user (maintenance) mode.

Command Line Installation

ownCloud can be installed entirely from the command line. After downloading the tarball and copying ownCloud into the appropriate directories, or after installing ownCloud packages (See Linux Package Manager Installation and Manual Installation on Linux) you can use occ commands in place of running the graphical Installation Wizard.



These instructions assume that you have a fully working and configured webserver. If not, please refer to the documentation on configuring configure-web-server for detailed instructions.

Apply correct permissions to your ownCloud directories; see strong_permissions. Then choose your occ options. This lists your available options:

```
{occ-command-example-prefix} occ
ownCloud is not installed - only a limited number of commands are available
ownCloud version 10.0.8

Usage:
 [options] command [arguments]

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

Available commands:
 check                Check dependencies of the server environment
 help                 Displays help for a command
 list                 Lists commands
 status               Show some status information
 app
  app:check-code      Check code to be compliant
 l10n
  l10n:createjs       Create javascript translation files for a given app
 maintenance
  maintenance:install  Install ownCloud
```

Command Description

Display your maintenance:install options.

```
{occ-command-example-prefix} help maintenance:install
ownCloud is not installed - only a limited number of commands are available
Usage:
```

```
maintenance:install [--database=["..."]] [--database-connection-string=["..."]] \
                    [--database-name=["..."]] [--database-host=["..."]] \
                    [--database-user=["..."]] [--database-pass=["..."]] \
                    [--database-table-prefix=["..."]] [--admin-user=["..."]] \
                    [--admin-pass=["..."]] [--data-dir=["..."]]
```

Options

--database	Supported database type (default: sqlite). The supported values are: • mysql: MySQL/MariaDB • oci: Oracle (*ownCloud Enterprise edition only*) • pgsql: PostgreSQL • sqlite: SQLite3 (*ownCloud Community edition only*)
--database -connection-string	An Oracle-specific connection string.  As soon as this parameter is provided, other parameters like database-host and database-name are not used and do not need to be provided. For example: **Example** `sales= (DESCRIPTION= (ADDRESS= (PROTOCOL=tcp)(HOST=sales-server)(PORT=1521)) (CONNECT_DATA= (SERVICE_NAME=sales.us.acme.com)))`
--database-name	Name of the database.
--database-host	Hostname of the database (default: localhost).
--database-user	User name to connect to the database.
--database-pass	Password of the database user.
--database-table -prefix	Prefix for all tables (default: oc_).
--admin-user	Password of the admin account.
--data-dir	Path to data directory (default: /var/www/owncloud/data).

Example

This example completes the installation:

```
cd /var/www/owncloud/
{occ-command-example-prefix} maintenance:install \
  --database "mysql" \
  --database-name "owncloud"  \
  --database-user "root" \
  --database-pass "password" \
  --admin-user "admin" \
  --admin-pass "password"
ownCloud is not installed - only a limited number of commands are available
ownCloud was successfully installed
```

Command Line Upgrade

These commands are available only after you have downloaded upgraded packages or tar archives, and before you complete the upgrade. List all options, like this example on CentOS Linux:

Command Description

```
{occ-command-example-prefix} upgrade --help
Usage:
upgrade [options]
```

Options

--major	Automatically update apps to new major versions during minor updates of ownCloud Server.
--no-app-disable	Skip disabling of third party apps.

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

```
{occ-command-example-prefix} upgrade
ownCloud 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 <activity> ...
Updated <activity> to 2.1.0
Update successful
Turned off maintenance mode
```

Note how it details the steps. Enabling verbosity displays timestamps:

```
{occ-command-example-prefix} upgrade -v
ownCloud or one of the apps require upgrade - only a limited number of commands
are available
2017-06-23T09:06:15+0000 Turned on maintenance mode
2017-06-23T09:06:15+0000 Checked database schema update
2017-06-23T09:06:15+0000 Checked database schema update for apps
2017-06-23T09:06:15+0000 Updated database
2017-06-23T09:06:15+0000 Updated <files_sharing> to 0.6.6
2017-06-23T09:06:15+0000 Update successful
2017-06-23T09:06:15+0000 Turned off maintenance mode
```

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

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

DAV Commands

A set of commands to create address books, calendars, and to migrate address books:

```
dav
 dav:cleanup-chunks            Cleanup outdated chunks
 dav:create-addressbook        Create a dav address book
 dav:create-calendar           Create a dav calendar
 dav:sync-birthday-calendar    Synchronizes the birthday calendar
 dav:sync-system-addressbook   Synchronizes users to the system address book
```



These commands are not available in single-user (maintenance) mode.

dav:cleanup-chunks cleans up outdated chunks (uploaded files) more than a certain number of days old. By default, the command cleans up chunks more than 2 days old. However, by supplying the number of days to the command, the range can be increased. For example, in the example below, chunks older than 10 days will be removed.

```
{occ-command-example-prefix} dav:cleanup-chunks 10

# example output
Cleaning chunks older than 10 days(2017-11-08T13:13:45+00:00)
Cleaning chunks for admin
  0 [>----------------------------]
```

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

```
{occ-command-example-prefix} dav:create-addressbook molly mollybook
```

This example creates a new calendar for molly:

```
{occ-command-example-prefix} dav:create-calendar molly mollycal
```

Molly will immediately see these on her Calendar and Contacts pages. Your existing calendars and contacts should migrate automatically when you upgrade. If something goes wrong you can try a manual migration. First delete any partially-migrated calendars or address books. Then run this command to migrate user's contacts:

```
{occ-command-example-prefix} dav:migrate-addressbooks [user]
```

Run this command to migrate calendars:

```
{occ-command-example-prefix} dav:migrate-calendars [user]
```

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

```
{occ-command-example-prefix} dav:sync-birthday-calendar bernie
```

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

```
{occ-command-example-prefix} dav:sync-system-addressbook
```

Database Conversion

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

```
db
 db:convert-type          Convert the ownCloud database to the newly configured one
```

You need:

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

This is example converts SQLite to MySQL/MariaDB:

```
{occ-command-example-prefix} db:convert-type mysql oc_dbuser 127.0.0.1
oc_database
```



For a more detailed explanation see converting database types.

Full Text Search

Use these commands when you manage full text search related tasks.

Command Description

```
search
  search:index:create    Create initial Search Index for one or all users.
                         This command could not update the search index correctly
                         after the initial indexing.
  search:index:rebuild   Rebuild the search index for a given User.
                         If you want to rebuild the whole index, run "search:index:reset"
                         and then "search:index:build --all"
  search:index:reset     Reset the index
  search:index:update    Update the search index by running all pending background
jobs.
```

Create the Full Text Search Index

The command search:index:create creates the initial full text search index for one, or all, users.

```
{occ-command-example-prefix} search:index:create <user_id> (<user_id>)...
```

Arguments

<user_id>	The id of the user (or space-separated list of user ids) to create a full text search index for. A full text search index is created for all users, if this value is omitted.

Example 1

This example creates a full text search index for the user with user id testuser.

```
{occ-command-example-prefix} search:index:create testuser

Indexing user testuser
```

Example 2

This example creates a full text search index for the users with user ids admin and testuser.

```
{occ-command-example-prefix} search:index:create admin testuser

Indexing user admin
Indexing user testuser
```

Rebuild the Full Text Search Index

The command search:index:rebuild rebuilds the full text search index for one, multiple, or all users.

```
{occ-command-example-prefix} search:index:rebuild <user_id> (<user_id>)...
```

Arguments

<user_id>	The id of the user (or space-separated list of user ids) to rebuild a full text search index for.
--all	Rebuild the index for *all* users.
-f --force	Use this option to reset the index without further questions.

Example 1

This example rebuilds the full text search index for the user with user id testuser.

Listing 8. Rebuild the index for a single user

```
{occ-command-example-prefix} search:index:rebuild testuser

Indexing user testuser

This will delete all full text search index data for testuser! Do you want to proceed?
  [0] no
  [1] yes
 > 1
Rebuilding full text search Index for testuser
```

Example 2

This example rebuilds the full text search index for the users with user ids admin and testuser.

Listing 9. Rebuild the index for multiple users

```
docker-compose exec owncloud occ search:index:rebuild admin testuser
This will delete all search index data for admin, testuser! Do you want to proceed?
  [0] no
  [1] yes
 > 1
Rebuilding Search Index for admin
Rebuilding Search Index for testuser
```

Rebuild the Entire Index

The entire index can be rebuilt by running the following two commands:

```
{occ-command-example-prefix} search:index:reset
{occ-command-example-prefix} search:index:build --all
```

Reset the Full Text Search Index

The command search:index:reset resets (recreates and clears) the full text search index for all users.

```
{occ-command-example-prefix} search:index:reset
```

Arguments

-f --force	Use this option to reset the index without further questions.

Example

```
{occ-command-example-prefix} search:index:reset
This will delete the whole search index! Do you want to proceed?
  [0] no
  [1] yes
 > 1
Search index has been reset.
```

Update the Full Text Search Index

The command search:index:update updates the full text search index by running all pending background jobs.

```
{occ-command-example-prefix} search:index:update
```

Arguments

-q --quiet	Suppress all output from the command.

Example

This example updates the full text search index for all users.

```
{occ-command-example-prefix} search:index:update
Start Updating the Elastic search index:
No pending jobs found.
```

Enable and Disable App Mode

To do an initial full indexing without the full text search_elastic app interfering, it can be put in passive mode.

```
{occ-command-example-prefix} config:app:set search_elastic mode \
   --value passive
```

When the search_elastic app is in passive mode:

- The administrator will be able to run occ commands.
- The search_elastic app will not index any changes by itself.
- Search results will still be based on the core search.

Switching back to active mode can be done by running the following command:

```
{occ-command-example-prefix} config:app:set search_elastic mode \
   --value active
```

Configure Full Text Search to Only Index Metadata

If you only want to use the search_elastic app as a more scalable filenames search, you can disable content indexing by setting nocontent to true (default is false), as in the

example below.

```
{occ-command-example-prefix} config:app:set search_elastic nocontent --value
true
```



if this setting is reverted to false after being set to true, all files must be reindexed. Setting it to true does *not* require reindexing.

Encryption

occ includes a complete set of commands for managing encryption.

```
encryption
 encryption:change-key-storage-root  Change key storage root
 encryption:decrypt-all              Disable server-side encryption and decrypt all files
 encryption:disable                  Disable encryption
 encryption:enable                   Enable encryption
 encryption: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:list-modules             List all available encryption modules
 encryption:migrate                  initial migration to encryption 2.0
 encryption:recreate-master-key      Replace existing master key with new one.
Encrypt the file system with
                                     newly created master key
 encryption:select-encryption-type   Select the encryption type. The encryption
types available are: masterkey and
                                     user-keys. There is also no way to disable it again.
 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

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

```
{occ-command-example-prefix} app:enable encryption
{occ-command-example-prefix} encryption:enable
{occ-command-example-prefix} encryption:status
 - enabled: true
 - defaultModule: OC_DEFAULT_MODULE
```

Change Key Storage Root

encryption:change-key-storage-root is for moving your encryption keys to a different folder. It takes one argument, newRoot, which defines your new root folder. The folder must exist, and the path is relative to your root ownCloud directory.

```
{occ-command-example-prefix} encryption:change-key-storage-root ../../etc/oc-keys
```

You can see the current location of your keys folder:

```
{occ-command-example-prefix} encryption:show-key-storage-root
Current key storage root:  default storage location (data/)
```

List Modules

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

Encrypt All

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

Arguments

-y or --yes	Answer yes to all questions. This argument automatically answers, potential, questions with "yes", which is particularly important for automated deployments with Ansible or similar tools.

Decrypt All

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

```
{occ-command-example-prefix} encryption:decrypt freda
```

Users must have enabled recovery keys on their Personal pages. You must first put your ownCloud server into single-user mode, using the maintenance commands, to prevent any user activity until decryption is completed.

Arguments

-m=[METHOD]	Accepts the methods: recovery or password If the *recovery* method is chosen, then the recovery password will be used to decrypt files. If the *password* method is chosen, then individual user passwords will be used to decrypt files.
-c=[COMMAND]	Accepts the commands: yes or no This lets the command know whether to ask for permission to continue or not.

Fix Encrypted Version

encryption:fix-encrypted-version fixes the encrypted version of files, if the encrypted file(s) are not downloadable, for a given user.

Arguments

user	The id of the user whose files need fixing.

Method Descriptions

Recovery method

This method reads the value from the environment variable OC_RECOVERY_PASSWORD. This variable bounds the value of recovery password set in the encryption page. If this variable is not set the recovery process will be halted. This has to be used for decrypting all users. While opting recovery method user should not forget to set OC_RECOVERY_PASSWORD in the shell.

Password method

This method reads the value from the environment variable OC_PASSWORD. This variable bounds the value of user password. The password which user uses to login to oC account. When password method is opted the user needs to set this variable in the shell.

Continue Option Description

The continue option can be used to by pass the permissions asked like yes or no while decrypting the file system. If the user is sure about what he/she is doing with the command and would like to proceed, then -c yes when provided to the command would not ask permissions. If -c no is passed to the command, then permissions would be asked to the user. It becomes interactive.

Use encryption:disable to disable your encryption module. You must first put your ownCloud server into single-user mode to prevent any user activity.

encryption:migrate migrates encryption keys after a major ownCloud version upgrade. You may optionally specify individual users in a space-delimited list. See encryption configuration to learn more.

encryption:recreate-master-key decrypts the ownCloud file system, replaces the existing master key with a new one, and encrypts the entire ownCloud file system with the new master key. Given the size of your ownCloud filesystem, this may take some time to complete. However, if your filesystem is quite small, then it will complete quite quickly. The -y switch can be supplied to automate acceptance of user input.

Federation Sync

Synchronize the address books of all federated ownCloud servers.

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

```
{occ-command-example-prefix} federation:sync-addressbooks
```



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

File Operations

occ has five commands for managing files in ownCloud.

```
files
 files:check-cache          Check if the target file exists in the primary storage
 files:checksums:verify     Get all checksums in filecache and compares them by
                            recalculating the checksum of the file.
 files:cleanup              Deletes orphaned file cache entries.
 files:scan                 Rescans the filesystem.
 files:transfer-ownership   All files and folders are moved to another user
                            - outgoing shares are moved as well (incoming shares are
                            not moved as the sharing user holds the ownership of the
respective files).
```



These commands are not available in single-user (maintenance) mode.

The files:check-cache command

The main purpose for this command is to clean the cache for objectstores (objectstore and files_primary_S3 apps) as primary storage, but it is not limited to only those. It can be used for any other storage type as long it is the primary storage.

Files in the primary storage could be deleted outside of ownCloud, leaving information in ownCloud's file cache. This command intends to check if the target file can be read from the primary backend storage and if not, allows you to remove the information cached.



Removing files directly from the primary storage is not supported and should not happen. As such, the cases where you need to run this command should be extremely rare. This is why this command is only provided to check for one file instead of scanning the whole of ownCloud's filesystem.

```
{occ-command-example-prefix} files:check-cache --help
 Usage:
  files:check-cache [options] [--] <uid> <target-file>
```

Arguments

uid	The user (user id) who owns the file
target-file	The file we want to check

Options

--remove	Remove the file from the cache if it's missing in the backend

Examples of checking files for user maria:

```
{occ-command-example-prefix} files:check-cache maria welcome.txt
welcome.txt has been accessed properly
```

```
{occ-command-example-prefix} files:check-cache maria
maria@smbhome/myfile.txt
Ignoring maria@smbhome/myfile.txt because it is shared or not inside the primary
storage
```

The files:checksums:verify command

ownCloud supports file integrity checking, by computing and matching checksums. Doing so ensures that transferred files arrive at their target in the exact state as they left their origin.

In some rare cases, wrong checksums are written to the database which leads to synchronization issues, such as with the Desktop Client. To mitigate such problems a new command is available: occ files:checksums:verify.

Executing the command recalculates checksums, either for all files of a user or within a specified filesystem path on the designated storage. It then compares them with the values in the database. The command also offers an option to repair incorrect checksum values (-r, --repair).



Executing this command might take some time depending on the file count.

Below is sample output that you can expect to see when using the command.

```
{occ-command-example-prefix} files:checksums:verify
This operation might take very long.
Mismatch for files/welcome.txt:
 Filecache:   SHA1:eeb2c08011374d8ad4e483a4938e1aa1007c089d
MD5:368e3a6cb99f88c3543123931d786e21 ADLER32:c5ad3a63
 Actual:  SHA1:da39a3ee5e6b4b0d3255bfef95601890afd80709
MD5:d41d8cd98f00b204e9800998ecf8427e ADLER32:00000001
Mismatch for thumbnails/9/2048-2048-max.png:
 Filecache:   SHA1:2634fed078d1978f24f71892bf4ee0e4bd0c3c99
MD5:dd249372f7a68c551f7e6b2615d49463 ADLER32:821230d4
 Actual:  SHA1:da39a3ee5e6b4b0d3255bfef95601890afd80709
MD5:d41d8cd98f00b204e9800998ecf8427e ADLER32:00000001
```

Options

-r, --repair	Repair filecache-entry with mismatched checksums.
-u, --user=USER	Specific user to check.
-p, --path=PATH	Path to check relative to user folder. [default: ""]. For example, if the user's id was "john" and the --path value was "tree/apple", the command would check the ownCloud directory /john/files/tree/apple.

The files:cleanup command

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

The files:scan command

The files:scan command

- Scans for new files.
- Scans not fully scanned files.
- Repairs file cache holes.
- Updates the file cache.

File scans can be performed per-user, for a space-delimited list of users, for groups of users, and for all users.

```
{occ-command-example-prefix} files:scan --help
Usage:
  files:scan [options] [--] [<user_id>]...
```

Arguments

user_id	Will rescan all files of the given user(s).

Options

--output=[OUTPUT]	The output format to use (plain, json or json_pretty, default is plain).
-p --path=[PATH]	Limit rescan to this path, eg. --path="/alice/files/Music", the user_id is determined by the path and the user_id parameter and --all are ignored.
--group=[GROUP]	Scan user(s) under the group(s). This option can be used as --group=foo --group=bar to scan groups foo and bar (multiple values allowed)
-g --groups=[GROUP]	Scan user(s) under the group(s). This option can be used as --groups=foo,bar to scan groups foo and bar (multiple values allowed separated by commas)
-q --quiet	Do not output any message.
--all	Will rescan all files of all known users.
--repair	Will repair detached filecache entries (slow).
--unscanned	Only scan files which are marked as not fully scanned.



If not using --quiet, statistics will be shown at the end of the scan.

The --path Option

When using the --path option, the path must be in one of the following formats:

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

For example:

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

In the example above, the user_id alice is determined implicitly from the path component given. To get a list of scannable mounts for a given user, use the following command:

```
{occ-command-example-prefix} files_external:list user_id
```



Mounts are only scannable at the point of origin. Scanning of shares including federated shares is not necessary on the receiver side and therefore not possible.



Mounts based on session credentials can not be scanned as the users credentials are not available to the occ command set.

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

The --repair Option

As noted above, repairs can be performed for individual users, groups of users, and for all users in an ownCloud installation. What's more, repair scans can be run even if no files are known to need repairing and if one or more files are known to be in need of repair. Two examples of when files need repairing are:

- If folders have the same entry twice in the web UI (known as a '*ghost folder*'), this can also lead to strange error messages in the desktop client.
- If entering a folder doesn't seem to lead into that folder.



We strongly suggest that you backup the database before running this command.

The --repair option can be run within two different scenarios:

- Requiring a downtime when used on all affected storages at once.
- Without downtime, filtering by a specified User Id.

The following commands show how to enable single user mode, run a repair file scan in bulk on all storages, and then disable single user mode. This way is much faster than running the command for every user separately, but it requires single user mode.

```
{occ-command-example-prefix} maintenance:singleuser --on
{occ-command-example-prefix} files:scan --all --repair
{occ-command-example-prefix} maintenance:singleuser --off
```

The following command filters by the storage of the specified user.

```
{occ-command-example-prefix} files:scan USERID --repair
```



If many users are affected, it could be convenient to create a shell script, which iterates over a list of User ID's.

The files:transfer-ownership command

You may transfer all files and shares from one user to another, including to a user who has never logged in yet. This is useful before removing a user. For example, to move all files from <source-user> to <destination-user>, use the following command:

```
{occ-command-example-prefix} files:transfer-ownership <source-user>
<destination-user>
```

You can also move a limited set of files from <source-user> to <destination-user> by making use of the --path switch, as in the example below. Ownership of folder/to/move and all files and folders which it contains will be transferred to <destination-user>.

```
{occ-command-example-prefix} files:transfer-ownership --path="folder/to/move" \
  <source-user> \
  <destination-user>
```

When using this command, please keep the following in mind:

1. The directory provided to the --path switch **must** exist inside data/<source-user>/files.
2. The directory and its contents won't be moved as-is between the users. It will be moved into the destination user's files directory, into a directory name which follows the format: transferred from <source-user> on <timestamp>. Using the example above, it will be stored under: data/<destination-user>/files/transferred from <source-user> on 20170426_124510/
3. Currently file versions can't be transferred. Only the latest version of moved files will appear in the destination user's account.

Files External

These commands replace the data/mount.json configuration file used in ownCloud releases before 9.0. Commands for managing external storage.

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

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

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



These commands are only available when the "External storage support" app (files_external) is enabled. It is not available in single-user (maintenance) mode.

files_external:list

List configured mounts.

Usage

```
files_external:list [--show-password] [--full] [-a|--all] [-s|--short] [--] [<user_id>]
```

Arguments

user_id	User ID to list the personal mounts for, if no user is provided admin mounts will be listed.

Options

--show-password	User to add the mount configurations for, if not set the mount will be added as system mount.
--full	Don’t save the imported mounts, only list the new mounts.
-a, --all	Show both system-wide mounts and all personal mounts.
-s, --short	Show only a reduced mount info.
--output=[OUTPUT]	The output format to use (plain, json or json_pretty, default is plain).

Example

```
{occ-command-example-prefix} files_external:list user_1 --short
+----------+-----------------+----------+
| Mount ID | Mount Point     | Type     |
+----------+-----------------+----------+
| 1        | /mount_1        | Personal |
| 2        | /mount_2        | Personal |
+----------+-----------------+----------+
```

files_external:applicable

Manage applicable users and groups for a mount.

Usage

```
files_external:applicable
    [--add-user     ADD-USER]
    [--remove-user  REMOVE-USER]
    [--add-group    ADD-GROUP]
    [--remove-group REMOVE-GROUP]
    [--remove-all]
    [--output       [OUTPUT]]
    [--]
    <mount_id>
```

Arguments

mount_id	Can be obtained using occ files_external:list.

Options

--add-user	user to add as applicable (multiple values allowed).
--remove-user	user to remove as applicable (multiple values allowed).
--add-group	group to add as applicable (multiple values allowed).
--remove-group	group to remove as applicable (multiple values allowed).
--remove-all	Set the mount to be globally applicable.
--output=[OUTPUT]	The output format to use (plain, json or json_pretty, default is plain).

files_external:backends

Show available authentication and storage backends.

Usage

```
files_external:backends [options]
    [--]
    [<type>]
    [<backend>]
```

Arguments

type	Only show backends of a certain type. Possible values are authentication or storage.
backend	Only show information of a specific backend.

Options

--output=[OUTPUT]	The output format to use (plain, json or json_pretty, default is plain.

files_external:config

Manage backend configuration for a mount.

Usage

```
files_external:config [options]
    [--]
    <mount_id>
    <key>
    [<value>]
```

Arguments

mount_id	The ID of the mount to edit.
key	Key of the config option to set/get.
value	Value to set the config option to, when no value is provided the existing value will be printed.

Options

--output=[OUTPUT]	The output format to use (*plain*, *json* or *json_pretty*. The default is plain).

files_external:create

Create a new mount configuration.

Usage

```
files_external:create [options]
  [--]
  <mount_point>
  <storage_backend>
  <authentication_backend>
```

Arguments

mount_point	Mount point for the new mount.
storage_backend	Storage backend identifier for the new mount, see occ files_external:backends for possible values.
authentication_backend	Authentication backend identifier for the new mount, see occ files_external:backends for possible values.

Options

--user=[USER]	User to add the mount configurations for, if not set the mount will be added as system mount.
-c, --config=[CONFIG]	Mount configuration option in key=value format (multiple values allowed).
--dry	Don't save the imported mounts, only list the new mounts.
--output=[OUTPUT]	The output format to use (plain, json or json`pretty). The default is plain.

Storage Backend Details

Storage Backend	Identifier
Windows Network Drive	windows_network_drive
WebDav	dav
Local	local
ownCloud	owncloud
SFTP	sftp
Amazon S3	amazons3
Dropbox	dropbox
Google Drive	googledrive
SMB / CIFS	smb

Authentication Details

Authentication method	Identifier, name, configuration
Log-in credentials, save in session	password::sessioncredentials
Log-in credentials, save in database	password::logincredentials

Authentication method	Identifier, name, configuration
User entered, store in database	password::userprovided (*)
Global Credentials	password::global
None	null::null
Builtin	builtin::builtin
Username and password	password::password
OAuth1	oauth1::oauth1 (*)
OAuth2	oauth2::oauth2 (*)
RSA public key	publickey::rsa (*)
OpenStack	openstack::openstack (*)
Rackspace	openstack::rackspace (*)
Access key (Amazon S3)	amazons3::accesskey (*)

(*) - Authentication methods require additional configuration.



Each Storage Backend needs its corresponding authentication methods.

files_external:delete

Delete an external mount.

Usage

```
files_external:delete [options] [--] <mount_id>
```

Arguments

mount_id	The ID of the mount to edit.

Options

-y, --yes	Skip confirmation.
--output=[OUTPUT]	The output format to use (plain, json or json_pretty, default is plain).

files_external:export

Usage

```
files_external:export [options] [--] [<user_id>]
```

Arguments

user_id	User ID to export the personal mounts for, if no user is provided admin mounts will be exported.

Options

-a, --all	Show both system wide mounts and all personal mounts.

files_external:import

Import mount configurations.

Usage

```
files_external:import [options] [--] <path>
```

Arguments

path	Path to a json file containing the mounts to import, use - to read from stdin.

Options

--user=[USER]	User to add the mount configurations for, if not set the mount will be added as system mount.
--dry	Don't save the imported mounts, only list the new mounts.
--output=[OUTPUT]	The output format to use (*plain*, *json* or *json_pretty*, default is *plain*).

files_external:option

Manage mount options for a mount.

Usage

```
files_external:option <mount_id> <key> [<value>]
```

Arguments

mount_id	The ID of the mount to edit.
key	Key of the mount option to set/get.
value	Value to set the mount option to, when no value is provided the existing value will be printed.

files_external:verify

Verify mount configuration.

Usage

```
files_external:verify [options] [--] <mount_id>
```

Arguments

mount_id	The ID of the mount to check.

Options

-c, --config=[CONFIG]	Additional config option to set before checking in key=value pairs, required for certain auth backends such as login credentials (multiple values allowed).
--output=[OUTPUT]	The output format to use (*plain*, *json* or *json_pretty*, default is plain).

Full Text Search

Use these commands when you manage full text search related tasks.

Command Description

```
search
  search:index:create    Create initial Search Index for one or all users.
                         This command could not update the search index correctly
                         after the initial indexing.
  search:index:rebuild   Rebuild the search index for a given User.
                         If you want to rebuild the whole index, run "search:index:reset"
                         and then "search:index:build --all"
  search:index:reset     Reset the index
  search:index:update    Update the search index by running all pending background
jobs.
```

Create the Full Text Search Index

The command search:index:create creates the initial full text search index for one, or all, users.

```
{occ-command-example-prefix} search:index:create <user_id> (<user_id>)...
```

Arguments

<user_id>	The id of the user (or space-separated list of user ids) to create a full text search index for. A full text search index is created for all users, if this value is omitted.

Example 1

This example creates a full text search index for the user with user id testuser.

```
{occ-command-example-prefix} search:index:create testuser

Indexing user testuser
```

Example 2

This example creates a full text search index for the users with user ids admin and testuser.

```
{occ-command-example-prefix} search:index:create admin testuser

Indexing user admin
Indexing user testuser
```

Rebuild the Full Text Search Index

The command search:index:rebuild rebuilds the full text search index for one, multiple, or all users.

```
{occ-command-example-prefix} search:index:rebuild <user_id> (<user_id>)...
```

Arguments

<user_id>	The id of the user (or space-separated list of user ids) to rebuild a full text search index for.
--all	Rebuild the index for *all* users.
-f --force	Use this option to reset the index without further questions.

Example 1

This example rebuilds the full text search index for the user with user id testuser.

Listing 10. Rebuild the index for a single user

```
{occ-command-example-prefix} search:index:rebuild testuser

Indexing user testuser

This will delete all full text search index data for testuser! Do you want to proceed?
  [0] no
  [1] yes
 > 1
Rebuilding full text search Index for testuser
```

Example 2

This example rebuilds the full text search index for the users with user ids admin and testuser.

Listing 11. Rebuild the index for multiple users

```
docker-compose exec owncloud occ search:index:rebuild admin testuser
This will delete all search index data for admin, testuser! Do you want to proceed?
  [0] no
  [1] yes
 > 1
Rebuilding Search Index for admin
Rebuilding Search Index for testuser
```

Rebuild the Entire Index

The entire index can be rebuilt by running the following two commands:

```
{occ-command-example-prefix} search:index:reset
{occ-command-example-prefix} search:index:build --all
```

Reset the Full Text Search Index

The command search:index:reset resets (recreates and clears) the full text search index for all users.

```
{occ-command-example-prefix} search:index:reset
```

Arguments

-f --force	Use this option to reset the index without further questions.

Example

```
{occ-command-example-prefix} search:index:reset
This will delete the whole search index! Do you want to proceed?
  [0] no
  [1] yes
 > 1
Search index has been reset.
```

Update the Full Text Search Index

The command search:index:update updates the full text search index by running all pending background jobs.

```
{occ-command-example-prefix} search:index:update
```

Arguments

-q --quiet	Suppress all output from the command.

Example

This example updates the full text search index for all users.

```
{occ-command-example-prefix} search:index:update
Start Updating the Elastic search index:
No pending jobs found.
```

Enable and Disable App Mode

To do an initial full indexing without the full text search_elastic app interfering, it can be put in passive mode.

```
{occ-command-example-prefix} config:app:set search_elastic mode \
   --value passive
```

When the search_elastic app is in passive mode:

- The administrator will be able to run occ commands.
- The search_elastic app will not index any changes by itself.
- Search results will still be based on the core search.

Switching back to active mode can be done by running the following command:

```
{occ-command-example-prefix} config:app:set search_elastic mode \
   --value active
```

Configure Full Text Search to Only Index Metadata

If you only want to use the search_elastic app as a more scalable filenames search, you can disable content indexing by setting nocontent to true (default is false), as in the example below.

```
{occ-command-example-prefix} config:app:set search_elastic nocontent --value
true
```



if this setting is reverted to false after being set to true, all files must be reindexed. Setting it to true does *not* require reindexing.

Group Commands

The group commands provide a range of functionality for managing ownCloud groups. This includes creating and removing groups and managing group membership. Group names are case-sensitive, so "Finance" and "finance" are two different groups.

The full list of commands is:

```
group
 group:add                  Adds a group
 group:add-member           Add members to a group
 group:delete               Deletes the specified group
 group:list                 List groups
 group:list-members         List group members
 group:remove-member        Remove member(s) from a group
```

Creating Groups

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

```
group:add groupname
```

This example adds a new group, called "Finance":

```
{occ-command-example-prefix} group:add Finance
 Created group "Finance"
```

Listing Groups

You can list the names of existing groups with the group:list command. The syntax is:

```
group:list [options] [<search-pattern>]
```

Groups containing the search-pattern string are listed. Matching is not case-sensitive. If you do not provide a search-pattern then all groups are listed.

Options

--output=[OUTPUT]	Output format (plain, json or json_pretty, default is plain) [default: "plain"].

This example lists groups containing the string "finance".

```
{occ-command-example-prefix} group:list finance
 - All-Finance-Staff
 - Finance
 - Finance-Managers
```

This example lists groups containing the string "finance" formatted with json_pretty.

```
{occ-command-example-prefix} group:list --output=json_pretty finance
[
  "All-Finance-Staff",
  "Finance",
  "Finance-Managers"
]
```

Listing Group Members

You can list the user IDs of group members with the group:list-members command. The syntax is:

```
group:list-members [options] <group>
```

Options

--output=[OUTPUT]	Output format (plain, json or json_pretty, default is plain) [default: "plain"].

This example lists members of the "Finance" group.

```
{occ-command-example-prefix} group:list-members Finance
 - aaron: Aaron Smith
 - julie: Julie Jones
```

This example lists members of the Finance group formatted with json_pretty.

```
{occ-command-example-prefix} group:list-members --output=json_pretty Finance
{
  "aaron": "Aaron Smith",
  "julie": "Julie Jones"
}
```

Adding Members to Groups

You can add members to an existing group with the group:add-member command. Members must be existing users. The syntax is:

```
group:add-member [-m|--member [MEMBER]] <group>
```

This example adds members "aaron" and "julie" to group "Finance":

```
{occ-command-example-prefix} group:add-member --member aaron --member julie
Finance
  User "aaron" added to group "Finance"
  User "julie" added to group "Finance"
```

You may attempt to add members that are already in the group, without error. This allows you to add members in a scripted way without needing to know if the user is already a member of the group. For example:

```
{occ-command-example-prefix} group:add-member --member aaron --member julie
--member fred Finance
  User "aaron" is already a member of group "Finance"
  User "julie" is already a member of group "Finance"
  User fred" added to group "Finance"
```

Removing Members from Groups

You can remove members from a group with the group:remove-member command. The syntax is:

```
group:remove-member [-m|--member [MEMBER]] <group>
```

This example removes members "aaron" and "julie" from group "Finance".

```
{occ-command-example-prefix} group:remove-member --member aaron --member
julie Finance
  Member "aaron" removed from group "Finance"
  Member "julie" removed from group "Finance"
```

You may attempt to remove members that have already been removed from the group, without error. This allows you to remove members in a scripted way without needing to know if the user is still a member of the group. For example:

```
{occ-command-example-prefix} group:remove-member --member aaron --member
fred Finance
  Member "aaron" could not be found in group "Finance"
  Member "fred" removed from group "Finance"
```

Deleting a Group

To delete a group, you use the group:delete command, as in the example below:

```
{occ-command-example-prefix} group:delete Finance
```

Integrity Check

Apps which have an official tag **must** be code signed. Unsigned official apps won't be installable anymore. Code signing is optional for all third-party applications.

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

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

```
{occ-command-example-prefix} integrity:sign-app \
  --privateKey=/Users/karlmay/contacts.key \
  --certificate=/Users/karlmay/CA/contacts.crt \
  --path=/Users/karlmay/Programming/contacts
```

Verify your app:

```
{occ-command-example-prefix} integrity:check-app --path=/pathto/app appname
```

When it returns nothing, your app is signed correctly. When it returns a message then there is an error.

integrity:sign-core is for ownCloud core developers only.



See code signing to learn more.

l10n, Create Javascript Translation Files for Apps

This command creates JavaScript and JSON translation files for ownCloud applications.



The command does not update existing translations if the source translation file has been updated. It only creates translation files when none are present for a given language.

```
l10n
 l10n:createjs              Create Javascript translation files for a given app
```

The command takes two parameters; these are:

- app: the name of the application.
- lang: the output language of the translation files; more than one can be supplied.

To create the two translation files, the command reads translation data from a source PHP translation file.

A Working Example

In this example, we'll create Austrian German translations for the Comments app.



This example assumes that the ownCloud directory is /var/www/owncloud and that it uses ownCloud's standard apps directory, app.

First, create a source translation file in /var/www/owncloud/apps/comments/l10n, called de_AT.php. In it, add the required translation strings, as in the following example. Refer to the developer documentation on creating translation files, if you're not familiar with creating them.

```
<?php
// The source string is the key, the translated string is the value.
$TRANSLATIONS = [
  "Share" => "Freigeben"
];
$PLURAL_FORMS = "nplurals=2; plural=(n != 1);";
```

After that, run the following command to create the translation.

```
{occ-command-example-prefix} l10n:createjs comments de_AT
```

This will generate two translation files, de_AT.js and de_AT.json, in /var/www/owncloud/apps/comments/l10n.

Create Translations in Multiple Languages

To create translations in multiple languages simultaneously, supply multiple languages to the command, as in the following example:

```
{occ-command-example-prefix} l10n:createjs comments de_AT de_DE hu_HU es fr
```

Logging Commands

These commands view and configure your ownCloud logging preferences.

```
log
 log:manage     Manage logging configuration
 log:owncloud   Manipulate ownCloud logging backend
```

Command Description

Run log:owncloud to see your current logging status:

```
{occ-command-example-prefix} log:owncloud
Log backend ownCloud: enabled
Log file: /opt/owncloud/data/owncloud.log
Rotate at: disabled
```

Options

--enable	Enable this logging backend.
--file=[FILE]	Set the log file path.
--rotate-size=[ROTATE-SIZE]	Set the file size for log rotation, 0 = disabled.

Use the --enable option to turn on logging. Use --file to set a different log file path. Set your rotation by log file size in bytes with --rotate-size; 0 disables rotation. Run log:manage to set your logging backend, log level, and timezone: The defaults are owncloud, Warning, and UTC.

Options for log:manage:

--backend=[BACKEND]	Set the logging backend [owncloud, syslog, errorlog].
--level=[LEVEL]	Set the log level [debug, info, warning, error, fatal].

Log level can be adjusted by entering the number or the name:

```
{occ-command-example-prefix} log:manage --level 4
{occ-command-example-prefix} log:manage --level error
```



Setting the log level to debug (0) can be used for finding the cause of an error, but should not be the standard as it increases the log file size.

Managing Background Jobs

Use the background:queue command to manage background jobs.

```
background:queue
 background:queue:delete     Delete a job from the queue
 background:queue:execute    Run a single background job from the queue
 background:queue:status     List queue status
```

Deleting a Background Job

The command background:queue:delete deletes a queued background job. It requires the job id of the job to be deleted.

```
background:queue:delete <Job ID>
```

Arguments

Job ID	ID of the job to be deleted



Deleting a job cannot be undone. Be sure that you want to delete the job before doing so.

This example deletes queued background job #12.

```
{occ-command-example-prefix} background:queue:delete 12

Job has been deleted.
```

Executing a Background Job

The command background:queue:execute executes a queued background job. It requires the job id of the job to be executed.

```
background:queue:execute [options] [--] <Job ID>
```

Arguments

Job ID	ID of the job to be deleted

Options

-f --force	Force run the job even if within timing interval
--accept-warning	No warning about the usage of this command will be displayed

This example executes queued background job #12.

```
{occ-command-example-prefix} background:queue:execute 12

This command is for maintenance and support purposes.
This will run the specified background job now. Regular scheduled runs of the job
will
continue to happen at their scheduled times.
If you still want to use this command please confirm the usage by entering: yes
yes
Found job: OCA\UpdateNotification\Notification\BackgroundJob with ID 12
Running job...
Finished in 0 seconds
```

List Queued Backgroundjobs

The command background:queue:status will list queued background jobs, including details when it last ran.

```
background:queue:status
```

This example lists the queue status:

```
{occ-command-example-prefix} background:queue:status

+----+-----------------------------------------------+-------------------------+--------------+
| Id | Job                                           | Last run                | Job Arguments |
+----+-----------------------------------------------+-------------------------+--------------+
| 1  | OCA\Files\BackgroundJob\ScanFiles             | 2018-06-13T15:15:04+00:00 |
|
| 2  | OCA\Files\BackgroundJob\DeleteOrphanedItems   | 2018-06-
13T15:15:04+00:00 |              |
| 3  | OCA\Files\BackgroundJob\CleanupFileLocks      | 2018-06-
13T15:15:04+00:00 |              |
| 4  | OCA\DAV\CardDAV\SyncJob                       | 2018-06-12T19:15:02+00:00 |
|
| 5  | OCA\Federation\SyncJob                        | 2018-06-12T19:15:02+00:00 |
|
| 6  | OCA\Files_Sharing\DeleteOrphanedSharesJob     | 2018-06-
13T15:15:04+00:00 |              |
| 7  | OCA\Files_Sharing\ExpireSharesJob             | 2018-06-12T19:15:02+00:00 |
|
| 8  | OCA\Files_Trashbin\BackgroundJob\ExpireTrash  | 2018-06-
13T15:15:04+00:00 |              |
| 9  | OCA\Files_Versions\BackgroundJob\ExpireVersions | 2018-06-
13T15:15:04+00:00 |              |
| 10 | OCA\UpdateNotification\Notification\BackgroundJob | 2018-06-
12T19:15:03+00:00 |              |
| 11 | OC\Authentication\Token\DefaultTokenCleanupJob | 2018-06-
13T15:15:04+00:00 |              |
+----+-----------------------------------------------+-------------------------+--------------+
```

Maintenance Commands

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

```
maintenance
 maintenance:data-fingerprint       Update the systems data-fingerprint after a
backup is restored
 maintenance:mimetype:update-db     Update database mimetypes and update
filecache
 maintenance:mimetype:update-js     Update mimetypelist.js
 maintenance:mode                   Set maintenance mode
 maintenance:repair                 Repair this installation
 maintenance:singleuser             Set single user mode
 maintenance:update:htaccess        Updates the .htaccess file
```

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

Web browsers to continue working.

```
{occ-command-example-prefix} maintenance:mode --on
{occ-command-example-prefix} maintenance:mode --off
```

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

```
{occ-command-example-prefix} maintenance:singleuser --on
Single user mode enabled
```

Turn it off when you're finished:

```
{occ-command-example-prefix} maintenance:singleuser --off
Single user mode disabled
```

Run maintenance:data-fingerprint to tell desktop and mobile clients that a server backup has been restored. This command changes the ETag for all files in the communication with sync clients, informing them that one or more files were modified. After the command completes, users will be prompted to resolve any conflicts between newer and older file versions.

Migration Steps Command

You can run migration steps with the migrations command.

```
{occ-command-example-prefix} migrations:execute <app> <version>
```

Arguments

app	Name of the app this migration command shall work on.
version	The version to execute.

Example

This example executes the migration step for the core app:

```
{occ-command-example-prefix} migrations:execute core 20181220085457
```

Mimetype Update Commands

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

Notifications

If you want to send notifications to users or groups use the following command.

```
1 notifications
2   notifications:generate   Generates a notification.
```

Command Description

```
{occ-command-example-prefix} notifications:generate [-u|--user USER] [-g|--group
GROUP] [-l|--link <linktext>] [--] <subject> [<message>]
```

Arguments:

subject	The notification subject - maximum 255 characters.
message	A more extended message - maximum 4000 characters.
linktext	A link to an HTML page.

Options

-u [USER] --user=[USER]	User id to whom the notification shall be sent.
-g [GROUP] --group=[GROUP]	Group id to whom the notification shall be sent.
-l [LINK] --link=[LINK]	A link associated with the notification.

At least one user or group must be set. A link can be useful for notifications shown in client apps. Example:

```
{occ-command-example-prefix} notifications:generate -g Office "Emergency Alert"
"Rebooting in 5min"
```

Poll Incoming Federated Shares For Updates

This command must be used if received federated shares are being referenced by desktop clients but not regularly accessed via the webUI. This is because, for performance reasons, federated shares do not update automatically. Instead, federated share directories are only updated when users browse them using the webUI.

ownCloud and system administrators can use the incoming-shares:poll command to poll federated shares for updates.



The command polls all received federated shares, so does not require a path.

```
{occ-command-example-prefix} incoming-shares:poll
```

When using federation, it is recommended to execute occ incoming-shares:poll regularly using Cron jobs. The time interval between executions is a trade-off between the availability of changes in federated shares and resource consumption; which naturally depends a lot on the number of federated shares and the frequency of changes within those shares.

Executing the command once every 12 hours *should* be safe enough for most instances. However, the interval can be reduced to once every 2 hours, for instances with a small number of federated shares.

Depending on the desired resource consumption, this value should be lowered or increased based on individual expectations. To find a value that fits a specific setup, it is recommended to execute the command once, measure the execution time and set the interval, so that the background job can finish before the next execution is triggered.

Security

Use these commands when you manage security related tasks. Routes displays all routes of ownCloud. You can use this information to grant strict access via firewalls, proxies or load balancers etc.

Command Description

```
security:routes [options]
```

Options

--output=[OUTPUT]	Output format (plain, json or json-pretty, default is plain).
--with-details	Adds more details to the output.

Example 1:

```
{occ-command-example-prefix} security:routes
```

```
+-------------------------------------------------------+----------------+
| Path                                                  | Methods        |
+-------------------------------------------------------+----------------+
| /apps/federation/auto-add-servers                       | POST           |
| /apps/federation/trusted-servers                      | POST           |
| /apps/federation/trusted-servers/<id>                   | DELETE         |
| /apps/files/                                        | GET            |
| /apps/files/ajax/download.php                         |                |
...
```

Example 2:

```
{occ-command-example-prefix} security:routes --output=json-pretty
```

```
[
  {
     "path": "\/apps\/federation\/auto-add-servers",
     "methods": [
        "POST"
     ]
  },
```

Example 3:

```
{occ-command-example-prefix} security:routes --with-details
```

```
+-----------------------------------------+---------+-----------------------------------------------------
+------------------------------+
| Path                                    | Methods | Controller                                        |
Annotations                    |
+-----------------------------------------+---------+-----------------------------------------------------
+------------------------------+
| /apps/files/api/v1/sorting                | POST    |
OCA\Files\Controller\ApiController::updateFileSorting | NoAdminRequired               |
| /apps/files/api/v1/thumbnail/{x}/{y}/{file} | GET     |
OCA\Files\Controller\ApiController::getThumbnail      |
NoAdminRequired,NoCSRFRequired |
...
```

The following commands manage server-wide SSL certificates. These are useful when you create federation shares with other ownCloud servers that use self-signed certificates.

```
security:certificates         List trusted certificates
security:certificates:import  Import trusted certificate
security:certificates:remove  Remove trusted certificate
```

This example lists your installed certificates:

```
{occ-command-example-prefix} security:certificates
```

Import a new certificate:

```
{occ-command-example-prefix} security:certificates:import /path/to/certificate
```

Remove a certificate:

```
{occ-command-example-prefix} security:certificates:remove [certificate name]
```

Sharing

This is an occ command to cleanup orphaned remote storages. To explain why this is necessary, a little background is required. While shares are able to be deleted as a normal matter of course, remote storages with shared:: are not included in this process.

This might not, normally, be a problem. However, if a user has re-shared a remote share which has been deleted it will. This is because when the original share is deleted, the remote re-share reference is not. Internally, the fileid will remain in the file cache and storage for that file will not be deleted.

As a result, any user(s) who the share was re-shared with will now get an error when trying to access that file or folder. That's why the command is available. So, to cleanup all orphaned remote storages, run it as follows:

```
{occ-command-example-prefix} sharing:cleanup-remote-storages
```

You can also set it up to run as a background job.



These commands are not available in single-user (maintenance) mode.

System

```
 system
  system:cron          Execute background jobs as cron
```

```
{occ-command-example-prefix} -h system:cron
-Usage:
  system:cron [options]
```

Options

-p, --progress	Shows a progress bar - for use in manual execution. Do not use when executing from crontab

To execute background jobs using cron, you can use the system:cron command, as in the following example:

```
{occ-command-example-prefix} system:cron
```

If the --progress or -p argument is specified, then progress output will be displayed in the console, as in the example below.

```
Executing: 12 - OCA\UpdateNotification\Notification\BackgroundJob
  13 [------------>--------------]
```

If neither of these arguments is provided, no output will be displayed by the command.

Displaying progress information is useful when you want visual confirmation that background jobs have been executed. However, in a non-interactive environment, such as crontab, it should not be used.

Updating an Existing System Cron Configuration

If you have already automated background jobs via Cron, you must update the relevant crontab entry using the example below as a guide.

```
# Instead of the following configuration
/usr/bin/php -f /path/to/your/owncloud/cron.php

# Use the following one instead
{occ-command-example-prefix} system:cron
```

This command does not work if:

- Maintenance or Admin-only (single user) modes are enabled
- Background jobs are disabled

Trashbin

These commands are only available when the 'Deleted files' app (files_trashbin) is enabled. These commands are not available in single-user (maintenance) mode.

```
trashbin
 trashbin:cleanup   Remove deleted files
 trashbin:expire    Expires the users trash bin
```

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

```
{occ-command-example-prefix} trashbin:cleanup
Remove all deleted files
Remove deleted files for users on backend Database
 freda
 molly
 stash
 rosa
 edward
```

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

```
{occ-command-example-prefix} trashbin:cleanup molly freda
Remove deleted files of   molly
Remove deleted files of   freda
```

trashbin:expire deletes only expired files according to the trashbin_retention_obligation setting in config.php (see the "Deleted Files" section documentation). The default is to delete expired files for all users, or you may list users in a space-delimited list.

User Commands

The user commands provide a range of functionality for managing ownCloud users. This includes: creating and removing users, resetting user passwords, displaying a report which shows how many users you have, and when a user was last logged in. The full list, of commands is:

```
user
 user:add                    Adds a user
 user:delete                 Deletes the specified user
 user:disable                Disables the specified user
 user:enable                 Enables the specified user
 user:inactive               Reports users who are known to owncloud,
                           but have not logged in for a certain number of days
 user:lastseen               Shows when the user was logged in last time
 user:list                   List users
 user:list-groups            List groups for a user
 user:modify                 Modify user details
 user:report                 Shows how many users have access
 user:resetpassword              Resets the password of the named user
 user:setting                Read and modify user application settings
 user:sync                   Sync local users with an external backend service
```

Creating Users

You can create a new user with the user:add command.

```
{occ-command-example-prefix} user:add \
   [--password-from-env] \
   [--display-name [DISPLAY-NAME]] \
   [--email [EMAIL]] \
   [-g|--group [GROUP]] \
   [--] \
   <uid>
```

Arguments

uid	User ID used to login (must only contain a-z, A-Z, 0-9, -, _ and @).
--password-from-env	Read the password from the OC_PASS environment variable. A password is **not required**, *if* an email address is provided. If a password is not provided, a temporary one will be generated. It cannot be set to 0.
--display-name=[DISPLAY-NAME]	This corresponds to the **Full Name** on the Users page in your ownCloud Web UI.

--email=[EMAIL]	Email address for the user (optional). The user will be emailed a link to set their password, *if* email is configured correctly.
-g [GROUP] --group=[GROUP]	The groups the user should be added to. The group will be created if it does not exist. Multiple values are allowed.

Command Examples

This example adds new user, Layla Smith, and adds her to the users and db-admins groups. If either group does not exist, it is created.

Listing 12. Create a user with a password, email address, and display name, and add them to two groups

```
{occ-command-example-prefix} user:add \
  --display-name="Layla Smith" \
  --group="users" \
  --group="db-admins" \
  --email=layla.smith@example.com layla
  Enter password:
  Confirm password:
  The user "layla" was created successfully
  Display name set to "Layla Smith"
  Email address set to "layla.smith@example.com"
  User "layla" added to group "users"
  User "layla" added to group "db-admins"
```

Listing 13. Create a user with a temporary password (the user will receive a link to set their password).

```
{occ-command-example-prefix} user:add \
  --display-name "Layla Smith" \
  --email "************" \
  --group "users" \
  --group "db-admins" layla

The user "layla" was created successfully
Display name set to "Layla Smith"
Email address set to "*************"
User layla added to group users
User layla added to group db-admins
```

Deleting A User

To delete a user, you use the user:delete command.

```
{occ-command-example-prefix} user:delete <uid>
```

Arguments

uid	The username.

```
{occ-command-example-prefix} user:delete fred
```

Disable Users

Admins can disable users via the occ command too:

```
{occ-command-example-prefix} user:disable <username>
```



Once users are disabled, their connected browsers will be disconnected. Use the following command to enable the user again:

Enable Users

```
{occ-command-example-prefix} user:enable <username>
```

Finding Inactive Users

To view a list of users who've not logged in for a given number of days, use the user:inactive command.

```
{occ-command-example-prefix} user:inactive [options] [--] <days>
```

Arguments

<days>	The number of days (integer) that the user has not logged in since.

Options

--output=[OUTPUT]	Output format (plain, json or json_pretty, default is plain) [default: "plain"].

The example below searches for users inactive for five days, or more.

```
{occ-command-example-prefix} user:inactive 5
```

By default, this will generate output in the following format:

```
- 0:
  - uid: admin
  - displayName: admin
  - inactiveSinceDays: 5
```

You can see a counting number starting with 0, the user's user id, display name, and

the number of days they've been inactive. If you're passing or piping this information to another application for further processing, you can also use the --output switch to change its format. Using the output option json will render the output formatted as follows.

```
[{"uid":"admin","displayName":"admin","inactiveSinceDays":5}]
```

Using the output option json_pretty will render the output formatted as follows.

```
[
    {
        "uid": "admin",
        "displayName": "admin",
        "inactiveSinceDays": 5
    }
]
```

Finding the User's Last Login

To view a user's most recent login, use the user:lastseen command:

```
{occ-command-example-prefix} user:lastseen <uid>
```

Arguments

uid	The username.

Example

```
{occ-command-example-prefix} user:lastseen layla
  layla's last login: 09.01.2015 18:46
```

Listing Users

You can list existing users with the user:list command.

```
{occ-command-example-prefix} user:list [options] [<search-pattern>]
```

User IDs containing the search-pattern string are listed. Matching is not case-sensitive. If you do not provide a search-pattern then all users are listed.

Options

--output=[OUTPUT]	Output format (plain, json or json-pretty, default is plain).

-a [ATTRIBUTES] --attributes=[ATTRIBUTES]	Adds more details to the output. Allowed attributes, multiple values possible: uid, displayName, email, quota, enabled, lastLogin, home, backend, cloudId, searchTerms [default: [displayName]]

This example lists user IDs containing the string ron

```
{occ-command-example-prefix} user:list ron
 - aaron: Aaron Smith
```

The output can be formatted in JSON with the output option json or json_pretty.

```
{occ-command-example-prefix} user:list --output=json_pretty
 {
  "aaron": "Aaron Smith",
  "herbert": "Herbert Smith",
  "julie": "Julie Jones"
 }
```

This example lists all users including the attribute enabled.

```
{occ-command-example-prefix} user:list -a enabled
 - admin: true
 - foo: true
```

Listing Group Membership of a User

You can list the group membership of a user with the user:list-groups command.

```
{occ-command-example-prefix} user:list-groups [options] [--] <uid>
```

Arguments

uid	User ID.

Options

--output=[OUTPUT]	Output format (plain, json or json-pretty, default is plain).

Examples

This example lists group membership of user julie:

```
{occ-command-example-prefix} user:list-groups julie
 - Executive
 - Finance
```

The output can be formatted in JSON with the output option json or json_pretty:

```
{occ-command-example-prefix} user:list-groups --output=json_pretty julie
[
  "Executive",
  "Finance"
]
```

Modify User Details

This command modifies either the users username or email address.

```
{occ-command-example-prefix} user:modify [options] [--] <uid> <key> <value>
```

Arguments

uid	User ID used to login.
key	Key to be changed. Valid keys are: displayname and email.
value	The new value of the key.

All three arguments are mandatory and can not be empty. Example to set the email address:

```
{occ-command-example-prefix} user:modify carla email foobar@foo.com
```

The email address of carla is updated to foobar@foo.com.

Generating a User Count Report

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

```
{occ-command-example-prefix} user:report
```

There are no arguments and no options beside the default once to parametrize the output.

```
{occ-command-example-prefix} user:report
+-----------------+----+
| User Report     |    |
+-----------------+----+
| Database        | 12 |
| LDAP            | 86 |
|                 |    |
| total users     | 98 |
|                 |    |
| user directories | 2  |
+-----------------+----+
```

Setting a User's Password

```
{occ-command-example-prefix} user:resetpassword [options] [--] <user>
```

	Password changes automatically log out **all** connected browsers/devices.

Arguments

uid	The user's name.

Options

--password-from-env	Read the password from the OC_PASS environment variable.
--send-email	The email-id set while creating the user, will be used to send link for password reset. This option will also display the link sent to user.
--output-link	The link to reset the password will be displayed.

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

	To use password-from-env you must run as "real" root, rather than sudo, because sudo strips environment variables.

	To use send-email, the ownCloud instance must have email access fully configured.

Examples

Add a new user, called Fred Jones:

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

You can reset any user's password, including administrators (see Reset Admin Password):

```
{occ-command-example-prefix} user:resetpassword layla
  Enter a new password:
  Confirm the new password:
Successfully reset password for layla
```

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

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

This example emails a password reset link to the user. Additionally, when the command completes, it outputs the password reset link to the console:

```
{occ-command-example-prefix} user:resetpassword \
  --send-email \
  --output-link \
  layla
The password reset link is: http://localhost:{std-port-
http}/index.php/lostpassword/reset/form/rQAlCjNeQf3aphA6Hraq2/layla
```

If the specified user does not have a valid email address set, then the following error will be output to the console, and the email will not be sent:

```
Email address is not set for the user layla
```

User Application Settings

To manage application settings for a user, use the user:setting command. This command provides the ability to:

- Retrieve all settings for an application
- Retrieve a single setting
- Set a setting value
- Delete a setting

```
{occ-command-example-prefix} user:setting [options] [--] <uid> [<app>] [<key>]
```

If you're new to the user:setting command, the descriptions for the app and key arguments may not be completely transparent. So, here's a lengthier description of both.

Argument	Description
app	When an value is supplied, user:setting limits the settings displayed, to those for that, specific, application - assuming that the application is installed, and that there are settings available for it. Some example applications are core, files_trashbin, and user_ldap. A complete list, unfortunately, cannot be supplied, as it is impossible to know the entire list of applications which a user could, potentially, install.
key	This value specifies the setting key to be manipulated (set, retrieved, or deleted) by the user:setting command.

Retrieving User Settings

To retrieve all settings for a user, you need to call the user:setting command and supply at least the user's username.

```
{occ-command-example-prefix} user:setting <uid> [<app>] [<key>]
```

Arguments

uid	User ID used to login.
app	Restrict listing the settings for a given app. [default: ""].
key	Setting key to set, get or delete [default: ""].

Example for all settings set for a given user

```
{occ-command-example-prefix} user:setting layla
  - core:
    - lang: en
  - login:
    - lastLogin: 1465910968
  - settings:
    - email: layla@example.tld
```

Here we see that the user has settings for the application core, when they last logged in, and what their email address is. Example for all settings set restricted to application core for a given user

```
{occ-command-example-prefix} user:setting layla core
 - core:
    - lang: en
```

In the output, you can see that one setting is in effect, lang, which is set to en.
Example for all settings set restricted to application core, key lang for a given user

```
{occ-command-example-prefix} user:setting layla core lang en
```

This will display the value for that setting, such as en.

Setting and Deleting a Setting

```
{occ-command-example-prefix} user:setting [options] [--] <uid> [<app>] [<key>]
```

Arguments

uid	User ID used to login.
app	Restrict the settings to a given app. [default: ""].
key	Setting key to set, get or delete [default: ""].

Options

--output=[OUTPUT]	Output format (plain, json or json-pretty, default is plain).
--ignore-missing-user	Use this option to ignore errors when the user does not exist.
--default-value=[DEFAULT-VALUE]	If no default value is set and the config does not exist, the command will exit with 1. Only applicable on get.
--value=[VALUE]	The new value of the setting.
--update-only	Only updates the value, if it is not set before, it is not being added.
--delete	Specify this option to delete the config.
--error-if-not-exists	Checks whether the setting exists before deleting it.



In case you want to change the email address, use the user:modify command.

Here's an example of how you would set the language of the user layla.

```
{occ-command-example-prefix} user:setting layla core lang --value=en
```

Deleting a setting is quite similar to setting a setting. In this case, you supply the username, application (or setting category) and key as above. Then, in addition, you supply the --delete flag.

```
{occ-command-example-prefix} user:setting layla core lang --delete
```

Syncing User Accounts

This command syncs users stored in external backend services, such as *LDAP*, *Shibboleth*, and *Samba*, with ownCloud's, internal user database. However, it's not essential to run it regularly, unless you have a large number of users whose account properties have changed in a backend outside of ownCloud. When run, it will pick up changes from alternative user backends, such as LDAP, where properties like cn or display name have changed, and sync them with ownCloud's user database. If accounts are found that no longer exist in the external backend, you are given the choice of either removing or disabling the accounts.



It's also one of the commands that you should run on a regular basis to ensure that your ownCloud installation is running optimally.



This command replaces the old show-remnants functionality, and brings the LDAP feature more in line with the rest of ownCloud's functionality.

Usage

```
user:sync [options] [--] [<backend-class>]
```

Synchronize users from a given backend to the accounts table.

Arguments:

backend-class	The quoted PHP class name for the backend, e.g., - LDAP: "OCA\User_LDAP\User_Proxy" - Samba: "OCA\User\SMB" - Shibboleth: "OCA\User_Shibboleth\UserBackend"

Options

-l, --list	List all enabled backend classes.
-u [UID] --uid=[UID]	Sync only the user with the given user id.
-s, --seenOnly	Sync only seen users.
-c, --showCount	Calculate user count before syncing.
-m [MISSING-ACCOUNT-ACTION] --missing-account-action[=MISSING-ACCOUNT-ACTION]	Action to take if the account isn't connected to a backend any longer. Options are disable and remove. Note that removing the account will also remove the stored data and files for that account
-r, --re-enable	When syncing multiple accounts re-enable accounts that are disabled in ownCloud but available in the synced backend.

Below are examples of how to use the command with an *LDAP*, *Samba*, and *Shibboleth* backend.

LDAP

```
{occ-command-example-prefix} user:sync "OCA\User_LDAP\User_Proxy"
```

Samba

```
{occ-command-example-prefix} user:sync "OCA\User\SMB" -vvv
```

Below are examples of how to use the command with the **LDAP** backend along with example console output.

Example 1

```
{occ-command-example-prefix} user:sync "OCA\User_LDAP\User_Proxy" -m disable
-r
  Analysing all users ...
     6 [============================]

  No removed users have been detected.

  No existing accounts to re-enable.

  Insert new and update existing users ...
     4 [============================]
```

Example 2

```
{occ-command-example-prefix} user:sync "OCA\User_LDAP\User_Proxy" -m disable
-r
  Analysing all users ...
     6 [============================]

  Following users are no longer known with the connected backend.
  Disabling accounts:
  9F625F70-08DD-4838-AD52-7DE1F72DBE30, Bobbie, bobbie@example.org
disabled
  53CDB5AC-B02E-4A49-8FEF-001A13725777, David, dave@example.org disabled
  34C3F461-90FE-417C-ADC5-CE97FE5B8E72, Carol, carol@example.org disabled

  No existing accounts to re-enable.

  Insert new and update existing users ...
     1 [============================]
```

Example 3

```
{occ-command-example-prefix} user:sync "OCA\User_LDAP\User_Proxy" -m disable
-r
  Analysing all users ...
     6 [============================]

  Following users are no longer known with the connected backend.
  Disabling accounts:
  53CDB5AC-B02E-4A49-8FEF-001A13725777, David, dave@example.org skipped,
already disabled
  34C3F461-90FE-417C-ADC5-CE97FE5B8E72, Carol, carol@example.org skipped,
already disabled
  B5275C13-6466-43FD-A129-A12A6D3D9A0D, Alicia3, alicia3@example.org
disabled

  Re-enabling accounts:
  9F625F70-08DD-4838-AD52-7DE1F72DBE30, Bobbie, bobbie@example.org
enabled

  Insert new and update existing users ...
     1 [============================]
```

Example 4

```
{occ-command-example-prefix} user:sync "OCA\User_LDAP\User_Proxy" -m disable
-r
  Analysing all users ...
     6 [============================]

  No removed users have been detected.

  Re-enabling accounts:
  53CDB5AC-B02E-4A49-8FEF-001A13725777, David, dave@example.org enabled
  34C3F461-90FE-417C-ADC5-CE97FE5B8E72, Carol, carol@example.org enabled
  B5275C13-6466-43FD-A129-A12A6D3D9A0D, Alicia3, alicia3@example.org
enabled

  Insert new and update existing users ...
     4 [============================]
```

Example 5

```
{occ-command-example-prefix} user:sync "OCA\User_LDAP\User_Proxy" -m remove
```

Syncing via cron job

Here is an example for syncing with LDAP four times a day on Ubuntu:

```
crontab -e -u www-data

* */6 * * * /usr/bin/php /var/www/owncloud/occ user:sync -vvv \
    --missing-account-action="disable" \
    -n "OCA\User_LDAP\User_Proxy"
```

Versions



These commands are only available when the "Versions" app (files_versions) is enabled. These commands are not available in single-user (maintenance) mode.

versions:cleanup

versions:cleanup can delete all versioned files, as well as the files_versions folder, for either specific users, or for all users.

```
{occ-command-example-prefix} versions:cleanup [<user_id>]...
```

Options

user_id	Delete versions of the given user(s), if no user is given all versions will be deleted.

The example below deletes all versioned files for all users:

```
{occ-command-example-prefix} versions:cleanup
Delete all versions
Delete versions for users on backend Database
  freda
  molly
  stash
  rosa
  edward
```

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

```
{occ-command-example-prefix} versions:cleanup freda molly
Delete versions of   freda
Delete versions of   molly
```

versions:expire

versions:expire deletes only expired files according to the versions_retention_obligation setting in config.php (see the File versions section in config_sample_php_parameters). The default is to delete expired files for all users, or you may list users in a space-delimited list.

```
{occ-command-example-prefix} versions:expire [<user_id>]...
```

Options

user_id	Expire file versions of the given user(s), if no user is given file versions for all users will be expired.

Apps Commands

This command reference covers the ownCloud maintained apps commands.

Activity

The activity commands are used for automating activity notifications in ownCloud server.

```
activity
 activity:send-emails  Send all pending activity emails now
```

activity:send-emails

The activity:send-emails command sends all pending activity emails immediately, regardless of the time they are scheduled.

```
{occ-command-example-prefix} activity:send-emails
```

Anti-Virus

Marketplace URL: Anti-Virus

Use these commands to configure the Anti-Virus app. Parametrisation must be done with the occ config command set.

List the Current Settings

```
{occ-command-example-prefix} config:list files_antivirus
```

Set the Setting

To set a new value, use the command below and replace <Key> and value <Value> accordingly.

```
{occ-command-example-prefix} config:app:set files_antivirus <Key> --value
=<Value> --update-only
```

Antivirus Mode [string]

Antivirus Configuration.

Key	av_mode

Default	'executable'
Possible Values	'executable' 'daemon' 'socket'

Antivirus Socket [string]

Antivirus Socket.

Key	av_socket
Default	'/var/run/clamav/clamd.ctl'

Antivirus Host [string]

Hostname or IP address of Antivirus Host.

Key	av_host
Default	

Antivirus Port [integer]

Port number of Antivirus Host, 1-65535.

Key	av_port
Default	
Possible Values	1-65535

Antivirus Command Line Options [string]

Extra command line options (comma-separated).

Key	av_cmd_options
Default	

Antivirus Path to Executable [string]

Path to clamscan executable.

Key	av_path
Default	'/usr/bin/clamscan'

Antivirus Maximum Filesize [integer]

File size limit, -1 means no limit.

Key	av_max_file_size
Default	'-1'
Possible Values	'-1' integer number

Antivirus Maximum Stream Lenth [integer]

Max Stream Length.

Key	av_stream_max_length
Default	'26214400'

Antivirus Action [string]

When infected files were found during a background scan.

Key	av_infected_action
Default	'only_log'
Possible Values	'only_log' 'delete'

Antivirus Scan Process [string]

Define scan process.

Key	av_scan_background
Default	'true'
Possible Values	'true' 'false'

Brute Force Protection

Marketplace URL: Brute-Force Protection

Use these commands to configure the Brute Force Protection app. Parametrisation must be done with the occ config command set. The combination of uid and IP address is used to trigger the ban.

List the Current Settings

```
{occ-command-example-prefix} config:list brute_force_protection
```

Set the Setting

To set a new value, use the command below and replace <Key> and value <Value> accordingly.

```
{occ-command-example-prefix} config:app:set brute_force_protection <Key>
--value=<Value> --update-only
```

Fail Tolerance [attempts]

Number of wrong attempts to trigger the ban.

Key	brute_force_protection_fail_tolerance
Default	3

Time Treshold [seconds]

Time in which the number of wrong attempts must occur to trigger the ban.

Key	brute_force_protection_time_threshold
Default	60

Ban Period [seconds]

Time how long the ban will be active if triggered.

Key	brute_force_protection_ban_period
Default	300

Calendar

Marketplace URL: Calendar

For commands for managing the calendar, please see the DAV Command section in the occ core command set.

Contacts

Marketplace URL: Contacts

For commands for managing contacts, please see the DAV Command section in the occ core command set.

Data Exporter

This app is only available as a git clone. See the Data Exporter description for more information how to install this app. Import and export users from one ownCloud instance in to another. The export contains all user-settings, files and shares.

Export User Data

```
instance:export:user <userId> <exportDirectory>
```

Arguments

userId	User to export.
exportDirectory	Path to the directory to export data to.

Import User Data

```
instance:import:user [options] [--] <importDirectory>
```

Arguments

userId	User to export.
importDirectory	Path to the directory to import data from.

Options

-a [UID] --as=[UID]	Import the user under a different user id.

Migrate Shares

```
instance:export:migrate:share <userId> <remoteServer>
```

Arguments

userId	The exported userId whose shares we want to migrate.
remoteServer	The remote ownCloud server where the exported user is now, for example "https://myown.server:{std-port-http}/owncloud".

LDAP Integration

Marketplace URL: LDAP Integration

```
ldap
 ldap:check-user              Checks whether a user exists on LDAP.
 ldap:create-empty-config     Creates an empty LDAP configuration
 ldap:delete-config           Deletes an existing LDAP configuration
 ldap:search                  Executes a user or group search
 ldap:set-config              Modifies an LDAP configuration
 ldap:show-config             Shows the LDAP configuration
 ldap:test-config             Tests an LDAP configuration
```

Search for an LDAP user, using this syntax:

```
{occ-command-example-prefix} ldap:search [--group] [--offset="..."] [--limit="..."]
search
```

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

```
{occ-command-example-prefix} ldap:search "rob"
```

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

```
{occ-command-example-prefix} 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

whitespace is ignored.

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

```
{occ-command-example-prefix} ldap:check-user robert
```

ldap:check-user will not run a check when it finds a disabled LDAP connection. This prevents users that exist on disabled LDAP connections from being marked as deleted. If you know for sure 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.

```
{occ-command-example-prefix} ldap:check-user --force robert
```

ldap:create-empty-config creates an empty LDAP configuration. The first one you create has no configID, like this example:

```
{occ-command-example-prefix} ldap:create-empty-config
 Created new configuration with configID ''
```

This is a holdover from the early days, when there was no option to create additional configurations. The second, and all subsequent, configurations that you create are automatically assigned IDs.

```
{occ-command-example-prefix} ldap:create-empty-config
  Created new configuration with configID 's01'
```

Then you can list and view your configurations:

```
{occ-command-example-prefix} ldap:show-config
```

And view the configuration for a single configID:

```
{occ-command-example-prefix} ldap:show-config s01
```

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

```
{occ-command-example-prefix} ldap:delete  s01
Deleted configuration with configID 's01'
```

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

```
{occ-command-example-prefix} ldap:set-config s01 ldapAttributesForUserSearch
"cn;givenname;sn;displayname;mail"
```

The command takes the following format:

```
ldap:set-config <configID> <configKey> <configValue>
```

All of the available keys, along with default values for configValue, are listed in the table below.

Configuration	Setting
hasMemberOfFilterSupport	
hasPagedResultSupport	
homeFolderNamingRule	
lastJpegPhotoLookup	0
ldapAgentName	cn=admin,dc=owncloudqa,dc=com
ldapAgentPassword	*
ldapAttributesForGroupSearch	
ldapAttributesForUserSearch	
ldapBackupHost	
ldapBackupPort	
ldapBase	dc=owncloudqa,dc=com
ldapBaseGroups	dc=owncloudqa,dc=com
ldapBaseUsers	dc=owncloudqa,dc=com
ldapCacheTTL	600
ldapConfigurationActive	1
ldapDynamicGroupMemberURL	
ldapEmailAttribute	
ldapExperiencedAdmin	0
ldapExpertUUIDGroupAttr	
ldapExpertUUIDUserAttr	
ldapExpertUsernameAttr	ldapGroupDisplayName cn
ldapGroupFilter	ldapGroupFilterGroups
ldapGroupFilterMode	0
ldapGroupFilterObjectclass	
ldapGroupMemberAssocAttr	uniqueMember
ldapHost	ldap://host
ldapIgnoreNamingRules	

Configuration	Setting
ldapLoginFilter	(& objectclass=inetOrgPerson(uid=%uid))
ldapLoginFilterAttributes	
ldapLoginFilterEmail	0
ldapLoginFilterMode	0
ldapLoginFilterUsername	1
ldapNestedGroups	0
ldapOverrideMainServer	
ldapPagingSize	500
ldapPort	389
ldapQuotaAttribute	
ldapQuotaDefault	
ldapTLS	0
ldapUserDisplayName	displayName
ldapUserDisplayName2	
ldapUserFilter	objectclass=inetOrgPerson
ldapUserFilterGroups	
ldapUserFilterMode	0
ldapUserFilterObjectclass	inetOrgPerson
ldapUuidGroupAttribute	auto
ldapUuidUserAttribute	auto
turnOffCertCheck	0
useMemberOfToDetectMembership	1

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

```
{occ-command-example-prefix} ldap:test-config s01
The configuration is valid and the connection could be established!
```

```
{occ-command-example-prefix} config:app:set user_ldap updateAttributesInterval
--value=7200
```

In the example above, the interval is being set to 7200 seconds. Assuming the above example was used, the command would output the following:

```
Config value updateAttributesInterval for app user_ldap set to 7200
```

If you want to reset (or unset) the setting, then you can use the following command:

```
{occ-command-example-prefix} config:app:delete user_ldap
updateAttributesInterval
```

Reuse Existing LDAP Accounts if Available

If you want to allow new LDAP logins to attempt to reuse existing oc_accounts entries that match the resolved username attribute, and have backend set to User_Proxy, then set the reuse_accounts config setting to yes.

Below is an example of how to do so.

```
{occ-command-example-prefix} config:app:set user_ldap reuse_accounts
--value=yes
```

This functionality is valuable for several reasons; these are:

- It handles the situation of when admins mistakenly delete one or more user mappings, and subsequent logins then create new accounts.
- It allows auto-provisioned users with Shibboleth to be moved over to an LDAP server, but be able to continue using ownCloud.



== This functionality will not work in the following situations:

1. No user or group account exists with the supplied username.
2. A user or group account exists, but it uses a different backend. ==

Market

Marketplace URL: Market

The market commands *install*, *uninstall*, *list*, and *upgrade* applications from the ownCloud Marketplace.

```
market
  market:install    Install apps from the marketplace. If already installed and
                    an update is available the update will be installed.
  market:uninstall  Uninstall apps from the marketplace.
  market:list       Lists apps as available on the marketplace.
  market:upgrade    Installs new app versions if available on the marketplace
```



The user running the update command, which will likely be your webserver user, requires write permission for the /apps respectively apps-external folder.



If they don’t have write permission, the command may report that the update was successful, but it may silently fail.

These commands are not available in single-user (maintenance) mode. For more details please see the Maintenance Commands section in the occ core command set.

Install an Application

Applications can be installed both from the ownCloud Marketplace and from a local file archive.

Install Apps From The Marketplace

To install an application from the Marketplace, you need to supply the app's id, which can be found in the app's Marketplace URL. For example, the URL for *Two factor backup codes* is https://marketplace.owncloud.com/apps/twofactor_backup_codes. So its app id is twofactor_backup_codes.

```
{occ-command-example-prefix} market:install <ids> [option]
```

Arguments

ids	Ids of the apps

Options

-l [LOCAL] --local=[LOCAL]	Optional path to a local app package.

Install Apps From a File Archive

To install an application from a local file archive, you need to supply the path to the archive, and that you pass the -l switch. Only zip, gzip, and bzip2 archives are supported.

Usage Example

```
# Install an app from the marketplace.
{occ-command-example-prefix} market:install twofactor_backup_codes

# Install an app from a local archive.
{occ-command-example-prefix} market:install -l /mnt/data/richdocuments-
2.0.0.tar.gz
```



The target directory has to be **accessable to the webserver user** and you have to **enable** the app afterwards with the occ app:enable command.

Uninstall an Application

To uninstall an application use the following commands:

```
{occ-command-example-prefix} market:uninstall <ids>
```

Arguments

ids	Ids of the apps

List Apps From The Marketplace

This command lists apps available on the marketplace. It returns the ids if the apps.

```
{occ-command-example-prefix} market:list
```

Upgrade an Application

Install new app versions if available on the marketplace by using following commands:

```
{occ-command-example-prefix} market:upgrade <ids> [options]
```

Arguments

ids	Ids of the apps

Options

-l [LOCAL] --local=[LOCAL]	Optional path to a local app package.
--major	Allow update to a new major version.

Password Policy

Marketplace URL: Password Policy

Command to expire a user or group of users' passwords.

Command Description

```
{occ-command-example-prefix} user:expire-password <uid> [<expiredate>]
```

Arguments

uid	User ID.
expiredate	The date and time when a password expires, e.g. 2019-01-01 14:00:00 CET or -1 days.



The expiry date can be provided using any of PHP's supported date and time formats.

Options

-a, --all	Will add password expiry to all known users. uid and group option are discarded if the option is provided by user.
-u [UID] --uid=[UID]	The uid of the user to expire the password for. To expire the password of multiple users, pass the -u or --uid option multiple times, as in this example: --uid "Alice" --uid "Bob".

-g [GROUP] --group=[GROUP]	Add password expiry to user(s) in one or more groups. This option can be used as --group foo --group bar to add expiry passwords for users in multiple groups.

If an expiry date is not supplied, the password will expire with immediate effect. This is because the password will be set as being expired 24 hours before the command was run. For example, if the command was run at 2018-07-**12** 13:15:28 UTC, then the password's expiry date will be set to 2018-07-**11** 13:15:28 UTC.

After the command completes, console output, similar to that below, confirms when the user's password is set to expire.

```
The password for frank is set to expire on 2018-07-12 13:15:28 UTC.
```

Command Examples

```
# The password for user "frank" will be set as being expired 24 hours before the command was run.
{occ-command-example-prefix} user:expire-password -u frank

# Expire the user "frank"'s password in 2 days time.
{occ-command-example-prefix} user:expire-password -u frank '+2 days'

# Expire the user "frank"'s password on the 15th of August 2005, at 15:52:01 in the local timezone.
{occ-command-example-prefix} user:expire-password --uid frank '2005-08-15T15:52:01+00:00'

# Expire the user "frank"'s password on the 15th of August 2005, at 15:52:01 UTC.
{occ-command-example-prefix} user:expire-password --uid frank '15-Aug-05 15:52:01 UTC'
```

Caveats

Please be aware of the following implications of enabling or changing the password policy's "**days until user password expires**" option.

- Administrators need to run the occ user:expire-password command to initiate expiry for new users.
- Passwords will never expire for users who have **not** changed their initial password, because they do not have a password history. To force password expiration use the occ user:expire-password command.
- A password expiration date will be set after users change their password for the first time. To force password expiration use the occ user:expire-password command.
- Passwords changed for the first time, will expire based on the **active** password policy. If the policy is later changed, it will not update the password's expiry date to reflect the new setting.
- Password expiration dates of users where the administrator has run the occ user:expire-password command **won't** automatically update to reflect the policy change. In these cases, Administrators need to run the occ user:expire-password command again and supply a new expiry date.

Ransomware Protection (Enterprise Edition only)

Marketplace URL: Ransomware Protection

Use these commands to help users recover from a Ransomware attack. You can find more information about the application in the Ransomware Protection documentation.

Command Description

```
{occ-command-example-prefix} ransomguard:scan <timestamp> <user>
```

Arguments

<timestamp> <user>	Report all changes in a user's account, starting from timestamp.

```
{occ-command-example-prefix} ransomguard:restore <timestamp> <user>
```

Arguments

<timestamp> <user>	Revert all operations in a user account after a point in time.

```
{occ-command-example-prefix} ransomguard:lock <user>
```

Arguments

<user>	Set a user account as read-only for ownCloud and other WebDAV clients when malicious activity is suspected.

```
{occ-command-example-prefix} ransomguard:unlock <user>
```

Arguments

<user>	Unlock a user account after ransomware issues have been resolved.

S3 Objectstore

Marketplace URL: S3 Object Storage

List objects, buckets or versions of an object

```
{occ-command-example-prefix} s3:list
```

Arguments

bucket	Name of the bucket; it`s objects will be listed.

object	Key of the object; it`s versions will be listed.

Create a bucket as necessary to be used

```
{occ-command-example-prefix} s3:create-bucket
```

Arguments

bucket	Name of the bucket to be created

Options

update-configuration	If the bucket exists the configuration will be updated.
accept-warning	No warning about the usage of this command will be displayed.

SAML/SSO Shibboleth Integration (Enterprise Edition only)

Marketplace URL: SAML/SSO Integration

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

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

Two-factor Authentication

Marketplace URL: 2-Factor Authentication

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

Command Description

```
{occ-command-example-prefix} twofactor:disable <username>
```

To re-enable two-factor authentication again, use the following command:

```
{occ-command-example-prefix} twofactor:enable <username>
```

Language Configuration

In normal cases, ownCloud will automatically detect the language of the Web-GUI. If this does not work as expected, or you want to make sure that ownCloud always starts with a given language, you can use the **default_language** configuration parameter.

This parameter can be set in *config/config.php*

Parameters

```
'default_language' => 'en',
```

Please keep in mind, that this will not effect a users language preference, which can be configured under menu:Settings[Personal > General > Language] once he has logged in.

More supported languages can be found in directory *<ownCloud_root>/settings/l10n*. List all files with *ls *.js*. The language code to be used is the filename without extension.

Example:

```
en_GB.js --> en_GB
```

Please see Wikipedia for a match of language code to country.

Legal Settings Configuration

Introduction

Because of one or more legal frameworks around the world, some ownCloud instances may need to display links to both an Imprint as well as a Privacy Policy on all pages (both in the Web UI and within email templates). An Imprint document is a legally mandated statement of the ownership and authorship of the ownCloud installation. You can think of an Imprint as a rather fancy "**About Us**" page or an enhanced "**Terms and Conditions**" page; in Germany, this is known as an "**Impressum**".



Imprint and Privacy Policy links are shown on all **public** pages and in e-mail footers. Authenticated pages, such as files app or settings, do not show them.

Some of the more global legal frameworks prominent are:

- The GDPR
- The Australian Privacy Act 1988
- The Canadian Personal Information Protection and Electronic Data Act (PIPEDA)
- The California Online Privacy Protection Act (CalOPPA)
- The Children's Online Privacy Protection Rule (COPPA)

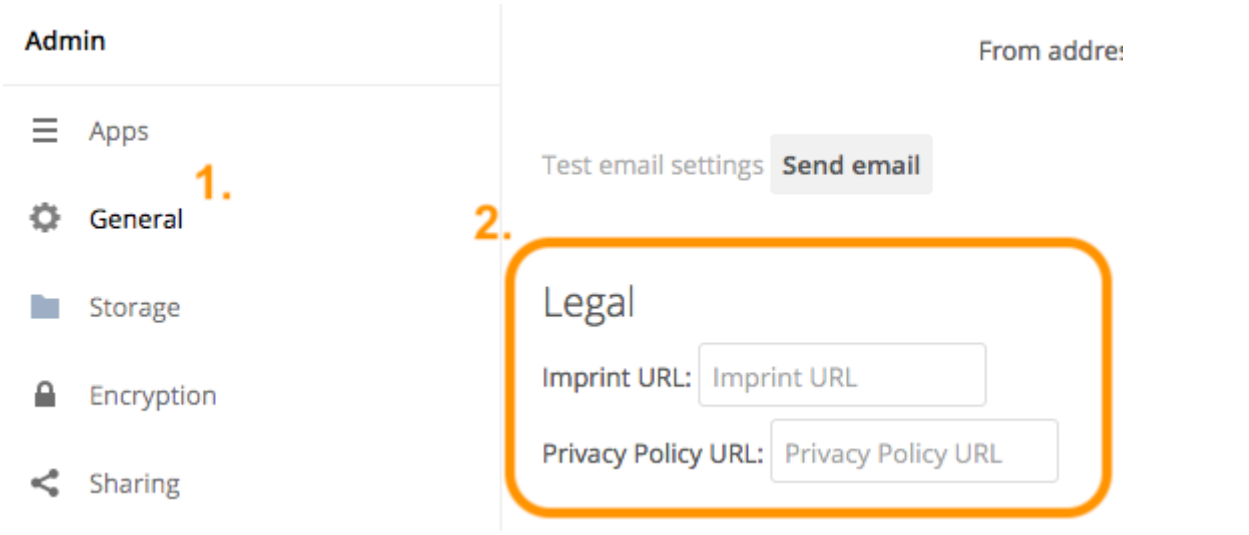
If you're required to link to either one of these, you can specify the link to them in two ways:

- Using the Web UI
- Using the Command Line

Using the Web UI

In the Web UI, under menu:Settings[Admin > General], under the heading "**Legal**", you can provide a link to an Imprint and a Privacy Policy URL, as you can see in the screenshot below.

Configuring Imprint and Privacy Policy URLs in the ownCloud Web UI.



The values entered will auto-save.

Using the Command Line

From the command line, you can use the occ config:app:get and occ config:app:set commands, as in the code sample below.

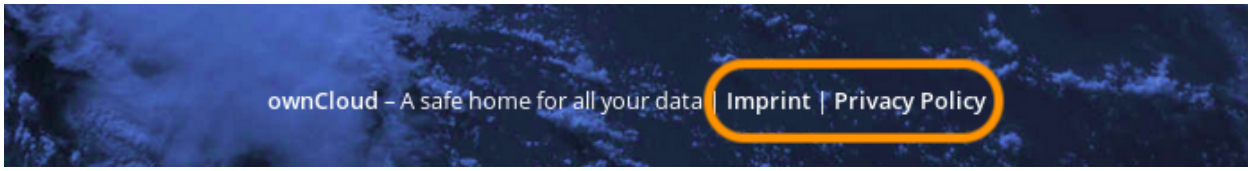
```
# Get the current values, if any, for the Imprint and Privacy Policy URLs
{occ-command-example-prefix} config:app:get core legal.imprint_url
{occ-command-example-prefix} config:app:get core legal.privacy_policy_url

# Set the Imprint and Privacy Policy URLs
{occ-command-example-prefix} config:app:set core legal.imprint_url
--value=new_value
{occ-command-example-prefix} config:app:set core legal.privacy_policy_url
--value=new_value
```

For more information about these commands, refer to the config command reference in the occ commands documentation.

Viewing the URLs

Customized email templates and themes that were built prior to ownCloud version 10.0.9 need to be updated.

Once the settings have been updated, you will see two links appear at the bottom the ownCloud login page, one for each option, as in the screenshot below.



Logging Configuration

Introduction

Use your ownCloud log to review system status, or to help debug problems. You may adjust logging levels, and choose between using the ownCloud log or your syslog.

Parameters

Logging levels range from **DEBUG**, which logs all activity, to **FATAL**, which logs only fatal errors.

- **0**: DEBUG: Debug, informational, warning, and error messages, and fatal issues.
- **1**: INFO: Informational, warning, and error messages, and fatal issues.
- **2**: WARN: Warning, and error messages, and fatal issues.
- **3**: ERROR: Error messages and fatal issues.
- **4**: FATAL: Fatal issues only.

By default the log level is set to **2** (WARN). Use **DEBUG** when you have a problem to diagnose, and then reset your log level to a less-verbose level, as **DEBUG** outputs a lot of information, and can affect your server performance.

Logging level parameters are set in the config/config.php file, or on the Admin page of your ownCloud Web GUI.

ownCloud

All log information will be written to a separate log file which can be viewed using the log viewer on your Admin page. By default, a log file named **owncloud.log** will be created in the directory which has been configured by the **datadirectory** parameter in config/config.php.

The desired date format can optionally be defined using the **logdateformat** parameter in config/config.php. By default the PHP date function parameter *c* is used, and therefore the date/time is written in the format *2013-01-10T15:20:25+02:00*. By using the date format in the example below, the date/time format will be written in the format *January 10, 2013 15:20:25*.

```
"log_type" => "owncloud",
"logfile" => "owncloud.log",
"loglevel" => "3",
"logdateformat" => "F d, Y H:i:s",
```

syslog

All log information will be sent to your default syslog daemon.

```
"log_type" => "syslog",
"logfile" => "",
"loglevel" => "3",
```

The syslog format can be changed to remove or add information. In addition to the %replacements% below %level% can be used, but it is used as a dedicated parameter to the syslog logging facility anyway.

```
'log.syslog.format' =>
'[%reqId%][%remoteAddr%][%user%][%app%][%method%][%url%] %message%',

  For the old syslog message format use:

'log.syslog.format' => '{%app%} %message%',
```

Conditional Logging Level Increase

You can configure the logging level to automatically increase to debug when the first condition inside a condition block is met. All conditions are optional !

- shared_secret: A unique token. If a http(s) request parameter named log_secret is added to the request and set to this token, 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.
- logfile: The log message invoked gets redirected to this logfile when a condition above is met.

Notes regarding the logfile key:

1. If no logfile is defined, the standard logfile is used.
2. Not applicable when using syslog.

> The following example demonstrates how all three conditions can look like. The first one that matches triggers the condition block writing the log entry to the defined logfile.

```
'log.conditions' => [
  [
    'shared_secret' => '57b58edb6637fe3059b3595cf9c41b9',
    'users' => ['user1', 'user2'],
    'apps' => ['comments'],
    'logfile' => '/tmp/test2.log'
  ]
],
```

Based on the conditional log settings above, following logs are written to the same logfile defined:

- Requests matching log_secret are debug logged.

```
curl -X PROPFIND -u sample-user:password \

https://your_domain/remote.php/webdav/?log_secret=57b58edb6637fe3059b3595cf9c41b9
```

- user1 and user2 gets debug logged.

- Access to app comments gets debug logged.

Request Tracing

ownCloud logs the X-REQUEST-ID header from desktop and mobile clients in the ownCloud log when sent with client requests.

The header helps when clients have a problem communicating with an ownCloud server, because:

1. The user can include the value in bug reports; and
2. System administrators can filter log files for the header value.

Storing this information makes searching more efficient, as system administrators don't have to rely solely on normal log entry elements, such as timestamps and IP addresses.

The Header's Value

The header's value is a UUID (version 4). These are generated from truly random (or pseudo-random) numbers by the client and do not contain *any* sensitive information. As a result it will not violate the user's privacy nor allow users to be tracked.

Required Server Configuration

Before the value can be stored in your web server's log files, your system administrator(s) need to configure two areas:

1. **The web server:** The web server's logging configuration needs to be adjusted, e.g., Apache's access and error log format, so that the value is stored in request log entries. An example of configuring Apache's CustomLog format is provided below.
2. **Load balancers:** All load balancers sitting in-between clients and your ownCloud instance(s), e.g., Traefik, Big-IP, need to be configured to pass the header through. This way it is possible to track ("trace") requests through larger environments. Please refer to your load balancer's configuration for details on how to adjust their configuration.

Web Server Configuration Example

Listing 14. Example for Apache

```
CustomLog /var/log/apache2/access.log "%h %l %u %t \"%r\" %>s %O
\"%{Referer}i\" \"%{User-Agent}i\" \"%{X-Request-ID}i\""
```



The exact log format chosen is entirely up to your system administrator(s).

Reverse Proxy Configuration

Introduction

ownCloud can be run through a reverse proxy, which can cache static assets such as images, CSS, or Javascript files, move the load of handling HTTPS to a different server or load balance between multiple servers.

Defining Trusted Proxies

For security, you must explicitly define the proxy servers that ownCloud is to trust.

Connections from trusted proxies will be specially treated to get the real client information, for use in access control and logging. Parameters are configured in config/config.php

Set the trusted_proxies parameter as an array of IP address to define the servers ownCloud should trust as proxies. This parameter provides protection against client spoofing, and you should secure those servers as you would your ownCloud server.

A reverse proxy can define HTTP headers with the original client IP address, and ownCloud can use those headers to retrieve that IP address. ownCloud 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 ownCloud, even when going through the trusted proxy! The correct value for this parameter is dependent on your proxy software.

Overwrite Parameters

The automatic hostname, protocol or webroot detection of ownCloud can fail in certain reverse proxy situations. This configuration allows the automatic detection to be manually overridden.

If ownCloud fails to automatically detect the hostname, protocol or webroot you can use the overwrite parameters inside the config/config.php. The overwritehost parameter is used to set the hostname of the proxy. You can also specify a port. The overwriteprotocol parameter is used to set the protocol of the proxy. You can choose between the two options HTTP and HTTPS. The overwritewebroot parameter is used to set the absolute web path of the proxy to the ownCloud folder. When you want to keep the automatic detection of one of the three parameters you can leave the value empty or don't set it. The overwritecondaddr parameter is used to overwrite the values dependent on the remote address. The value must be a **regular expression** of the IP addresses of the proxy. This is useful when you use a reverse SSL proxy only for HTTPS access and you want to use the automatic detection for HTTP access.

Example

Multiple Domains Reverse SSL Proxy

If you want to access your ownCloud installation http://domain.tld/owncloud via a multiple domains reverse SSL proxy https://ssl-proxy.tld/domain.tld/owncloud with the IP address 10.0.0.1 you can set the following parameters inside the config/config.php.

With an Apache as reverse proxy (ssl-proxy.tld) you can use this configuration:

```
ProxyPass "/domain.tld/owncloud" "http://domain.tld/owncloud"
ProxyPassReverse "/domain.tld/owncloud" "http://domain.tld/owncloud"
```



If you want to use the SSL proxy during installation you have to create config/config.php manually, otherwise you have to extend the existing $CONFIG array.

Server Security

In this section you will find all the details you need to configure ownCloud securely.

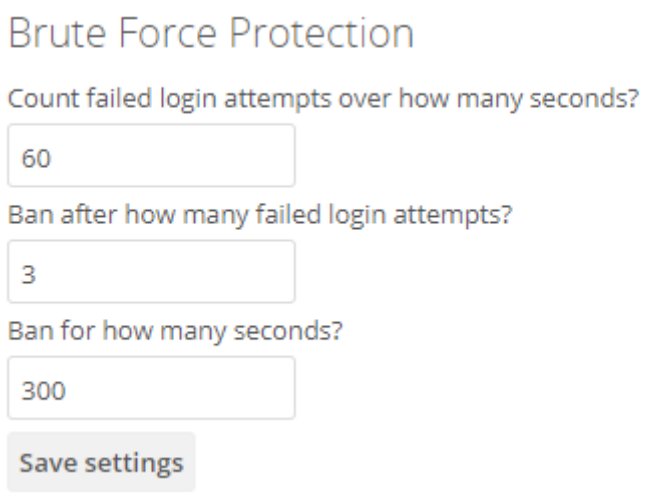
- OAuth2
- Password Policy

- Brute-Force Protection
- Hardware Security Module Daemon

Brute-Force Protection

The Brute-Force Protection extension allows administrators to specify a maximum number of unsuccessful user account login attempts. On reaching the unsuccessful login limit, ownCloud temporarily bans further login attempts to those user accounts from the originating IP address. The time frame of the ban is configurable by ownCloud administrators.

To configure this app in the web interface, navigate to admin → settings → admin/security.



To configure this app on the command line you can use occ commands.

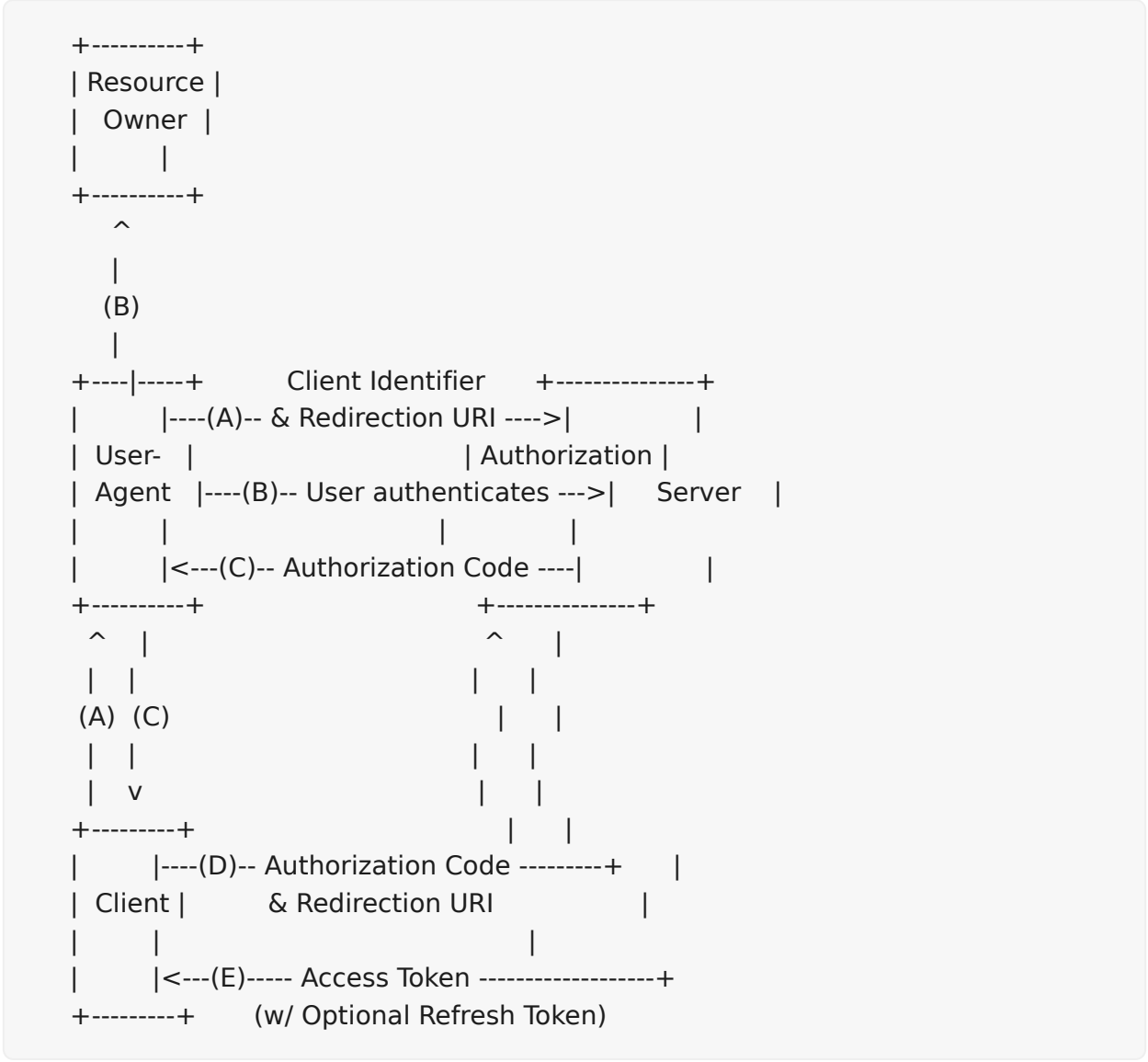
OAuth2

What is it?

OAuth2 is summarized in RFC 6749 as follows:

> The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf.

Here is an overview of how the process works:



The OAuth2 App

OAuth2 support is available in ownCloud via an OAuth2 application which is available from the ownCloud Marketplace. The app aims to:

1. Connect ownCloud clients (both desktop and mobile) in a standardized and secure way.
2. Make 3rd party software integrations easier by providing an unified authorization interface.

Requirements

To use the OAuth2 app, your ownCloud installation will need to meet the following dependencies:

- **Apache:** If you are hosting your ownCloud installation using the Apache web server, then mod_rewrite and mod_headers modules must be installed and enabled.
  1. **Redis:** You will need to have a Redis server available, ideally the latest, stable, version
  2. **PHP:** You PHP installation must have the phpredis extension (>= 4.2) installed and enabled.



We strongly recommend that you configured PHP to store sessions in Redis. Here is an example configuration.

```
; Redis configuration
SESSION_SAVE_HANDLER=redis
SESSION_SAVE_PATH=tcp://[REDIS_SERVER]:6397
REDIS_SESSION_LOCKING_ENABLED=1
REDIS_SESSION_LOCK_RETRIES=750
REDIS_SESSION_LOCK_WAIT_TIME=20000
```

The configuration parameters above are taken from the available ones. According to an issue in the phpredis repository, the default values are too small for common usage. It suggested using the ones above instead. However, please adjust these values to meet your needs.

Installation

To install the application, place the content of the OAuth2 app inside your installation's app directory, or use the Market application.

Basic Configuration

To enable token-only based app or client logins in config/config.php set token_auth_enforced to true.

Restricting Usage

- Enterprise installations can limit the access of authorized clients, preventing unwanted clients from connecting.

Endpoints

Description	URI
Authorization URL	/index.php/apps/oauth2/authorize
Access Token URL	/index.php/apps/oauth2/api/v1/token

Protocol Flow

Client Registration

The clients first have to be registered in the admin settings: /settings/admin?sectionid=authentication. You need to specify a name for the client (the name is unrelated to the OAuth 2.0 protocol and is just used to recognize it later) and the redirection URI. A client identifier and client secret are generated when adding a new client, which both consist of 64 characters. For further information about client registration, please refer to the official client registration RFC from the IETF.

Authorization Request

For every registered client an authorization request can be made. The client redirects the resource owner to the authorization URL and requests authorization. The following URL parameters have to be specified:

Parameter	Required	Description
response_type	yes	Needs to be code because at this time only the authorization code flow is implemented.
client_id	yes	The client identifier obtained when registering the client.
redirect_uri	yes	The redirection URI specified when registering the client.
state	no	Can be set by the client "to maintain state between the request and callback". See `RFC 6749`_ for more information.

For further information about client registration, please refer to the official authorization request RFC from the IETF.

Authorization Response

After the resource owner's authorization, the app redirects to the redirect_uri specified in the authorization request and adds the authorization code as URL parameter code. An authorization code is valid for 10 minutes. For further information about client registration, please refer to the official authorization response RFC from the IETF.

Access Token Request

With the authorization code, the client can request an access token using the access token URL. Client authentication is done using basic authentication with the client identifier as username and the client secret as a password. The following URL parameters have to be specified:

Parameter	Required	Description
grant_type		Either authorization_code or refresh_token.
code	if the grant type authorization_code is used.	
redirect_uri	if the grant type authorization_code is used.	
refresh_token	if the grant type refresh_token is used.	

For further information about client registration, please refer to the official access token request RFC from the IETF.

Access Token Response

The app responses to a valid access token request with a JSON response like the following. An access token is valid for 1 hour and can be refreshed with a refresh token.

```
{
   "access_token" :
"1vtnuo1NkIsbndAjVnhl7y0wJha59JyaAiFIVQDvcBY2uvKmj5EPBEhss0pauzdQ",
   "token_type" : "Bearer",
   "expires_in" : 3600,
   "refresh_token" :
"7y0wJuvKmj5E1vjVnhlPBEhha59JyaAiFIVQDvcBY2ss0pauzdQtnuo1NkIsbndA",
   "user_id" : "admin",
   "message_url" :
"https://www.example.org/owncloud/index.php/apps/oauth2/authorization-
successful"
}
```


For further information about client registration, please refer to the official access token response RFC from the IETF.



For a succinct explanation of the differences between access tokens and authorization codes, check out this answer on StackOverflow.

Limitations

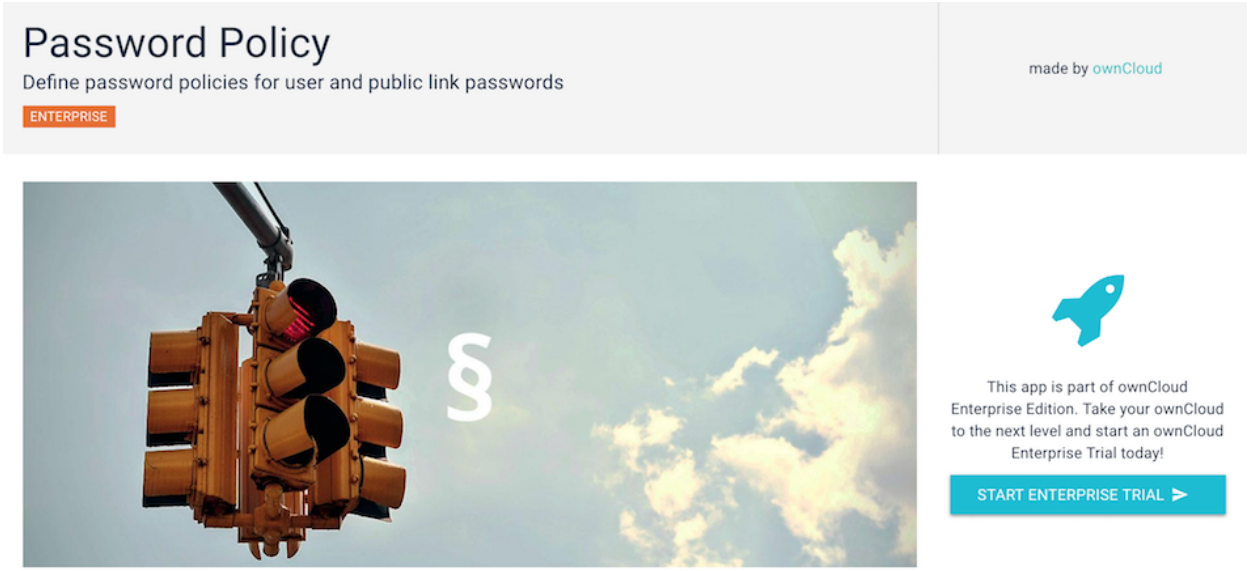
- Since the app does not handle user passwords, only master key encryption works (similar to the Shibboleth app).
- Clients cannot migrate accounts from Basic Authorization to OAuth2, if they are currently using the user_ldap backend.
- It is not possible to explicitly end user sessions when using OAuth2. Have a read through User Authentication with OAuth 2.0 to find out more.

Further Reading

- User Authentication with OAuth 2.0
- The problem with OAuth for Authentication.
- Session Authentication vs Token Authentication
- OAuth 2.0 Token Revocation

Password Policy

The Password Policy App



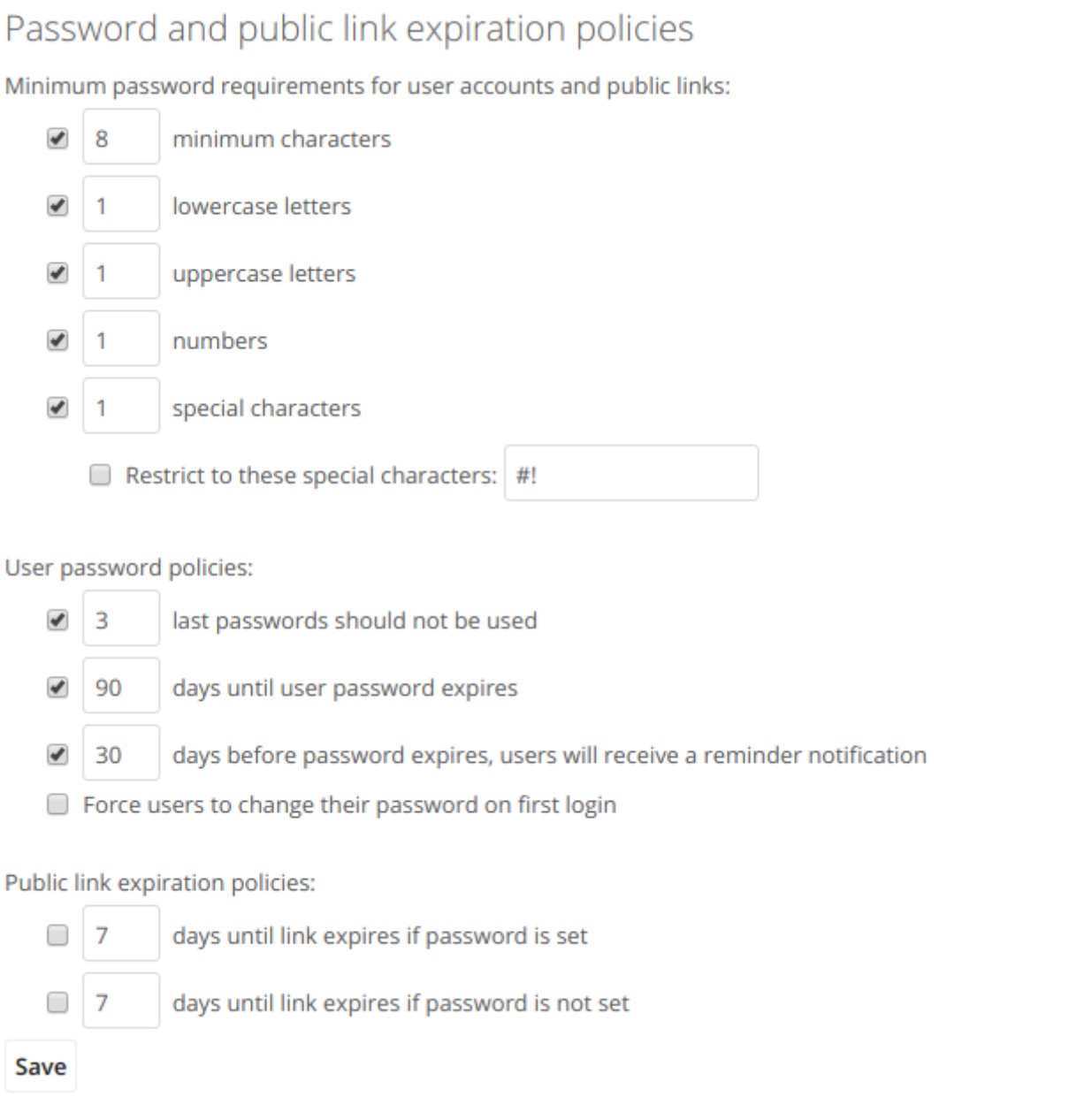
From the 2.0.0 release of the Password Policy app, ownCloud administrators (both enterprise **and** community edition) have the option of installing and enabling the application. The Password Policy application enables administrators to define password requirements for user passwords and public links.

Some of policy rules apply to both user passwords and public links, and some apply to just one or the other. The table below shows where each option can be used.

Setting	User Passwords	Public Links
Specify valid password requirements	*	*
Disallow usage of a number of previous passwords	*	
Specify a password expiration period	*	
Forced password change on first login	*	
Disallowing passwords that match a configurable number of previous passwords (defaults to the previous 3).	*	
Users can be notified a configurable number of days before their password expires	*	
	Users will be notified when their password has expired.	*
	Specify expiration dates for public link shares	
*	Specify the number of days until link expires if a password is set	

Setting	User Passwords	Public Links
*		Specify the number of days until link expires if a password is **not** set
	*	

Here is an example of what an administrator will see:



Active user sessions will **not** end when passwords expire. However, a password change will be forced when the user session expires (e.g., on logout). OAuth2 tokens for app or client authentication, and App passwords are not affected.

Installing and enabling the application also extends the occ command to support password policy management.

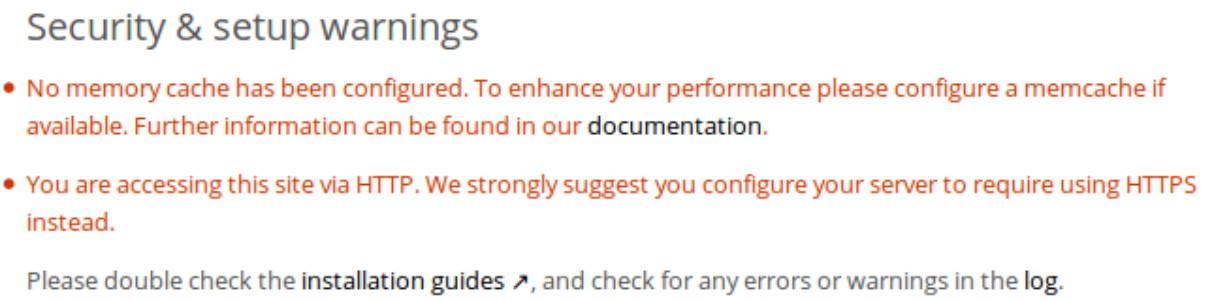


After enabling the "**days until user password expires**" policy setting in the web UI, administrators need to run the occ user:expire-password command to set an initial password change date for all existing users.

Warnings on Admin Page

Introduction

Your ownCloud 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.



Cache Warnings

No memory cache has been configured. To enhance your performance please configure a memcache if available.

ownCloud supports multiple PHP caching extentions:

- APCu
- 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 caching_configuration.

Transactional file locking is disabled

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

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

You are accessing this site via HTTP

> You are accessing this site via HTTP. We strongly suggest you configure your server to require using HTTPS instead.

Please take this warning seriously; using HTTPS is a fundamental security measure. You must configure your Web server to support it, and then there are some settings in the **Security** section of your ownCloud Admin page to enable. The following pages describe how to enable HTTPS on the Apache webserver.

- Enable SSL on Apache
- Use HTTPS

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

Some environments are not passing a valid PATH variable to ownCloud. The PHP FPM tips provides the information about how to configure your environment.

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

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

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

/dev/urandom is not readable by PHP

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

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

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

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

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

Outdated NSS / OpenSSL version

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

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

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

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

Some files have not passed the integrity check

Please refer to the Fixing Invalid Code Integrity Messages documentation how to debug this issue.

Your database does not run with "READ COMMITED" transaction isolation level

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

Please refer to MySQL / MariaDB with Binary Logging Enabled) how to configure your database for this requirement.

The HSM (Hardware Security Module) Daemon (hsmdaemon)

Introduction

The hsmdaemon is a daemon, provided by ownCloud, to delegate encryption to an HSM (Hardware Security Module). This can be necessary, as PHP cannot, directly, interface with a PKCS11 stack; neither with an API wrapper, because one does not exist, nor via the OpenSSL bindings. Because of this, a separate process is needed to decrypt anything with the private key stored in an HSM.



When using hsmdaemon with an HSM, the keys *may* still be stored on the same physical machine as ownCloud.



For hsmdaemon support, you need ownCloud Enterprise Edition >= 10.2. We recommend consulting with us when deploying storage encryption with an HSM.

Running exec() to decrypt the key with a command line command to do the encryption might leak the HSM credentials if the admin lists the currently running processes. To prevent that, an HSM daemon will be used that can open a session to the HSM upon startup.

This daemon will be used by ownCloud to decrypt the current master key upon request. The communication happens via UNIX sockets or TCP sockets and is

authorized by a shared token that the daemon stores in the ownCloud database via a REST/JSON route.

ownCloud internally uses OpenSSL to en-/decrypt keys and needs to be extended to support en-/decrypt operations via the new daemon. The current solution encrypts the ownCloud master key with a key from the HSM.



From the technical point of view the Crypt class is extended to handle the key generation in the HSM device and also to get the key from HSM. For the read/write operation on a file, the request goes to the HSM and then, based on the keys fetched from HSM, the files are encrypted or decrypted. The keys are not replaced.

How The HSM Daemon Interacts with ownCloud

Upon startup, the daemon will generate a token and send it to ownCloud via a new REST/JSON route. After connecting with the HSM daemon, an unsophisticated, line-based, protocol is used (every line ends with CRLF):

1. ownCloud sends the token read from database.
2. The daemon compares the received token with its token and returns an "OK" line.
3. ownCloud then sends the data it wants to decrypt as a Base64-encoded, one-line string.
4. The daemon returns the decrypted data as a Base64-encoded one-line string.

Doing so ensures that an evil admin will need to wiretap the communication between either the database or the HSM daemon and ownCloud.

Quick Overview

HSM support consists of two core parts:

1. An actual HSM PKCS11 module.
2. An hsmdaemon that provides a JWT-protected web API for the PKCS11 stack to generate key pairs and decrypt data.

Deployment Recommendation

We recommend running hsmdaemon on every web server to reduce latency.

Installation

Integrating the hsmdaemon with ownCloud requires 3 steps; these are:

1. Install a PKCS11 Module
2. Install and Configure the hsmdaemon
3. Configure ownCloud



The installation instructions in this guide have been designed to work with ownCloud's supported operating systems. If you are using a different operating system or distribution, please adjust the instructions to suit your environment.

Install a PKCS11 Module

Install Using a Preconfigured PKCS11 Module

At least one PKCS11 library is necessary. This is typically provided by an HSM vendor. If a PKCS11 library is not available, you can use the software HSM, *SoftHSM2*.

Initialise the Token

Now we can initialize the token:

```
softhsm2-util --init-token --slot 0 --label "My token 1"
```

It will ask for two PINs, an SO and a User pin. See https://www.opendnssec.org/softhsm/, for more information.

Install PKCS11 CLI tools (optional)

To use the PKCS11 API on the CLI, we need to install OpenSC.

- Debian and Ubuntu
- openSUSE and SUSE Linux Enterprise Server
- Fedora and Red Hat Enterprise Linux and Centos

Initialise on Debian and Ubuntu

To install OpenSC on Debian and Ubuntu, run the following command:

```
sudo apt install -y opensc
```

Initialise on openSUSE and SUSE Linux Enterprise Server

To install OpenSC on openSUSE and SUSE Linux Enterprise Server, run the following command:

```
sudo sudo zypper install -y --auto-agree-with-licenses opensc
```

Initialise on Fedora and Red Hat Enterprise Linux and Centos

To install OpenSC on Fedora and Red Hat Enterprise Linux and Centos, run the following command:

```
sudo yum install --assumeyes opensc
```

List Tokens

You can list the available tokens using pkcs11-tool, by running the following command.

```
sudo pkcs11-tool --module </path/to/libsofthsm2.so> -l --pin <user-pin> -O
```

The Module Parameter

The module parameter is either the library provided by the HSM vendor, or libsofthsm2 which was installed with SoftHSM 2. If you are using libsofthsm2, the path

to libsofthsm2.so for each of the supported distributions is available below.

Distribution	Path
Debian and Ubuntu	/usr/lib/softhsm/libsofthsm2.so
openSUSE and SUSE Linux Enterprise Server	/usr/lib64/pkcs11/libsofthsm2.so
Fedora and Red Hat Enterprise Linux and Centos	/usr/lib64/pkcs11/libsofthsm2.so



See the OpenSC Wiki for more information.

Install and Configure the hsmdaemon

Installing hsmdaemon requires several steps. These are:

1. Install the hsmdaemon Binary
2. Copy the Config File
3. Install the System Service
4. Configure the PKCS 11 Module Path
5. Configure Slot and Pin
6. Test Key Generation
7. Configure Other Options

Install the hsmdaemon Binary

After you've obtained the hsmdaemon from ownCloud, you need to:

1. Move the hsmdaemon binary to a directory located in your system path.
2. Make the hsmdaemon binary Executable
3. Copy the Config File



If you are not sure which directories are in your system path, run the following script to see a complete list:

```
OFS=$IFS && IFS=':'
for i in $(echo $PATH); do echo $i; done;
IFS=$OFS;
```

You should see a list similar to the following:

```
/usr/local/sbin
/usr/local/bin
/usr/sbin
/usr/bin
/sbin
/bin
```

Copy the Config File

The default location that hsmdaemon looks for its config file is /etc/hsmdaemon/hsmdaemon.toml. To create it from the example config file available in provided package, run the following commands.

```
mkdir /etc/hsmdaemon                          # Create the hsmdaemon configuration directory
cp hsmdaemon.toml /etc/hsmdaemon/hsmdaemon.toml   # Copy the example config file
chown root /etc/hsmdaemon/hsmdaemon.toml          # Set the owner of the file to root
chmod 750 /etc/hsmdaemon/hsmdaemon.toml           # Allow only the root and users in the root group to read & write the configuration file
```

Install the System Service

Now that the binary is available and the configuration file is in place, hsmdaemon must be installed as a system service. To do this, run it with the install option, as in the example below.

```
./hsmdaemon install
```

If it installs successfully, then you should see the following console output:

```
Install HSM Daemon:                          [  OK  ]
```

It should now be running and set to start automatically at boot time.



The daemon is managed using the following three commands:

- sudo service hsmdaemon start
- sudo service hsmdaemon stop and
- sudo service hsmdaemon status.

Configure the PKCS11 Module Path

To set the path to the PKCS11 module, update the line below in /etc/hsmdaemon/hsmdaemon.toml, with the appropriate path on your system.

```
[pkcs11]
module = "/usr/lib/x86_64-linux-gnu/softhsm/libsofthsm2.so" # softhsm v2
```

List Available Slots

This command lists the available slots.

```
hsmdaemon listslots
{"level":"debug","ts":"2019-02-
14T09:27:02.068+0100","caller":"hsmdaemon/keymanager.go:27","msg":"initialize
pkcs11 module","module":"/usr/lib/softhsm/libsofthsm2.so"}
{"level":"info","ts":"2019-02-
14T09:27:02.087+0100","caller":"hsmdaemon/keymanager.go:65","msg":"Slots
found","slotIds":[550099622,1989683358,2]}
Available slots:
Slot: 550099622,
   Slot info:
      Description:      SoftHSM slot ID 0x20c9daa6
      Manufacturer ID:  SoftHSM project
      Hardware version: 2.2
      Firmware version: 2.2
      Token present:    yes
      Flags:
   Token info:
      Manufacturer ID:    SoftHSM project
      Model:              SoftHSM v2
      Hardware version:   2.2
      Firmware version:   2.2
      Serial number:      e8ba06bca0c9daa6
      Initialized:        yes
      User PIN init.:     yes
      Label:              oc token without pin
      MaxSessionCount:    0
      SessionCount:       18446744073709551615
      MaxRwSessionCount:  0
      RwSessionCount:     18446744073709551615
      MaxPinLen:          255
      MinPinLen:          4
      TotalPublicMemory:  18446744073709551615
      FreePublicMemory:   18446744073709551615
      TotalPrivateMemory: 18446744073709551615
      FreePrivateMemory:  18446744073709551615
      UTCTime:            2019021408270200
      Flags: CKF_RNG CKF_LOGIN_REQUIRED CKF_RESTORE_KEY_NOT_NEEDED
CKF_USER_PIN_COUNT_LOW
Slot: 1989683358,
   Slot info:
      Description:      SoftHSM slot ID 0x7698289e
      Manufacturer ID:  SoftHSM project
      Hardware version: 2.2
      Firmware version: 2.2
```



See the OpenSC Wiki for more information.

Configure the Slot and Pin

Ask the customer which slot to use and if a PIN is needed. Update /etc/hsmdaemon/hsmdaemon.toml with the information that the customer provides, in the pkcs11 section, as in the example below.

```
[pkcs11]
module = "/usr/lib/x86_64-linux-gnu/softhsm/libsofthsm2.so" # softhsm v2
pin = "1234"          # The user pin supplied when running softhsm2-util --init-token,
comment it out , or leave empty if no pin is necessary
slot = 1989683358     # Find your slot id with `sudo hsmdaemon listslots`
```

Testing

Test Key Generation



If no PIN is supplied, generating a new key might be protected by an operator card that has to be inserted in the HSM. In this case, coordinate testing and final master key generation with your HSM team.

For testing key generation, run the command hsmdaemon genkey test, as in the following example.

```
hsmdaemon genkey test
Id: 9bac3719-2b8d-11e9-aeab-0242b5ece4c3, label: test
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAl1BO4vsI+xDk+x0nccl7
HQhMR/hwfa0+N8fyYNI8yzTTmYDqz9aaF20qG48+mjC0AUEt2kfKo94xM3UeEw4c
st4j1dpRJtmAJThcuN8OH3sa+3MeXWgGuWxjB1lxEEOqax2A6XzllDlbDsogwkOL
hSkUU9AaMRBtF8fASJGtJDP+iXwdb7OsFg78PS1wBAISYSUwk06xY7LwWIxge+hY
4oU+5x4itusdO6rz6kbcJtmUyDUb8DhKnN6OdkhnifUZLBG9HQyTa5OM+BAabbFZ
mTM2gZlUnGKXN7c4kaBPFt1IfjjVYu7pvj3B2uxUf4GywuSuWGWnAy89FqeXteRV
jwIDAQAB
-----END PUBLIC KEY-----
```

Test Data Encryption

For testing data encryption, run the hsmdaemon encrypt command, as in the following example.

```
# The first argument is the "Id:" value from running the genkey command above.
# The second is the base64-encoded data to be encrypted.
sudo hsmdaemon encrypt 9bac3719-2b8d-11e9-aeab-0242b5ece4c3 Zm9vYmFy
```

If successful, you should see output similar to the below example.

{"level":"debug","ts":"2019-03-20T12:43:40.540+0100","caller":"hsmdaemon/keymanager.go:27","msg":"initialize pkcs11 module","module":"/usr/lib/softhsm/libsofthsm2.so"}
{"level":"debug","ts":"2019-03-20T12:43:40.545+0100","caller":"hsmdaemon/keymanager.go:205","msg":"openHSMSession","slotID":858597139}
{"level":"info","ts":"2019-03-20T12:43:40.549+0100","caller":"hsmdaemon/keymanager.go:621","msg":"Fetching private key","keyID":"9bac3719-2b8d-11e9-aeab-0242b5ece4c3"}
{"level":"debug","ts":"2019-03-20T12:43:40.549+0100","caller":"hsmdaemon/keymanager.go:641","msg":"Got uuid","string":"13d34146-4b02-11e9-adbd-0023ae27c404"}
WcezVb2N6bF8wlDooKZcmFn3tZgoIpoFGx6wQetx9sp1nK7JW2Y4OKt7P+0VKKlFO7yXaffVDD2Q6jZZCQukQVRV1zJrwbI9xU3YlOAwJFPP+WM/dZ1vdUwi7L05wq8UpL13LJWlMkvd1eIqKJS7apMnFk2hbnxXP6UKZmI++1tXvqbAc6fwhcB5J+JG6lmS4RwnD+eJC3dq5t00zzdI6vuIM/y3UT7ESklmHl5bKl+N+d6yk6qLxnFnIJweL+M3Tf13+XPNAh5JxZpheJPvN3oL28uX76aizy4BCLnRgQ/ryUQeDF+a4zNF22sMwBh4Pt46KrYGNDZAnQpVzmkrZQ==

Test Showing Keys

To show an existing key, use the showkey command with the key's id, as in the following example.

```
sudo hsmdaemon showkey 9bac3719-2b8d-11e9-aeab-0242b5ece4c3
```

Configure Other Options (optional)

For more options see the self-documented default config file hsmdaemon.toml.



During ownCloud config you might want to run the hsmdaemon service in the foreground to see what is going on. You can do so, using the following command (which also shows example console output, formatted for readability).

```
./hsmdaemon
{
   "level": "info",
   "ts": "2019-02-14T09:32:59.081+0100",
   "caller": "hsmdaemon/hsmdaemon.go:146",
   "msg": "Server listening",
   "host": "localhost",
   "port": 8513,
   "version": "0.0.7",
   "build": "2019-02-08T10:47:55+00:00"
}
```

Configure ownCloud



If anyone accesses ownCloud while encryption is enabled, it will automatically generate the keys. To prevent this, shut down the web server until encryption is appropriately configured.

To configure ownCloud to work with the hsmdaemon requires the following steps:

- Generate a Secret for the hsmdaemon REST API
- Configure HSM-based Encryption
- Initialize and Check Generated Keys

Generate a Secret for the hsmdaemon REST API

Generate a shared secret to use for the hsmdaemon.

```
cat /proc/sys/kernel/random/uuid
7a7d1826-b514-4d9f-afc7-a7485084e8de
```

Use this secret for hsmdaemon in /etc/hsmdaemon/hsmdaemon.toml

```
[jwt]
secret = "7a7d1826-b514-4d9f-afc7-a7485084e8de"
```

Set the generated secret for ownCloud:

```
{occ-command-example-prefix} config:app:set \
   encryption hsm.jwt.secret \
   --value '7a7d1826-b514-4d9f-afc7-a7485084e8de'
```

If the command succeeds, you should see the following console output:

```
Config value hsm.jwt.secret for app encryption set to 7a7d1826-b514-4d9f-afc7-
a7485084e8de
```

Configure HSM-based Encryption

Enable HSM mode and enable encryption by running the commands in the following example.

```
occ config:app:set encryption hsm.url --value 'http://localhost:8513'
occ app:enable encryption
occ encryption:enable
```

If the commands are successful, you should see the following console output:

```
Config value hsm.url for app encryption set to http://localhost:8513

encryption enabled

Encryption enabled

Default module: OC_DEFAULT_MODULE
```

If you want to use a single master key run

```
occ encryption:select-encryption-type masterkey
```

Initialize and Check Generated Keys

Now start your web server, and log in with any user to initialize the keys, have a look at the output of the hsmdaemon to see key generation and decryption requests. Check that the private key /path/to/data/files_encryption/OC_DEFAULT_MODULE/ is less than **1000 bytes**. If it is not, then something is not configured correctly. You have to wipe all keys and reset the database flags for encryption to get a clean start for the ownCloud setup.

ownCloud Server Tuning

Using Cron to Perform Background Jobs

See Background Jobs for a description and the benefits.

Enable Memory Caching

Caching improves performance by storing data, code, and other objects in memory. Memory cache configuration for ownCloud is no longer automatically available from ownCloud 8.1 but must be installed and configured separately. ownCloud supports Redis, APCu, and Memcached as memory caching backends. See Memory Caching, for further details.

Use Redis-based Transactional File Locking

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

Redis Tuning

Redis tuning improves both file locking (if used) and memory caching (when using Redis). Here is a brief guide for tuning Redis to improve the performance of your ownCloud installation, when working with sizeable instances.

TCP-Backlog

If you raised the TCP-backlog setting, the following warning appears in the Redis logs:

```
WARNING: The TCP backlog setting of 20480 cannot be enforced because
/proc/sys/net/core/somaxconn is set to the lower value of..
```

If so, please consider that newer versions of Redis have their own TCP-backlog value set to 511, and that you have to increase if you have many connections. In high requests-per-second environments, you need a significant backlog to avoid slow clients connection issues.



The Linux kernel will silently truncate the TCP-backlog setting to the value of /proc/sys/net/core/somaxconn. So make sure to raise both the value of somaxconn and tcp_max_syn_backlog, to get the desired effect.

To fix this warning, set the value of net.core.somaxconn to 65535 in /etc/rc.local, so that it persists upon reboot, by running the following command.

```
sudo echo sysctl -w net.core.somaxconn=65535 >> /etc/rc.local
```

After the next reboot, 65535 connections will be allowed, instead of the default value.

Transparent Huge Pages (THP)

If you are experiencing latency problems with Redis, the following warning may appear in your Redis logs:

```
WARNING you have Transparent Huge Pages (THP) support enabled in your kernel.
This creates both latency and memory usage issues with Redis.
```

If so, unfortunately, when a Linux kernel has Transparent Huge Pages enabled, Redis incurs a significant latency penalty after the fork call is used, to persist information to disk. Transparent Huge Pages are the cause of the following issue:

1. A fork call is made, resulting in two processes with shared huge pages being created.
2. In a busy instance, a few event loops cause commands to target a few thousand pages, causing the copy-on-write of almost the entire process memory.
3. Big latency and memory usage result.

As a result, make sure to disable Transparent Huge Pages using the following command:

```
echo never > /sys/kernel/mm/transparent_hugepage/enabled
```

Redis Latency Problems

If you are having issues with Redis latency, please refer to the official Redis guide on how to handle them.

Database Tuning

Using MariaDB/MySQL Instead of SQLite

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

See the section Linux Database Configuration for how to configure ownCloud for MySQL or MariaDB. If your installation is already running on SQLite then it is possible to convert to MySQL or MariaDB using the steps provided in database conversion.

Tune MariaDB/MySQL

A comprehensive guide to tuning MySQL and MariaDB is outside the scope of the ownCloud documentation. However, here are three links that can help you find further information:

- MySQLTuner.
- Percona Tools for MySQL
- Optimizing and Tuning MariaDB.

Tune PostgreSQL

A comprehensive guide to tuning PostgreSQL is outside the scope of the ownCloud documentation. However, here are three links that can help you find further information:

- Five Steps to PostgreSQL Performance
- Tuning the autovacuum proceff for tables with huge update workloads (oc_filecache)

SSL / Encryption App

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

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

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

Webserver Tuning

Tune Apache

Enable HTTP/2 Support

If you want to improve the speed of an ownCloud installation, while at the same time increasing its security, you can enable HTTP/2 support for Apache. Please be aware that most browsers require HTTP/2 to be used with SSL enabled.

Apache Processes

An Apache process uses around 12MB of RAM. Apache should be configured so that the maximum number of HTTPD processes times 12MB is lower than the amount of RAM. Otherwise the system begins to swap and the performance goes down.

Use KeepAlive

The KeepAlive directive enables persistent HTTP connections, allowing multiple requests to be sent over the same TCP connection. Enabling it reduces latency by as much as 50%. We recommend to keep the KeepAliveTimeout between 3 and 5. Higher numbers can block the Server with inactive connections. In combination with the periodic checks of the sync client the following settings are recommended:

```
KeepAlive On
KeepAliveTimeout 3
MaxKeepAliveRequests 200
```

Hostname Lookups

```
# cat /etc/httpd/conf/httpd.conf
   ...
HostnameLookups off
```

Log files

Log files should be switched off for maximum performance. To do that, comment out the CustomLog directive. However, keep ErrorLog set, so errors can be tracked down.

Using Third Party PHP Components

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

Managing Third Party Parameters

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

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

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

```
<?php

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

Virus Scanner Support

Overview

ClamAV is the only *officially* supported virus scanner available for use with ownCloud. It:

- Operates on all major operating systems, including *Windows*, *Linux*, and *macOS*

- Detects all forms of malware including *Trojan horses*, *viruses*, and *worms*
- Scans *compressed files*, *executables*, *image files*, *Flash*, *PDF*, as well as many others

What's more, ClamAV's Freshclam daemon automatically updates its malware signature database at scheduled intervals. However, other scanners can be used, so long as they:

1. Can receive data streams via pipes on the command-line and return an exit code.
2. Return a parseable result on stdout.

How ClamAV Works With ownCloud

ownCloud integrates with antivirus tools by connecting to them via:

- A URL and port
- A socket
- Streaming the data from the command-line via a pipe with a configured executable

In the case of ClamAV, ownCloud's Antivirus extension sends files as streams to a ClamAV service (which can be on the same ownCloud server or another server within the same network) which in turn scans them and returns a result to stdout.



Individual chunks are **not** scanned. The whole file is scanned when it is moved to the final location.

The information is then parsed, or an exit code is evaluated if no result is available to determine the response from the scan. Based on ownCloud's evaluation of the response (or exit code) an appropriate action is then taken, such as recording a log message or deleting the file.



Scanner exit status rules are used to handle errors when ClamAV is run in CLI mode. Scanner output rules are used in daemon/socket mode.

Things To Note

1. Files are checked when they are uploaded or updated (whether because they were edited or saved) but *not* when they are downloaded.
2. ownCloud doesn't support a cache of previously scanned files.
3. If the app is either not configured or is misconfigured, then it rejects file uploads.
4. If ClamAV is unavailable, then the app rejects file uploads.
5. A file size limit applies both to background jobs and to file uploads.

Configuring the ClamAV Antivirus Scanner

You can configure your ownCloud server to automatically run a virus scan on newly-uploaded files using the Antivirus App.



ClamAV must be installed before installing and configuring Antivirus App for Files.

Installing ClamAV

As always, Linux distributions install and configure ClamAV in different ways. Below you can find the instructions for installing it on Debian or Red Hat-based distributions.

Debian, Ubuntu, Linux Mint

Install ClamAV on Debian, Ubuntu — and their many variants — with the following command:

```
sudo apt install clamav clamav-daemon
```

This automatically creates the default configuration files and launches the clamd and freshclam daemons. You shouldn't have to do anything else, though it is a good idea to review the ClamAV documentation, as well as ClamAV's settings in /etc/clamav/.

Red Hat 7 and CentOS 7

On Red Hat 7 and related systems, you must install the Extra Packages for Enterprise Linux (EPEL) repository, and then install ClamAV. To do so, run the following commands:

```
yum install epel-release
yum install clamav clamav-scanner clamav-scanner-systemd clamav-server
clamav-server-systemd clamav-update
```



Regardless of the operating system, we recommend that you enable verbose logging in both clamd.conf and freshclam.conf until you get any kinks with your ClamAV installation worked out.

Configuring and Running ClamAV

After installing ClamAV and the related tools, you will now have 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. Running either man clamd.conf or man freshclam.conf provides detailed information on all the available configuration options.



Refer to /etc/passwd and /etc/group when you need to verify the ClamAV user and group.

When you're finished editing the configuration files, you must enable the clamd service file and start clamd. You can do so using the following commands:

```
systemctl enable clamav-daemon.service
systemctl start clamav-daemon.service
```

When successful, output similar to the following renders to the console:

```
Synchronizing state of clamav-daemon.service with SysV service script with
/lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable clamav-daemon
```

Configure the Port

To configure the port that ClamAV listens on, add the following line in /etc/clamav/clamd.conf:

```
TCPSocket 3310
```

Then, restart the ClamAV daemon as follows:

```
sudo /etc/init.d/clamav-daemon restart
```



Enable verbose logging in scan.conf and freshclam.conf until it is running the way you want.

Automating ClamAV Virus Database Updates

To update your malware database and get the latest malware signatures, you need to run freshclam frequently. Do this by running freshclam or sudo freshclam on Debian-based distributions.

We recommend you do this, post-installation, to download your first set of malware signatures. If you want to adjust freshclam's behavior, edit /etc/clamav/freshclam.conf and make any changes you believe are necessary.

After that, create a cron job to automate the process. For example, to run it every hour at 47 minutes past the hour, add the following in the applicable user's crontab:

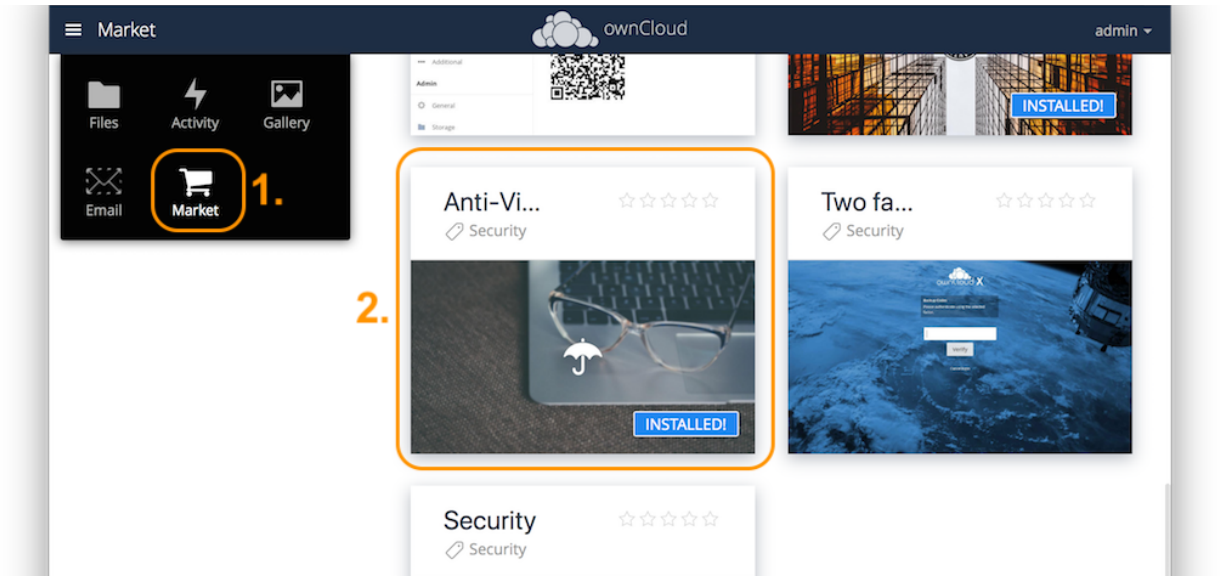
```
# m   h  dom mon dow  command
47  *  *   *   *  /usr/bin/freshclam --quiet
```



Please avoid any multiples of 10, because those are when the ClamAV servers are hit the hardest for updates.

Install the Anti-Virus App

The Anti-Virus app needs to be installed from the ownCloud Market (it's available in the _ "Security"_ category). You can access the ownCloud Market via the App Menu (or App Switcher).





The Anti-Virus app can also be downloaded, installed, and enabled manually.



If the app is enabled but either not configured or incorrectly configured it will **strictly reject all uploads** for the whole instance

ClamAV can be configured in two ways:

1. By using the occ config:app:set command.
2. By using the Antivirus Configuration panel

Configure ClamAV Using occ

All of the configuration settings for ClamAV are configurable by passing the relevant key and value to the occ config:app:set files_antivirus command. For example:

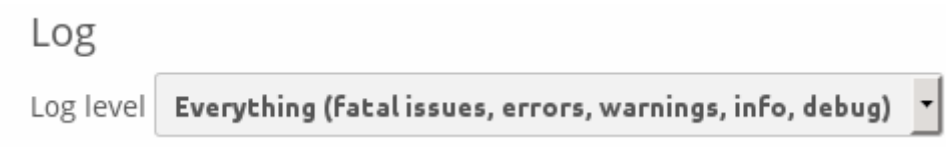
```
{occ-command-example-prefix} config:app:set \
   files_antivirus av_socket --value="/var/run/clamav/clamd.ctl"
```

Available Configuration Settings

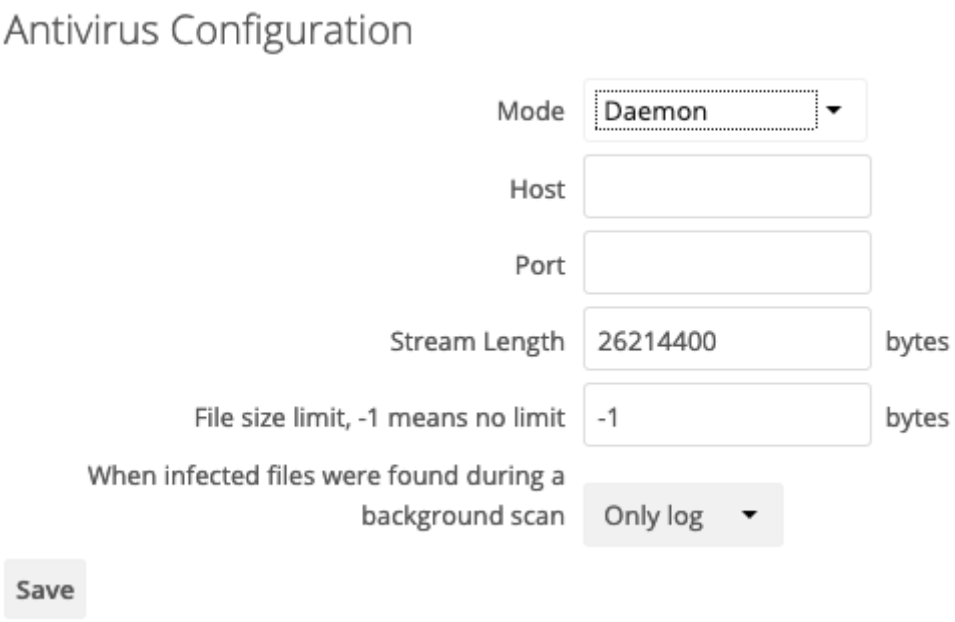
Setting	Description	Default
av_cmd_options	Extra command line options (comma-separated) to pass to ClamAV.	
av_host	The hostname or IP address of the Antivirus server.	
av_infected_action	The action to take when infected files were found during a background scan. It can be set to one of only_log and delete.	only_log
av_max_file_size	The maximum file size limit; -1 means no limit.	-1
av_mode	The operating mode. It can be set to one of executable, daemon, and socket.	executable
av_path	The path to the clamscan executable.	/usr/bin/clamscan
av_port	The port number of the Antivirus server. Allowed values are 1 - 65535.	
av_socket	The name of ClamAV's UNIX socket file.	/var/run/clamav/clamd.ctl
av_stream_max_length	The maximum stream length that ClamAV will accept.	26214400

Configure ClamAV Using The Antivirus Configuration Panel

Once ClamAV is installed, select menu:Settings[General (Admin)] and, in the "**Log**" section, set btn:[Log level] to "*Everything (fatal issues, errors, warnings, info, debug)*".



Now, navigate to menu:Settings[Security (Admin)], where you'll find the "**Antivirus Configuration**" panel. There, as below, you'll see the configuration options which ownCloud passes to ClamAV.



Mode Configuration

ClamAV runs in one of three modes:

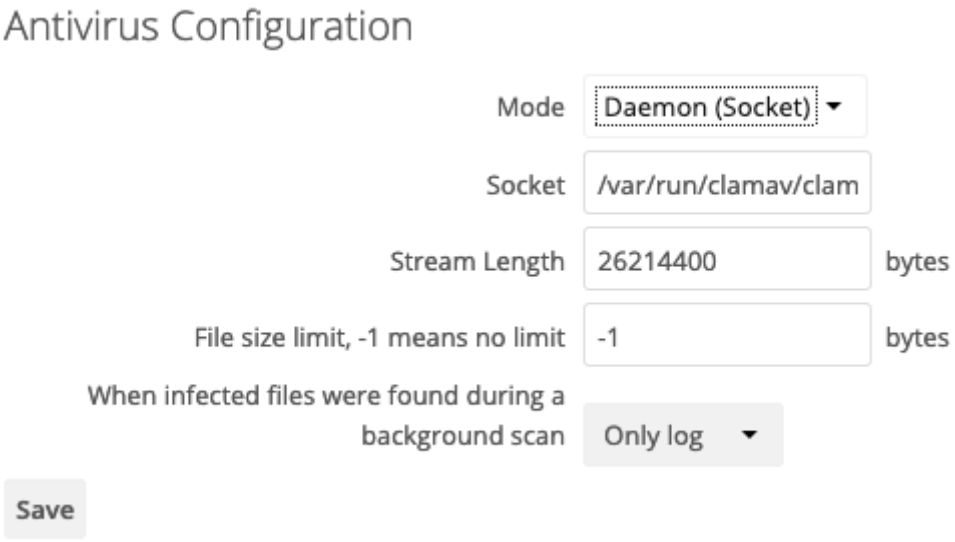
- Daemon (Socket)
- Daemon
- Executable

Daemon (Socket)

In this mode, ClamAV runs in the background on the same server as the ownCloud installation, or the socket can be made available via a share mount. When there is no activity, clamd places a minimal load on your system. However, if your users upload large volumes of files, you will see high CPU usage. Please keep this in mind.



You must run freshclam at least once for ClamAV to generate the socket.



First, set btn:[Mode] to "**Daemon (Socket)**". ownCloud 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 it, as in the example below:

```
sudo ss -a | grep -iq clamav && echo "ClamAV is running"
```



If you don't have ss installed, you may have netstat installed. If so, you can run the following to check if ClamAV is running:

```
netstat -a | grep -q clam && echo "ClamAV is running"
```

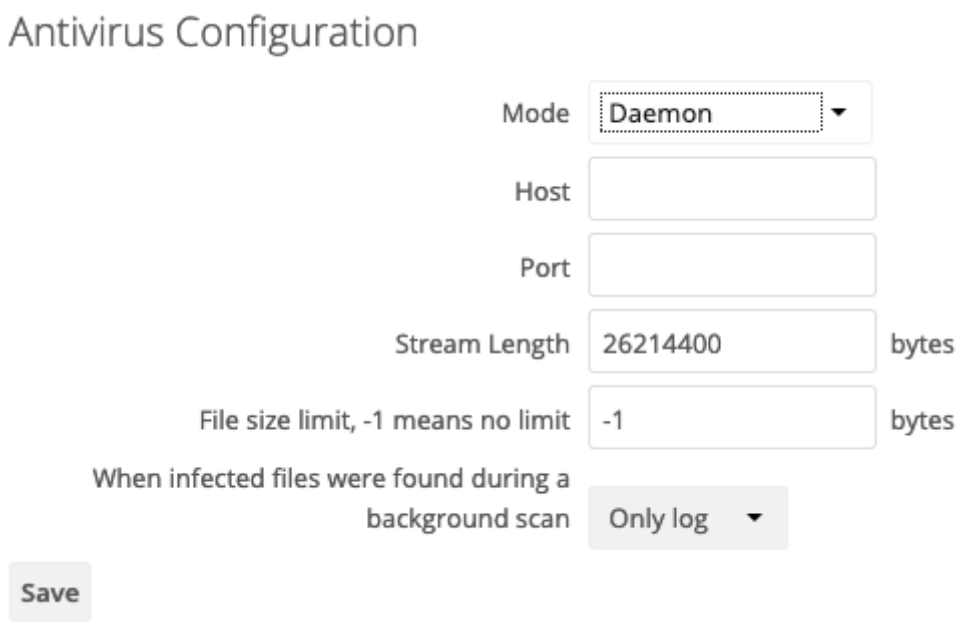
The Stream Length value sets the number of bytes to read in one pass; 10485760 bytes (ten megabytes) is the default. This value should be no larger than the PHP memory_limit settings or physical memory if memory_limit is set to -1 (no limit).

When infected files were found during a background scan gives you the choice of either:

- Logging any alerts without deleting the files
- Immediately deleting infected files

Daemon

In this mode, ClamAV runs on a different server. This is a good option for ownCloud servers with high volumes of file uploads.



First, set btn:[Mode] to "**Daemon**". Then, you need to set btn:[Host] to the hostname or IP address of the remote server running ClamAV, and set btn:[Port] to the server's port number.



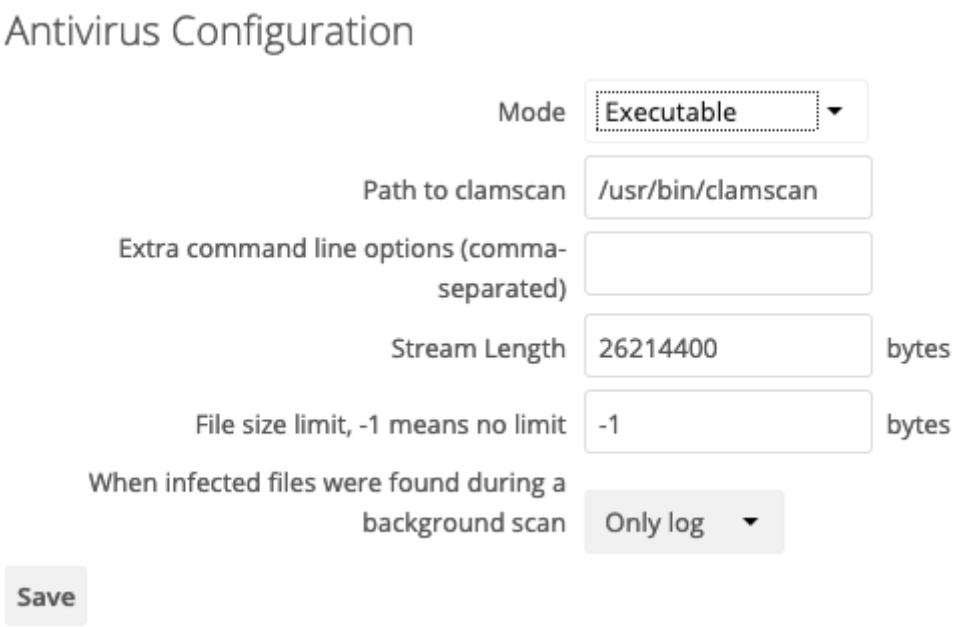
The port number is the value of TCPSocket in /etc/clamav/clamd.conf.

Executable

In this mode, ClamAV runs on the same server as the ownCloud installation, with the clamscan command running only when a file is uploaded.



clamscan is slow and not always reliable for on-demand usage; it is better to use one of the daemon modes.

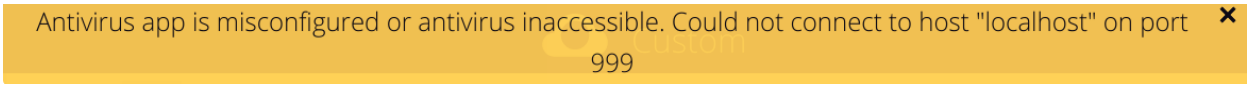


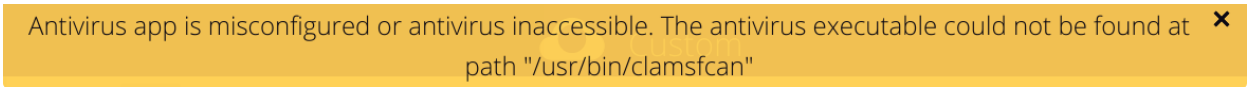
First, set btn:[Mode] to "**Executable**". Then, set btn:[Path to clamscan] to the path to clamscan, which is the interactive ClamAV scanning command, on your server. ownCloud should automatically find it. However, if it doesn't, run which clamscan to find the command's path.

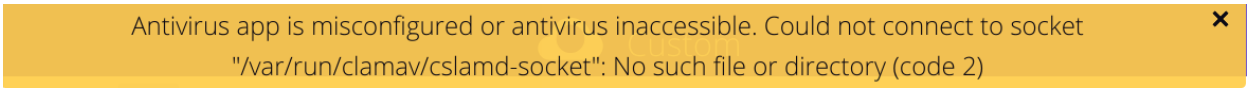
When you are satisfied with how ClamAV is operating, you might want to go back and change all of your logging to less verbose levels.

Configuration Warnings

The Antivirus App shows one of three warnings if it is misconfigured or ClamAV is not available. You can see an example of all three below.







What To Do With Infected Files Found During Scans

When infected files are detected during a scan, the app can take one of two actions:

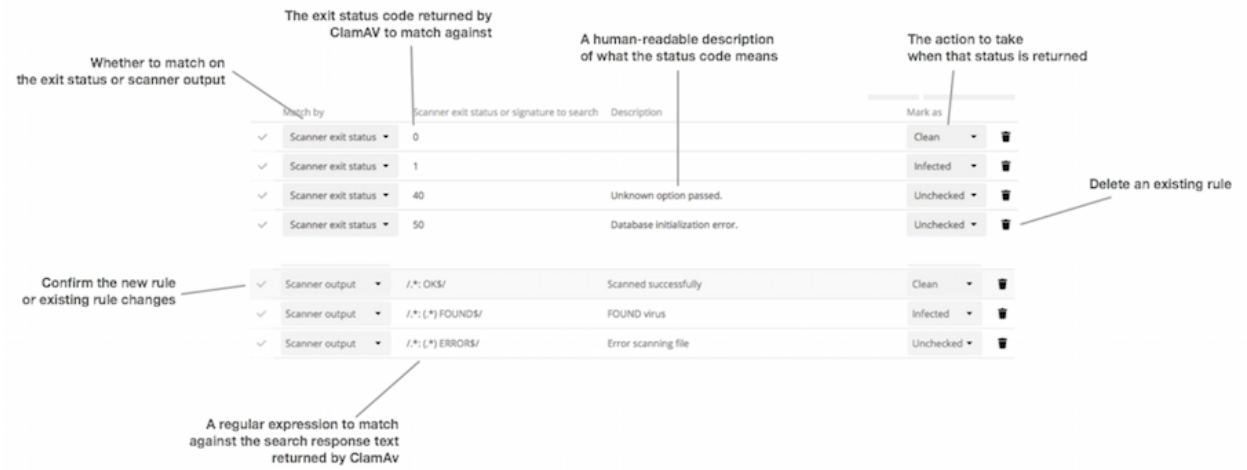
- **Only log**: Log the event.
- **Delete file**: Delete the detected file.

Set btn:[When infected files were found during a background scan] to the value that

suits your needs.

Rule Configuration

ownCloud provides the ability to customize how it reacts to the response given by an antivirus scan. To do so, under menu:Admin[Security (Admin)] click btn:[Advanced], which you can see in the screenshot below, you can view and change the existing rules. You can also add new ones.



Rules can match on either an exit status (e.g., 0, 1, or 40) or a pattern in the string returned from ClamAV (e.g., `/.: (.) FOUND$/`).

Here are some points to bear in mind about rules:

- Scanner exit status rules are used to handle errors when ClamAV is run in CLI mode while
- scanner output rules are used in daemon/socket mode.
- Daemon output is parsed by regexp.
- In case there are no matching rules, the status is: Unknown, and a warning will be logged.

Default Ruleset

The default rule set for ClamAV is populated automatically with the following rules:

Exit Status or Signature	Description	Marks File As
0		Clean
1		Infected
40	Unknown option passed	Unchecked
50	Database initialization error	Unchecked
52	Not supported file type	Unchecked
53	Can’t open directory	Unchecked
54	Can’t open file	Unchecked
55	Error reading file	Unchecked
56	Can’t stat input file	Unchecked

Exit Status or Signature	Description	Marks File As
57	Can’t get absolute path name of current working directory	Unchecked
58	I/O error	Unchecked
62	Can’t initialize logger	Unchecked
63	Can’t create temporary files/directories	Unchecked
64	Can’t write to temporary directory	Unchecked
70	Can’t allocate memory (calloc)	Unchecked
71	Can’t allocate memory (malloc)	Unchecked
/.*: OK$/		Clean
/.**: (.)** FOUND$/		Infected
/.**: (.)** ERROR$/		Unchecked

The rules are always checked in the following order:

1. Infected
2. Error
3. Clean

In case there are no matching rules, the status would be Unknown and a warning would be logged.

Update An Existing Rule

To match on an exit status, change the "**Match by**" dropdown list to "**Scanner exit status**" and in the "**Scanner exit status or signature to search**" field, add the status code to match on.

To match on the scanner’s output, change the "**Match by**" dropdown list to "**Scanner output**" and in the "**Scanner exit status or signature to search**" field, add the regular expression to match against the scanner’s output.

Then, while not mandatory, add a description of what the status or scan output means. After that, set what ownCloud should do when the exit status or regular expression you set matches the value returned by ClamAV. To do so change the value of the dropdown in the "**Mark as**" column.

The dropdown supports the following three options:

Option	Description
Clean	The file is clean and contains no viruses
Infected	The file contains a virus
Unchecked	No action should be taken

With all these changes made, click the btn:[check mark] on the left-hand side of the

"**Match by**" column, to confirm the change to the rule.

Add A New Rule

To add a new rule, click the button marked btn:[Add a rule] at the bottom left of the rules table. Then follow the process outlined in Update An Existing Rule.

Delete An Existing Rule

To delete an existing rule, click the btn:[rubbish bin] icon on the far right-hand side of the rule that you want to delete.

User

In this section, you will find all the information you need for user-related configuration in ownCloud.

- Users Page in ownCloud
- LDAP Authentication
- Password Reset for an Admin
- Password Reset for a User
- FTP, SMB, IMAP User Authentication
- User Provisioning API
- User Roles in ownCloud

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 ownCloud Web interface, so that the ownCloud server can email a reset link to you.
2. Ask another ownCloud server admin to reset it for you.

If neither of these is an option, then you have a third option, and that is using the occ command. occ is in the owncloud directory, for example /var/www/owncloud/occ. occ has a command for resetting all user passwords, user:resetpassword. It is best to run occ as the HTTP user, as in this example on Ubuntu Linux:

```
$ {occ-command-example-prefix} /var/www/owncloud/occ user:resetpassword
admin
Enter a new password:
Confirm the new password:
Successfully reset password for admin
```

If your ownCloud username is not admin, then substitute your ownCloud username.

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

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

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



Password changes automatically log out **all** connected browsers/devices.

Resetting a User Password

The ownCloud login screen displays a **Wrong password. Reset it?** message after a user enters an incorrect password, and then ownCloud automatically resets their password. However, if you are using a read-only authentication backend such as LDAP or Active Directory, this will not work. In this case you may specify a custom URL in your config.php file to direct your user to a server than can handle an automatic reset:

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



Password changes automatically log out **all** connected browsers/devices.

User Authentication with IMAP, SMB, and FTP

Overview

You may configure additional user backends in ownCloud's configuration file (config/config.php) using the following syntax:

```
<?php

"user_backends" => [
   0 => [
      "class"     => ...,
      "arguments" => [
         0 => ...
      ],
   ],
],
```



A non-blocking or correctly configured SELinux setup is needed for these backends to work, if SELinux is enabled on your server. Please refer to the SELinux configuration for further details.

Currently the External user support app (user_external), *which is not enabled by default*, provides three backends. These are:

- IMAP
- SMB
- FTP

See Installing and Managing Apps for more information.

IMAP

Provides authentication against IMAP servers.

Option	Value/Description
Class	OC_User_IMAP.
Arguments	A mailbox string as defined in the PHP documentation.
Dependency	PHP's IMAP extension. See Manual Installation on Linux
	for instructions on how to install it.

Example

```
<?php

"user_backends" => [
   0 => [
      "class"     => "OC_User_IMAP",
      "arguments" => [
         // The IMAP server to authenticate against
         '{imap.gmail.com:993/imap/ssl}',
         // The domain to send email from
         'example.com'
      ],
   ],
],
```



The second arguments parameter ensures that only users from that domain are allowed to login. When set, after a successful login, the domain will be stripped from the email address and the rest used as an ownCloud username. For example, if the email address is guest.user@example.com, then guest.user will be the username used by ownCloud.

SMB

Provides authentication against Samba servers.

Option	Value/Description
Class	OC_User_SMB.
Arguments	The samba server to authenticate against.
Dependency	PECL's smbclient extension or smbclient.

Example

```
<?php

"user_backends" => [
   [
      "class"     => "OC_User_SMB",
      "arguments" => [
         'localhost'
      ],
   ],
],
```

FTP

Provides authentication against FTP servers.

Option	Value/Description
Class	OC_User_FTP.
Arguments	The FTP server to authenticate against.
Dependency	PHP's FTP extension. See Source Installation. for instructions on how to install it.

Example

```
<?php

"user_backends" => [
   [
      "class"     => "OC_User_FTP",
      "arguments" => [
         'localhost'
      ],
   ],
],
```

User Authentication with LDAP

Introduction



Please check both the advanced and expert configurations carefully before using in production.

ownCloud ships with an LDAP application, called user_auth_ldap. This application allows LDAP users (including those from Active Directory) to appear in your ownCloud user listings. These users can authenticate to ownCloud with their LDAP credentials so that separate ownCloud user accounts don't need to be created for them. You will manage their ownCloud group memberships, quotas, and sharing permissions just like any other ownCloud user.



The PHP LDAP module is required. It is supplied by php7.2-ldap on Debian/Ubuntu and php-ldap on CentOS/Red Hat/Fedora. Please check for the correct version, based on your installation of PHP.

The LDAP application supports:

- LDAP group support
- File sharing with ownCloud users and groups
- Access via WebDAV and ownCloud Desktop Client
- Versioning, external Storage and all other ownCloud features
- Seamless connectivity to Active Directory, with no extra configuration required
- Support for primary groups in Active Directory
- Only read access to your LDAP (edit or delete of users on your LDAP is not supported)



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



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

Configuration

First, enable the LDAP user and group backend app on the Apps page in ownCloud. 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 light appears when the configuration is correct. Hover your cursor over the fields to see some pop-up tooltips.

Server Tab

Start with the Server tab. You may configure multiple servers if you have them. At a minimum, you must supply the LDAP server's hostname. If your server requires authentication, enter your credentials on this tab.



The base DN and port are mandatory.

Server configuration

Configure one or more LDAP servers. Click btn:[Delete Configuration] to remove the active configuration.

Host

The hostname or IP address of the LDAP server. It can also be an 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.

  Example:

  - 389

Use StartTLS support
: Check this option to use Transport Layer Security (TLS) when interacting with the LDAP server.



When using this option, ownCloud assumes that the CA certificates are properly setup. The connection will be encrypted, if the server supports it.

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=owncloudsystemuser,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. You can either enter this value manually, or click btn:[Detect Base DN] to have ownCloud attempt to determine the value.

  Example:

  - dc=my-company,dc=com

User Filter

Use this to control which LDAP users are listed as ownCloud users on your ownCloud server. In order to control which LDAP users can log in to your ownCloud server, use the Login filter. You may bypass the form fields and enter a raw LDAP filter if you prefer.

Only those object classes
: ownCloud determines the object classes that are typically available for user objects in your LDAP. ownCloud automatically selects the object class that returns the highest number of users. You may select multiple object classes.

Only from those groups
: If your LDAP server supports the memberof-overlay in LDAP filters, you can define that only users from one or more certain groups are allowed to appear in user listings in ownCloud. By default, no value is selected. You may select multiple groups.



Group membership is configured by adding memberUid, uniqueMember or member attributes to an LDAP group (see Group Member association) below. To efficiently look up the groups, a user who is a member of the LDAP server must support a memberof-overlay. It allows using the virtual memberOf or isMemberOf attributes of an LDAP user in the user filter. If your LDAP server does not support the memberof-overlay in LDAP filters, the input field is disabled. Please contact your LDAP administrator.

- Active Directory uses memberOf and is enabled by default.
- OpenLDAP uses memberOf. Reverse Group Membership Maintenance needs to be enabled.
- Oracle uses isMemberOf and is enabled by default.

Edit raw filter instead

Clicking on this text toggles the filter mode, and you can enter the raw LDAP filter directly. Example:

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

x users found

This is an indicator that tells you approximately how many users will be listed in ownCloud. The number updates automatically after any changes.

Active Directory offers "*Recursive retrieval of all AD group memberships of a user*". This means that you would be able to search the group you enter and all the other child groups from this group for users. Enter this filter to access this feature for a single group:

```
(&
   (objectClass=user)

(memberof:1.2.840.113556.1.4.1941:=CN=<groupname>,DC=example,DC=co
m)
)
```

Enter your group name instead of the <groupname> placeholder. If you want to search multiple groups with this feature, adjust your filter like this:

```
(&
   (objectClass=user)
   (|

(memberOf:1.2.840.113556.1.4.1941:=CN=<groupname1>,CN=Users,DC=exa
mple,DC=com)

(memberOf:1.2.840.113556.1.4.1941:=CN=<groupname2>,CN=Users,DC=exa
mple,DC=com)
   )
)
```

You can add as many groups to recurse by using the format: (|(m1)(m2)(m3).....). Here is the description from Microsoft (point #10):

> The string 1.2.840.113556.1.4.1941 specifies LDAP_MATCHING_RULE_IN_CHAIN. This applies only to DN attributes. This is an extended match operator that walks the chain of ancestry in objects all the way to the root until it finds a match. **This reveals group nesting.** It is available only on domain controllers with Windows Server 2003 SP2 or Windows Server 2008 (or above).

For more information, see the following from Technet:

- Active Directory: LDAP Syntax Filters
- Active Directory Week: Explore Group Membership with PowerShell

Login Filter

The settings in the Login Filter tab determine which LDAP users can log in to your ownCloud 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 User Filter 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 ownCloud.

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 raw filter instead
Clicking on this text toggles the filter mode, and you can enter the raw LDAP filter directly. Example:

The %uid placeholder is replaced with the login name entered by the user upon login.

Examples:

- Only username:

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

- Username or email address:

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

Group Filter

By default, no LDAP groups will be available in ownCloud. The settings in the group filter tab determine which groups will be available in ownCloud. You may also elect to enter a raw LDAP filter instead.

Only those object classes
ownCloud will determine the object classes that are typically available for group objects in your LDAP server. ownCloud 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 those groups
ownCloud will generate a list of available groups found in your LDAP server. From these groups, you can select the group or groups that get access to your ownCloud server.

Edit raw filter instead

> 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 ownCloud. The number updates automatically after any change.

Advanced Settings

The LDAP Advanced Setting section contains options that are not needed for a working connection. This provides controls to disable the current configuration, configure replica hosts, and various performance-enhancing options.

The Advanced Settings are structured into three parts:

- Connection Settings
- Directory Settings
- Special Attributes

Connection Settings

Configuration Active
: Enables or Disables the current configuration. By default, it is turned off. When ownCloud 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. ownCloud 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 supplied, 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 ownCloud only connect to the **backup server**. This is useful for planned downtimes for example **Upgrades or Updates of the Main Server**. **Backup Server Handling** When ownCloud is not able to contact the main LDAP server, ownCloud assumes it is offline and will not try to connect again for the time specified in" **Cache Time-To-Live**".

Turn off SSL certificate validation
: Turns off SSL certificate checking.



Use it for testing only!

Cache Time-To-Live

A cache is introduced to avoid unnecessary LDAP traffic, for example caching usernames so they don't have to be looked up for every page, and speeding up loading of the Users page. Saving the configuration empties the cache. The time is given in seconds. Note that almost every PHP request requires a new connection to the LDAP server. If you require fresh PHP requests we recommend defining a minimum lifetime of 15s or so, rather than completely eliminating the cache.

Examples:

- Ten minutes: 600
- One hour: 3600

See the Caching section below for detailed information on how the cache operates.

Directory Settings

User Display Name Field

The attribute that should be used as display name in ownCloud.

Examples:

- displayName
- givenName
- sn

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).

Examples:

- mail
- userPrincipalName
- sAMAccountName

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.

Examples:

- 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.

Examples:

- displayName
- mail

Group Display Name Field

The attribute that should be used as ownCloud group name. ownCloud allows a limited set of characters (a-zA-Z0-9.-_@). Once a group name is assigned it cannot be changed.

Examples:

- 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.

Examples:

- 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.

Examples:

- 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. ownCloud detects the value automatically. You should only change it if you have a very valid reason and know what you are doing.

Examples:

- member with FDN for Active Directory or for objectclass groupOfNames groups
- memberUid with RDN for objectclass posixGroup groups
- uniqueMember with FDN for objectclass groupOfUniqueNames groups



The Group Member association is used to efficiently query users of a certain group, e.g., on the userManagement page or when resolving all members of a group share.

Dynamic Group Member URL

The LDAP attribute that on group objects contains an LDAP search URL that determines what objects belong to the group. An empty setting disables dynamic group membership functionality. See Configuring Dynamic Groups for more details.

Nested Groups

This makes the LDAP connector aware that groups could be stored inside existing group records. By default a group will only contain users, so enabling this option isn't necessary. However, if groups are contained inside groups, and this option is not enabled, any groups contained within other groups will be ignored and not returned in search results.

Paging Chunk Size

This sets the maximum number of records able to be returned in a response when ownCloud requests data from LDAP. If this value is greater than the limit of the underlying LDAP server (such as 3000 for Microsoft Active Directory) the LDAP server will reject the request and the search request will fail. Given that, it is important to set the requested chunk size to a value no larger than that which the underlying LDAP server supports.

Special Attributes

Quota Field

The name of the LDAP attribute to retrieve the user quota limit from, e.g., ownCloudQuota. *Note:* any quota set in LDAP overrides quotas set in ownCloud's user management page.

Quota Default

Override ownCloud's default quota **for LDAP users** who do not have a quota set in the Quota Field, e.g., 15 GB. Please bear in mind the following, when using these fields to assign user quota limits. It should help to alleviate any, potential, confusion.

1. After installation ownCloud uses an unlimited quota by default.
2. Administrators can modify this value, at any time, in the user management page.
3. However, when an LDAP quota is set it will override any values set in ownCloud.
4. If an LDAP per/attribute quota is set in the active directory, it will override the LDAP Quota Default value.



Administrators are not allowed to modify the user quota limit in the user management page when steps 3 or 4 are in effect. At this point, updates are only possible via LDAP. See the LDAP Schema for ownCloud Quota

Email Field

Set the user's email from an LDAP attribute, e.g., mail. Leave it empty for default behavior.

User Home Folder Naming Rule

By default, the ownCloud server creates the user directory in your ownCloud data directory and gives it the ownCloud username, e.g., /var/www/owncloud/data/5a9df029-322d-4676-9c80-9fc8892c4e4b, if your data directory is set to /var/www/owncloud/data.

It is possible to override this setting and name it after an LDAP attribute value, e.g., attr:cn. The attribute can return either an absolute path, e.g., /mnt/storage43/alice, or a relative path which must not begin with a /, e.g., CloudUsers/CookieMonster. This relative path is then created inside the data directory (e.g., /var/www/owncloud/data/CloudUsers/CookieMonster).

Since ownCloud 8.0.10 and up 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. For older versions you may enforce the home folder rule with the occ command, like this example on Ubuntu:

```
{occ-command-example-prefix} config:app:set user_ldap
enforce_home_folder_naming_rule --value=1
```

Since ownCloud 10.0 the home folder naming rule is only applied when first provisioning the user. This prevents data loss due to re-provisioning the users home folder in case of unintentional changes in LDAP.

Expert Settings

In "**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 ownCloud 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 ownCloud, i.e., that it is checking all other activated user backends (including local ownCloud 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 ownCloud. 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.

Examples:

- uid

Override UUID detection

By default, ownCloud 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 that it is unique. Leave it empty for default behavior. Changes will have effect only on newly mapped LDAP users and groups.

It also will take effect when a user 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 ownCloud in production use and clearing the bindings the (see User and Group Mapping` section below).

Examples:

- cn

Username-LDAP User Mapping

ownCloud 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 ownCloud. This requires a mapping from an ownCloud username to an LDAP user.

The created username is mapped to the UUID of the LDAP user. Additionally, the DN is cached to reduce LDAP interaction, but it is not used for identification. If the DN changes, the change will be detected by ownCloud by checking the UUID value.

The same is valid for groups. The internal ownCloud name is used all over in ownCloud. Clearing the mappings will have leftovers everywhere. Never clear the mappings in a production environment, but only in a testing or experimental server.



Clearing the mappings is not configuration sensitive, it affects all LDAP configurations!

Testing the Configuration

The "**Test Configuration**" button checks the values as currently given in the input fields. You do not need to save before testing. By clicking on the button, ownCloud will try to bind to the ownCloud 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.

Syncing Users

While users who match the login and user filters can log in, only synced users will be found in the sharing dialog. Whenever users log in, their display name, email, quota, avatar and search attributes will be synced to ownCloud. If you want to keep the metadata up to date you can set up a cron job, using the occ user:sync command. Versions of ownCloud before 10.0 imported all users when the users page was loaded, but this is no longer the case.



We recommend creating a Cron job, to automate regularly syncing LDAP users with your ownCloud database. If you have many users, you do not have to sync all of them to update a small number of users. You can use the OCS User Sync API to sync individual users. It provides a way to trigger user sync from outside of ownCloud.

How Often Should the Job Run?

This depends on the amount of users and speed of the update, but we recommend *at least* once per day. You can run it more frequently, but doing so may generate too much load on the server.

Reuse Existing User and Group LDAP Accounts

New LDAP logins can attempt to reuse *existing* user and group accounts if:

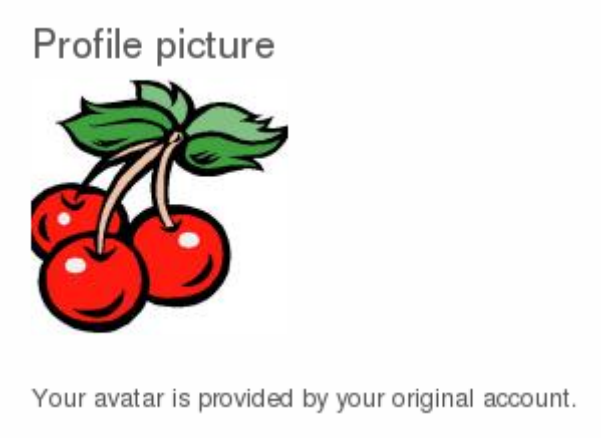
- They match the resolved username attribute.
- They have User_Proxy set as their backend.

To enable this functionality, the reuse_accounts config setting must be set to yes. To enable it, run the following command.

```
{occ-command-example-prefix} config:app:set user_ldap reuse_accounts
--value=yes
```

ownCloud Avatar Integration

ownCloud 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 ownCloud Personal pages. Avatars managed in ownCloud 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 ownCloud. LDAP avatars override ownCloud avatars, and when an LDAP avatar is deleted then the most recent ownCloud avatar replaces it.

Photos served from LDAP are automatically cropped and resized in ownCloud. This affects only the presentation, and the original image is not changed.

Troubleshooting, Tips and Tricks

SSL Certificate Verification (LDAPS, TLS)

A common mistake with SSL certificates is that they may not be known to PHP. If you have trouble with certificate validation, make sure that:

- You have the certificate of the server installed on the ownCloud server.
- The certificate is listed in the system's LDAP configuration file, usually /etc/ldap/ldap.conf.
- If you are using LDAPS, make sure that the port is correctly configured (the default port is 636)
- If you get the error "**Lost connection to LDAP server**" or "**No connection to LDAP server**", double-check the connection parameters and try connecting to LDAP with tools like ldapsearch. If using LDAPS or TLS, make sure the certificate is readable by the user that is used to serve ownCloud.

Microsoft Active Directory

Compared to earlier ownCloud versions, no further tweaks need to be done to make ownCloud work with Active Directory. ownCloud will automatically find the correct configuration in the set-up process.

memberOf / Read MemberOf permissions

If you want to use memberOf within your filter you might need to give your querying user the permissions to use it. For Microsoft Active Directory this is described here.

Duplicating Server Configurations

In case you have a working configuration and want to create a similar one or "snapshot" configurations before modifying them you can do the following:

1. Go to the "**Server**" tab
2. On "**Server Configuration**" choose "**Add Server Configuration**"
3. Answer the question "**Take over settings from recent server configuration?**" with "**yes**".
4. (optional) Switch to "**Advanced**" tab and uncheck "**Configuration Active**" in the "**Connection Settings**", so the new configuration is not used on Save
5. Click btn:[Save]

Now you can modify and enable the configuration.

Performance Tips

Filter out Deactivated Users

With this filter you can filter out the deactivated users and show only active users.

```
!(userAccountControl:1.2.840.113556.1.4.803:=2)
```

Here is what the full filter can look like.

```
&(|(objectclass=organizationalPerson))
  (!(userAccountControl:1.2.840.113556.1.4.803:=2))
  (|(|(memberof=CN=Domain
Users,CN=Users,DC=dp,DC=mosreg,DC=ru)(primaryGroupID=513)))
)
```

Caching

Using caching to speed up lookups. The ownCloud cache is populated on demand, and remains populated until the **Cache Time-To-Live** for each unique request expires. User logins are not cached, so if you need to improve login times set up a slave LDAP server to share the load.

You can adjust the "**Cache Time-To-Live**" value to balance performance and freshness of LDAP data. All LDAP requests will be cached for 10 minutes by default, and you can alter this with the "**Cache Time-To-Live**" setting. The cache answers each request that is identical to a previous request, within the time-to-live of the original request, rather than hitting the LDAP server.

The "**Cache Time-To-Live**" is related to each single request. After a cache entry expires there is no automatic trigger for re-populating the information, as the cache is populated only by new requests, for example by opening the User administration page, or searching in a sharing dialog.

There is one trigger which is automatically triggered by a certain background job which keeps the user-group-mappings up-to-date, and always in cache.

Under normal circumstances, all users are never loaded at the same time. Typically the loading of users happens while page results are generated, in steps of 30 until the limit is reached or no results are left. For this to work on an oC-Server and LDAP-Server, "**Paged Results**" must be supported, which assumes PHP >= 5.6.



Please ensure that you're using the minimum supported PHP version ({minimum-php-version}).

ownCloud 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.

LDAP Indexing

Turn on indexing. Deciding which attributes to index depends on your configuration and which LDAP server you are using. See the openLDAP tuning guide for openLDAP, and How to Index an Attribute in Active Directory for Active Directory.

Use Precise Base DNs

The more precise your base DN, the faster LDAP can search because it has fewer branches to search.

Use Precise Filters

Use good filters to further define the scope of LDAP searches, and to intelligently direct your server where to search, rather than forcing it to perform needlessly-general searches.

ownCloud LDAP Internals

Some parts of how the LDAP backend works are described here.

User and Group Mapping

In ownCloud, 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 are 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.

From ownCloud 5, the internal user name and a visible display name are separated. This is not the case for group names, yet, i.e., a group name cannot be altered.

That means that your LDAP configuration should be good and ready before putting it into production. The mapping tables are filled early, but as long as you are testing, you can empty the tables any time.



Do not do this in production.

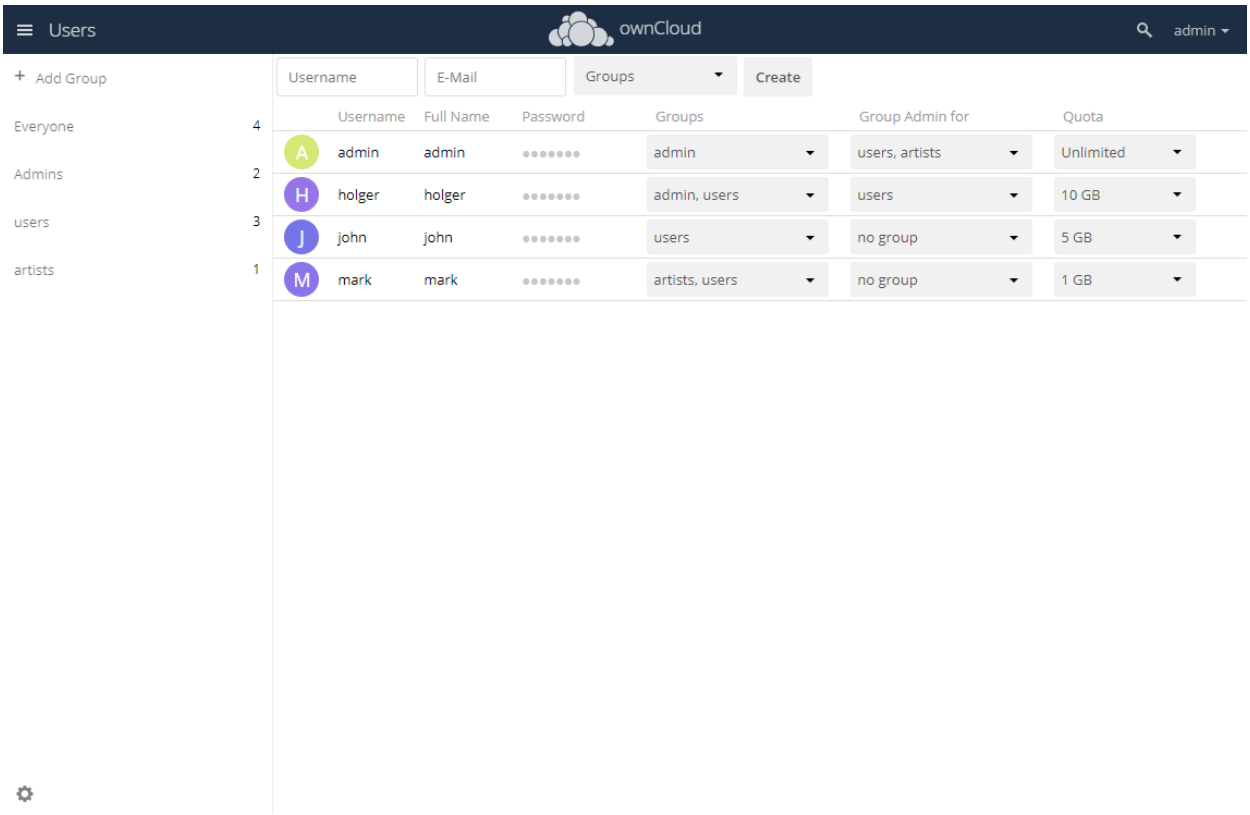
Handling with Backup Server

When ownCloud is not able to contact the main LDAP server, ownCloud 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 ownCloud will connect to it instead. When you have scheduled downtime, check btn:[Disable Main Server] to avoid unnecessary connection attempts.

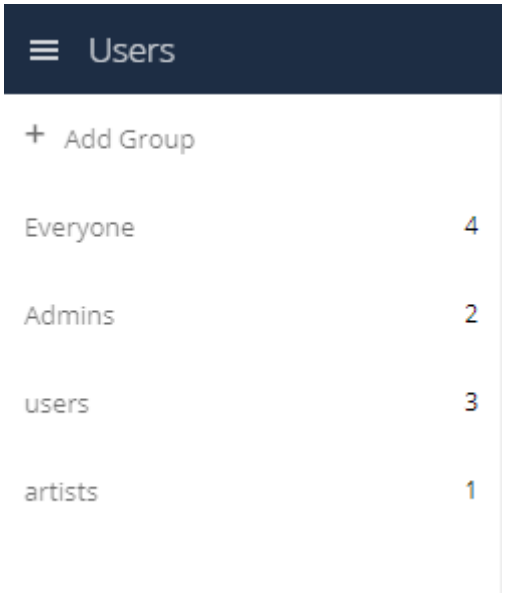
User Management

Default View

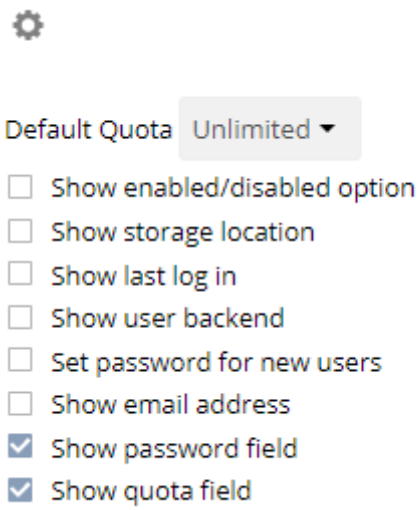
The **default view** displays basic information about your users.



The **Group filter** on the left sidebar lets you quickly filter users by their group memberships, and create new groups.



Click the btn:[gear] icon on the lower left sidebar to view the avaiable settings.



User accounts have the following **properties**:

Login Name (Username)
: The unique ID of an ownCloud user, and it cannot be changed.

Full Name
: The user's display name that appears on file shares, the ownCloud Web interface, and emails. Admins and users may change the Full Name anytime. If the Full Name is not set it defaults to the login name.

Password
: The admin sets the new user's first password. Both the user and the admin can change the user's password at anytime.

E-Mail
: The admin sets the new user's E-Mail. The user then get's an E-Mail to set his Password. Both the user and the admin can change the user's E-Mail at anytime.

Groups
: You may create groups, and assign group memberships to users. By default new users are not assigned to any groups.

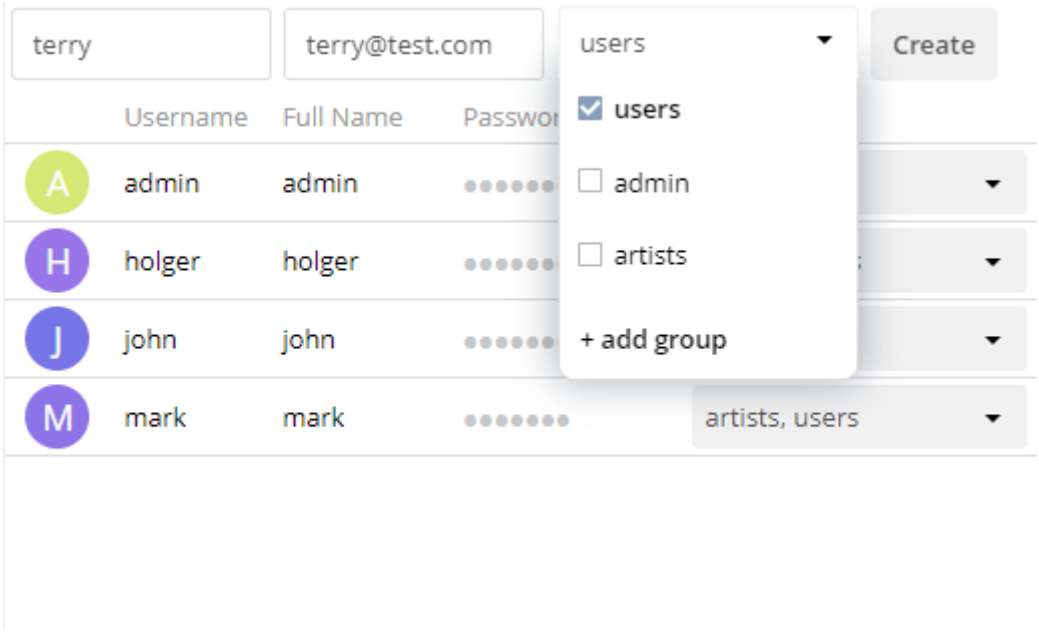
Group Admin
: Group admins are granted administrative privileges on specific groups, and can add and remove users from their groups.

Quota
: The maximum disk space assigned to each user. Any user that exceeds the quota cannot upload or sync data. You have the option to include external storage in user quotas.

Creating a New User

To create a user account:

- Enter the new user's **Login Name** and their **E-Mail**
- Optionally, assign **Groups** memberships
- Click the btn:[Create] button

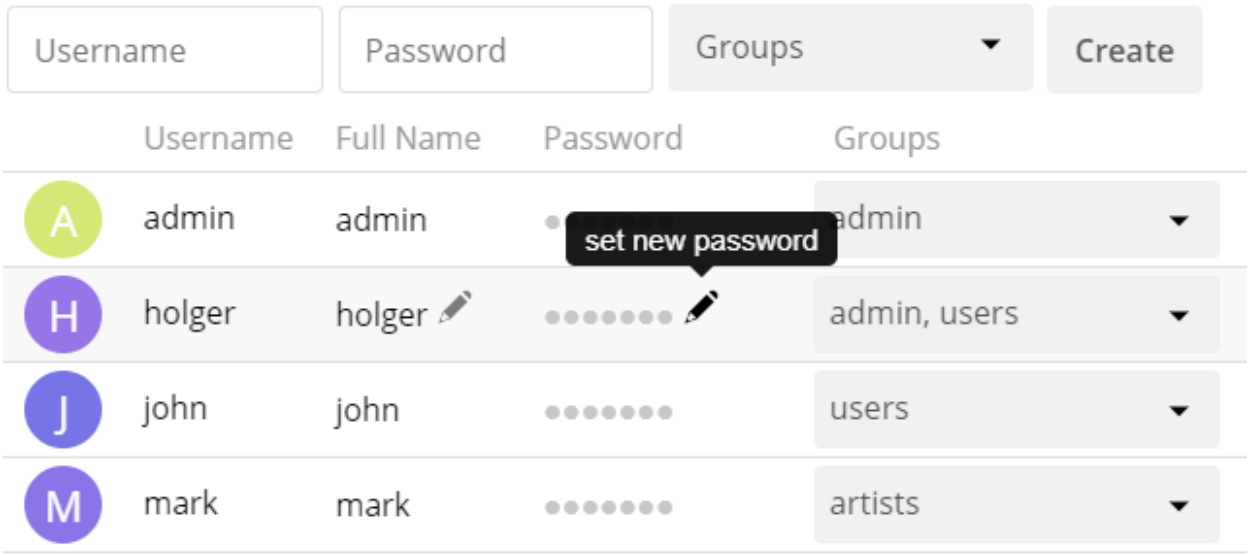


Login names may contain letters (a-z, A-Z), numbers (0-9), dashes (-), underscores (_), periods (.) and at signs (@). After creating the user, you may fill in their **Full Name** if it is different than the login name, or leave it for the user to complete.

Password Reset

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 btn:[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.



See Encryption Configuration.

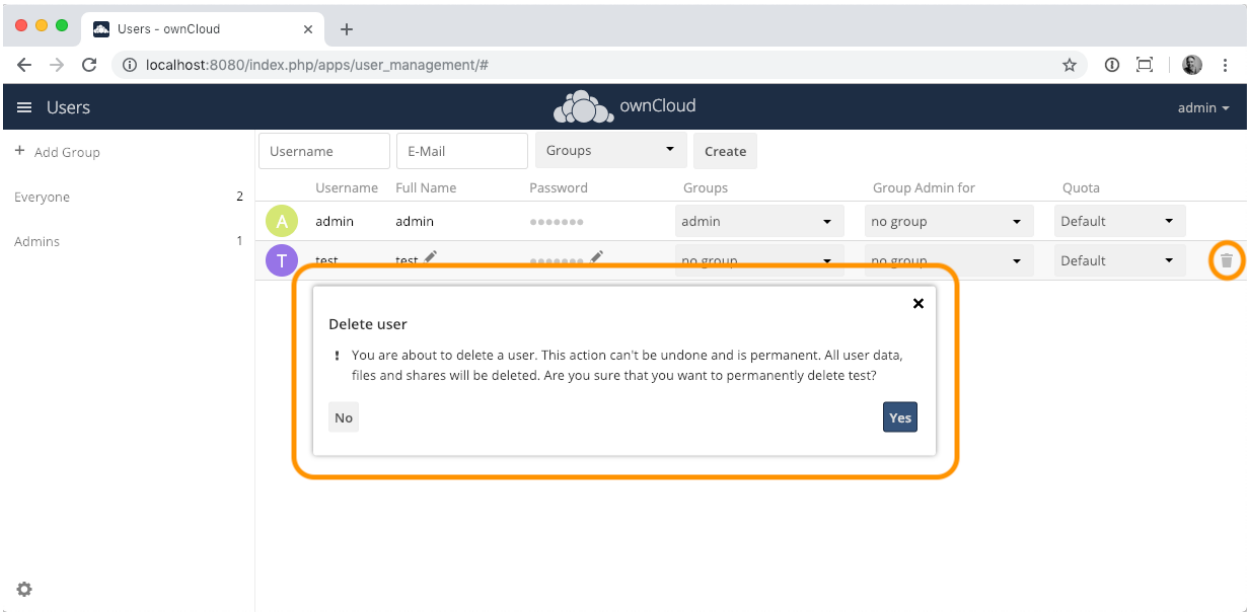
Renaming a User

Each ownCloud 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 btn:[pencil] icon
- Enter the user's new display name

Deleting Users



To delete a user, hover your cursor over their name on the **Users** page, and click the trashcan icon that appears at the far right. You'll then see a confirmation dialog appear, asking if you're sure that you want to delete the user.

If you click btn:[Yes], the user is permanently deleted, including all of the files owned by the user, including all files they have shared. If you need to preserve the user's files and shares, you must first download them from your ownCloud Files page, (which compresses them into a zip file).

Alternatively, you can use a sync client to copy them to your local computer. If you click btn:[No], the confirmation dialog will disappear and the user is not deleted.



See File Sharing Configuration to learn how to create persistent file shares that survive user deletions.

Granting Administrator Privileges

ownCloud has two types of administrators:

- **ownCloud Administrators** have full rights on your ownCloud server, and can access and modify all settings. To assign the ownCloud Administrators role to a user, simply add them to the admin group.
- **Group Administrators**. Group administrators have the rights to create, edit and delete users in their assigned groups. Use the dropdown menus in the Group Admin column to assign group admin privileges.

Managing Groups

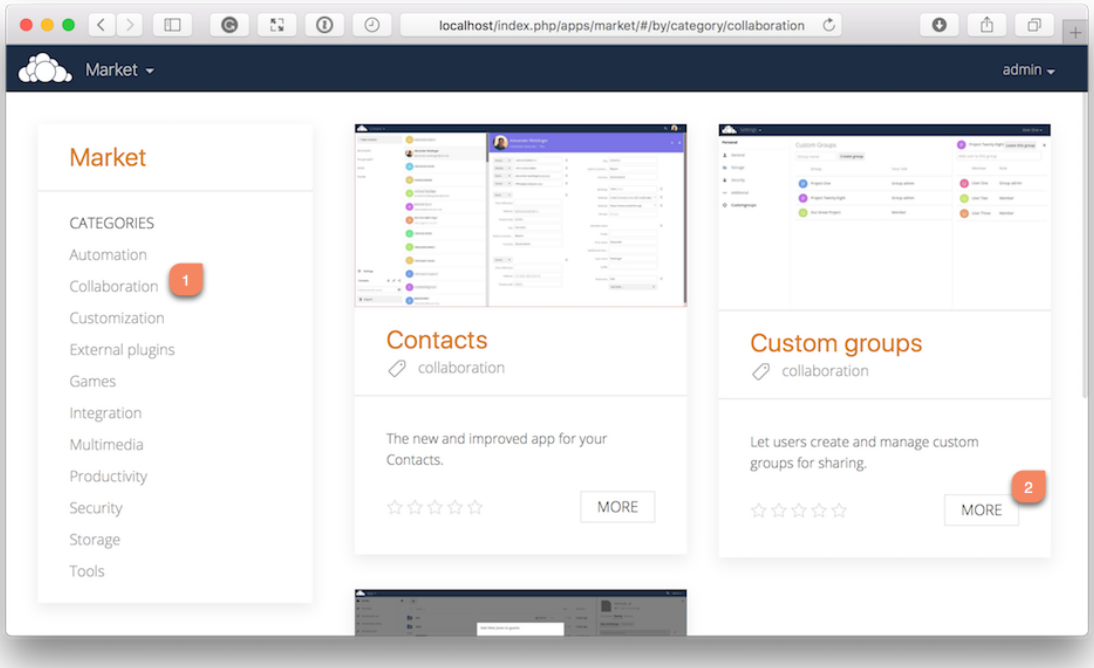
You can assign new users to groups when you create them, and create new groups when you create new users. You may also use the **Add Group** button at the top of the left pane to create new groups. New group members will immediately have access to file shares that belong to their new groups.

Enabling Custom Groups

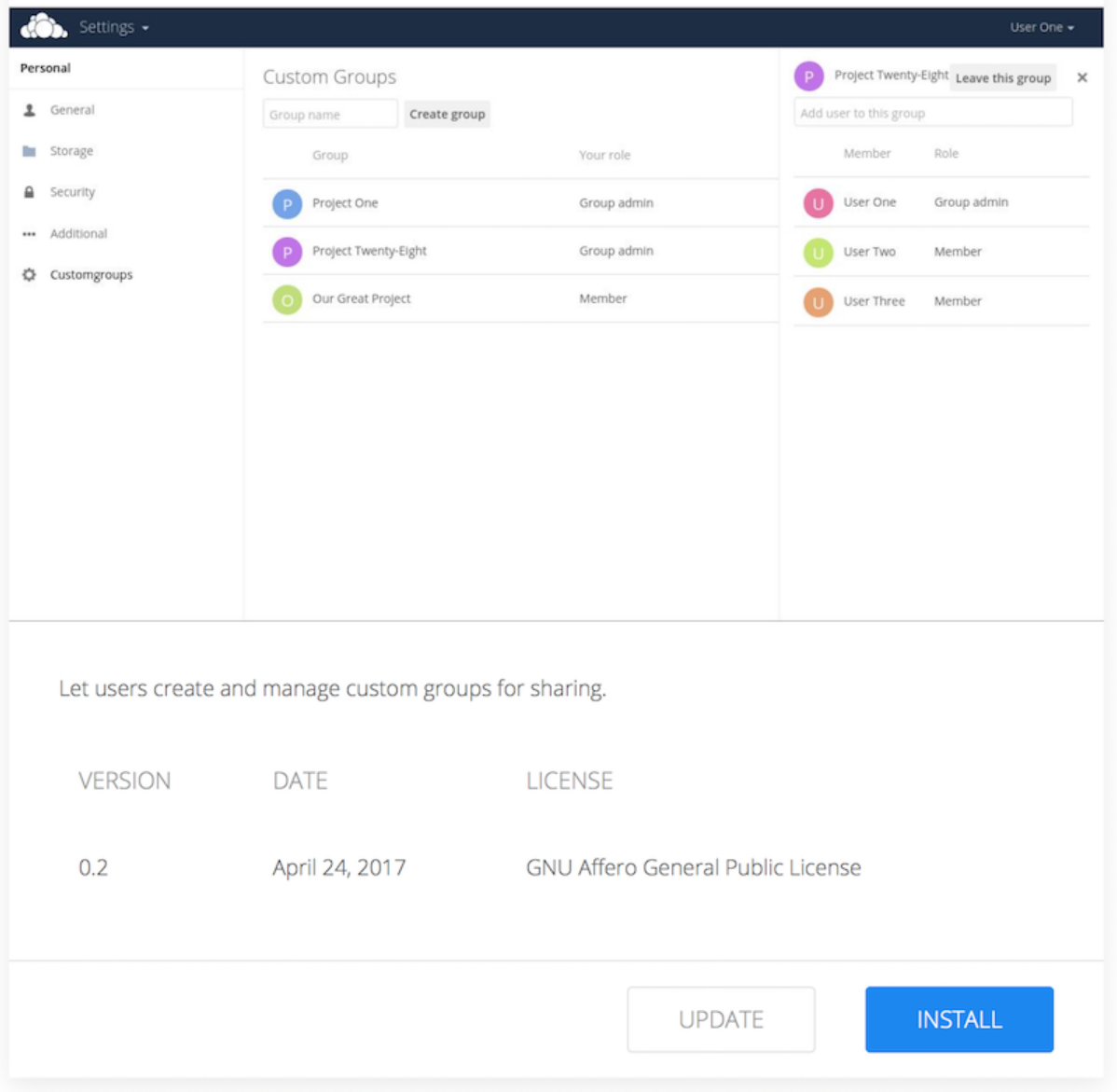
In previous versions of ownCloud, files and folders could only be shared with individual users or groups created by administrators. This wasn't the most efficient way to work. From ownCloud 10.0, users can create groups on-the-fly, through a feature called "Custom Groups", enabling them to share content in a more flexible way.

To enable Custom Groups:

1. From the ownCloud Market, which you can find in version 10.0 under the Apps menu, click btn:[Market].
2. Click btn:[Collaboration] (1), to filter the list of available options and click the btn:[Custom groups] application (2).



3. Click btn:[INSTALL] in the bottom right-hand corner of the Custom Groups application.



With this done, Custom Group functionality will be available in your ownCloud installation.

Overriding Default Behavior

Disabling Administrators from Administering Custom Groups

Depending on your Custom Groups and ownCloud's global settings, configured by the ownCloud admin, Custom Groups may behave differently:

- Creating or renaming a Custom Group using an existing name of another Custom Group can be allowed or not depending on administrative settings.
- Custom Group creation can be limited to ownCloud **group admins**.
- Disable administration of Custom Groups by ownCloud administrators. This is enabled by setting customgroups.disallow-admin-access-all to true in config/config.php.

Hide Custom Groups App Based On Group Membership

The app can be hidden from the user's personal settings page if the user belongs to one or more disallowed groups, To specify the disallowed groups, list them against the customgroups.disallowed-groups key in config/config.php, as in the following example.

```
// Hide the Custom Groups app for users in the "guest_app" group.
'customgroups.disallowed-groups' => ['guest_app'],
```

Setting Storage Quotas

There are 4 types of quota settings in ownCloud when dealing with LDAP users.

Quota Field

Found in menu:User Authentication[the Advanced Tab > Special Attributes], this setting overwrites the rest. If set, this is what will be set for an LDAP user's quota in ownCloud.

Quota Default

Found in menu:User Authentication[the Advanced Tab > Special Attributes], this is the fallback option if no quota field is defined.

User Quota

This is what you set in the web UI drop down menu, and is how you set user quota.

Default Quota

This will be set if no quota is set, and is found in menu:Users Tab[Gear Wheel > Default Quota]. If **Quota Field** is not set, but **Quota Default** is, and a systems administrator tries to set a quota for an LDAP user with **User Quota**, it will not work, since it is overridden by **Quota Default**.

Click the btn:[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 normal abbreviations for your storage values such as 500 MB, 5 GB, 5 TB, and so on.

External Storage Quota

You now have a configurable option in config.php that controls whether external storage is counted against user's quotas. This is still experimental, and may not work as expected. The default is to not count external storage as part of user storage quotas. If you prefer to include it, then change the default false to true.:

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

Storage Space Considerations

Metadata (such as thumbnails, temporary files, and encryption keys) takes up about 10% of disk space, but is not counted against user quotas. Users can check their used and available space on their Personal pages. Only files that originate with users count against their quotas, and not files shared with them that originate from other users. For example, if you upload files to a different user's share, those files count against your quota. If you re-share a file that another user shared with you, that file does not count against your quota, but the originating user's.

Encrypted files are a little larger than unencrypted files; the unencrypted size is calculated against the user's quota.

Deleted files that are still in the trash bin do not count against quotas. The trash bin is set at 50% of quota. Deleted file aging is set at 30 days. When deleted files exceed 50% of quota then the oldest files are removed until the total is below 50%.

Versions

When version control is enabled, the older file versions are not counted against quotas.

Public Links

When a user creates a public link share via URL, and allows uploads, any uploaded files count against that user's quota.

User Provisioning API

Introduction

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 ownCloud
- Group admin users can also query ownCloud and perform the same functions as an admin for groups they manage.
- Query for active ownCloud applications, application info, and to enable or disable an app.

HTTP requests can be used via a Basic Auth header to perform any of the functions listed above. The Provisioning API app is enabled by default. The base URL for all calls to the share API is **owncloud_base_url/ocs/v1.php/cloud**.

Instruction Set For Users

Add User

Create a new user on the ownCloud server. Authentication is done by sending a basic HTTP authentication header.

Syntax

Request Path	Method	Content Type
ocs/v1.php/cloud/users	POST	text/plain

Argument	Type	Description
email	string	The email address for the user (*optional*). The user will be emailed a link to set their password, *if* email is configured correctly.

Argument	Type	Description
groups	array	Groups to add the user to (*optional*). Any group that does not already exist, will be created.
password	string	The required password for the new user. A password is **not required**, *if* an email address is provided. If a password is not provided, a temporary one will be generated.
userid	string	The required username for the new user.

Status Codes

- 100 - successful
- 101 - invalid input data
- 102 - username already exists
- 103 - unknown error occurred whilst adding the user
- 104 - group does not exist

Usage Examples

Listing 15. Create the user "Frank" with password "frankspassword".

```
curl -X POST http://admin:secret@example.com/ocs/v1.php/cloud/users \
  -d userid="Frank" \
  -d password="frankspassword"
```

Listing 16. Create the user "Frank" with password "frankspassword" and add him to the "finance" and "management" groups.

```
curl -X POST http://admin:secret@example.com/ocs/v1.php/cloud/users \
  -d userid="Frank" \
  -d password="frankspassword" \
  -d groups[]="finance" -d groups[]="management"
```

Listing 17. Create the user "Frank" with password "frankspassword", email address "frank@example.com" and add him to the "finance" and "management" groups. The user will be emailed with a link to change the password.

```
curl -X POST http://admin:secret@example.com/ocs/v1.php/cloud/users \
  -d userid="Frank" \
  -d email="frank@example.com" \
  -d password="frankspassword" \
  -d groups[]="finance" -d groups[]="management"
```

XML Output

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

Get Users

Retrieves a list of users from the ownCloud server. Authentication is done by sending a Basic HTTP Authorization header.

Request Path	Method	Content Type
ocs/v1.php/cloud/users	GET	text/plain

Argument	Type	Description
search	string	optional search string
limit	int	optional limit value
offset	int	optional offset value

Status Codes

- 100 - successful

Get Users Example

```
# Returns list of users matching the search string.
curl http://admin:secret@example.com/ocs/v1.php/cloud/users?search=Frank
```

XML Output

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

Get User

Retrieves information about a single user. Authentication is done by sending a Basic HTTP Authorization header.

Request Path	Method	Content Type
Syntax: ocs/v1.php/cloud/users/{userid}	GET	text/plain

Argument	Type	Description
userid	int	Id of the user to retrieve

Status Codes

- 100 - successful

Get User Example

```
# Returns information on the user "Frank"
curl http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank
```

XML Output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <status>ok</status>
    <statuscode>100</statuscode>
    <message/>
  </meta>
  <data>
    <enabled>true</enabled>
    <quota>
      <free>81919008768</free>
      <used>5809166</used>
      <total>81924817934</total>
      <relative>0.01</relative>
    </quota>
    <email>user@example.com</email>
    <displayname>Frank</displayname>
    <home>/mnt/data/files/Frank</home>
    <two_factor_auth_enabled>false</two_factor_auth_enabled>
  </data>
</ocs>
```

Edit User

Edits attributes related to a user. Users are able to edit *email*, *displayname* and *password*; admins can also edit the quota value.



The Basic Authorization HTTP header must be used to authenticate this request, using the credentials of a user who has sufficient access rights to make the request.

Request Path	Method	Content Type
ocs/v1.php/cloud/users/{userid}	PUT	text/plain

Argument	Type	Description
key	string	the field to edit (email, quota, display, password)
value	mixed	the new value for the field

Status Codes

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

Edit User Example

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

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

XML Output

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

Enable User

Enables a user on the ownCloud server.



The Basic Authorization HTTP header must be used to authenticate this request, using the credentials of a user who has sufficient access rights to make the request.

Request Path	Method	Content Type
ocs/v1.php/cloud/users/{userid}/enable	PUT	text/plain

Argument	Type	Description
userid	string	The id of the user to enable

Status Codes

- 100 - successful
- 101 - failure

Enable User Example

```
# Enable the user with the userid "Frank"
curl -X PUT http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/enable
```

XML Output

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

Disable User

Disables a user on the ownCloud server.



The Basic Authorization HTTP header must be used to authenticate this request, using the credentials of a user who has sufficient access rights to make the request.

Request Path	Method	Content Type
ocs/v1.php/cloud/users/{userid}/disable	PUT	text/plain

Argument	Type	Description
userid	string	The id of the user to disable

Status Codes

- 100 - successful
- 101 - failure

Disable User Example

```
# Disable the user "Frank"
curl -X PUT http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/disable
```

XML Output

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

Delete User

Deletes a user from the ownCloud server. Authentication is done by sending a Basic HTTP Authorization header.

Request Path	Method	Content Type
ocs/v1.php/cloud/users/{userid}	DELETE	text/plain

Argument	Type	Description
userid	string	The id of the user to delete

Status Codes

- 100 - successful
- 101 - failure

Delete User Example

```
# Deletes the user "Frank"
curl -X DELETE http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank
```

XML Output

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

Get Groups

Retrieves a list of groups the specified user is a member of.



The Basic Authorization HTTP header must be used to authenticate this request, using the credentials of a user who has sufficient access rights to make the request.

Request Path	Method	Content Type
ocs/v1.php/cloud/users/{userid}/groups	GET	text/plain

Argument	Type	Description
userid	string	The id of the user to retrieve groups for

Status Codes

- 100 - successful

Get Groups Example

```
# Retrieves a list of groups of which "Frank" is a member
curl http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/groups
```

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 To Group

Adds the specified user to the specified group. Authentication is done by sending a Basic HTTP Authorization header.

Request Path	Method	Content Type
ocs/v1.php/cloud/users/{userid}/groups	POST	text/plain

Argument	Type	Description
userid	string	The id of the user to retrieve groups for
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

Add To Group Example

```
# Adds the user "Frank" to the group "newgroup"
curl -X POST
http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/groups -d
groupid="newgroup"
```

XML Output

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

Remove From Group

Removes the specified user from the specified group. Authentication is done by sending a Basic HTTP Authorization header.

Request Path	Method	Content Type
ocs/v1.php/cloud/users/{userid}/groups	DELETE	text/plain

Argument	Type	Description
userid	string	The id of the user to retrieve groups for
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

Remove From Group Example

```
# Removes the user "Frank" from the group "newgroup"
curl -X DELETE
http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/groups -d
groupid="newgroup"
```

XML Output

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

Create Sub-admin

Makes a user the sub-admin of a group. Authentication is done by sending a Basic HTTP Authorization header.

Request Path	Method	Content Type
ocs/v1.php/cloud/users/{userid}/subadmins	POST	text/plain

Argument	Type	Description
userid	string	The id of the user to be made a sub-admin
groupid	string	the group of which to make the user a sub-admin

Status Codes

- 100 - successful
- 101 - user does not exist
- 102 - group does not exist
- 103 - unknown failure

Create Sub-admin Example

```
# Makes the user "Frank" a sub-admin of the "group" group
curl -X POST
https://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/subadmins -d
groupid="group"
```

XML Output

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

Remove Sub-admin

Removes the sub-admin rights for the user specified from the group specified. Authentication is done by sending a Basic HTTP Authorization header.

Request Path	Method	Content Type
ocs/v1.php/cloud/users/{userid}/subadmins	DELETE	text/plain

Argument	Type	Description
userid	string	the id of the user to retrieve groups for
groupid	string	the group from which to remove the user's sub-admin rights

Status Codes

- 100 - successful
- 101 - user does not exist
- 102 - user is not a sub-admin of the group / group does not exist
- 103 - unknown failure

Remove Sub-admin Example

```
# Removes "Frank's" sub-admin rights from the "oldgroup" group
curl -X DELETE
https://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/subadmins -d
groupid="oldgroup"
```

XML Output

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

Get Sub-admin Groups

Returns the groups in which the user is a sub-admin. Authentication is done by sending a Basic HTTP Authorization header.

Request Path	Method	Content Type
ocs/v1.php/cloud/users/{userid}/subadmins	GET	text/plain

Argument	Type	Description
userid	string	The id of the user to retrieve sub-admin groups for

Status Codes

- 100 - successful
- 101 - user does not exist
- 102 - unknown failure

Get Sub-admin Groups Example

```
# Returns the groups of which "Frank" is a sub-admin
curl -X GET
https://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/subadmins
```

XML Output

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

Get Groups

Retrieves a list of groups from the ownCloud server. Authentication is done by sending a Basic HTTP Authorization header.

Request Path	Method	Content Type
ocs/v1.php/cloud/groups	GET	text/plain

Argument	Type	Description
search	string	optional search string
limit	int	optional limit value
offset	int	optional offset value

Status Codes

- 100 - successful

Get Groups Example

```
# Returns list of groups matching the search string.
curl http://admin:secret@example.com/ocs/v1.php/cloud/groups?search=admi
```

XML Output

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

Add Group

Adds a new group. Authentication is done by sending a Basic HTTP Authorization header.

Request Path	Method	Content Type
ocs/v1.php/cloud/groups	POST	text/plain

Argument	Type	Description
groupid	string	the new group’s name

Status Codes

- 100 - successful
- 101 - invalid input data
- 102 - group already exists
- 103 - failed to add the group

Add Group Example

```
# Adds a new group called "newgroup"
curl -X POST http://admin:secret@example.com/ocs/v1.php/cloud/groups -d
groupid="newgroup"
```

XML Output

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

Get Group

Retrieves a list of group members. Authentication is done by sending a Basic HTTP Authorization header.

Request Path	Method	Content Type
ocs/v1.php/cloud/groups/{groupid}	GET	text/plain

Argument	Type	Description
groupid	string	The group id to return members from

Status Codes

- 100 - successful

Get Group Example

```
# Returns a list of users in the "admin" group
curl http://admin:secret@example.com/ocs/v1.php/cloud/groups/admin
```

XML Output

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

Get Sub-admins

Returns sub-admins of the group. Authentication is done by sending a Basic HTTP Authorization header.

Request Path	Method	Content Type
ocs/v1.php/cloud/groups/{groupid}/subadmins	GET	text/plain

Argument	Type	Description
groupid	string	The group id to get sub-admins for

Status Codes

- 100 - successful
- 101 - group does not exist
- 102 - unknown failure

Get Sub-admins Example

```
# Return the sub-admins of the group: "mygroup"
curl
https://admin:secret@example.com/ocs/v1.php/cloud/groups/mygroup/subadmins
```

XML Output

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

Delete Group

Removes a group. Authentication is done by sending a Basic HTTP Authorization header.

Request Path	Method	Content Type
ocs/v1.php/cloud/groups/{groupid}	DELETE	text/plain

Argument	Type	Description
groupid	string	the group to delete

Status Codes

- 100 - successful
- 101 - group does not exist
- 102 - failed to delete group

Delete Group Example

```
# Delete the group "mygroup"
curl -X DELETE
http://admin:secret@example.com/ocs/v1.php/cloud/groups/mygroup
```

XML Output

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

Get Apps

Returns a list of apps installed on the ownCloud server. Authentication is done by sending a Basic HTTP Authorization header.

Request Path	Method	Content Type
ocs/v1.php/cloud/apps/	GET	text/plain

Argument	Type	Description
filter	string	Whether to retrieve enabled or disable
		apps. Available values are enabled
		and disabled.

Status Codes

- 100 - successful
- 101 - invalid input data

Get Apps Example

```
# Gets enabled apps
curl http://admin:secret@example.com/ocs/v1.php/cloud/apps?filter=enabled
```

XML Output

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

Get App Info

Provides information on a specific application. Authentication is done by sending a Basic HTTP Authorization header.

Request Path	Method	Content Type
ocs/v1.php/cloud/apps/{appid}	GET	text/plain

Argument	Type	Description
appid	string	The app to retrieve information for

Status Codes

- 100 - successful

Get App Info Example

```
# Get app info for the "files" app
curl http://admin:secret@example.com/ocs/v1.php/cloud/apps/files
```

XML Output

```
<?xml version="1.0"?>
<ocs>
 <meta>
  <statuscode>100</statuscode>
  <status>ok</status>
 </meta>
 <data>
  <info/>
  <remote>
   <files>appinfo/remote.php</files>
   <webdav>appinfo/remote.php</webdav>
   <filesync>appinfo/filesync.php</filesync>
  </remote>
  <public/>
  <id>files</id>
  <name>Files</name>
  <description>File Management</description>
  <licence>AGPL</licence>
  <author>Robin Appelman</author>
  <require>4.9</require>
  <shipped>true</shipped>
  <standalone></standalone>
  <default_enable></default_enable>
  <types>
   <element>filesystem</element>
  </types>
 </data>
</ocs>
```

Enable

Enable an app. Authentication is done by sending a Basic HTTP Authorization header.

Request Path	Method	Content Type
ocs/v1.php/cloud/apps/{appid}	POST	text/plain

Argument	Type	Description
appid	string	The id of the app to enable

Status Codes

- 100 - successful

Enable Example

```
# Enable the "files_texteditor" app
curl -X POST
http://admin:secret@example.com/ocs/v1.php/cloud/apps/files_texteditor
```

XML Output

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

Disable

Disables the specified app. Authentication is done by sending a Basic HTTP Authorization header.

Request Path	Method	Content Type
ocs/v1.php/cloud/apps/{appid}	DELETE	text/plain

Argument	Type	Description
appid	string	The id of the app to disable

Status Codes

- 100 - successful

Disable Example

```
Disable the "files_texteditor" app
curl -X DELETE
http://admin:secret@example.com/ocs/v1.php/cloud/apps/files_texteditor
```

XML Output

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

ownCloud Roles

The following information is not an in-depth guide, but more of a high-level overview of each type.

Anonymous

- Is not a regular user.
- Has access to specific content made available via public links.
  - Can be password-protected (optional, enforced, policy-enforced).
  - Can have an expiration date (optional, enforced, enforced dependent on password).
- Has no personal space
- Has no file ownership (ownership of uploaded/created files is directed to sharer).
- Has no use of clients.
- Quota is that of the sharer.
- Permissions are those granted by the sharer for specific content, e.g., *view-only*, *edit*, and *File Drop*.
- Can only use file and viewer apps, such as PDF Viewer and Collabora Online.

Guest

- Is a regular user with restricted permissions, identified via e-mail address.
- Has no personal space.
- Has no file ownership (ownership of uploaded/created files is directed to sharer).
- Has access to shared space. The permissions are granted by the sharer.
- Is not bound to the inviting user.
  - Can log in as long as shares are available.
  - Becomes deactivated when no shares are left; this is the shared with guests filter.
  - Reactivated when a share is received.
  - Administrators will be able to automate user cleanup ("**disabled for x days**").
- Can use all clients.

- Fully auditable in the enterprise edition.
- Can be promoted to group administrator or administrator, but will still have no personal space.
- Apps are specified by the admin (whitelist).



The Shared with Guests Filter

This filter makes it easy for sharers to view and remove their shares with a guest, which also removes their responsibility for guests. When all of a guest's shares are removed, the guest is then disabled and can no longer login.

Standard User

- Is a regular user (from LDAP, ownCloud user backend, or another backend).
- Has personal space. Permissions are granted by the administrator.
- Shared space: Permissions as granted by sharer.
- Apps: All enabled, might be restricted by group membership.

Federated User

- Is not an internal user.
- Can trust a federated system.
- Has access to shared space through users on the considered ownCloud system.
- Can share data with the considered system (accept-/rejectable).

ownCloud Group Administrator

- Is a regular user, such as from LDAP, an ownCloud user backend, or another backend.
- Can manage users in their groups, such as adding and removing them, and changing quota of users in the group.
- Can add new users to their groups and can manage guests.
- Can enable and disable users.
- Can impersonate users in their groups.
- Custom group creation may be restricted to group admins.

ownCloud Administrator

- Is a regular user (from LDAP, ownCloud user backend, or another backend).
- Can configure ownCloud features via the UI, such as sharing settings, app-specific configurations, and external storages for users.
- Can manage users, such as adding and removing, enabling and disabling, quota and group management.
- Can restrict app usage to groups, where applicable.
- Configurable access to log files.
- Mounting of external shares and local shares (of external filesystems) is disabled by default.

System Administrator

- Is not an ownCloud user.

- Has access to ownCloud code (e.g., config.php and apps folders) and command-line tool (occ occ).
- Configures and maintains the ownCloud environment (*PHP*, *Webserver*, *DB*, *Storage*, *Redis*, *Firewall*, *Cron*, and *LDAP*, etc.).
- Maintains ownCloud, such as updates, backups, and installs extensions.
- Can manage users and groups, such as via occ.
- Has access to the master key when storage encryption is used.
- **Storage admin:** Encryption at rest, which prevents the storage administrator from having access to data stored in ownCloud.
- **DB admin:** Calendar/Contacts etc. DB entries not encrypted.

Auditor

- Is not an ownCloud user.
- Conducts usage and compliance audits in enterprise scenarios.
- App logs (especially Auditlog) can be separated from ownCloud log. This separates the Auditor and Sysadmin roles. An audit.log file can be enabled, which the Sysadmin can't access.
- **Best practice:** parse separated log to an external analyzing tool.

Guests (Enterprise only)

Share with external users conveniently just by entering an email address in the sharing dialog. Recipients receive an email containing an activation link. They can log in using their email address as user name and the password they chose during activation. Guests may even use the ownCloud desktop clients and mobile apps to connect to ownCloud and work on shared contents.

Guest users do not have storage space and can only work in contents that are shared with them. Have a look at our informational YouTube video below, for an introduction to the Guests app.

▶ https://www.youtube.com/watch?v=L42PBHgqKVI *(YouTube video)*

Installation

Go to Market and install the Guests app if not already installed with your enterprise bundle. The Guests app requires the email server to be configured in your ownCloud because you need to be able to invite your guests by mail.

Configuration

Check your Guests app's configuration in the sharing section of the admin settings. There you can change the Guest's **group name** and add or exclude apps to the app **whitelist** of the Guests app. Guests can not access apps that are not in that list.

Troubleshooting

If for some reason you don't see some buttons, please try a different browser to exclude the script/adblocking add-on as a cause. If you as a guest user can not open a PDF document for example in your ownCloud, but you can download it - please check the White List of your Guests app in the sharing section of the admin settings. You have to specify that the guest users can access the required app.

OpenID Connect (OIDC)

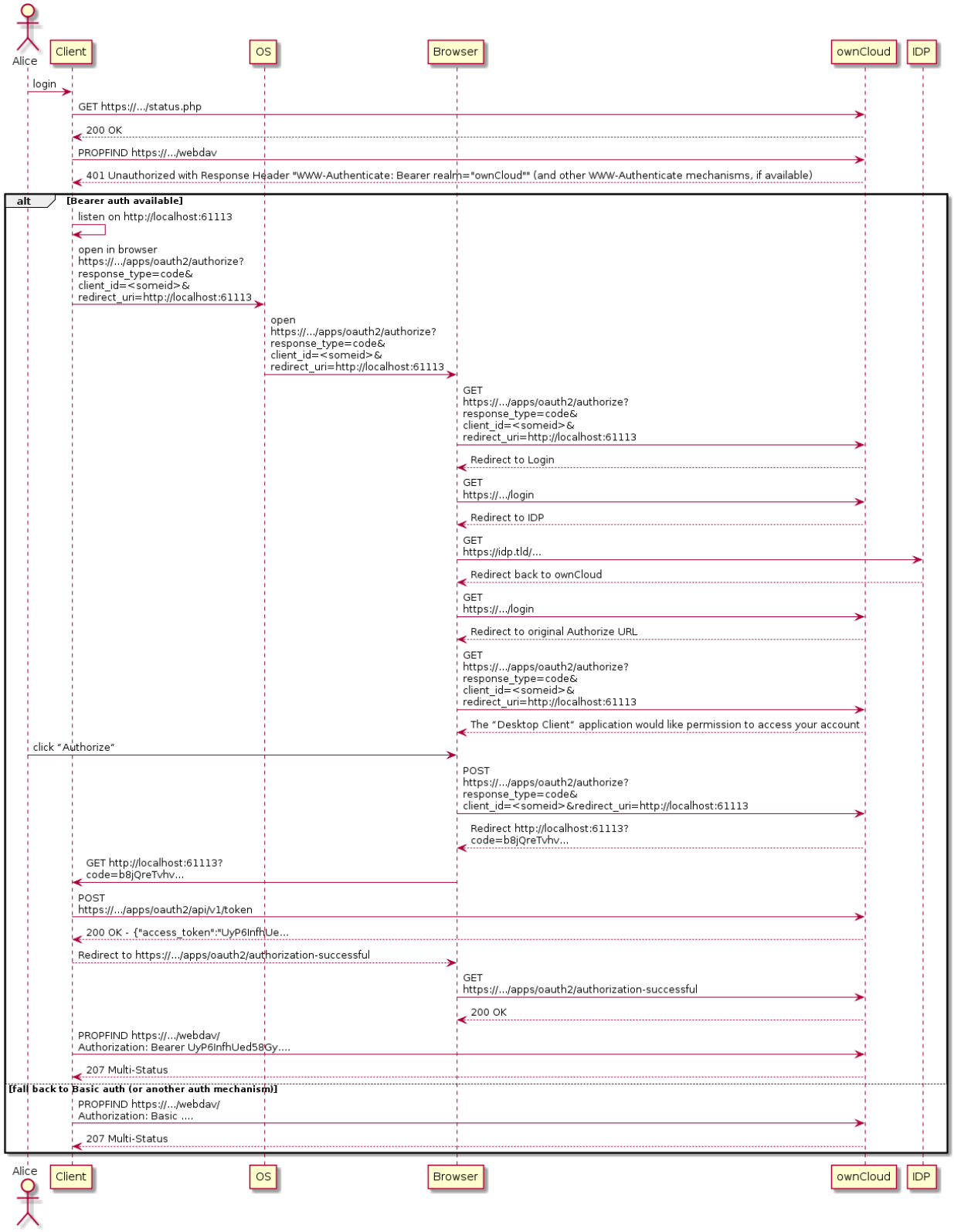
Introduction

> OpenID Connect 1.0 is a simple identity layer on top of the OAuth 2.0 protocol. It allows Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the End-User in an interoperable and REST-like manner. OpenID Connect allows clients of all types, including Web-based, mobile, and JavaScript clients, to request and receive information about authenticated sessions and end-users. The specification suite is extensible, allowing participants to use optional features such as encryption of identity data, discovery of OpenID Providers, and session management, when it makes sense for them.
>
> — https://openid.net/connect/, "What is OpenID Connect?"

Prerequisites

A distributed Memcache setup is required to operate this app - such as Redis or Memcached properly. Please follow the documentation on how to set up caching.

Code Flow



Configure ownCloud

Please refer to the OIDC configuration example in the sample configuration documentation for how to configure ownCloud to work with OIDC.

Setup Within the OpenID Provider

When registering ownCloud as and OpenID Client, use https://cloud.example.net/index.php/apps/openidconnect/redirect as the redirect URL. In case OpenID Connect Front-Channel Logout 1.0 is supported, please enter https://cloud.example.net/index.php/apps/openidconnect/logout as the logout URL within the client registration of the OpenID Provider.

Setup Service Discovery

To allow other clients to use OpenID Connect when talking to ownCloud, please set up a redirect on the webserver to point .well-known/openid-configuration to /index.php/apps/openidconnect/config. Here is an .htaccess example

```
RewriteRule ^\.well-known/openid-configuration
/index.php/apps/openidconnect/config [R=301,L]
```

Maintenance

In this section, you will find all that you need to help you maintain your ownCloud installation.

How to Upgrade Your ownCloud Server

Introduction

We recommend that you keep your ownCloud server up to date. When an update is available for your ownCloud server, you will see a notification at the top of your ownCloud Web interface. When you click the btn:[notification], it will bring you here.

Before beginning an upgrade, please keep the following points in mind:

- Review the release notes for important information about the needed migration steps during that upgrade to help ensure a smooth upgrade process.
- Check ownCloud's mandatory requirements (such as PHP versions and extensions), which can change from one version to the next. Ensure that you review them and update your server(s), if required, before upgrading ownCloud.
- Upgrading is disruptive, as your ownCloud server will be put into maintenance mode.
- Large installations may take several hours to complete the upgrade.
- Review any installed third-party apps for compatibility with the new ownCloud release.
- Downgrading **is not supported** as it risks corrupting your data. If you want to revert to an older ownCloud 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 ownCloud forums to resolve your issue without downgrading.



If required, you can skip major releases when upgrading your ownCloud installation. However, we recommend that you first upgrade to the latest point release of your respective minor version, e.g., *10.2.1*. See Upgrading Across Skipped Releases for more information.

If you are on ownCloud 8.2.11, 9.0.9, 9.1.X, 10.0.X, 10.1.X, 10.2.X, or 10.3.X you can go directly to the latest server version.

Here are some examples:

Version	Can Upgrade to 10.4.0?	Requirements
10.3.2	Yes	
10.2.1	Yes	
10.1.1	Yes	
10.0.10	Yes	
9.1.8	Yes	
9.1.0	Yes	
9.0.9	Yes	
9.0.8	**No**	Must upgrade to 9.0.9 first.
8.2.11	Yes	
8.2.10	**No**	Must upgrade to 8.2.11 first.
7.0.15	**No**	Must upgrade to 8.0.16, then to 8.1.12, and then to 8.2.11 first.
7.0.10	**No**	Must upgrade to 7.0.15, then to 8.0.16, then to 8.1.12, and then to 8.2.11 first.

Prerequisites

We strongly recommend that you always maintain regular backups as well as make a fresh backup before every upgrade. We also recommend that you review any installed third-party apps for compatibility with the new ownCloud release. Ensure that they are all disabled before beginning the upgrade. After the upgrade is complete re-enable any which are compatible with the new release.



Unsupported apps may disrupt your upgrade.

Upgrade Options

There are two ways to upgrade your ownCloud server:

1. **(Recommended)** Perform a manual upgrade, using the latest ownCloud release.
2. **(Discouraged)** Use your distribution's package manager, in conjunction with our official ownCloud repositories. **Note:** This approach should not be used unattended nor in clustered setups. We discourage upgrades with Linux Package Manager because you might encounter unwanted side effects.



Enterprise customers: refer to Installing & Upgrading ownCloud Enterprise Edition for more information.

Manual ownCloud Upgrade

Preparation

This section describes how to manually upgrade your ownCloud installation.

Enable Maintenance Mode

Put your server in maintenance mode and **disable Cron jobs**. Doing so prevents new logins, locks the sessions of logged-in users, and displays a status screen so that users know what is happening.

There are two ways to enable maintenance mode.

1. The **preferred** method is to use the occ command — which you must run as your webserver user.

```
# Enable maintenance mode using the occ command.
{occ-command-example-prefix} maintenance:mode --on
```

2. The other way is by changing the value in your config.php file and replacing 'maintenance' ⇒ false, to 'maintenance' ⇒ true,.



In a clustered environment please check that all nodes are in maintenance mode.

Stop the Webserver

With those steps completed, stop your webserver.

```
# Stop the web server
sudo service apache2 stop
```

Backup Your Existing Installation

First, backup the following items:

1. The complete ownCloud directory
2. The ownCloud server database

```
# This example assumes Ubuntu Linux and MariaDB
# Rename owncloud directory
mv /var/www/owncloud /var/www/owncloud-old-version-number
# Backup the Database
mysqldump -u<username> -p<password> <databasename> > <ownCloud-
Version-Dump.sql>
```

In clustered environment please check that all nodes are in maintenance mode.

Review Third-Party Apps

Review any installed third-party apps for compatibility with the new ownCloud release. Ensure that they are all disabled before beginning the upgrade. Third party apps are all apps that are not distributed by ownCloud or not listed in Supported Apps in

ownCloud.

1. Disable via Command Line

```
# This command lists all apps by <app-id> and app version
{occ-command-example-prefix} app:list

# This command disables the app with the given <app-id>
{occ-command-example-prefix} app:disable <app-id>
```

2. Disable via Browser
   Goto menu:Settings[Admin > Apps] and disable all third-party apps

Download the Latest Installation

Download the latest ownCloud server release where your current installation was. In this example /var/www/

```
cd /var/www/
wget https://download.owncloud.org/community/owncloud-10.2.1.tar.bz2
```



Enterprise users must download their new ownCloud archives from their accounts on https://customer.owncloud.com/owncloud/.

To download the tarball from https://customer.owncloud.com you will need a command like this:

```
wget https://username:password@customer.owncloud.com/link-to-tarball
```

Upgrade



For this description we assume that your existing ownCloud installation is located in the default location: /var/www/owncloud. The path might differ, depending on your installation.

Extract the New Source

Extract the new server release in the location of your original ownCloud installation.

```
tar -xvf owncloud-10.2.1.tar.bz2
```

With the new source files now in place of the old ones, next copy the config.php file from your old ownCloud directory to your new ownCloud directory:

```
sudo cp /var/www/owncloud-old-version-number/config/config.php
/var/www/owncloud/config/config.php
```

If you keep your data/ directory *inside* your owncloud/ directory, move it from your old version of ownCloud to your new version:

```
mv /var/www/owncloud-old-version-number/data /var/www/owncloud/data
```

If you keep your data **outside** of your owncloud directory, 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.

Market and Marketplace App Upgrades

Before getting too far into the upgrade process, please be aware of how the Market app and its configuration options affect the upgrade process.



The Market app — and other apps from the Marketplace — will not be updated when you upgrade ownCloud, if upgrade.automatic-app-update is set to true in config.php.

In addition, if there are installed apps (whether compatible or incompatible with the next version, or missing source code) and the Market app is enabled but there is no available internet connection, these apps will need to be manually updated once the upgrade is finished.

Copy Old Apps

If you are using third party or enterprise applications, look in your new /var/www/owncloud/apps/ directory to see if they are present. If not, copy them from your old apps/ directory to your new one.

Permissions

To finalize the preparation of the upgrade, you need to set the correct ownership of the new ownCloud files and folders.

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

Start the Upgrade

With the apps disabled and ownCloud in maintenance mode, start the upgrade process from the command line:

```
# Here is an example on Ubuntu Linux.
# Execute this within the ownCloud root folder.
{occ-command-example-prefix} upgrade
```

The upgrade operation can take anywhere from a few minutes to a few hours, depending on the size of your installation. When it is finished you will see either a success message, or an error message which indicates why the process did not complete successfully.

Disable Maintenance Mode

Assuming your upgrade succeeded, next disable maintenance mode.

```
# Disable maintenance mode using the occ command.
{occ-command-example-prefix} maintenance:mode --off
```

Restart the Webserver

With all that done, restart your web server:

```
sudo service apache2 start
```

Finalize the Installation

With maintenance mode disabled, login and:

- Check that the version number reflects the new installation.
  It can be reviewed at the bottom of menu:Settings[Admin > General].
- Check that your other settings are correct.
- Go to the menu:Settings[Admin > Apps] page and review the core apps to make sure the right ones are enabled.
- After the upgrade is complete, re-enable any third-party apps that are compatible with the new release.

  1. Enable via Command Line

     ```
     # This command enables the app with the given <app-id>
     {occ-command-example-prefix} app:enable <app-id>
     ```

  2. Enable via Browser

Go to menu:Settings[Admin > Apps > "Show disabled apps"] and enable all compatible third-party apps

+ WARNING: Install or enable unsupported apps at your own risk.

Rollback

If you need to rollback your upgrade, see the Restoring ownCloud documentation.

Troubleshooting

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

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

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

In the unlikely case that files do not show up in the web-ui after the upgrade, use the files:scan command to make them visible again. Here is an example of how to do so:

```
{occ-command-example-prefix} files:scan --all
```

See the owncloud.org support page for further resources for both home and enterprise users.

Sometimes, ownCloud can get *stuck in a upgrade*. This is usually due to the process taking too long and encountering a PHP time-out. Stop the upgrade process this way:

```
{occ-command-example-prefix} maintenance:mode --off
```

Then start the manual process:

```
{occ-command-example-prefix} upgrade
```

If this does not work properly, try the repair function:

```
{occ-command-example-prefix} maintenance:repair
```

Upgrade ownCloud From Packages

Upgrade Quickstart

The alternative to a manual upgrade is configuring your system to use ownCloud's Open Build Service repository. Then stay current by using your Linux package manager to install fresh ownCloud packages. After installing upgraded packages you must run a few more steps to complete the upgrade. These are the basic steps to upgrading ownCloud:

- Disable all third-party apps.
- Make a fresh backup.
- Upgrade your ownCloud packages.
- Run occ upgrade



The optional parameter to skip migration tests was removed in ownCloud 10.0. See Testing a Migration for background information.

- Apply strong permissions to your ownCloud directories.
- Take your ownCloud server out of maintenance mode.
- Re-enable third-party apps.



If required, you can skip major releases when upgrading your ownCloud installation. However, we recommend that you first upgrade to the latest point release of your respective minor version, e.g., *10.2.1*. See Upgrading Across Skipped Releases for more information.

If you are on ownCloud 8.2.11, 9.0.9, 9.1.X, 10.0.X, 10.1.X, 10.2.X, or 10.3.X you can go directly to the latest server version.

Here are some examples:

Version	Can Upgrade to 10.4.0?	Requirements
10.3.2	Yes	
10.2.1	Yes	
10.1.1	Yes	
10.0.10	Yes	
9.1.8	Yes	
9.1.0	Yes	
9.0.9	Yes	
9.0.8	**No**	Must upgrade to 9.0.9 first.
8.2.11	Yes	
8.2.10	**No**	Must upgrade to 8.2.11 first.
7.0.15	**No**	Must upgrade to 8.0.16, then to 8.1.12, and then to 8.2.11 first.
7.0.10	**No**	Must upgrade to 7.0.15, then to 8.0.16, then to 8.1.12, and then to 8.2.11 first.



When upgrading from oC 9.0 to 9.1 with existing Calendars or Address books please have a look at the release notes for important information about the needed migration steps during that upgrade.

Upgrade Tips

Upgrading ownCloud from our Open Build Service repository is just like any normal Linux upgrade. For example, on Debian or Ubuntu Linux this is the standard system upgrade command:

```
apt-get update && apt-get upgrade
```

Or you can upgrade just ownCloud with this command:

```
apt-get update && apt-get install owncloud-files
```

On Fedora, CentOS, and Red Hat Linux use yum to see all available updates:

```
yum check-update
```

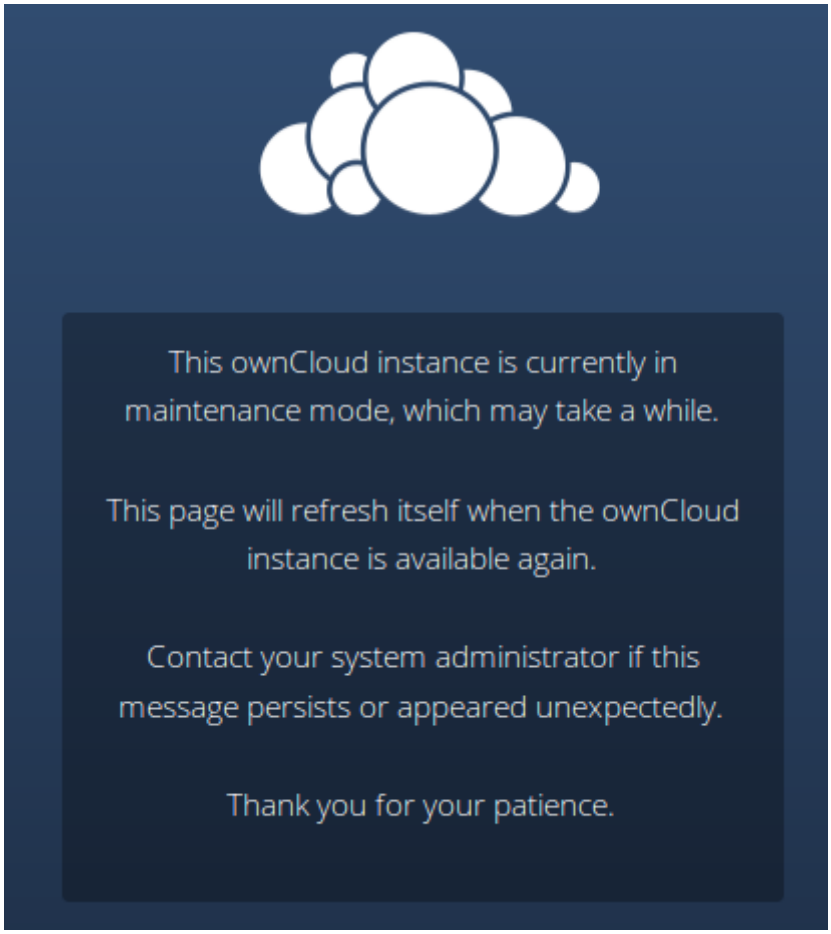
You can apply all available updates with this command:

```
yum update
```

Or update only ownCloud:

```
yum update owncloud-files
```

Your Linux package manager only downloads the current ownCloud packages. Then your ownCloud server is immediately put into maintenance mode. You may not see this until you refresh your ownCloud page.



Then use occ to complete the upgrade. You must run occ as your HTTP user. This example is for Debian/Ubuntu:

```
{occ-command-example-prefix} upgrade
```

This example is for CentOS/RHEL/Fedora:

```
{occ-command-example-prefix} upgrade
```

The optional parameter to skip migration tests during this step was removed in

ownCloud 10.0.



See Testing a Migration for background information, and Using the OCC command to learn more about occ.

Setting Strong Directory Permissions

After upgrading, verify that your ownCloud directory permissions are set accordingly.

Upgrading Across Skipped Releases

It is best to update your ownCloud installation with every new point release (e.g., 8.1.10), and to never skip any major release (e.g., don't skip 8.2.x between 8.1.x and 9.0.x). If you have skipped any major release you can bring your ownCloud current with these steps:

1. Add the repository of your current version (e.g., 8.1.x)
2. Upgrade your current version to the latest point release (e.g., 8.1.10) via your package manager
3. Run the occ upgrade routine (see Upgrade Quickstart above)
4. Add the repository of the next major release (e.g., 8.2.x)
5. Upgrade your current version to the next major release (e.g., 8.2.8) via your package manager
6. Run the occ upgrade routine (see Upgrade Quickstart above)
7. Repeat from step 4 until you reach the last available major release (e.g., 9.1.x)

You'll find repositories of previous ownCloud major releases in the ownCloud Server Changelog.

Upgrading ownCloud with the Updater App

Introduction

The Updater app automates many of the steps of upgrading an ownCloud 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 manual installations.



When upgrading from oC 9.0 to 9.1 with existing Calendars or Adressbooks please have a look at the release notes of oC 9.0 for important info about this migration.



- The Updater app is not enabled and not supported in ownCloud Enterprise edition.
- The Updater app is not included in the Linux packages on our Open Build Service, but only in the tar and zip archives.
- When you install ownCloud from packages you should keep it updated with your package manager.

Downgrading is not supported and risks corrupting your data! If you want to revert to an older ownCloud 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 ownCloud forums to see if your issue can be resolved without downgrading.

We strongly recommend that you make regular backups, as well as a fresh backup before every upgrade.



The Updater app *does not* backup your database or data directory.

We also recommend that you review any installed third-party apps for compatibility with the new ownCloud release. Ensure that they are all disabled before beginning the upgrade. After the upgrade is complete, re-enable any which are compatible with the new release.



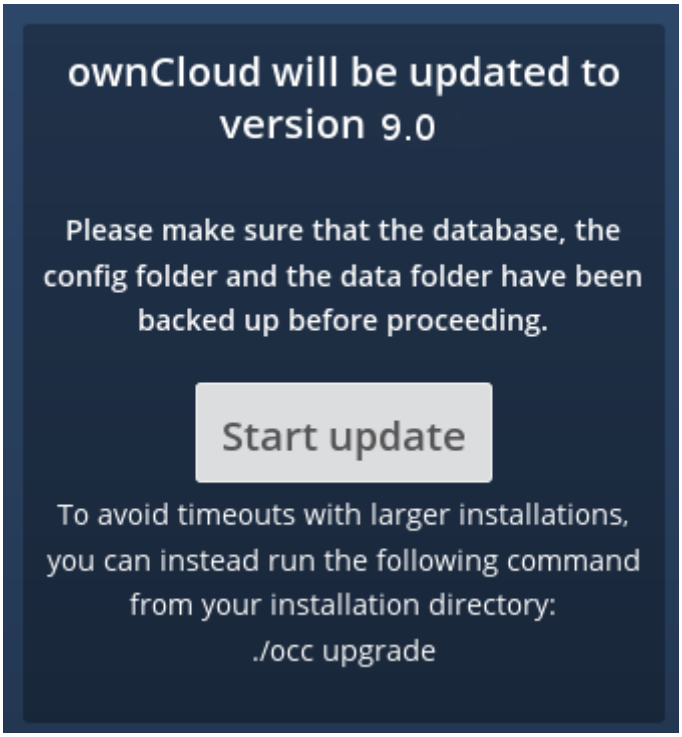
Unsupported apps may disrupt your upgrade.

The Updater app performs these operations:

- Creates an updater_backup directory under your ownCloud data directory
- Downloads and extracts updated package content into the updater_backup/packageVersion directory
- Makes a copy of your current ownCloud instance, except for your data directory, to updater_backup/currentVersion-randomstring
- Moves all directories except data and config from the current instance to updater_backup/tmp
- Moves all directories from updater_backup/packageVersion to the current version
- Copies your old config.php to the new config/ directory

Using the Updater app to update your ownCloud installation is just a few steps:

1. You should see a notification at the top of any ownCloud page when there is a new update available.
2. Even though the Updater app backs up important directories, you should always have your own current backups (See Backing up ownCloud for details.)
3. Verify that the HTTP user on your system can write to your whole ownCloud directory; see the Setting Permissions for Updating section below.
4. Navigate to your Admin page and click the btn:[Update Center] button under Updater. This takes you to the Updater control panel.
5. Click btn:[Update], and carefully read the messages. If there are any problems it will tell you. The most common issue is directory permissions; your HTTP user needs write permissions to your whole ownCloud directory. (See Set Strong Directory Permissions.) Another common issue is SELinux rules (see SELinux Configuration.) Otherwise you will see messages about checking your installation and making backups.
6. Click Proceed, and then it performs the remaining steps, which takes a few minutes.
7. If your directory permissions are correct, a backup was made, and downloading the new ownCloud archive succeeded you will see the following screen. Click the btn:[Start Update] button to complete your update:





If you have a large ownCloud installation and have shell access, you should use the occ upgrade command, running it as your HTTP user, instead of clicking the btn:[Start Update] button, in order to avoid PHP timeouts.

This example is for Ubuntu Linux:

```
{occ-command-example-prefix} upgrade
```

The optional parameter to skip migration tests during this step was removed in ownCloud 10.0. See Testing a Migration for more information.

1. It runs for a few minutes, and when it is finished displays a success message, which disappears after a short time.

Refresh your Admin page to verify your new version number. In the Updater section of your Admin page you can see the current status and backups. These are backups of your old and new ownCloud installations, and do not contain your data files. If your update works and there are no problems you can delete the backups from this screen.

If the update fails, then you must update manually.

Setting Permissions for Updating

For hardened security, we highly recommend setting the permissions on your ownCloud directory as strictly as possible, immediately after the initial installation. However, these strict permissions will prevent the Updater app from working, as it needs your whole ownCloud directory to be owned by the HTTP user.

So to set the appropriate permissions for updating, run the code below. Replace the ocpath variable with the path to your ownCloud directory, and replace the htuser and htgroup variables with your HTTP user and group.

```
#!/bin/bash
# Sets permissions of the owncloud instance for updating

ocpath='/var/www/owncloud'
htuser='www-data'
htgroup='www-data'

chown -R ${htuser}:${htgroup} ${ocpath}
```

You can find your HTTP user in your HTTP server configuration files. Or you can use PHP Version and Information. Look for the User/Group line.

- 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.

After the update is completed, re-apply the strong directory permissions immediately.

Command Line Options

The Updater app includes command-line options to automate updates, to create checkpoints and to roll back to older checkpoints. You must run it as your HTTP user. This example on Ubuntu Linux displays command options:

```
{occ-command-example-prefix} updater/application.php list
```

See usage for commands, like this example for the upgrade:checkpoint command:

```
{occ-command-example-prefix} updater/application.php upgrade:checkpoint -h
```

You can display a help summary:

```
{occ-command-example-prefix} updater/application.php --help
```

When you run it without options it runs a system check:

```
{occ-command-example-prefix} owncloud/updater/application.php
ownCloud updater 1.0 - CLI based ownCloud server upgrades
Checking system health.
- file permissions are ok.
Current version is 9.0.0.12
No updates found online.
Done
```

Create a checkpoint:

```
{occ-command-example-prefix} updater/application.php upgrade:checkpoint
--create
Created checkpoint 9.0.0.12-56d5e4e004964
```

List checkpoints:

```
{occ-command-example-prefix} updater/application.php upgrade:checkpoint --list
[source,console]
```

Restore an earlier checkpoint:

```
{occ-command-example-prefix} updater/application.php \
   upgrade:checkpoint --restore=9.0.0.12-56d5e4e004964
```

Add a line like this to your crontab to automatically create daily checkpoints:

```
2 15 * * * {occ-command-example-prefix}
/path/to/owncloud/updater/application.php
upgrade:checkpoint --create > /dev/null 2>&1
```

updater.secret value in config.php

When running the updater, you will be prompted to add a hashed secret into your config.php file. On the updater web interface, you then need to enter the unhashed secret into the web form.

In case you forgot your password/secret, you can re-create it by changing config.php. You can run this on your shell:

```
php -r 'echo password_hash("Enter a random password here",
PASSWORD_DEFAULT)."\n";'
```

Please replace Enter a random password here with your own. Then add this into your config.php:

```
'updater.secret' => 'The value you got from the above hash command',
```

Upgrade PHP on RedHat 7 and CentOS 7

Introduction

You should, almost, always upgrade to the latest version of PHP supported by ownCloud, if and where possible. And if you're on a version of PHP older than {minimum-php-version} you **must** upgrade. This guide steps you through upgrading your installation of PHP to one of the supported PHP versions ({supported-php-versions}) on Red Hat or CentOS 7.

Upgrade PHP to Version 7.1

To upgrade to PHP 7.1 you first need to subscribe to the Red Hat Software Collections channel repository to download and install the PHP 7.1 package in RHEL 7 (if you've not done this already). This documentation uses the same command as you will find there.



Ensure that you have subscription-manager installed. If you don't, yet, have it installed, do so with the following command:

```
# Install subscription manager
yum install --assumeyes subscription-manager

# Add the required repositories for the PHP packages
subscription-manager repos --enable rhel-server-rhscl-7-rpms
```

Install the Required Packages

Then, proceed by installing the required PHP 7.1 packages. You can use the command below to save you time.

```
yum install \
  rh-php71 \
  rh-php71-php \
  rh-php71-php-cli \
  rh-php71-php-curl \
  rh-php71-php-devel \
  rh-php71-php-gd \
  rh-php71-php-intl \
  rh-php71-php-ldap \
  rh-php71-php-mbstring \
  rh-php71-php-mysqlnd \
  rh-php71-php-opcache
  rh-php71-php-pdo \
  rh-php71-php-pear \
  rh-php71-php-xml \
  rh-php71-php-xmlrpc \
  rh-php71-php-zip
```

Enable PHP 7.1 and Disable PHP 5.6

Next, you need to enable PHP 7.1 and disable PHP 5.6 system-wide. To enable PHP 7.1 system-wide, run the following command:

```
cp /opt/rh/rh-php71/enable /etc/profile.d/rh-php71.sh source /opt/rh/rh-
php71/enable
```

Then, you need to disable loading of the PHP 5.6 Apache modules. You can do this either by changing their names, as in the example below, or deleting the files.

```
mv /etc/httpd/conf.d/php.conf /etc/httpd/conf.d/php56.off
mv /etc/httpd/conf.modules.d/10-php.conf /etc/httpd/conf.modules.d/10-php56.off
```

Update the Apache Configuration Files

With that done, you next need to copy the PHP 7.1 Apache modules into place; that being the two Apache configuration files and the shared object file.

```
cp /opt/rh/httpd24/root/etc/httpd/conf.d/rh-php71-php.conf /etc/httpd/conf.d/
cp /opt/rh/httpd24/root/etc/httpd/conf.modules.d/15-rh-php71-php.conf
/etc/httpd/conf.modules.d/
cp /opt/rh/httpd24/root/etc/httpd/modules/librh-php71-php7.so /etc/httpd/modules/
```

Upgrade PHP to Version 7.2

To upgrade to PHP 7.2 you first need to subscribe to the Red Hat Software Collections channel repository to download and install the PHP 7.2 package in RHEL 7 (if you've not done this already). This documentation uses the same command as you will find there.



Ensure that you have subscription-manager installed. If you don't, yet, have it installed, do so with the following command:

```
# Install subscription manager
yum install --assumeyes subscription-manager

# Add the required repositories for the PHP packages
subscription-manager repos --enable rhel-server-rhscl-7-rpms
```

Install the Required Packages

Then, proceed by installing the required PHP 7.2 packages. You can use the command below to save you time.

```
yum install \
  rh-php72 \
  rh-php72-php \
  rh-php72-php-cli \
  rh-php72-php-curl \
  rh-php72-php-devel \
  rh-php72-php-gd \
  rh-php72-php-intl \
  rh-php72-php-ldap \
  rh-php72-php-mbstring \
  rh-php72-php-mysqlnd \
  rh-php72-php-opcache
  rh-php72-php-pdo \
  rh-php72-php-pear \
  rh-php72-php-xml \
  rh-php72-php-xmlrpc \
  rh-php72-php-zip
```

Enable PHP 7.2 and Disable PHP 5.6

Next, you need to enable PHP 7.2 and disable PHP 5.6 system-wide. To enable PHP 7.2 system-wide, run the following command:

```
cp /opt/rh/rh-php72/enable /etc/profile.d/rh-php72.sh source /opt/rh/rh-
php72/enable
```

Then, you need to disable loading of the PHP 5.6 Apache modules. You can do this either by changing their names, as in the example below, or deleting the files.

```
mv /etc/httpd/conf.d/php.conf /etc/httpd/conf.d/php56.off
mv /etc/httpd/conf.modules.d/10-php.conf /etc/httpd/conf.modules.d/10-php56.off
```

Update the Apache Configuration Files

With that done, you next need to copy the PHP 7.2 Apache modules into place; that being the two Apache configuration files and the shared object file.

```
cp /opt/rh/httpd24/root/etc/httpd/conf.d/rh-php72-php.conf /etc/httpd/conf.d/
cp /opt/rh/httpd24/root/etc/httpd/conf.modules.d/15-rh-php72-php.conf
/etc/httpd/conf.modules.d/
cp /opt/rh/httpd24/root/etc/httpd/modules/librh-php72-php7.so /etc/httpd/modules/
```

Upgrade PHP to Version 7.3

To upgrade to PHP 7.3 you first need to subscribe to the Red Hat Software Collections channel repository to download and install the PHP 7.3 package in RHEL 7 (if you've not done this already). This documentation uses the same command as you will find there.



Ensure that you have subscription-manager installed. If you don't, yet, have it installed, do so with the following command:

```
# Install subscription manager
yum install --assumeyes subscription-manager

# Add the required repositories for the PHP packages
subscription-manager repos --enable rhel-server-rhscl-7-rpms
```

Install the Required Packages

Then, proceed by installing the required PHP 7.3 packages. You can use the command below to save you time.

```
yum install \
  rh-php73 \
  rh-php73-php \
  rh-php73-php-cli \
  rh-php73-php-curl \
  rh-php73-php-devel \
  rh-php73-php-gd \
  rh-php73-php-intl \
  rh-php73-php-ldap \
  rh-php73-php-mbstring \
  rh-php73-php-mysqlnd \
  rh-php73-php-opcache
  rh-php73-php-pdo \
  rh-php73-php-pear \
  rh-php73-php-xml \
  rh-php73-php-xmlrpc \
  rh-php73-php-zip
```

Enable PHP 7.3 and Disable PHP 5.6

Next, you need to enable PHP 7.3 and disable PHP 5.6 system-wide. To enable PHP 7.3 system-wide, run the following command:

```
cp /opt/rh/rh-php73/enable /etc/profile.d/rh-php73.sh source /opt/rh/rh-
php73/enable
```

Then, you need to disable loading of the PHP 5.6 Apache modules. You can do this either by changing their names, as in the example below, or deleting the files.

```
mv /etc/httpd/conf.d/php.conf /etc/httpd/conf.d/php56.off
mv /etc/httpd/conf.modules.d/10-php.conf /etc/httpd/conf.modules.d/10-php56.off
```

Update the Apache Configuration Files

With that done, you next need to copy the PHP 7.3 Apache modules into place; that being the two Apache configuration files and the shared object file.

```
cp /opt/rh/httpd24/root/etc/httpd/conf.d/rh-php73-php.conf /etc/httpd/conf.d/
cp /opt/rh/httpd24/root/etc/httpd/conf.modules.d/15-rh-php73-php.conf
/etc/httpd/conf.modules.d/
cp /opt/rh/httpd24/root/etc/httpd/modules/librh-php73-php7.so /etc/httpd/modules/
```

Restart Apache

Finally, you need to restart Apache to make the changes permanent, as in the command below.

```
service httpd restart
```

Upgrade Marketplace Applications

Introduction

To upgrade Marketplace applications, please refer to the documentation below, as applicable for your ownCloud setup.

Single-Server Environment

To upgrade Marketplace applications when running ownCloud in a single server environment, you can use use the Market app, specifically by running market:upgrade. This will install new versions of your installed apps if updates are available in the marketplace.



The user running the update command, which will likely be your webserver user, needs write permission for the /apps folder. If they don't have write permission, the command may report that the update was successful, however it may silently fail.

Clustered / Multi-Server Environment

The Market app, both the UI and command line, are not, *currently*, designed to operate on clustered installations. Given that, you will have to update the applications on each server in the cluster individually. There are several ways to do this. But here is a concise approach:

1. Download the latest server release (whether the tarball or the zip archive).
2. Download your installed apps from the ownCloud marketplace.
3. Combine them together into one installation source, such as *a Docker or VM image*, or *an Ansible script*, etc.
4. Apply the combined upgrade across all the cluster nodes in your ownCloud setup.

Backing up ownCloud

When you backup your ownCloud server, there are four things that you need to copy:

1. Your config/ directory.
2. Your data/ directory.

3. Your ownCloud database.
4. Your custom theme files, if you have any. (See Theming ownCloud)

When you install your ownCloud server from our Open Build Service packages (or from distro packages, which we do not recommend) **do not backup your ownCloud server files**, which are the other files in your owncloud/ directory such as core/, 3rdparty/, apps/, lib/, and all the rest of the ownCloud files. If you restore these files from backup they may not be in sync with the current package versions, and will fail the code integrity check. This may also cause other errors, such as white pages.

When you install ownCloud from the source tarballs this will not be an issue, and you can safely backup your entire ownCloud installation, with the exception of your ownCloud database. Databases cannot be copied, but you must use the database tools to make a correct database dump.

To restore your ownCloud installation from backup, see Restoring ownCloud.

Backing Up the config/ and data/ Directories

Simply copy your config/ and data/ folder to a place outside of your ownCloud environment. This example uses rsync to copy the two directories to /oc-backupdir:

```
rsync -Aax config data /oc-backupdir/
```

There are many ways to backup normal files, and you may use whatever method you are accustomed to.

Backup Database

You can't just copy a database, but must use the database tools to make a correct database dump.

MySQL/MariaDB

MySQL or MariaDB, which is a drop-in MySQL replacement, is the recommended database engine. To backup MySQL/MariaDB:

```
mysqldump --single-transaction -h [server] -u [username] -p [password] [db_name]
> owncloud-dbbackup_`date +"%Y%m%d"`.bak
```

Example:

```
mysqldump --single-transaction -h localhost -u username -p password owncloud >
owncloud-dbbackup_`date +"%Y%m%d"`.bak
```

SQLite

```
sqlite3 data/owncloud.db .dump > owncloud-dbbackup_`date +"%Y%m%d"`.bak
```

PostgreSQL

```
PGPASSWORD="password" pg_dump [db_name] -h [server] -U [username] -f
owncloud-dbbackup_`date +"%Y%m%d"`.bak
```

Restoring Files From Backup When Encryption Is Enabled

If you need to restore files from backup, which were backed up when encryption was enabled, here's how to do it.



This is **not officially supported**. ownCloud officially supports either restoring the full backup or restoring nothing — not restoring individual parts of it.

- Restore the file from backup.
- Restore the file's encryption keys from your backup.
- Run occ files:scan, which makes the scanner find it.



In the DB it will:

- Have the "size" set to the encrypted size, which is wrong (and bigger);
- The "encrypted" flag will be set to 0

- Retrieve the encrypted flag value
- Update the encrypted flag.



There's no need to update the encrypted flag for files in either files_versions or files_trashbin, because these aren't scanned or found by occ files:scan.

- Download the file once as the user; the file's size will be corrected automatically.

This process might not be suitable across all environments. If it's not suitable for yours, you might need to run an OCC command that does the scanning.

Retrieve the Encrypted Flag Value

1. In the backup database, retrieve the numeric_id value for the storage where the file was located from the oc_storages table and store the value for later reference. For example, if you have the following in your oc_storages table, then numeric_id you should use is 3, if you need to restore a file for user1.

```
+------------------------------+------------+-----------+--------------+
| id                           | numeric_id | available | last_checked |
+------------------------------+------------+-----------+--------------+
| home::admin                  |          1 |         1 |         NULL |
| local::/var/www/owncloud/data/ |          2 |         1 |         NULL |
| home::user1                  |          3 |         1 |         NULL |
+------------------------------+------------+-----------+--------------+
```

2. In the live database instance, find the fileid of the file to restore by running the query below, substituting the placeholders for the retrieved values, and store the value for later reference.

```
SELECT fileid
FROM oc_filecache
WHERE path = 'path/to/the/file/to/restore'
  AND storage = <numeric_id>
```

3. Retrieve the backup, which includes the data folder and database.
4. Retrieve the required file from your backup and copy it to the real instance.
5. In the backup database, retrieve the file's encrypted value, by running the query below and store the value for later reference. The example query assumes the storage was the same and the file was in the same location. If not, you will need to track down where the file was before.

```
SELECT encrypted
FROM oc_filecache
WHERE path = 'path/to/the/file/to/restore'
  AND storage = <numeric_id>
```

6. Update the live database instance with retrieved information, by running the following query, substituting the placeholders for the retrieved values:

```
UPDATE oc_filecache
  SET encrypted = <encrypted>
  WHERE fileid = <fileid>.
```

Maintenance Mode Configuration

You must put your ownCloud server into maintenance mode before performing upgrades, and for performing troubleshooting and maintenance. Please see Using the occ Command to learn how to put your server into the various maintenance modes (maintenance:mode, maintenance:singleuser, and maintenance:repair) with the occ command.

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:

```
{occ-command-example-prefix} maintenance:mode --on
```

You may also put your server into this mode by editing config/config.php. Change "maintenance" ⇒ false to "maintenance" ⇒ true:

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

Then change it back to false when you are finished.

Data Exporter

Important Information



This app is currently in beta stage, the functionality is officially not supported.
Please file any issues here.



The app is not available on the marketplace.
To use this app, you must git clone it from the data_exporter repository and run make all in the apps root directory to install all dependencies.

Description

A set of occ command line tools to export and import users with their shares from one ownCloud instance in to another. Please see what is exported for export details and known limitations for limitation details. Please see the Data Exporter Commands description for details using the occ commands.



To use data exporter, you must install and enable the data exporter app on both, the source and the target instance first.

Use Cases

- Manual zero-downtime migration of users and their shares from one instance in to another.
- Migrate from instances with different storages (POSIX to S3).
- Service GDPR-Requests by providing all files and metadata of a user in a single package.
- Merge users from different instances.

Usage Example

Export user1 from a source instance to a target instance while preserving all shares with users on the source instance. For this example, both instances must be able to reach each other via federation.



Test if you can create remote-shares before starting this process.

Export the User on the Source Instance

This will create a folder /tmp/export/user1 which contains all the files and metadata of the user.

```
{occ-command-example-prefix} instance:export:user user1 /tmp/export
```

Copy the Export to the Target Instance

Copy the created export to the target instance, for example using scp:

```
scp -rp /tmp/export root@newinstance.com:/tmp/export
```

Import the User on the Target Instance

This imports the user in to the target instance while converting all his outgoing-shares to federated shares pointing to the source instance:

```
{occ-command-example-prefix} instance:import:user /tmp/export/user1
```

Recreate all Shares to Point to the Target Instance

user1 now lives on a target instance, therefore it is necessary to recreate all shares so that they point to the target instance. To do so run this command on the source instance:

```
{occ-command-example-prefix} instance:export:migrate:share user1
https://newinstance.com
```

Delete the User on the Source Instance

Finally delete user1 on the source instance:



This can not be undone!



If the user is stored in the ownCloud database, you need to manually reset his password on the target instance. See known limitations for further information.

```
{occ-command-example-prefix} user:delete user1
```

What is Exported

- Files (Local)
- Meta-data (Username, Email, Personal Settings)
- Shares (Local, Link-shares, Group-Shares)
- Versions

Known Limitations

- External storages, comments and tags are not exported
- If a user is stored in the ownCloud database (not-LDAP etc.) the password must be manually reset by the admin as passwords can not be migrated.
- Versions import in to S3 does not preserve the version timestamp.
- Import alias (import using another username) currently does not work and breaks share-import.
- Shares import requires federation to be correctly setup between both servers and share-api to be enabled.
- A share’s state will be always "accepted" regardless of the state in the old server.
- Remote shares from both directions need to be manually accepted.
- Federated shares from other servers are not migrated.
- Password protected link-shares are not imported correctly, user needs to reset the password.
- Group shares require the group to be present on the target-system or else the share will be ignored silently.
- If link-shares require a password on the new server but do so on the old the import

process will crash.

As this is an early version some limitations might be fixed in the future while others can not be circumvented.

How To Manually Move a Data Directory

Introduction

If you need to move your ownCloud data directory from its current location to another location — **without** using a symbolic link — this section steps through how to do so.

Assumptions

This guide assumes that:

- The current folder is: /var/www/owncloud/data
- The new folder is: /mnt/owncloud/data
- You're using Apache as your webserver
- The ownCloud database name is owncloud

Please change the paths above to reflect your environment.

Description of Steps

The following steps are necessary to move the data directory.

1. Stop Apache
2. Enable maintenance mode for your instance
3. Use Rsync to sync the files from the current to the new directory
4. Double-check the directory permissions on the new directory
5. Change the ownCloud configuration to point to the new data directory
6. Disable maintenance mode for your instance
7. Restart Apache

Look at each section below for a detailed description.

Apache and Rsync

To save time, here are the commands which you can copy/paste for Apache and rsync:

```
sudo service apache2 stop

sudo service apache2 start

sudo rsync -avz /var/www/owncloud/data /mnt/owncloud
```



Check your commands for how to start or stop your webserver if you are not on Ubuntu/Debian.

Enable and Disable Maintenance Mode

It is necessary to enable maintenance mode to avoid running cron jobs. To enable maintenance mode, run the following command.

```
{occ-command-example-prefix} maintenance:mode --on
```

To disable maintenance mode of your instance run the following command:

```
{occ-command-example-prefix} maintenance:mode --off
```

Fix Hard-coded Database Path Variables

Open a database command line client to enter database commands and activate your ownCloud database.

```
use owncloud;
```

Update the oc_storages Table

Run the SQL below:

```
UPDATE oc_storages
  SET id='local::/mnt/owncloud/data/'
  WHERE id='local::/var/www/owncloud/data/';
```

Update the oc_accounts Table

You next need to update the home column in the oc_accounts table. This column contains the absolute path for user folders, e.g., /mnt/owncloud/data/my_user/files.

If a user does not have the path already set, you have to identify the users id and set the path with the following command, user by user. This example assumes the user name is my_user and their id is 1.

Run the SQL below:

```
UPDATE oc_accounts SET home='/mnt/owncloud/data/my_user/files'
  WHERE id=1;
```

For all users who already have a path like /var/www/owncloud/data/ in your database, you can use the REPLACE command:

```
UPDATE oc_accounts
  SET home = REPLACE(
    home,
    '/var/www/owncloud/data/',
    '/mnt/owncloud/data/'
  );
```

For more information follow the complete MySQL REPLACE command syntax.



Please don't copy and paste this example verbatim — nor any of the others. They are examples only.

Update the oc_jobs Table

The next area to check is the oc_jobs table. The logrotate process may have hard-coded a non-standard (or old) value for the data path. To check it, run the SQL below and see if any results are returned:

```
SELECT * FROM oc_jobs
 WHERE class = 'OC\Log\Rotate';
```

If results are returned, run the SQL below to update them, changing the id value as appropriate.

```
UPDATE oc_jobs
 SET argument = REPLACE(
  argument,
  '\\/var\\/www\\/owncloud\\/data\\/',
  '\\/mnt\\/owncloud/data\\/'
 )
 WHERE id = <id of the incorrect record>;
```



The old data path will be written with \/. Therefore you must add one, additional, backslash, like this: \\/.

Fix the Application Settings

Individual apps may reference the data directory separately from the core system configuration. For those apps, you have to change the configured path. Run the following command to list app configs.

```
{occ-command-example-prefix} config:list
```

Here is an example of the output which you may see:

```
{
   "apps": {
      "fictitious": {
         "enabled": "yes",
         "installed_version": "2.3.2",
         "types": "filesystem",
         "datadir": "/var/www/owncloud/data"
      }
   }
}
```

In the example above, the app "fictitious" sets the data directory to /var/www/owncloud/data. Change this value by using the following command:

```
{occ-command-example-prefix} config:app:set --value /mnt/owncloud/data fictitious datadir
```



You have to repeat this for all apps found defining the data directory as key.

Fix the config.php Settings

To fix the config.php settings:

1. Change the datadirectory key in your config.php to the new path. To do so, start an editor of your choice and open /var/www/owncloud/config/config.php
2. Change the value of the key from 'datadirectory' ⇒ '/var/www/owncloud/data', to 'datadirectory' ⇒ '/mnt/owncloud/data',.

Encryption

In this section you will find all the details you need to maintain encryption in ownCloud.

Migrating User Key Encryption to Master Key Encryption

Introduction

Why should you move away from User Key-based encryption?

While it is a bit more secure than a central encryption approach, User key-based encryption has some disadvantages. It blocks some additional functions such as the integration of an online editor like LibreOffice or OnlyOffice into ownCloud and can cause problems when sharing files with groups. See Limitations of User-Key Based Encryption for more details. Therefore Master-key-based encryption is now the recommended setup for all new installations.



User key-based encryption is planned to be removed from ownCloud in the near future. As an existing customer, you will be able to continue to use this solution as long as ownCloud 10.x is supported.

Pre-Conditions

The decryption workflow described here will only work with the following pre-conditions:

- The admin recovery key password is activated and available to the ownCloud administrator
- Users have opted-in to enable the admin recovery key password
- The recovery key password has been supplied by the admin on the users page

Please see How To Enable Users File Recovery Keys for more details.



A notification to the users (e.g. through the announcement app) prior to the migration process is recommended, as the instance will not be available during this task.

Steps to Migrate from User Key-based to Master Key-based Encryption

There are several steps you need to follow to ensure a smooth and complete transition:

1. Disable User Key-based encryption
2. Remove the encryption records from the ownCloud database
3. Remove the files_encryption directory
4. Encrypt the filesystem using Master Key-based encryption

Disable User Key-based Encryption

The first part of the migration process is to decrypt all files and to disable encryption in ownCloud, which requires three commands to be executed. These commands are:

1. occ encryption:decrypt-all,
2. occ encryption:disable and
3. occ app:disable.

You can see an example of calling the commands listed below, configured to require no user interaction.

```
{occ-command-example-prefix} encryption:decrypt-all --continue=yes && \
 {occ-command-example-prefix} encryption:disable --no-interaction && \
 {occ-command-example-prefix} app:disable --no-interaction encryption
```



The decryption of the files by the ownCloud administrator requires the current passwords of all users! This only works when users have enabled password recovery and if an admin recovery password is available.

Remove the Encryption Records from the ownCloud Database

Once your ownCloud files are unencrypted, and encryption has been disabled, you need to remove the encryption records from the database. There is, currently, no occ command to handle this, so it has to be done manually. Specifically, you need to remove all records from the oc_appconfig table where the appid column is set to encryption.

In the examples below, you can see how to do this using MySQL. If you are not using MySQL, please use the commands specific to your database vendor.

```
SELECT * FROM `oc_appconfig` WHERE `appid` LIKE 'encryption'
```

Remove the files_encryption Directory

With the database updated, next, the files_encryption directory needs to be removed. Below is an example of how to do so, to save you time.

```
cd <your owncloud root directory>
find ./data* -name files_encryption -exec rm -rvf {} \;
```

Encrypt the Filesystem Using Master Key-based Encryption

Now, your ownCloud files can be encrypted using Master Key-based encryption. This requires the following steps:

1. The encryption app needs to be enabled
2. Encryption needs to be enabled
3. The encryption type needs to be set to Master Key
4. Re-encryption of the ownCloud filesystem.

The following example shows how to do this on the command line.

```
{occ-command-example-prefix} app:enable encryption && \
 {occ-command-example-prefix} encryption:enable && \
 {occ-command-example-prefix} encryption:select-encryption-type masterkey -y
&& \
 {occ-command-example-prefix} encryption:encrypt-all --yes
```

Verify the Encrypted Files

With the files encrypted using Master Key-based encryption, you should now verify that everything worked properly. To do so, run a SELECT query in your database which returns all files from the oc_appconfig table where the appid column is set to encryption. You should see a number of records, as in the output of the example below.

```
select * from `oc_appconfig` where appid='encryption';
encryption|recoveryKeyId|recoveryKey_73facda6
encryption|publicShareKeyId|pubShare_73facda6
encryption|masterKeyId|master_73facda6
encryption|installed_version|1.3.1
encryption|types|filesystem
encryption|enabled|yes
encryption|useMasterKey|1
```

Disable Single User Mode

With encryption migrated from User Key-based encryption to Master Key-based, disable single user mode, if you enabled it before beginning the migration.

```
{occ-command-example-prefix} maintenance:singleuser --off
```

Post Note



It is possible, that after migration, some or all users see a re-synchronisation of their data from the server to the desktop client - especially for shared folders.

Migrating to a Different Server

Introduction

If the need arises, ownCloud can be migrated to a different server. A typical use case would be a hardware change or a migration from the Enterprise appliance to a physical server. All migrations have to be performed with ownCloud in maintenance mode. Online migration is supported by ownCloud only when implementing industry-standard clustering and high-availability solutions **before** ownCloud is installed for the first time.

To start, let's work through a potential use case. A configured ownCloud instance runs reliably on one machine, but for some reason the instance needs to be moved to a new machine. Depending on the size of the ownCloud instance the migration might take several hours.

For the purpose of this use case, it is assumed that:

1. The end users reach the ownCloud instance via a virtual hostname (such as a DNS CNAME record) which can be pointed at the new location.
2. The authentication method (e.g., LDAP) remains the same after the migration.



During the migration, do not make any changes to the original system, except for putting it into maintenance mode. This ensures, should anything unforeseen happen, that you can go back to your existing installation and resume availability of your installation while debugging the problem.

How to Migrate

Firstly, set up the new machine with your desired Linux distribution. At this point you can either install ownCloud manually via the compressed archive, or with your Linux package manager.

Then, on the original machine turn on maintenance mode and then stop ownCloud. After waiting 6 - 7 minutes for all sync clients to register that the server is in maintenance mode, stop the web server that is serving ownCloud.

After that, create a database dump from the database, copy it to the new machine, and import it into the new database. Then, copy only your data, configuration, and database files from your original ownCloud instance to the new machine.



You must keep the data/ directory's original file path during the migration. However, you can change it before you begin the migration, or after the migration's completed.

The data files should keep their original timestamp otherwise the clients will re-download all the files after the migration. This step might take several hours, depending on your installation. This can be done on a number of sync clients, such as by using rsync with -t option

With ownCloud still in maintenance mode and before changing the DNS CNAME record, start up the database and web server on the new machine. Then point your web browser to the migrated ownCloud instance and confirm that:

1. You see the maintenance mode notice
2. That a log file entry is written by both the web server and ownCloud
3. That no error messages occur.

If all of these things occur, then take ownCloud out of maintenance mode and repeat.

After doing this, log in as an admin and confirm that ownCloud functions as normal.

At this point, change the DNS CNAME entry to point your users to the new location. And with the CNAME entry updated, you now need to update the trusted domains.

Managing Trusted Domains

All URLs used to access your ownCloud server must be white-listed in your config.php file, under the trusted_domains setting. Users are allowed to log into ownCloud only when they point their browsers to a URL that is listed in the trusted_domains setting.



This setting is important when changing or moving to a new domain name. You may use IP addresses and domain names.

A typical configuration looks like this:

```
'trusted_domains' => [
  0 => 'localhost',
  1 => 'server1.example.com',
  2 => '192.168.1.50',
],
```

The loopback address, 127.0.0.1, is automatically white-listed, so as long as you have access to the physical server you can always log in. In the event that a load-balancer is in place, there will be no issues as long as it sends the correct X-Forwarded-Host header.

Example Migration

The following is an example migration with assumptions to make this migration work:

- Ubuntu 16.04+
- SSH with PermitRootLogin set to yes
- Database used is MySQL / MariaDB

Preparation

If not already available on the new server, make sure SSH is installed:

```
sudo apt install ssh -y
```

Next, edit ssh-config and enable root ssh login.

```
nano /etc/ssh/sshd_config
PermitRootLogin yes
```

And then restart SSH.

```
sudo service ssh restart
```

Lastly, install ownCloud on the new server.

Enable Maintenance Mode

The first step is to enable maintenance mode. To do that, use the following commands:

```
cd /var/www/owncloud/
{occ-command-example-prefix} maintenance:mode --on
```

After that's done, then wait a few minutes and stop your web server, in this case Apache:

```
sudo service apache2 stop
```

Transfer the Database

Now, you have to transfer the database from the old server to the new one. To do that, first backup the database.

```
cd /var/www/owncloud/
mysqldump --single-transaction -h localhost \
   -u admin -ppassword owncloud > owncloud-dbbackup.bak
```

Then, export the database to the new server.

```
rsync -Aaxt owncloud-dbbackup.bak root@new_server_address:/var/www/owncloud
```

With that completed, import the database on new server.

```
mysql -h localhost -u admin -ppassword owncloud < owncloud-dbbackup.bak
```



You can find the values for the mysqldump command in your config.php, in your owncloud root directory. [server]= dbhost, [username]= dbuser, [password]= dbpassword, and [db_name]= dbname.



For InnoDB tables only

The –single-transaction flag will start a transaction before running. Rather than lock the entire database, this will let mysqldump read the database in the current state at the time of the transaction, making for a consistent data dump.



For Mixed MyISAM / InnoDB tables

Either dumping your MyISAM tables separately from InnoDB tables or use –lock-tables instead of –single-transaction to guarantee the database is in a consistent state when using mysqldump.

Transfer Data and Configure the New Server

The following ownCloud directories will be synced to the target instance: apps, config and data.

```
rsync -Aavxt apps config data root@new_server_address:/var/www/owncloud
```



If you have an additional apps directory like apps-external, this directory needs to be added to the sync list above.



If you want to move your data directory to another location on the target server, it is advised to do this as a second step. Please see the data directory migration document for more details.

Finish the Migration

Now it's time to finish the migration. To do that, on the new server, first verify that ownCloud is in maintenance mode.

```
{occ-command-example-prefix} maintenance:mode
```

Next, start up the database and web server on the new machine.

```
sudo service mysql start
sudo service apache2 start
```

With that done, point your web browser to the migrated ownCloud instance, and confirm that you see the maintenance mode notice, and that no error messages occur. If both of these occur, take ownCloud out of maintenance mode.

```
{occ-command-example-prefix} maintenance:mode --off
```

And finally, log in as admin and confirm normal function of ownCloud. If you have a domain name, and you want an SSL certificate, we recommend certbot.

Reverse the Changes to ssh-config

Now you need to reverse the change to ssh-config. Specifically, set PermitRootLogin to no and restart ssh. To do that, run the following command:



This is a security measure and improves SSH security.

```
sudo service ssh restart
```

Update DNS and Trusted Domains

Finally, update the DNS' CNAME entry to point to your new server. If you have not only migrated physically from server to server but have also changed your ownCloud server's domain name, you also need to update the domain in the Trusted Domain setting in config.php, on the target server.

Restoring ownCloud

Introduction

When you install ownCloud from packages, follow these steps to restore your ownCloud installation. Start with a fresh ownCloud package installation in a new, empty directory. Then restore these items from your backup (see backup):

1. Your config/ directory.
2. Your data/ directory.
3. Your ownCloud database.
4. Your custom theme files, if you have any. (See Theming ownCloud)

When you install ownCloud from the source tarballs you may safely restore your entire ownCloud installation from backup, with the exception of your ownCloud database. Databases cannot be copied, but you must use the database tools to make a correct restoration.

When you have completed your restoration, see Set Strong Directory Permissions.

Restore Directories

Simply copy your configuration and data folder to your ownCloud environment. You could use this command, which restores the backup example in backup:

```
sudo rsync -Aax config data /var/www/owncloud/
```

There are many ways to restore normal files from backups, and you may use whatever method you are accustomed to.

Restore Database



This guide assumes that your previous backup is called owncloud-dbbackup.bak.

MySQL

MySQL is the recommended database engine. To restore MySQL:

```
{occ-command-example-prefix} maintenance:mode --on
sudo mysql -h [server] -u [username] -p[password] [db_name] < owncloud-
dbbackup.bak
{occ-command-example-prefix} maintenance:data-fingerprint
{occ-command-example-prefix} maintenance:mode --off
```

SQLite

```
sudo rm data/owncloud.db
sudo sqlite3 data/owncloud.db < owncloud-dbbackup.bak
```

```
PGPASSWORD="password" pg_restore -c -d owncloud -h [server] -U [username]
owncloud-dbbackup.bak
```

What is the Appliance?

If you don't know a lot about Linux, only have a small IT staff, or are your IT staff — even if that's only in your spare time — the ownCloud X Appliance will let you get started using ownCloud quickly and easily.

The Appliance:

- Provides a pre-packaged, easy to deploy ownCloud, ready for you in most popular virtual machine formats, including *ESX*, *VirtualBox*, *KVM* and *VMware*.
- Contains the ownCloud 10 virtual image, and all the additional software you need to get up and running on ownCloud in minutes; this includes: *ownCloud X Server and Enterprise Apps*, *Apache 2*, *PHP*, and *MySQL*.
- Scales up to 500 users. Depending on the intensity and pattern of use, this can vary from 400 up to 600 users.

Some configurations, such as SAML IDPs, or LDAP or AD instances, may need additional configuration to connect.

How to Install the Appliance

Introduction

The installation process is a little involved, but not too much. To keep it succinct, you need to:

- Download and Launch the appliance
- Step through the configuration wizard
- Activate the configured appliance
- After that, you can access the running instance of ownCloud and further configure it to suit your needs.



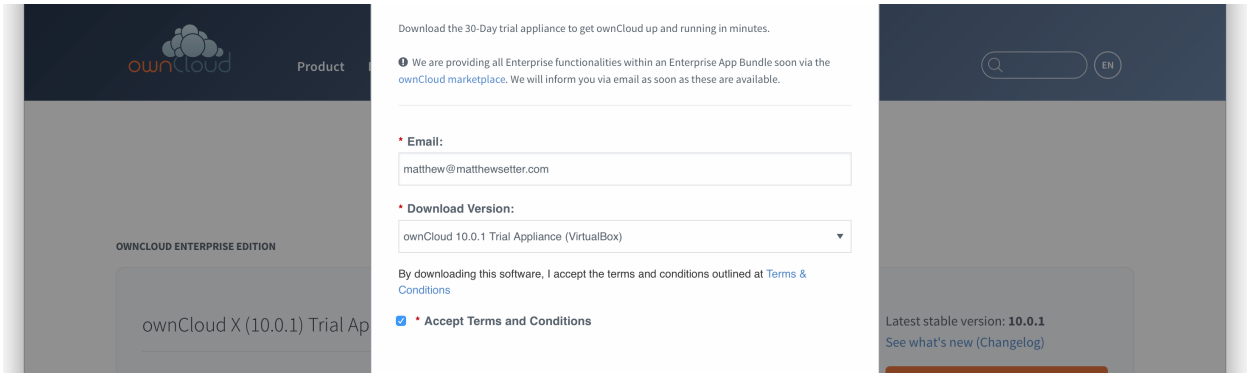
It's recommended to setup the appliance with a working DHCP Server and access to the internet.

The appliance has to be activated with a license that you will receive from Univention via email. This license has to be imported into the appliance via the **web interface**.

Download the Appliance

First off, you need to download the ownCloud X Appliance from the ownCloud download page and click btn:[DOWNLOAD NOW]. This will display a form, which you can see a sample of below, which you'll need to fill out. It will ask you for the following details:

- Email address
- Download version (*ESXi*, *VirtualBox*, *VMware*, *KVM*)
- Your first, last, and company names, and your country of origin



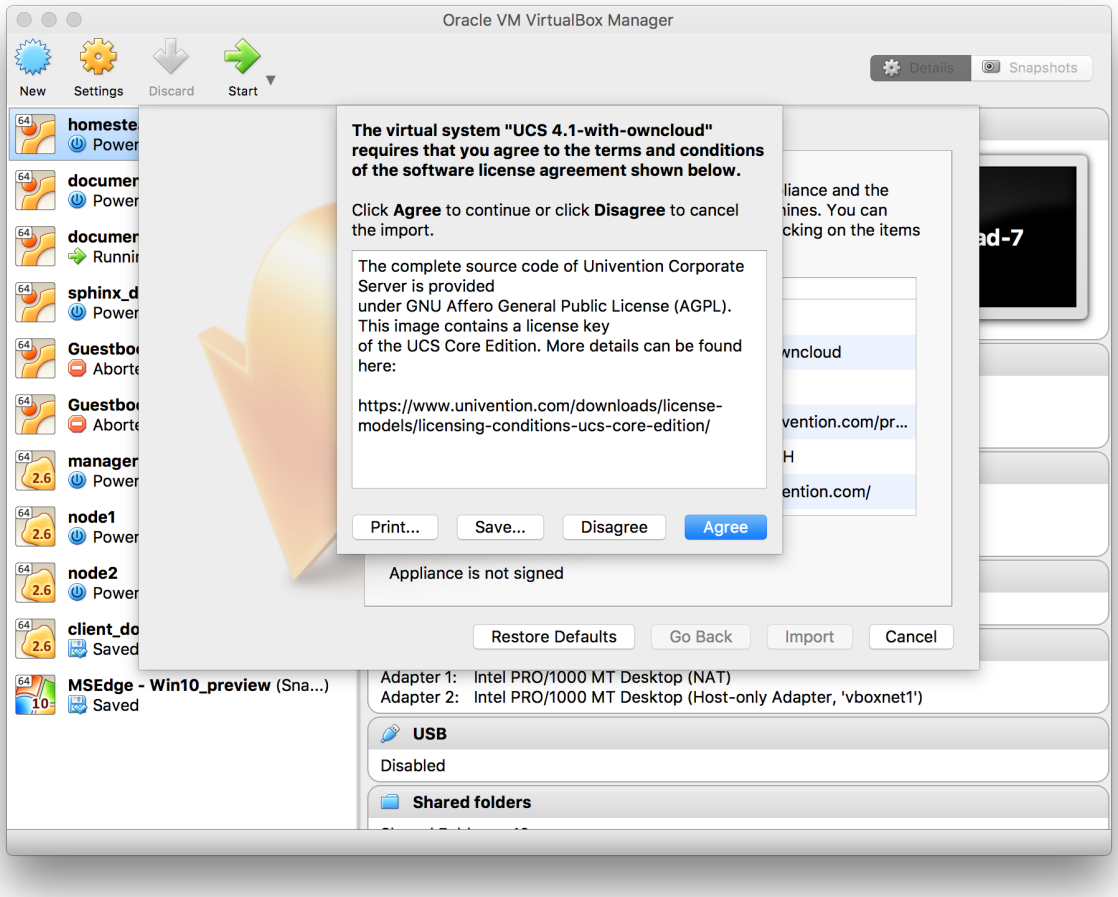
After you've filled out the form, click btn:[DOWNLOAD OWNCLOUD] to begin the download of the virtual appliance.

The virtual appliance files are around 1.4GB in size, so may take some time, depending on your network bandwidth.

You can also download it from the owncloud.org page.

Launch the Appliance

Once you've downloaded the virtual appliance file, import it into your virtualization software, accept the T's & C's of the license agreement, and launch it. The example below shows this being done using VirtualBox.



If you try to install an ownCloud appliance in your domain after removing an existing one, please remember to remove the original one from you DNS configuration.

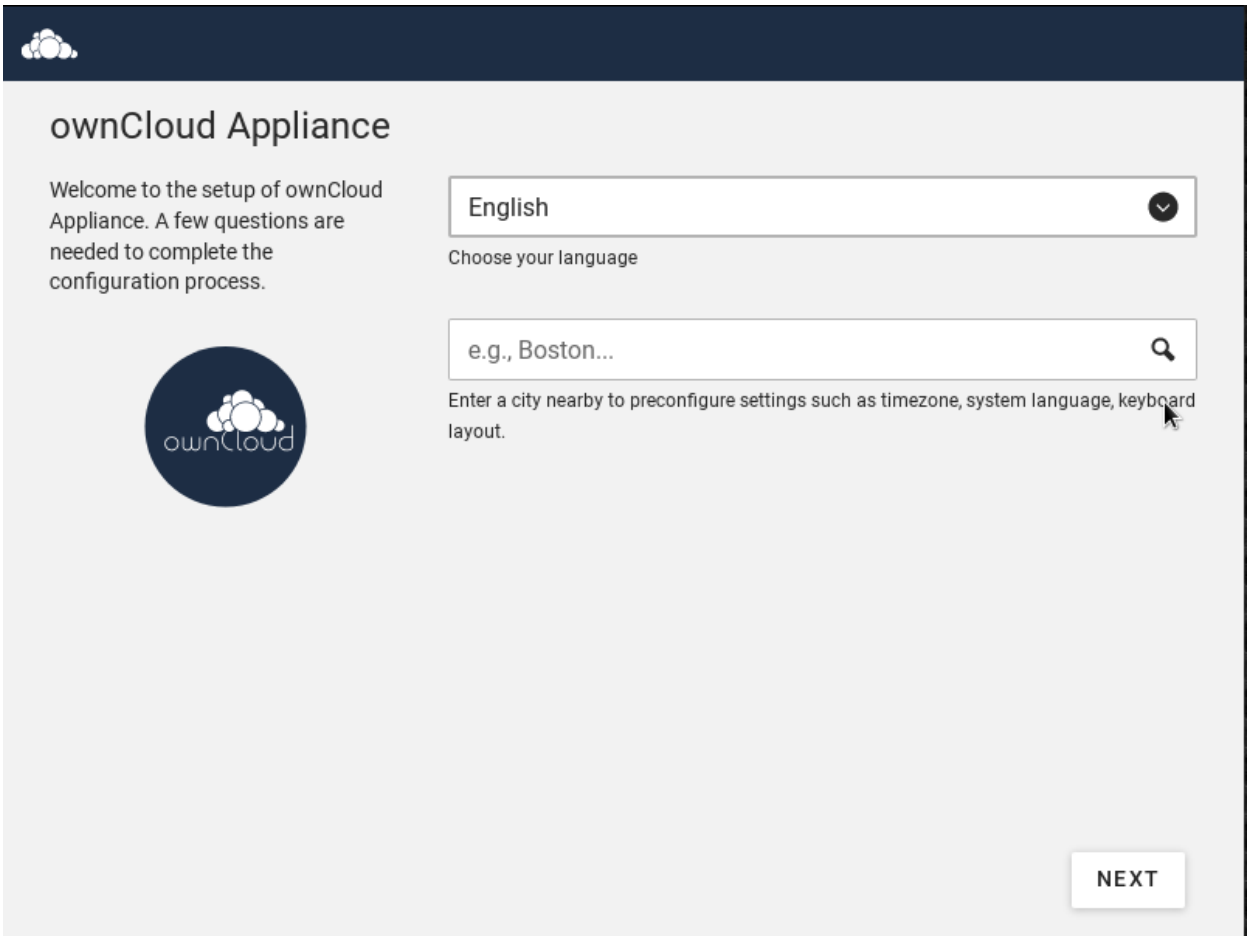
Don't Forget the **IP Address** and the **Administrator Password**. You will need them to

use the Appliance.

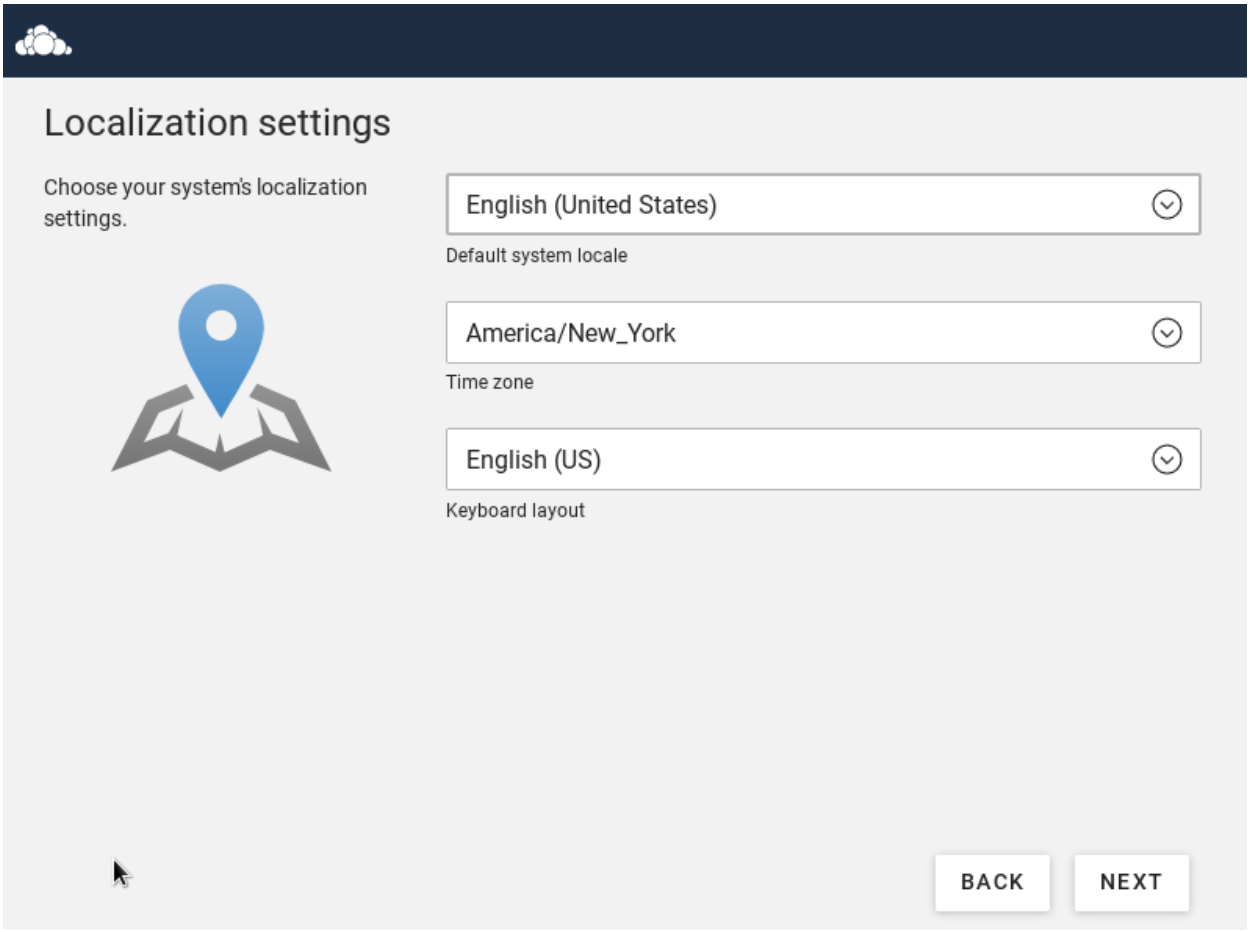
Configuration wizard

Once imported, start the appliance. Doing so launches the installer wizard which helps you specify the core configuration.

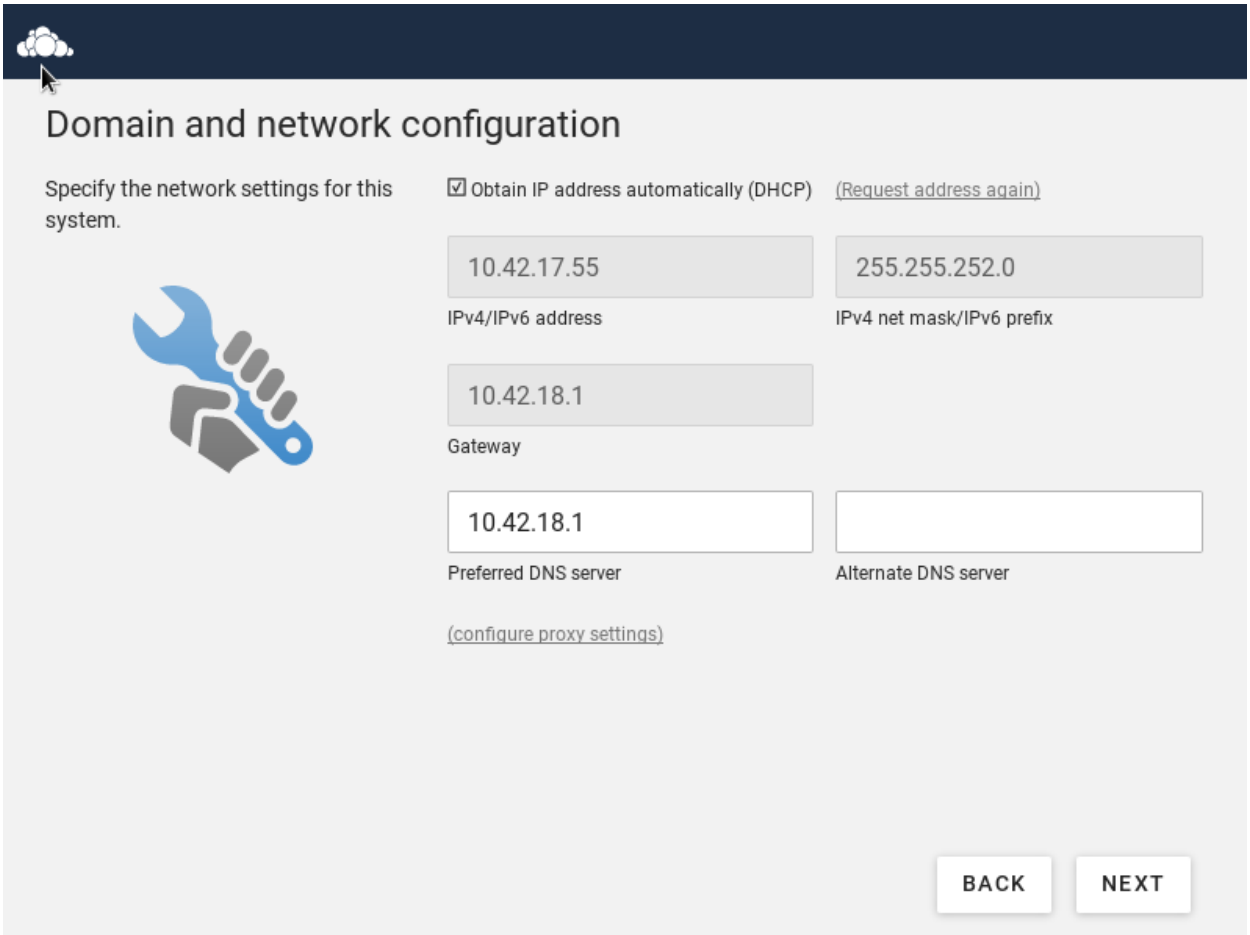
Follow this screenshot guide to securely and easily configure your appliance.



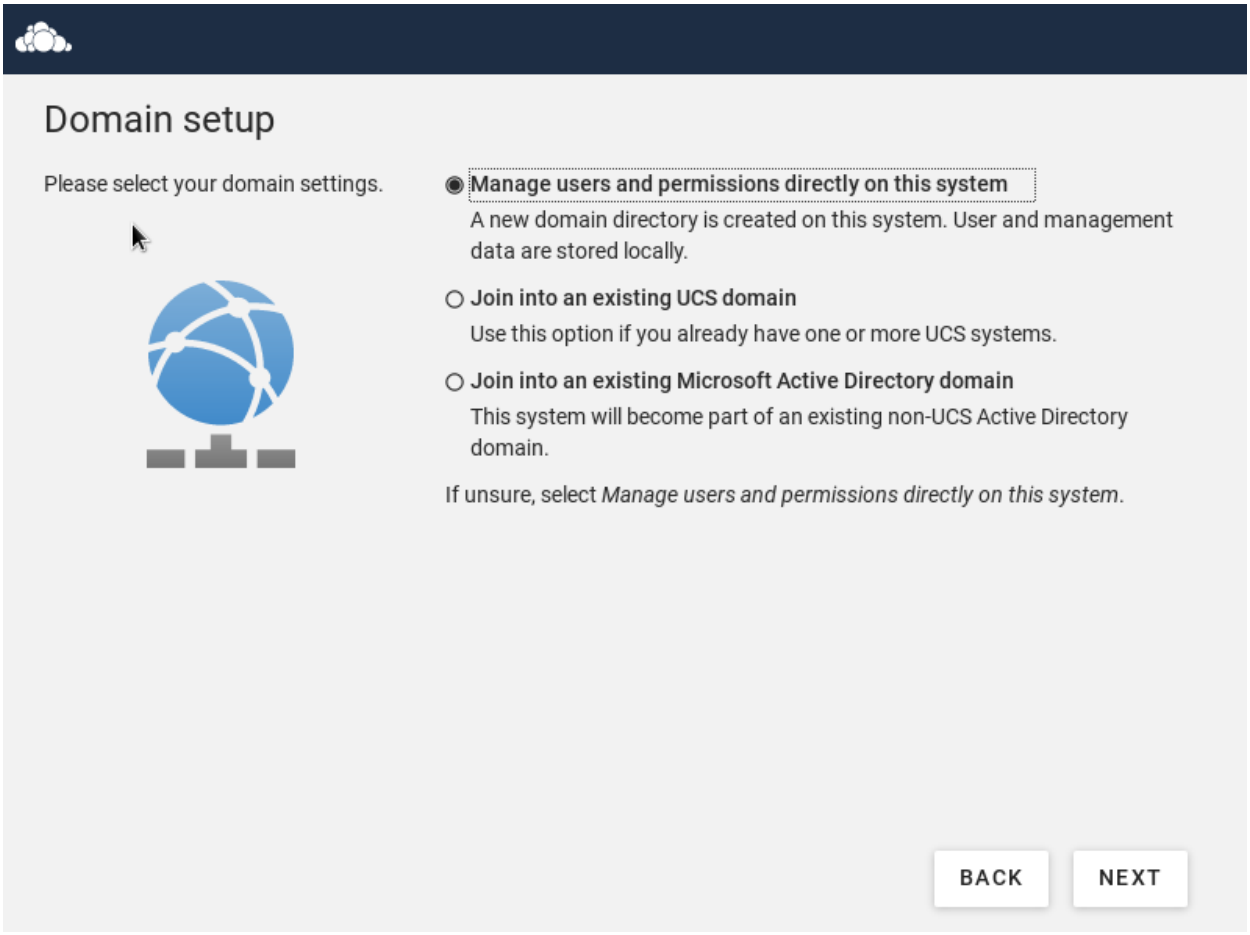
Here, you can choose your **language**. Currently there are 2 options: English and German. You can set your city, which will then automatically set the localization settings in the next screen.



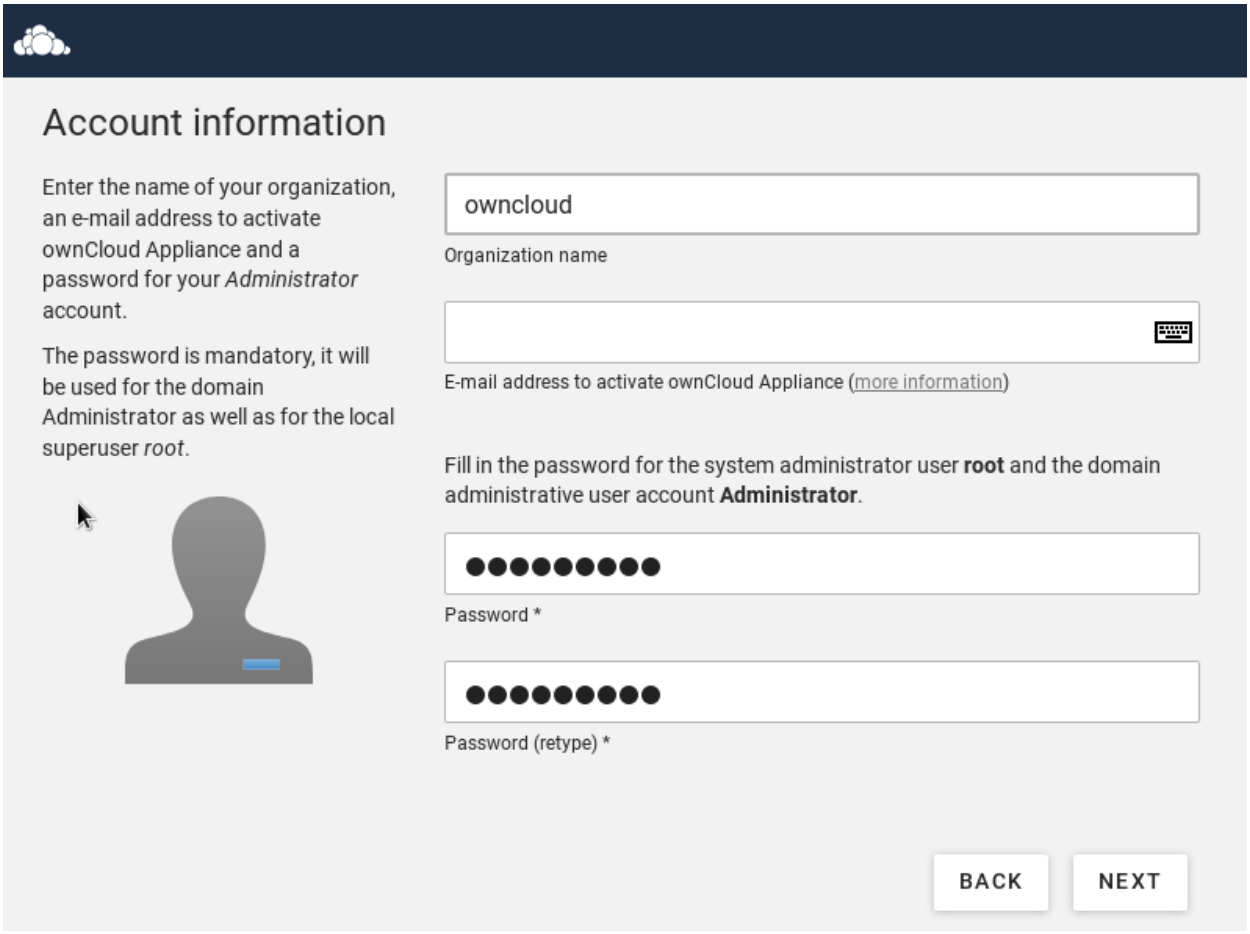
Here, you can set your default **language**, **time zone** and **keyboard layout**. This will be set automatically if you enter your City in the previous screen.



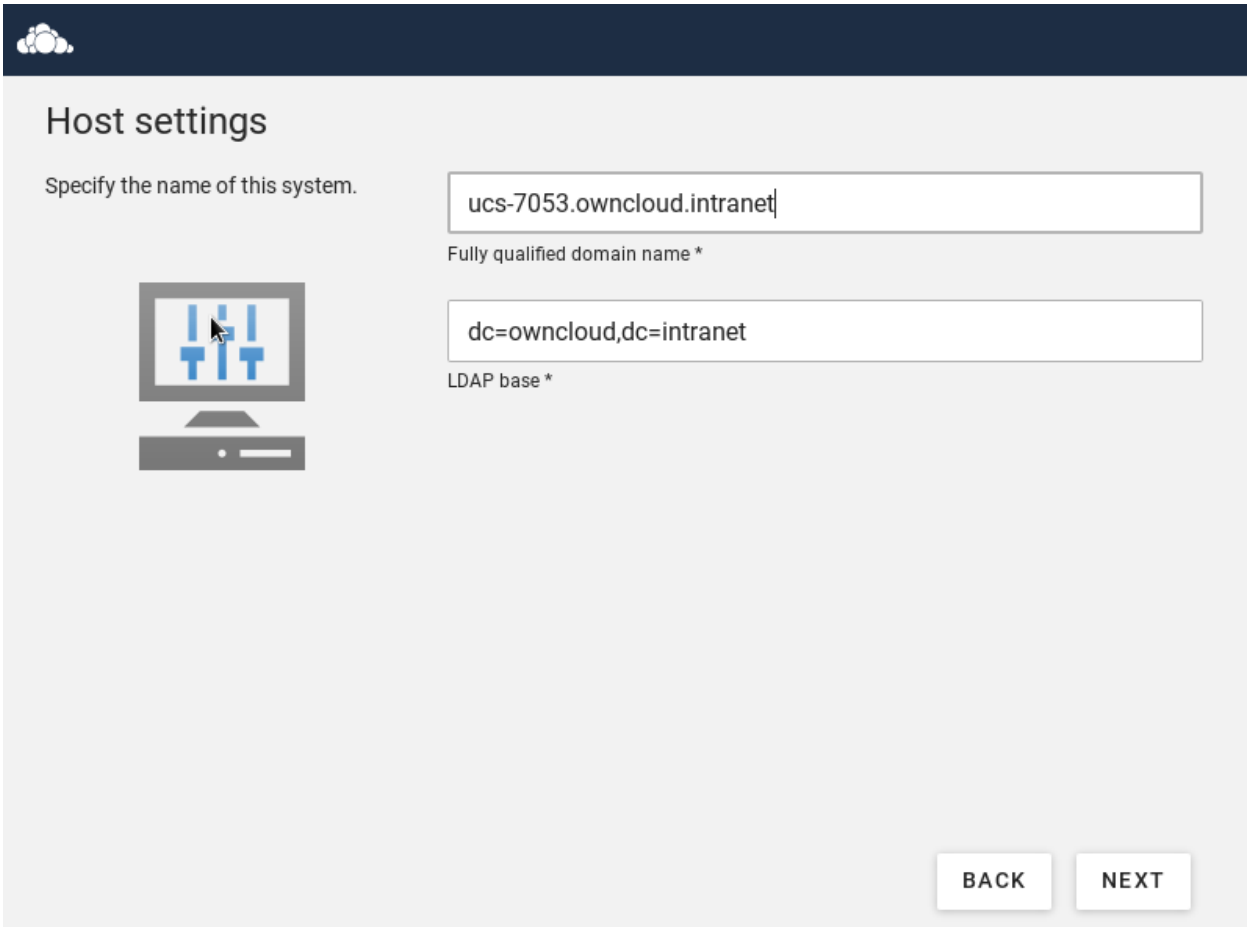
Here, you will see the automatically obtained **network configuration** if you have a DHCP server in your network. If not - you will have to set this yourself. You can also enter a alternate DNS server if you need one.



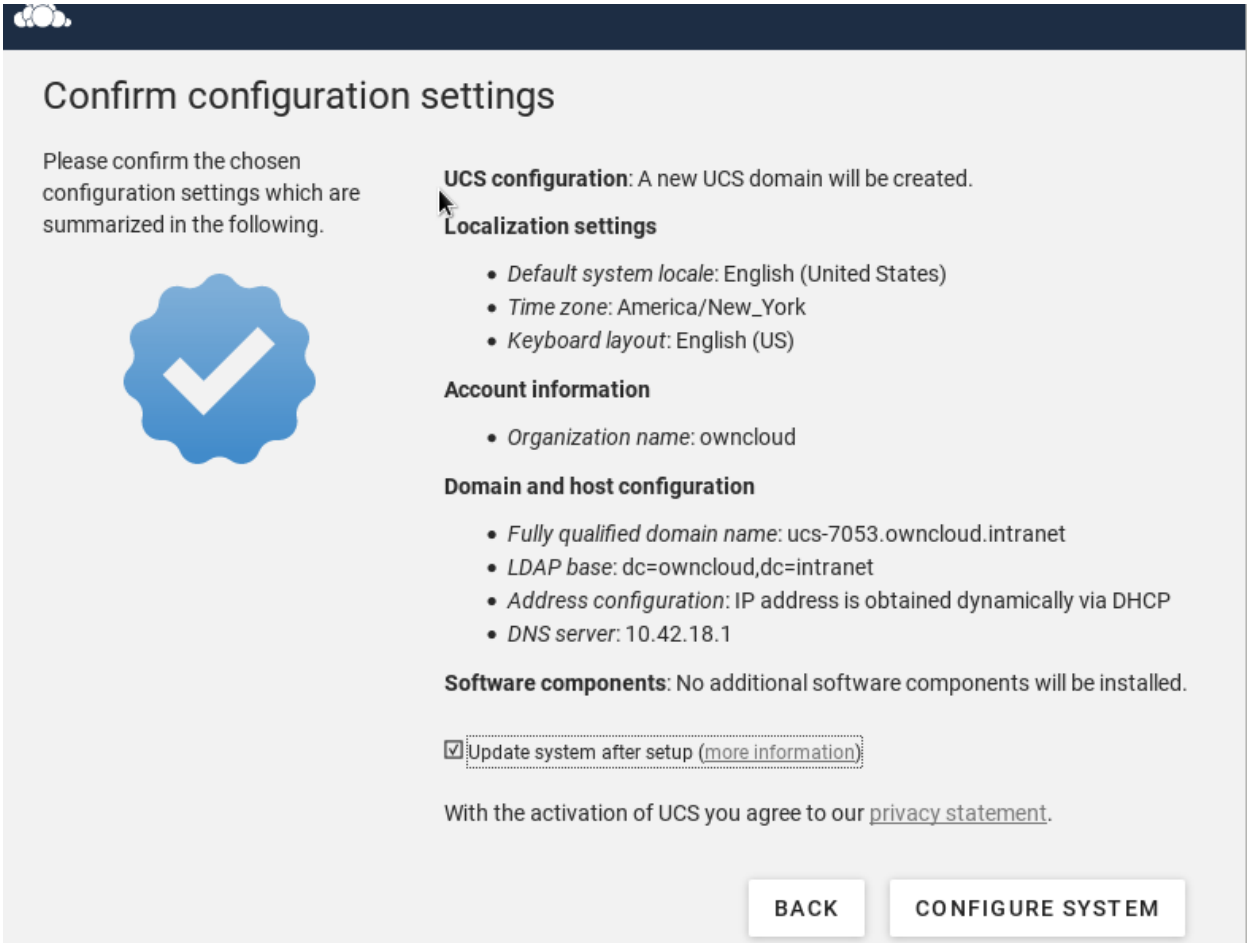
This is an important setting. **Choose the default option** if you don’t have deep knowledge about Microsoft Active directory and the univention system.



The second important setting during this setup: the **Administrator password**. You will need this to log in to your appliance and administer it. Please **write this password down**. Setting your email address here is optional, since you can set it later on.

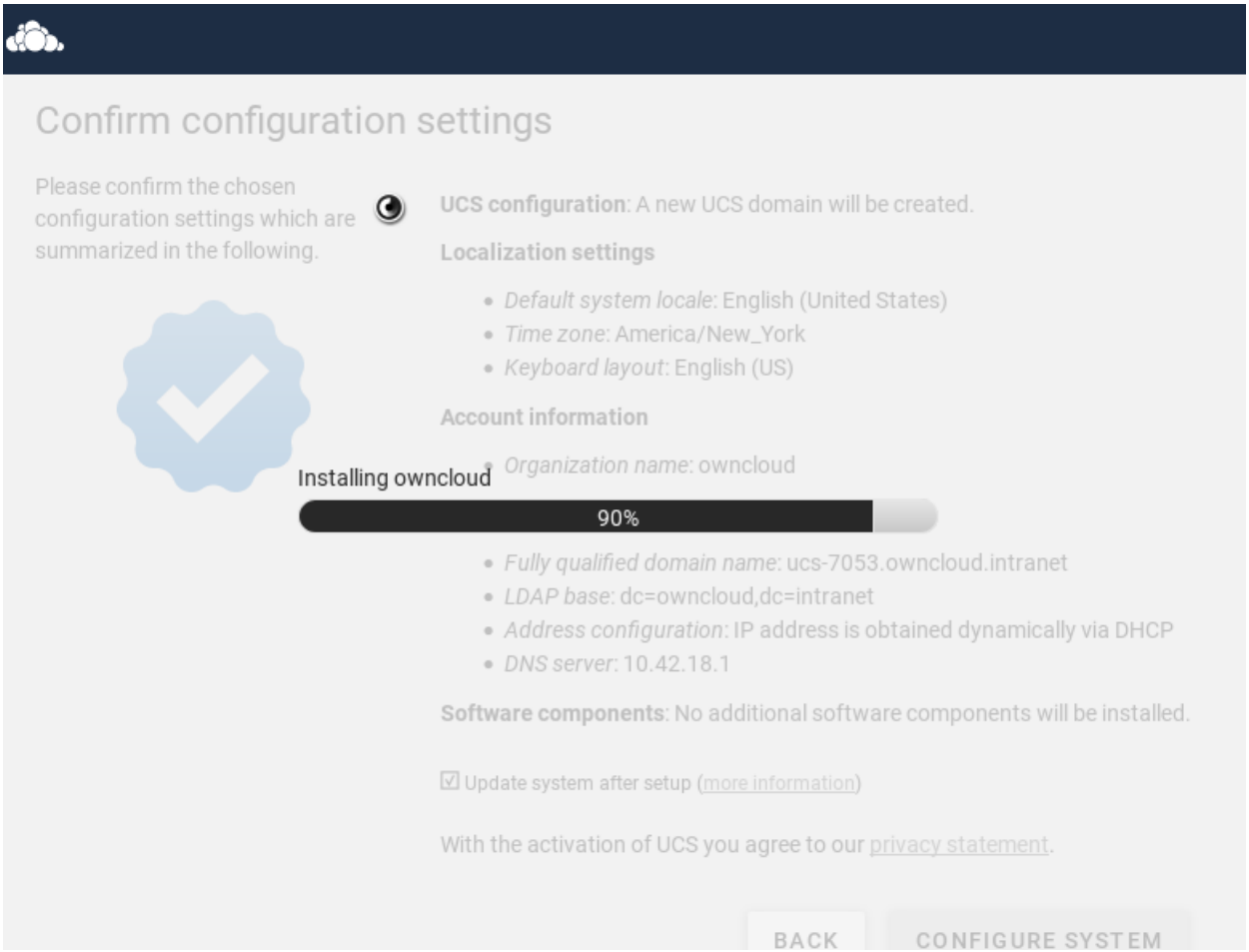


Here, you can set or change the **FQDN** to your custom address.

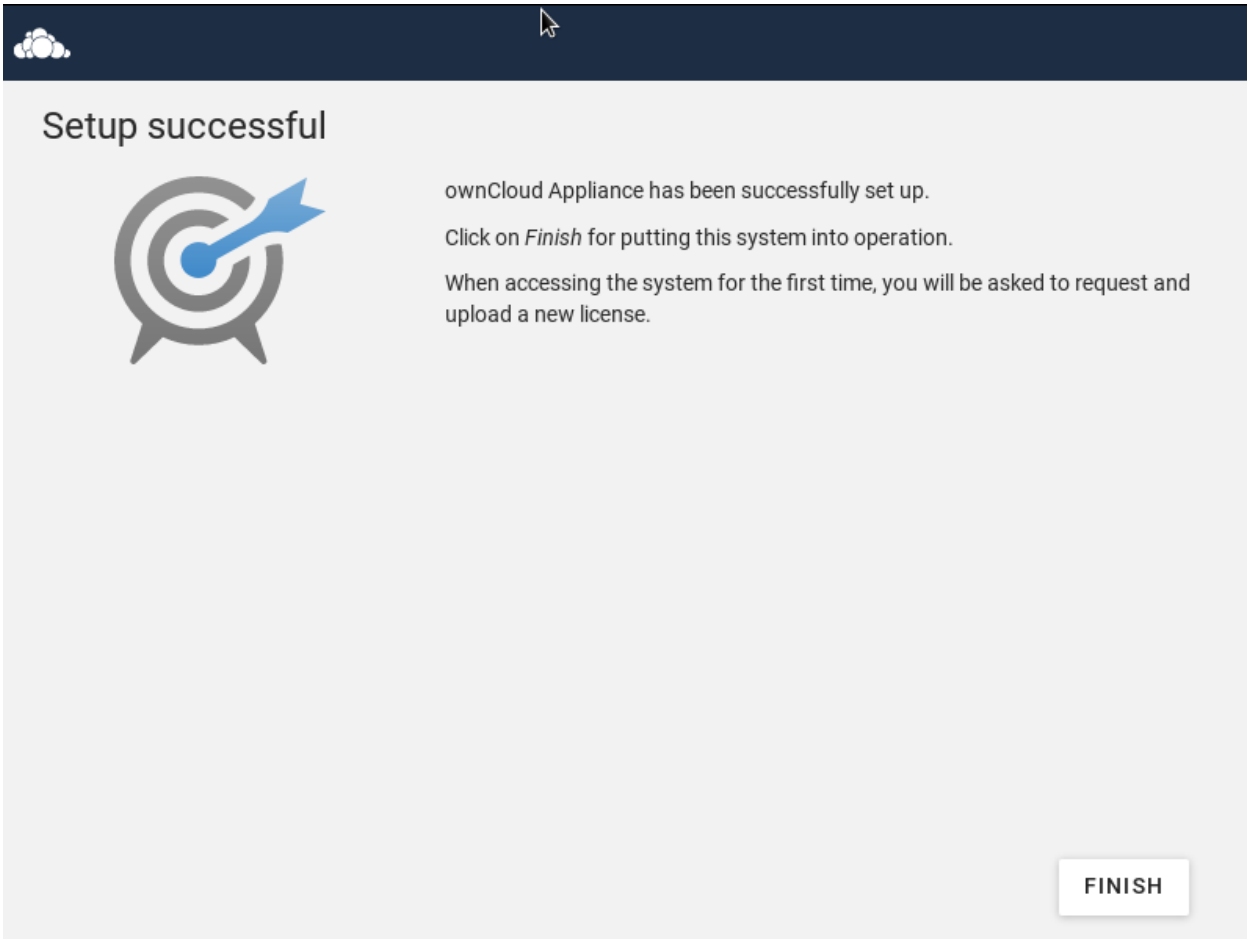


Here, you get a finalized confirmation screen of what you have entered / set and you

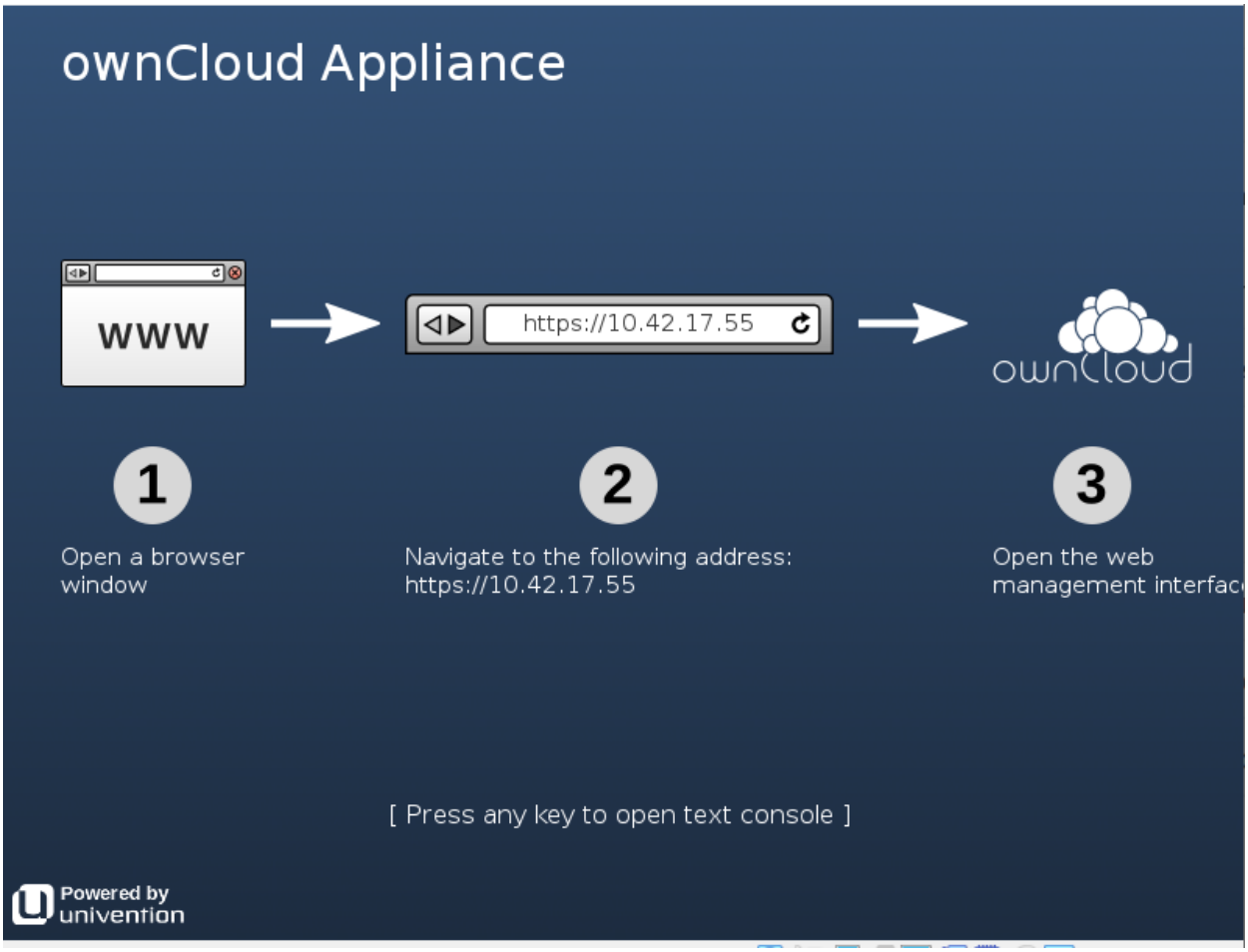
can finish the process. Note that if you let the check box to update your system in - the installation will take **considerably longer**. Keep his in mind. You can apply the updates later on if you choose to skip it during the installation.



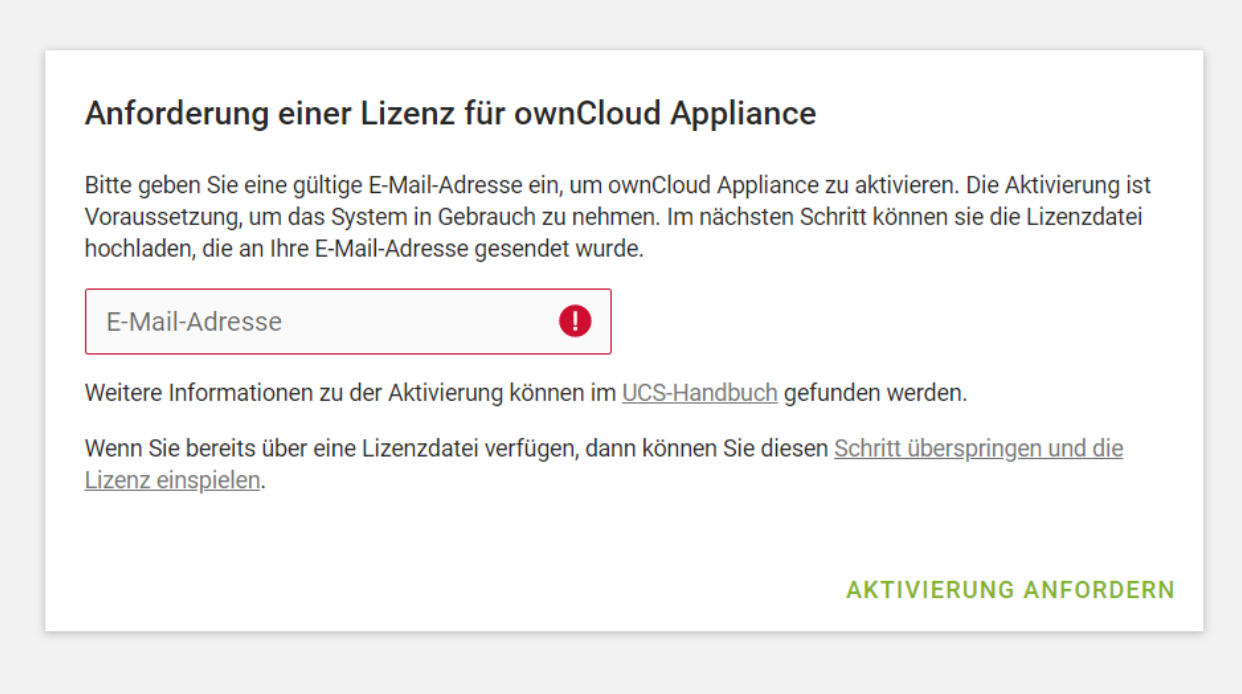
Wait until the setup is finished.



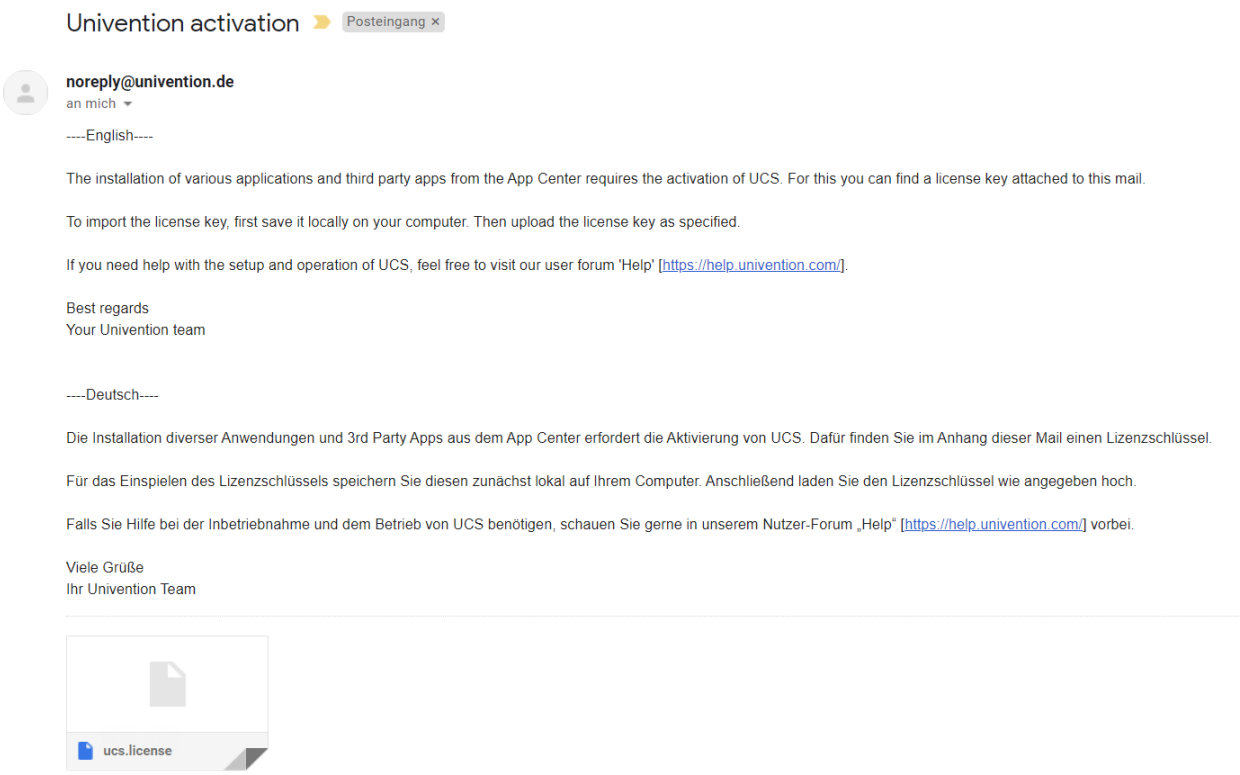
When the installation is complete, you will see this screen informing you that the installation was successful.



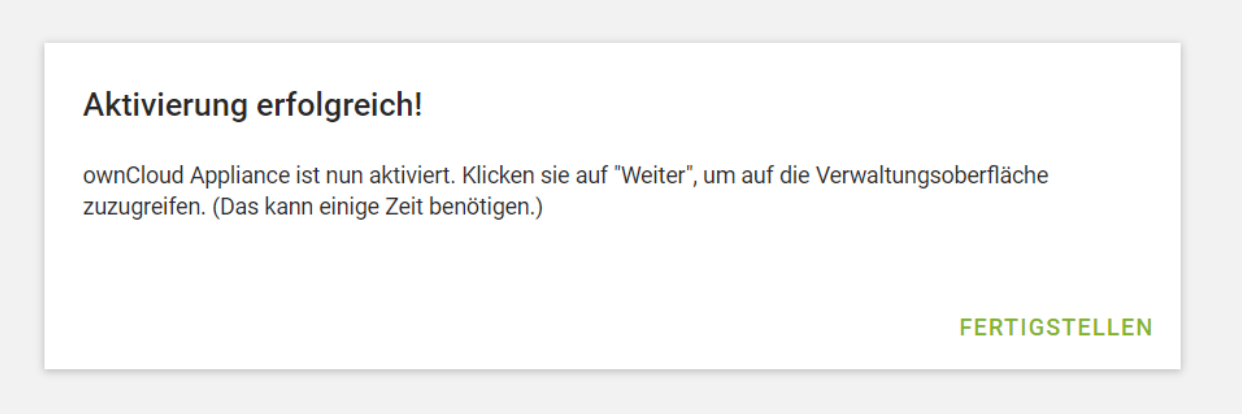
The VM will show you this screen, showing the ip **address** you have to navigate to in order to **activate** your appliance



Enter your email-address to receive a **license** to activate your Appliance. Without activation you **can not login** in to the appliance.



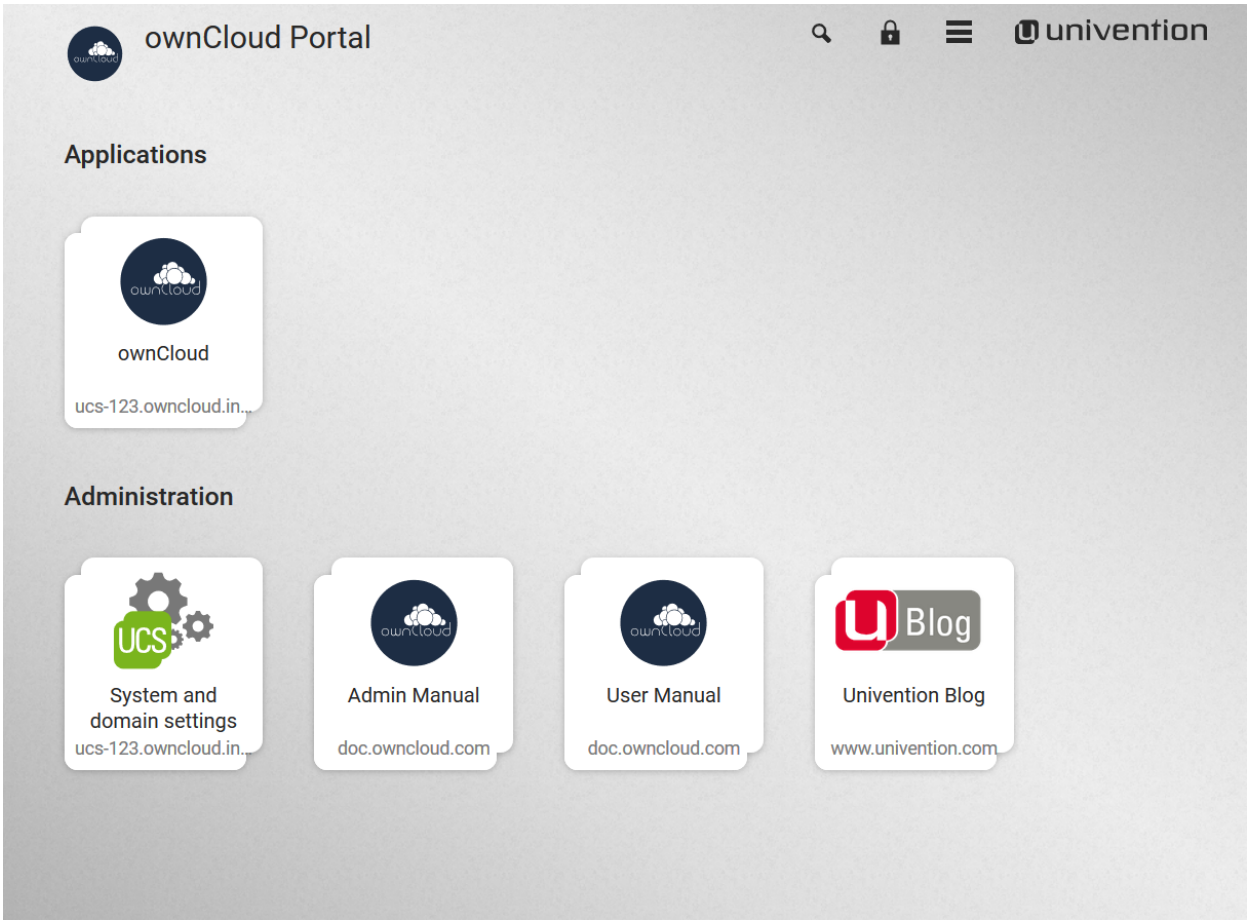
You will receive the email shortly. **Download** the license and **import** it in to the appliance.



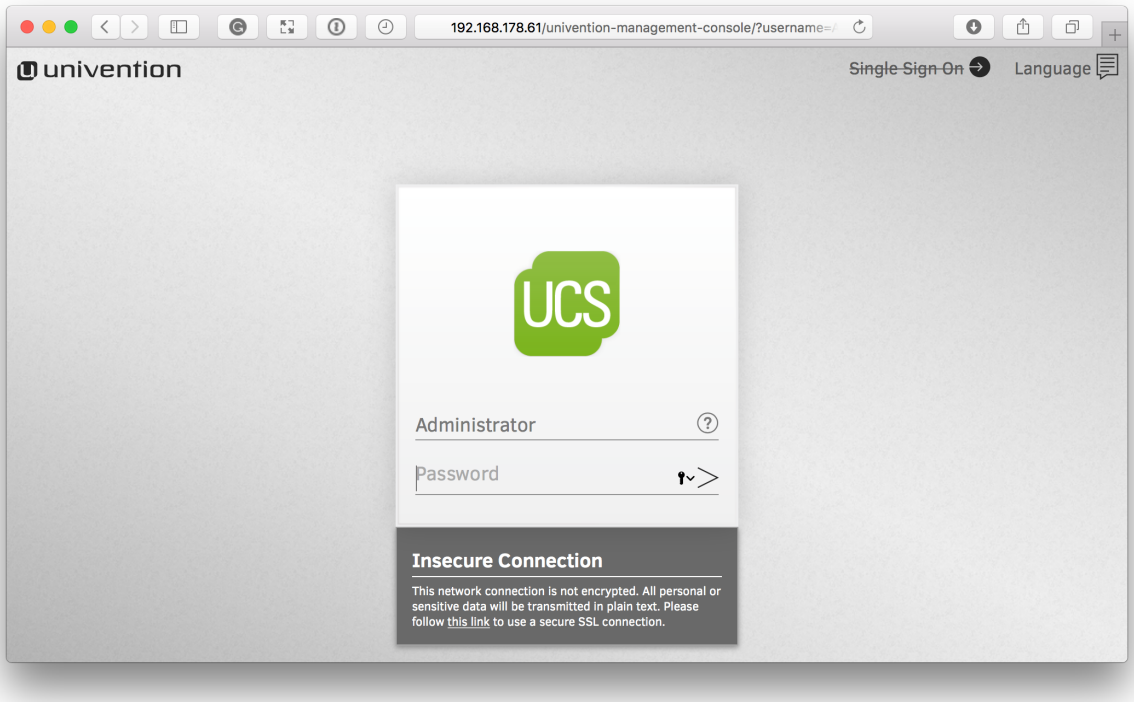
Once activated, you will see this screen, informing you that the appliance was successfully activated.

Administer the Appliance

Once activated, you should be redirected to the portal, which you can see below.

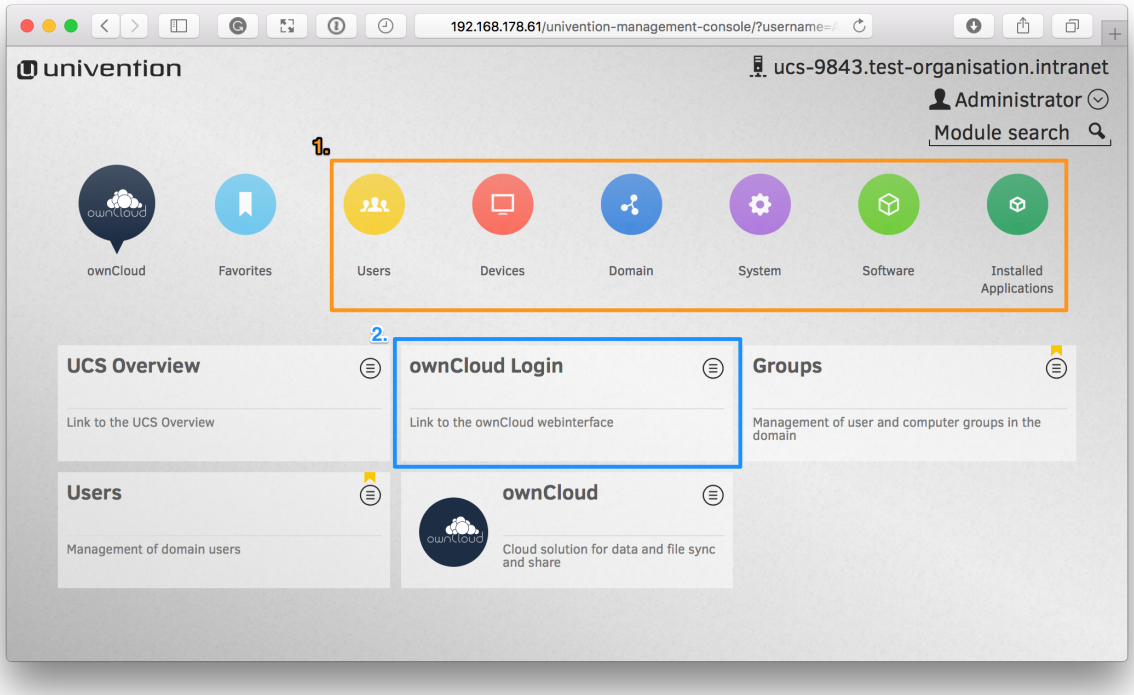


If you want to create new users and groups, or download apps from the Univention appcenter click on the btn:[System and domain settings]. Login as the "**Administrator**" using the password that you supplied during the configuration wizard earlier.



If you are not redirected to the appliance login page, you can open it using the following url: https://<ip address of the virtual machine>/univention-management-console.

After you've done so, you will now be at the Univention management console, which you can see below.



The management console allows you to manage the virtual appliance (1), covering such areas as: *users*, *devices*, *domains*, and *software*. You will also be able to access the ownCloud web interface (2).

The default username for the ownCloud is: owncloud and so is the password. The

password is **not** the password you supplied during the configuration wizard.

For security reasons rpcbind should be disabled in the appliance. An open, from the internet accessable portmapper service like rpcbind can be used by an attacker to perform DDoS-Reflection-Attacks. Furthermore, the attacker can obtain information about your system, for example running rpc-services, or existing network shares. The German IT security agency "BSI" reported, that systems with an open rpcbind service were used to perform DDoS-Reflection-Attacks against other systems.



If you want to create NFS shares on the appliance and give someone permission to access them, then you can enable rpcbind again.

Active Directory Integration

In case you have tested the appliance with your Active Directory environment, removed the appliance and now want to include it again - you might run into some issues.

The solution is to clean up the previous DNS entries in your Domain Controller. After that, you should be able to include the appliance again in your Active Directory environment.

Appliance Configuration

In this section you will find all the details you need to configure the ownCloud appliance..

Login Information and Custom Paths

Welcome to the ownCloud Appliance. Here are the login credentials.

```
username: owncloud
password: owncloud
```

Login to the Appliance via command line or SSH with the root account.

```
username: root
password: <Administrator password>
```

Login into the ownCloud docker container with this Univention command:

```
univention-app shell owncloud
```

Set ownCloud as the default page instead of the Univention Portal

```
ucr set apache2/startsite=/owncloud
service apache2 restart
```

ownCloud’s data directory is under the following path:

```
/var/lib/univention-appcenter/apps/owncloud/data
```

ownCloud's config directory, containing config.php:

```
/var/lib/univention-appcenter/apps/owncloud/conf
```

File extension blacklist for the Ransomware app:

```
/var/lib/univention-
appcenter/apps/owncloud/data/custom/ransomware_protection/blacklist.txt.dist
```

App Settings

Configurable Options

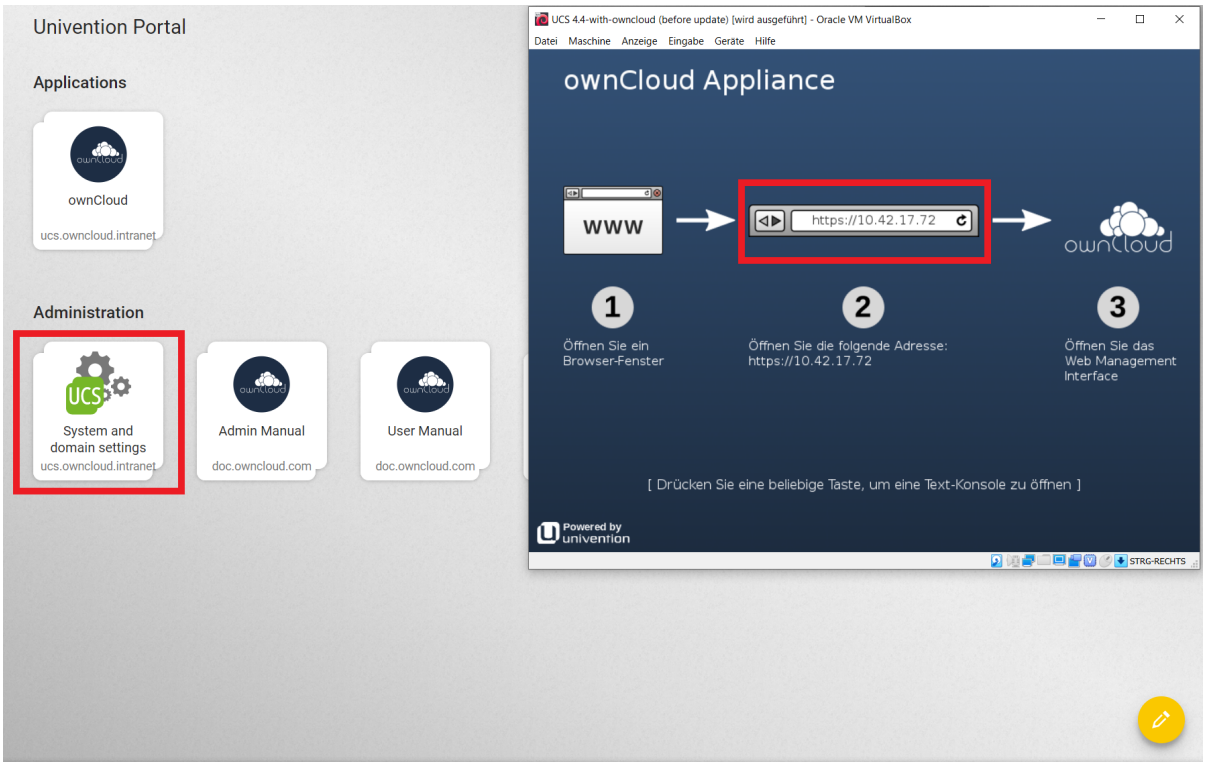
You can configure certain the ownCloud app in the Univention Portal:

- Enterprise License Key
- Marketplace API Key
- Language
- ownCloud Domain
- ownCloud SubURL
- Log Level
- Password Reset

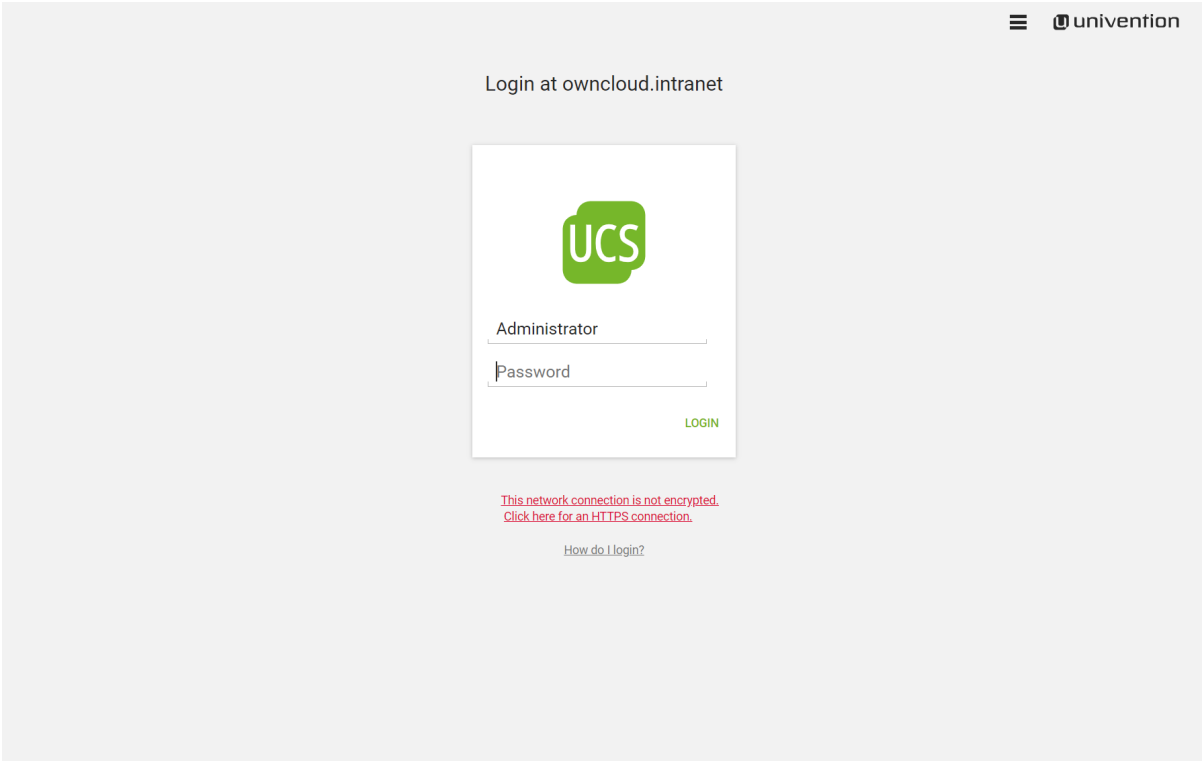
Access the settings:

Here is how you can access these settings:

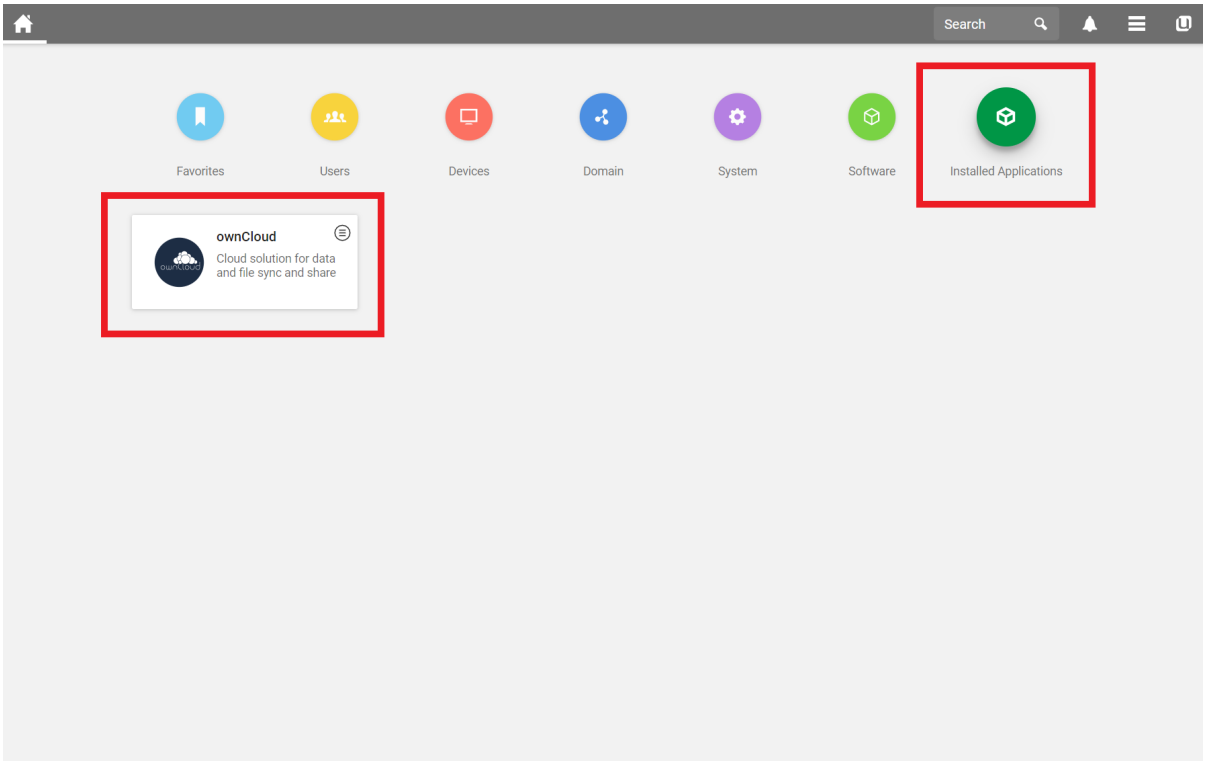
1. Go to the Portal Page of your Appliance and select **System Settings**.



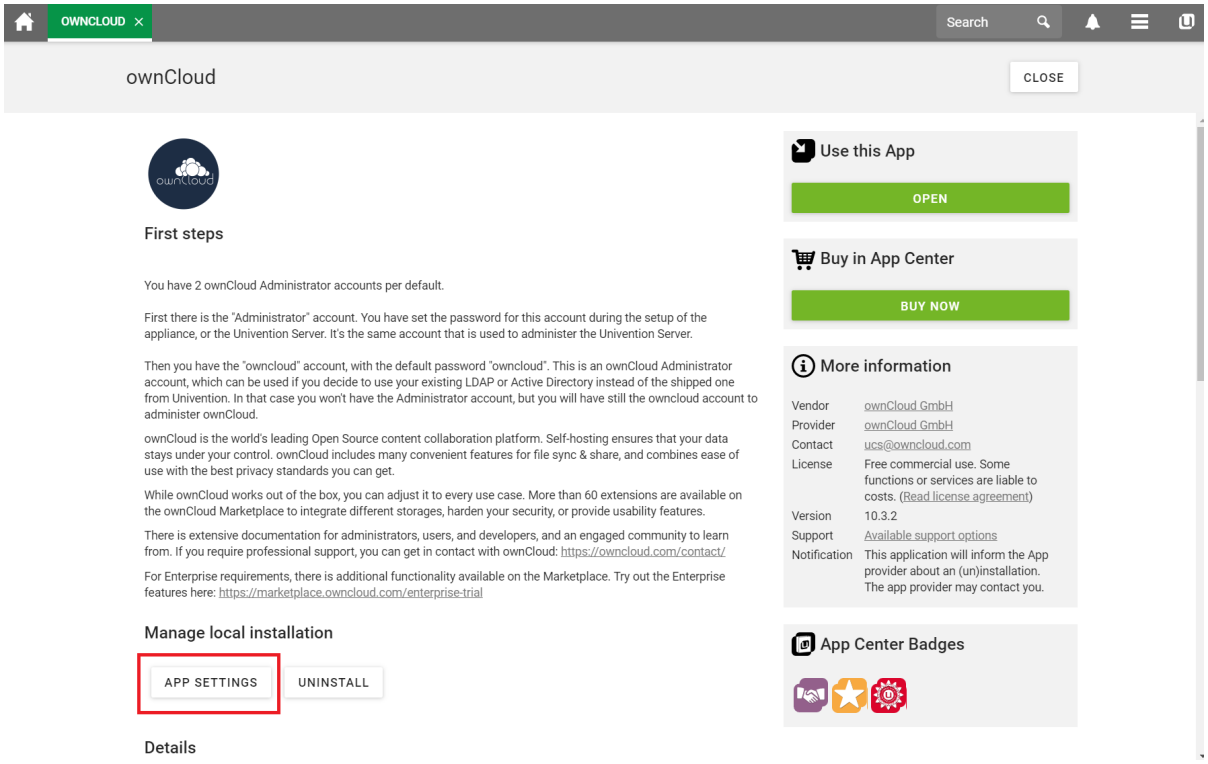
2. Login as the Appliance Administrator.



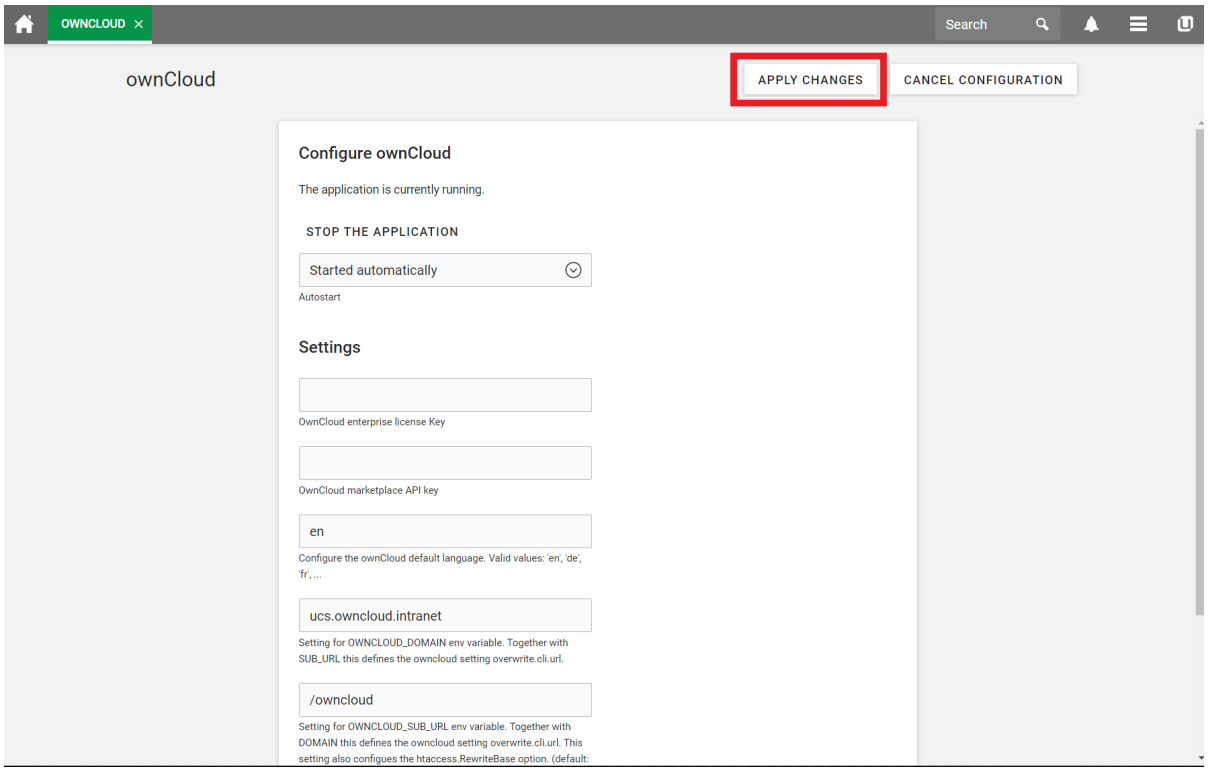
3. Go to **Installed Applications** and select **ownCloud**.



4. Go in to **App Settings**.



5. After changing these Settings, don’t forget to **save** them.



How to add certificates

If you want to use your own SSL certificates for the appliance, you have to follow these three steps:

1. Create the certificates and deposit them on your appliance.
2. Connect to your appliance either directly on the command line of your virtual machine or via ssh connection to your appliance.
3. Execute the following commands:

```
ucr set apache2/ssl/certificate="/etc/myssl/cert.pem"
ucr set apache2/ssl/key="/etc/myssl/private.key"
```

Remember to adjust the path and filename to match your certificate.

Once you've completed these steps, restart Apache using the following command:

```
sudo service apache2 restart
```

Now your certificates will be used to access your appliance. If you want to limit the access to your server exclusively to HTTPS, use this command:

```
sudo ucr set apache2/force_https=yes
```

For further information please visit our partner site at Univention.

Firewall Protected Environment

If you are considering setting up the appliance in an environment with a firewall, please create rules that permit access to the following hosts. If your DNS is not working, you can use the IP addresses instead. If you are using Google as your DNS server (IP=8.8.8.8), you have to permit access to it too.

Firewall Rules:

- 176.9.114.147
- 5.9.68.237
- 8.8.8.8
- docker.software-univention.de
- marketplace.owncloud.com
- owncloud.com
- owncloud.org
- software-univention.de

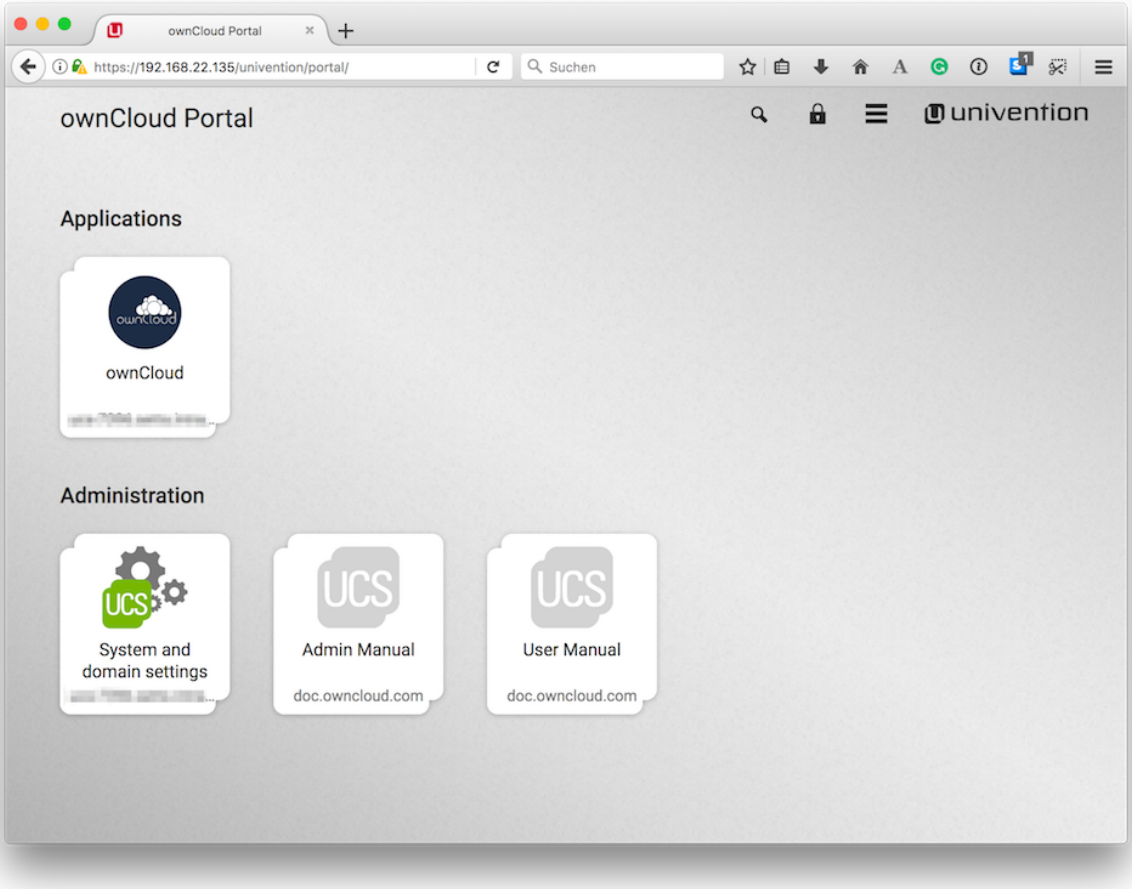
Adding Users and Groups in UCS for ownCloud

Introduction

If you want to add users and groups to your ownCloud installation via the UCS (Univention Corporate Server) UI, here's a concise guide showing how.

Login to the Univention Management Console

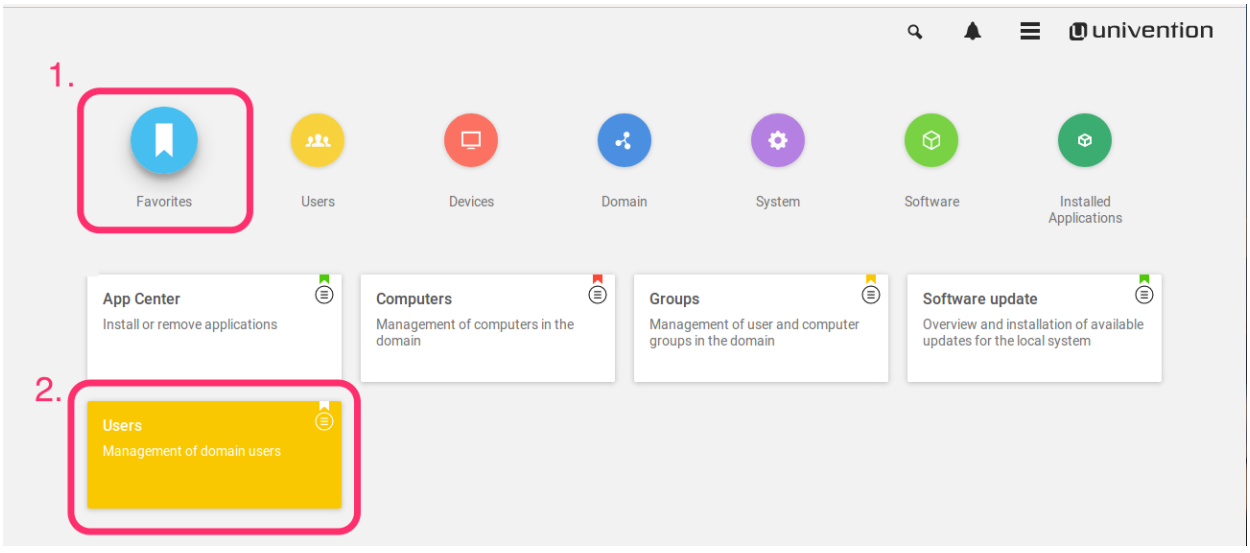
After logging in to the Univention server, under "**Administration**", click the first option, labeled btn:[System and domain settings].



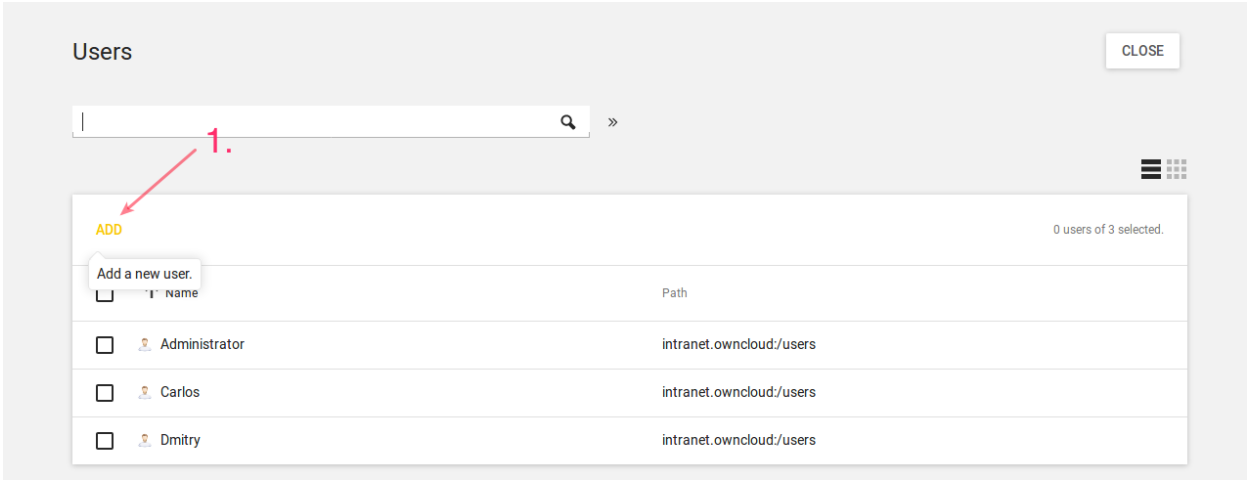
This takes you to the Univention Management Console.

Create the User

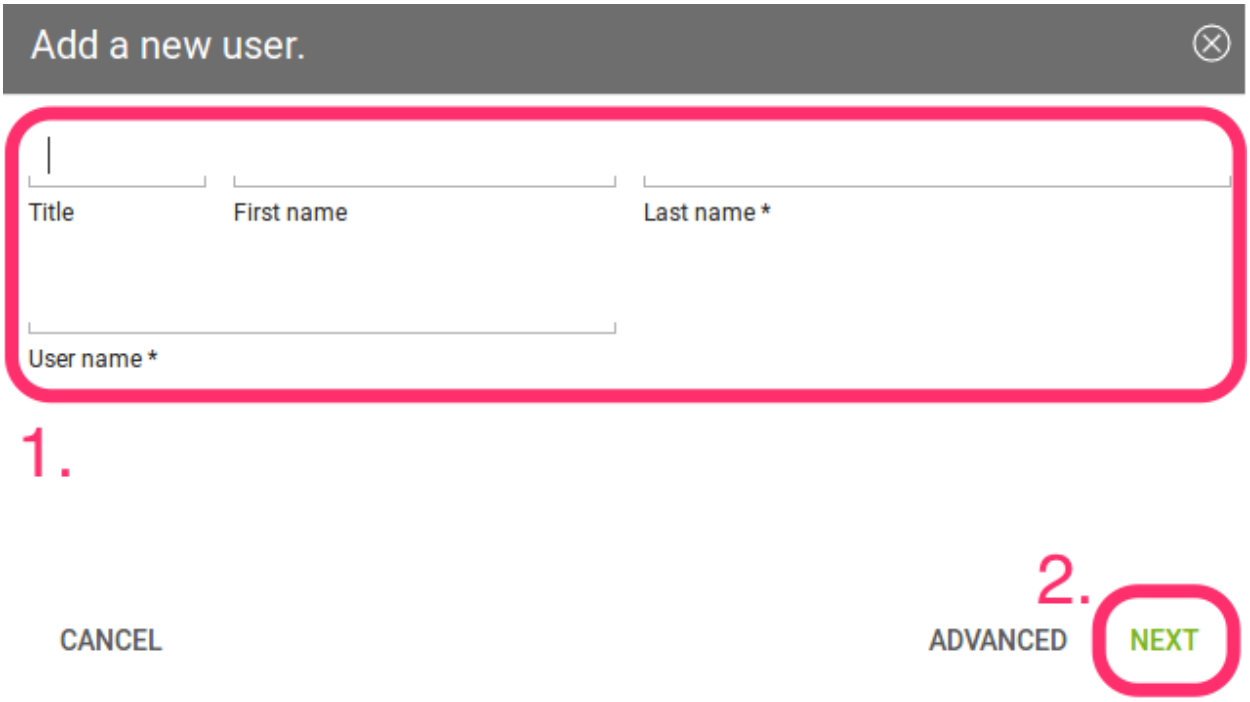
Once there, click btn:[Users].



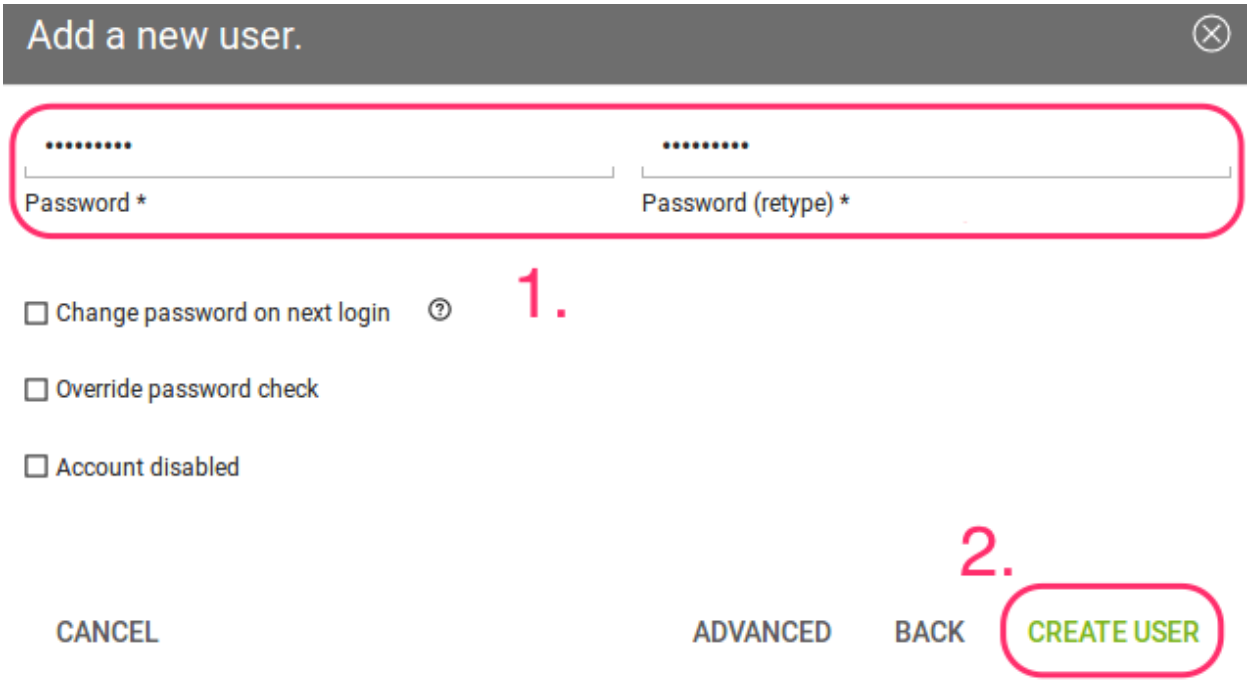
In the screen that appears, add a new user by clicking btn:[ADD] in the top left-hand corner of the users table.



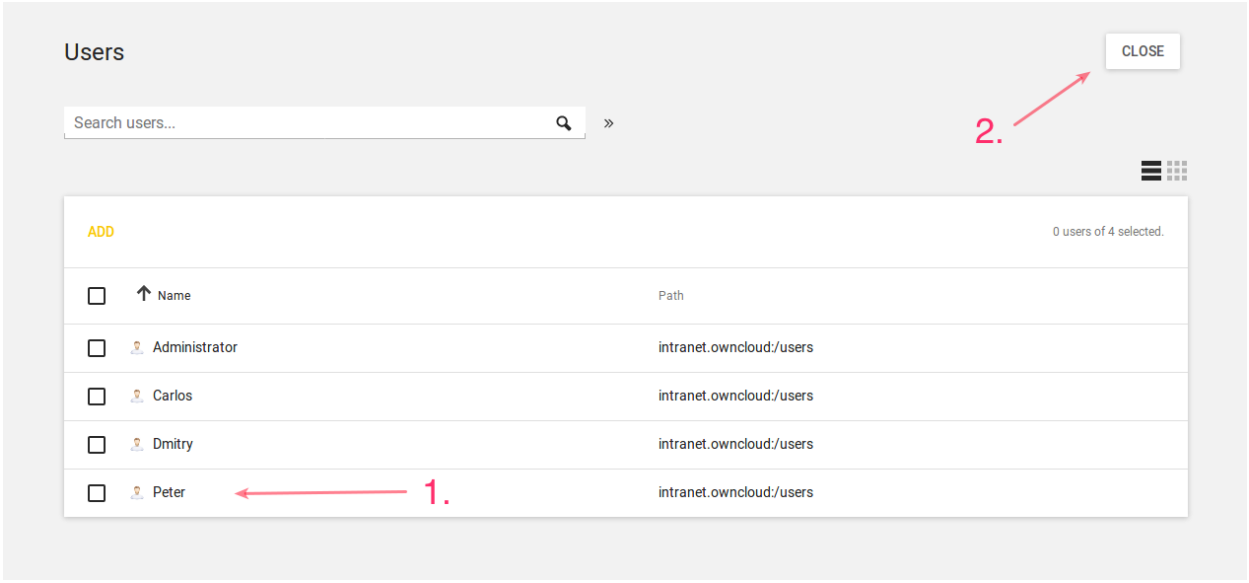
This opens up a new user dialog, where you can supply the relevant details for the new user. Enter a username and optionally a first name, last name, and a title. Then click btn:[NEXT].



In the next dialog that appears, enter and confirm the password. You can, optionally, choose some further options, if desired. Then click btn:[CREATE USER].

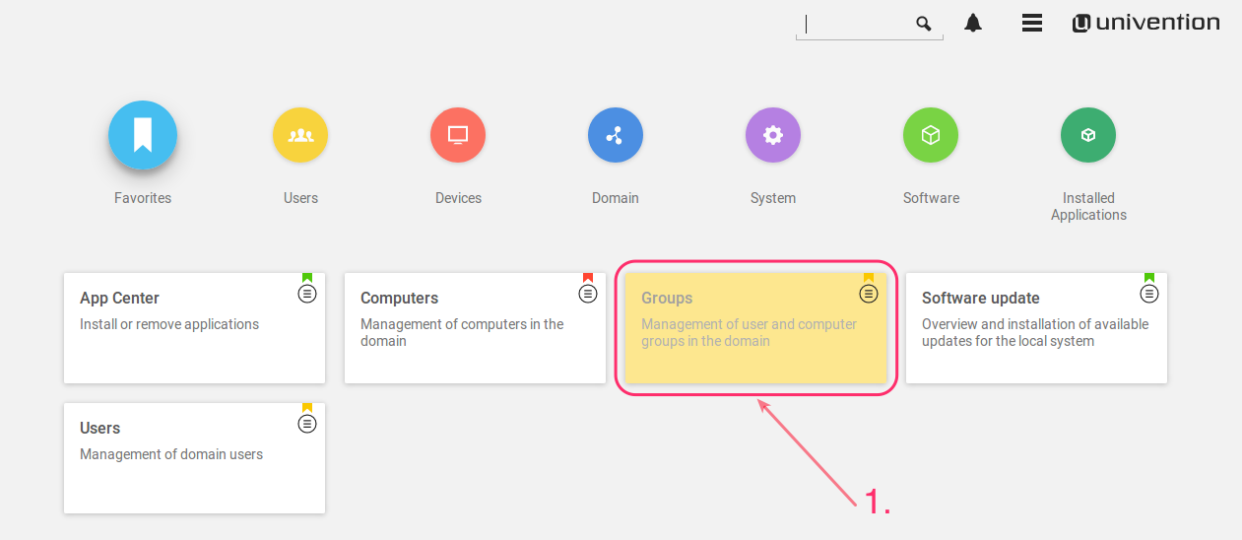


The new user will have been created, so click the btn:[CLOSE] button, in the top right-hand corner, to go back to "**Favorites**".

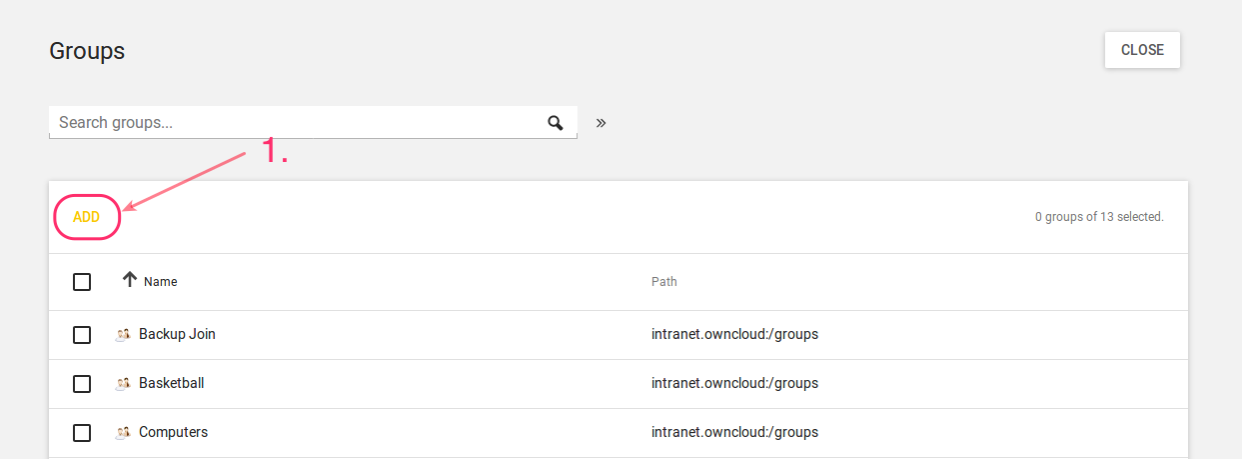


Create the Group

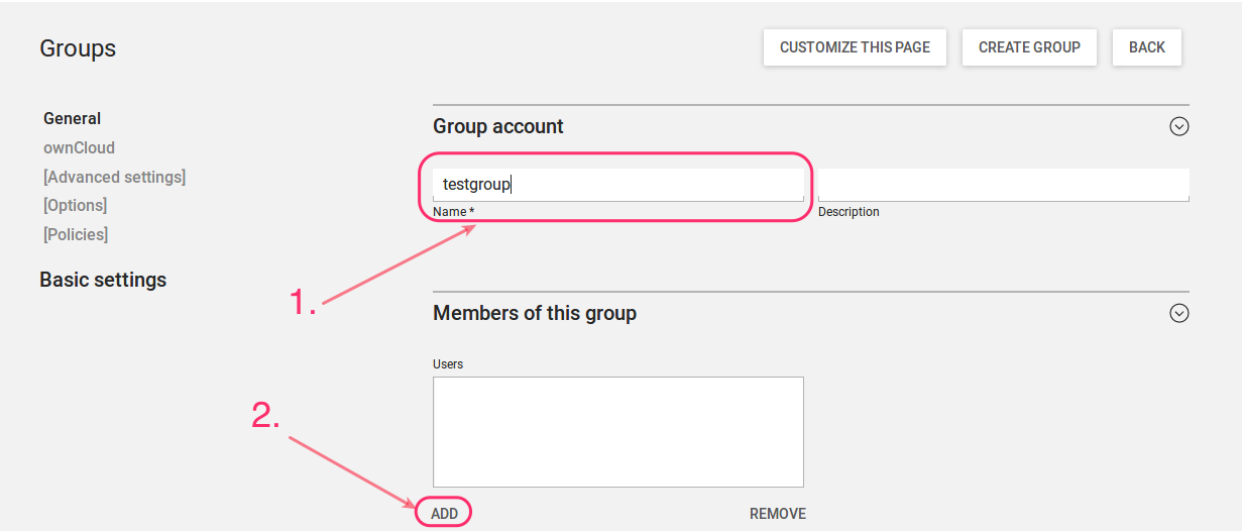
Now it’s time to create a new group. Click btn:[Groups], which is located between "**Computers**" and "**Software Update**".



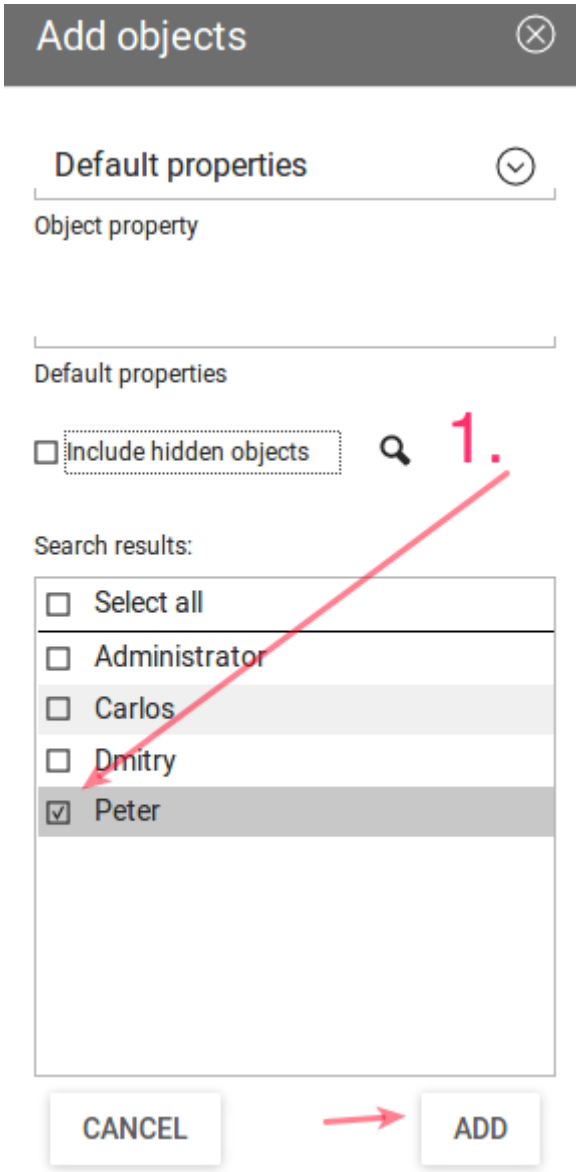
From there, click btn:[ADD], located on the left-hand side of the groups table.



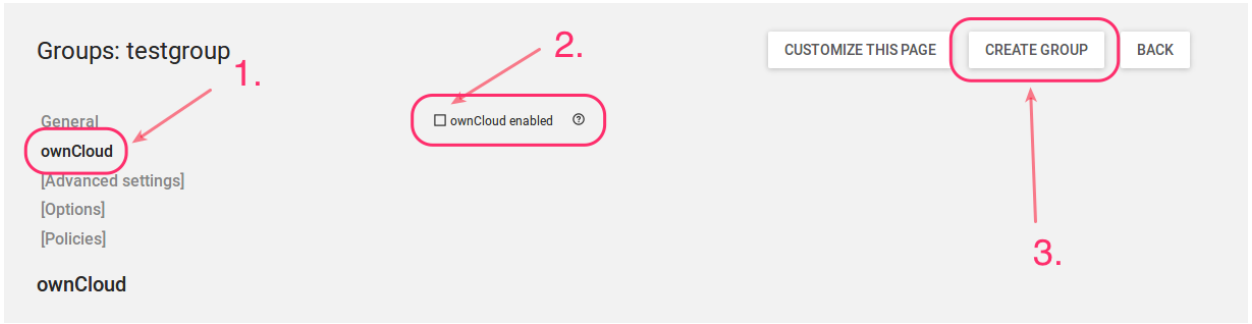
In the next dialog that appears, first enter the name of the group and optionally a description. Then, under "**Members of this group**", click btn:[ADD].



This opens up an "**Add objects**" (or "**Add new group**") dialog. Find the user, in the list at the bottom, that you want to add to the group, check the checkbox next to their name, and click btn:[ADD].



After that, click on btn:[ownCloud] in the left-hand side navigation, and check the option btn:[ownCloud enabled]. And lastly, click btn:[CREATE GROUP].



With that done, the new user and group are now available in your ownCloud installation.

Depending on your installation, you will either see these changes immediately or you will have to wait for the user sync to be done. This happens ever 10 minutes by default.

The ownCloud X Appliance Enterprise Trial

The appliance contains the community edition of ownCloud but can be easily upgraded to the enterprise edition. This upgrade gives you access to a free, 30-day trial of the

enterprise edition and all it's features. All you need is an email address to get started. Here are the necessary steps:

- Visit https://marketplace.owncloud.com/enterprise-trial
- Enter your email address and chose a password
- Click on "*Complete Process*"
- Check your email and activate your account
- Log in with your credentials at https://marketplace.owncloud.com
- Copy the API key

Now you have to go to your ownCloud installation and enable the Market app

- To enable enterprise features Select "*Add API Key*" and paste your key
- Start the Enterprise trial



If you don't see the button to install the "*Enterprise App Bundle*" select "*Clear cache*" and refresh the page.

Now you have access to the full ownCloud enterprise experience.

Working on Documents in the ownCloud Appliance

Introduction

Creating and editing documents in ownCloud can be achieved with either Collabora or OnlyOffice. It's your choice which one you prefer to use.

This guide covers the setup and update of the two office apps.



Access with **HTTPS** using a **domain name** is required. Add the IP address and the domain name of your appliance to your /etc/hosts file, or have it added to your existing DNS server, if you don't want to use the Appliance as your DNS server.

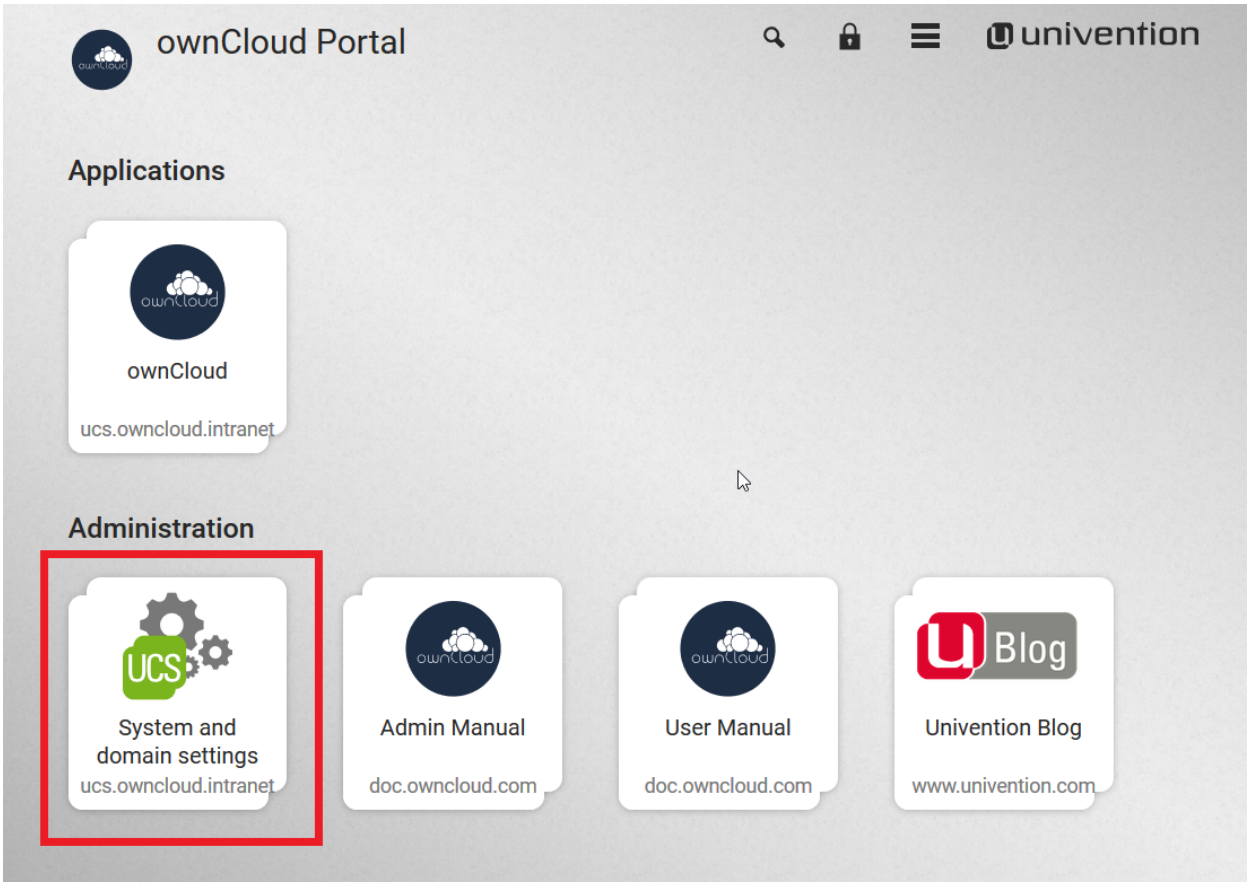
Appcenter

First you have to get to the Appcenter. Here are the steps to do that:

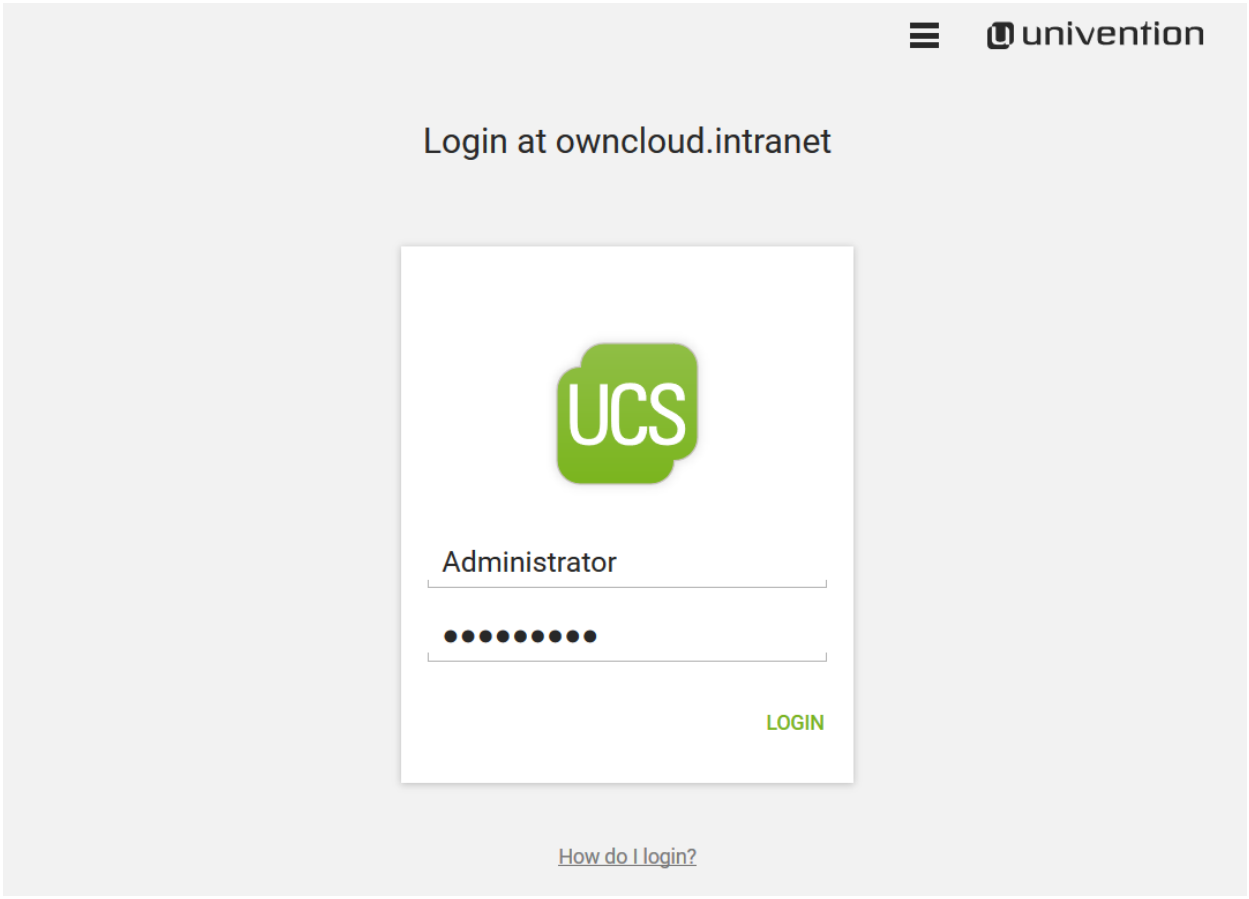
1. Connect to your appliance using IP address or domain name.

```
https://172.16.40.100
# or
https://ucs-2341.CompanyName.com
```

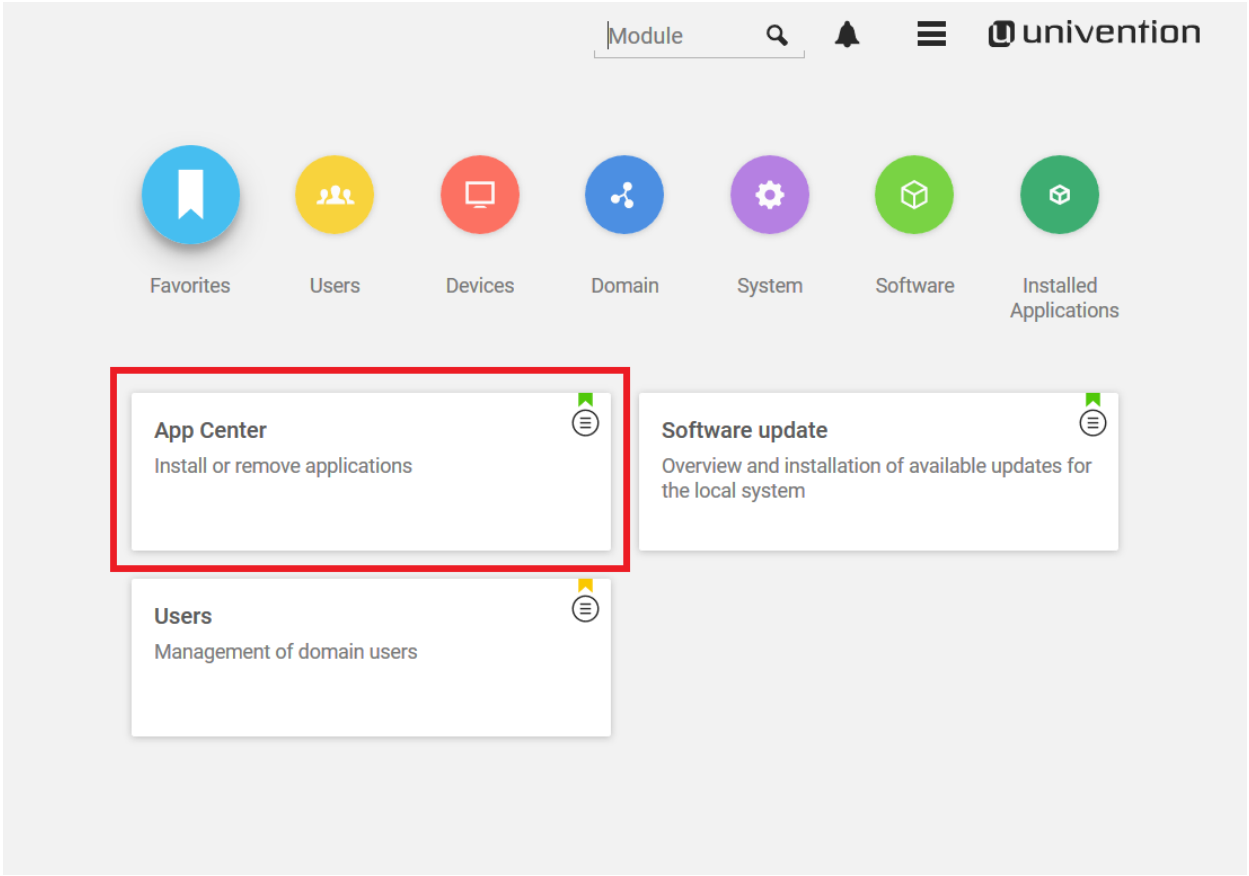
- Login into the management console
  - Click on the btn:[Domain and System] settings



- Type in the Administrator as username and the password you set.



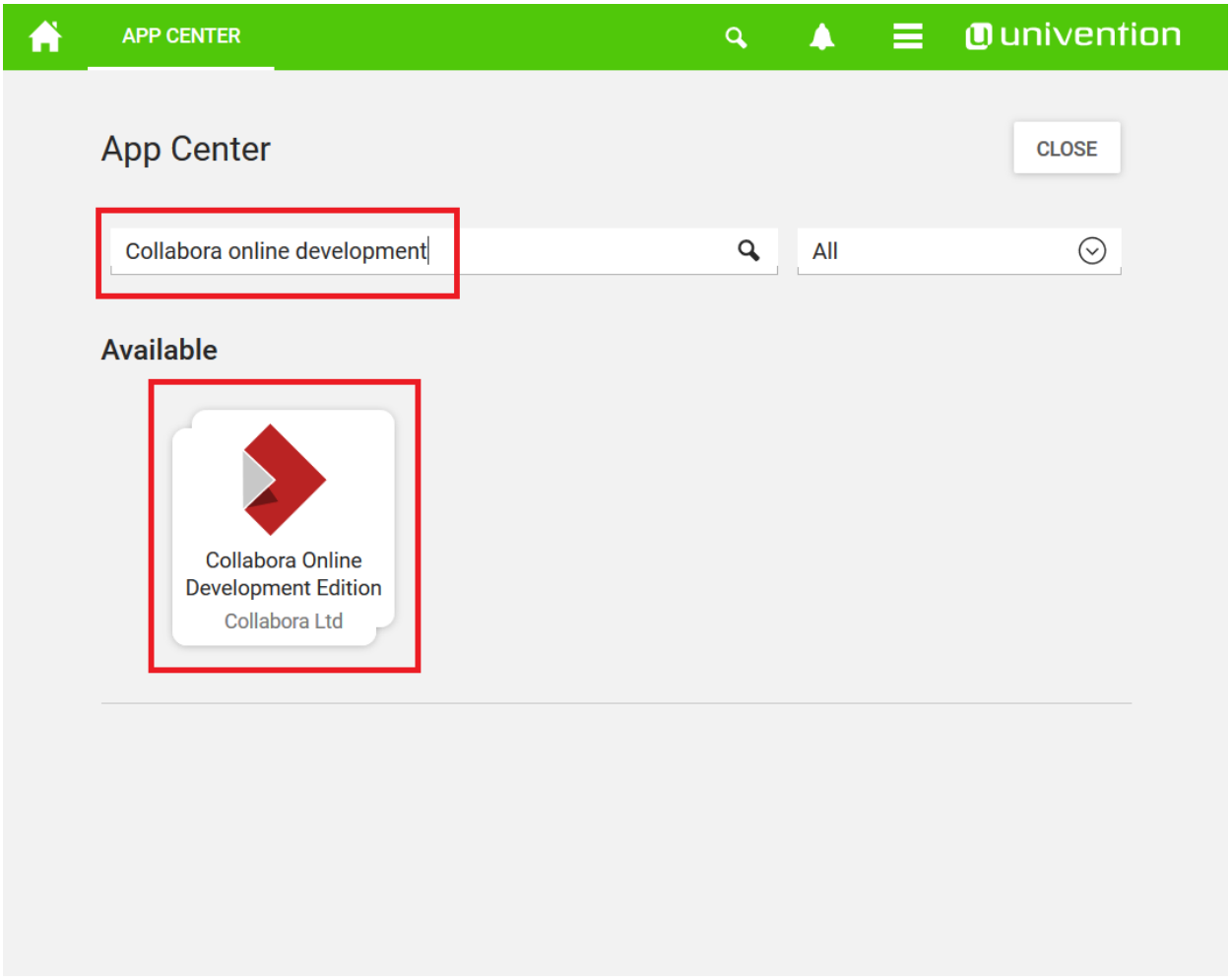
- Now you can access the **Appcenter**".

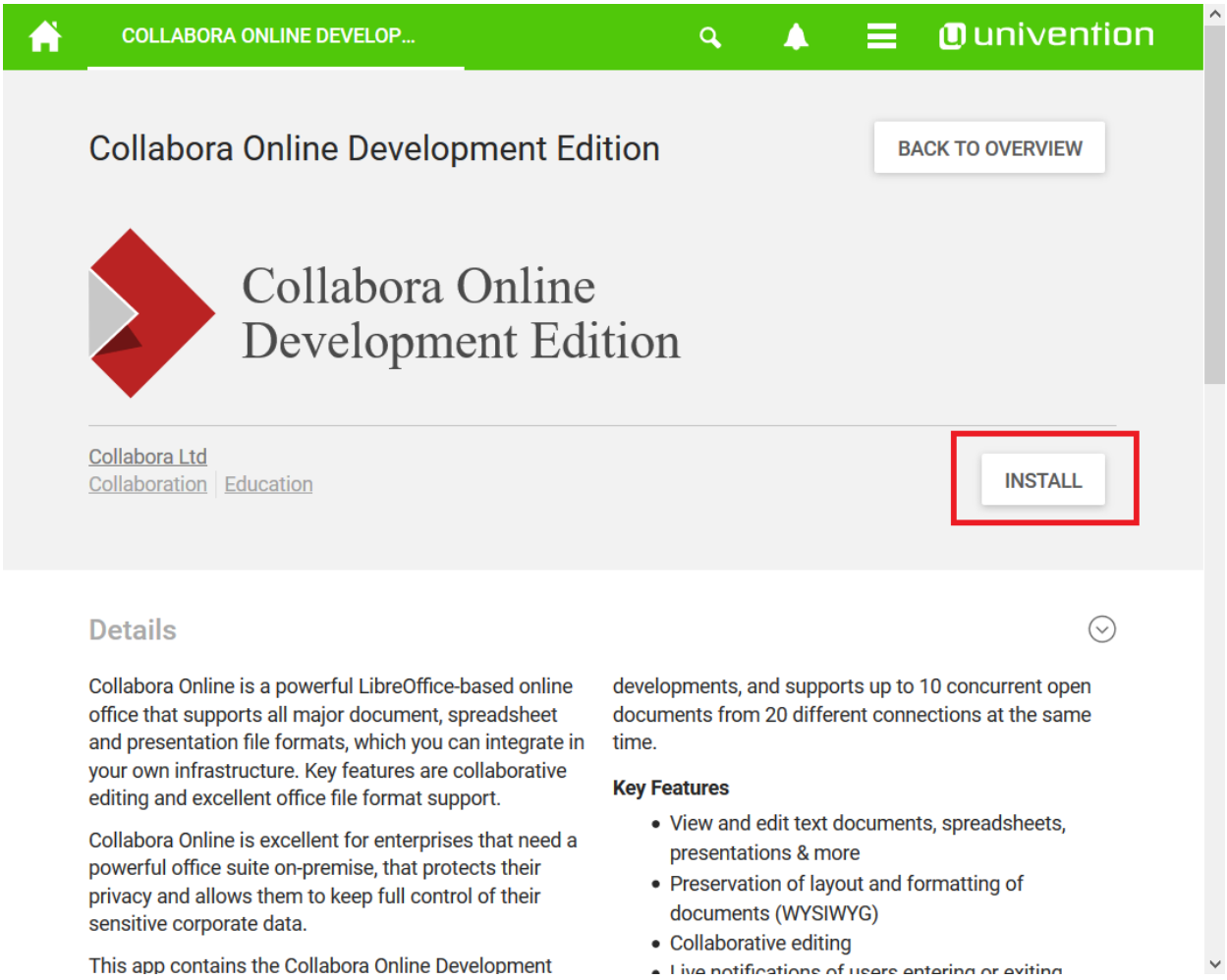


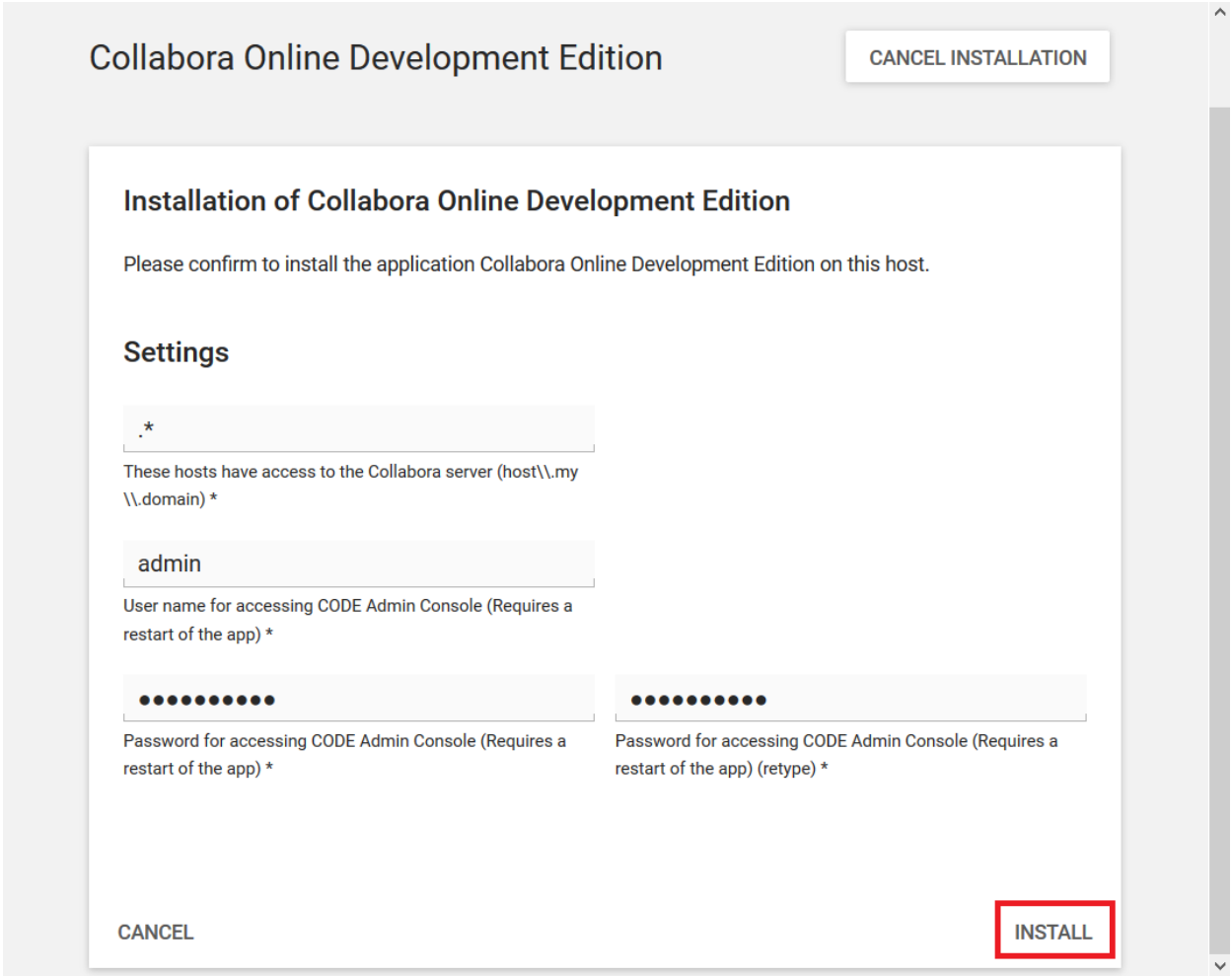
From here on it's your choice to install Collabora or OnlyOffice.

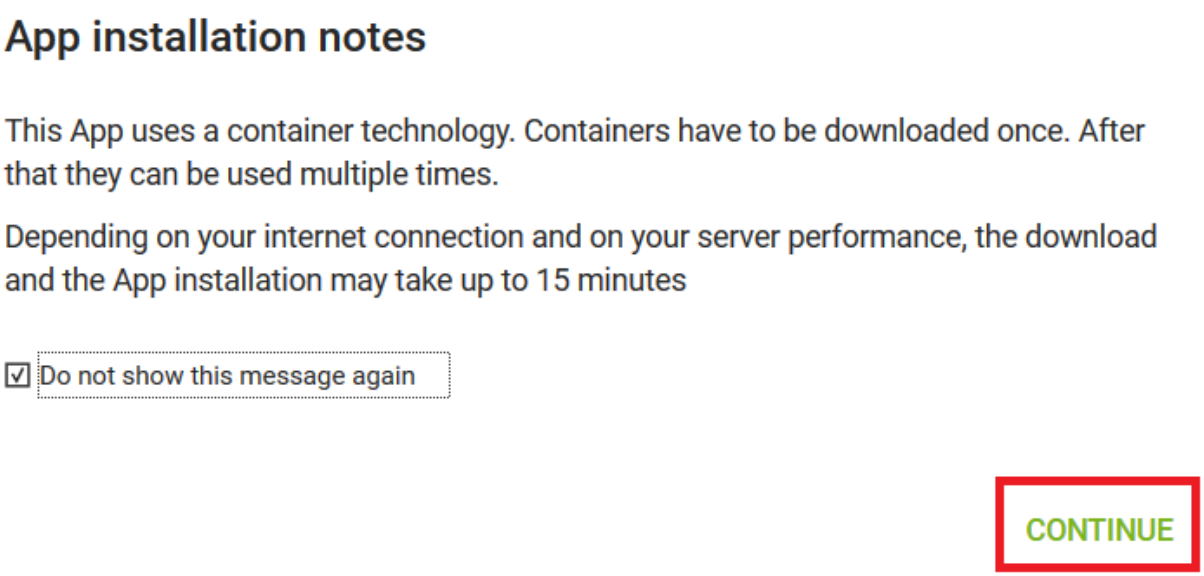
How to Install Collabora

- Install Collabora in UCS.

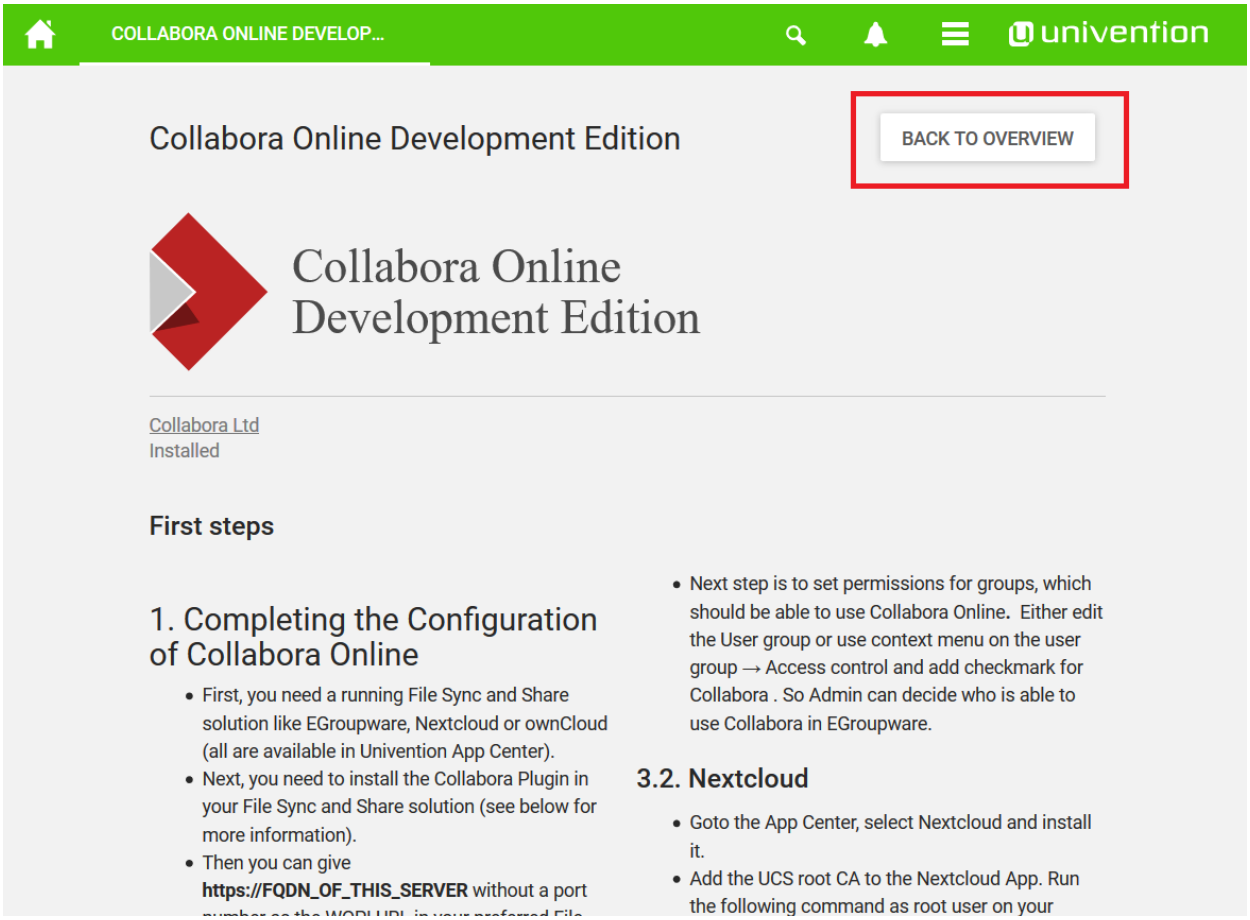

APP CENTER
univention
App Center
CLOSE
Collabora online development
All
Available
Collabora Online Development Edition
Collabora Ltd

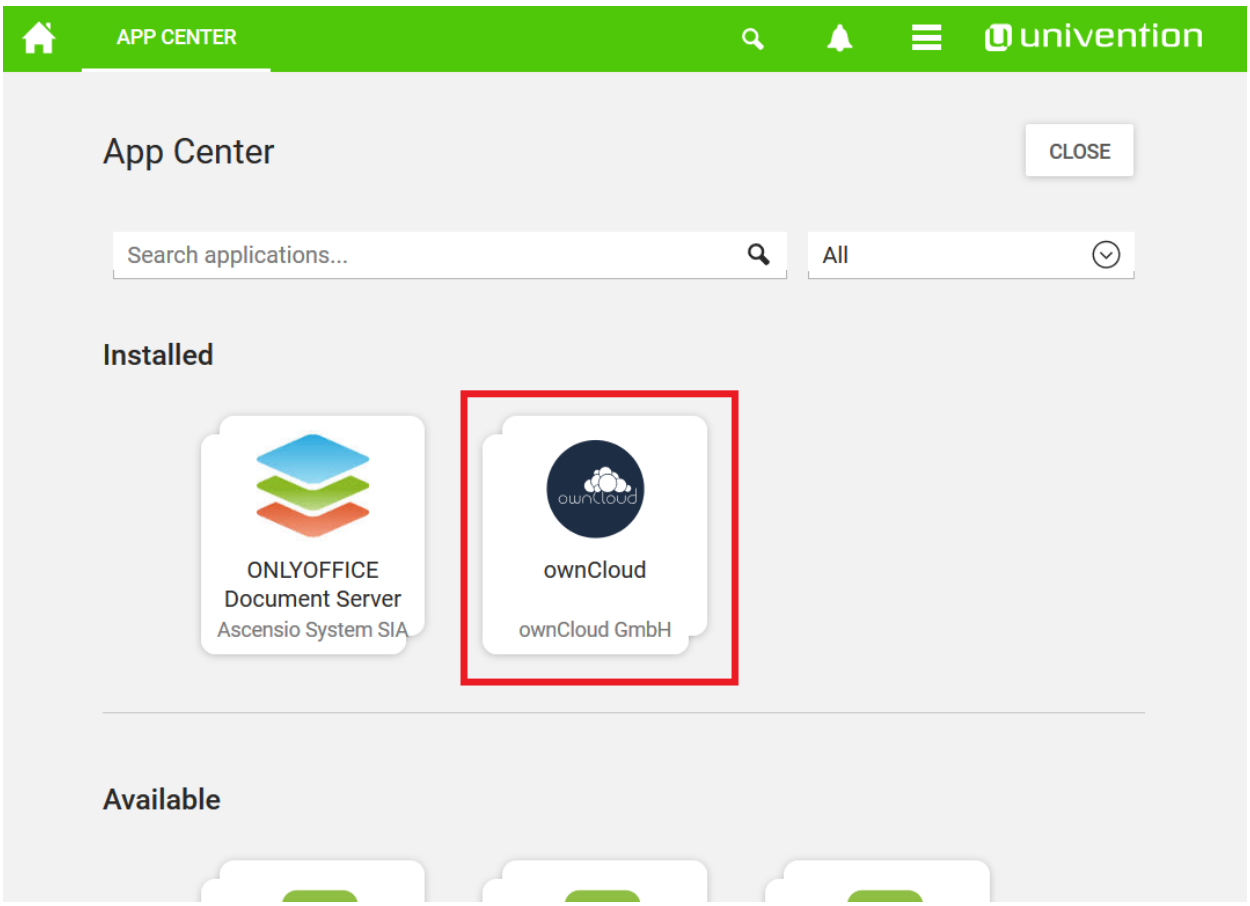

COLLABORA ONLINE DEVELOP...
univention
Collabora Online Development Edition
BACK TO OVERVIEW
Collabora Online Development Edition
Collabora Ltd
Collaboration | Education
INSTALL
Details
Collabora Online is a powerful LibreOffice-based online office that supports all major document, spreadsheet and presentation file formats, which you can integrate in your own infrastructure. Key features are collaborative editing and excellent office file format support.
Collabora Online is excellent for enterprises that need a powerful office suite on-premise, that protects their privacy and allows them to keep full control of their sensitive corporate data.
This app contains the Collabora Online Development
developments, and supports up to 10 concurrent open documents from 20 different connections at the same time.
Key Features
View and edit text documents, spreadsheets, presentations & more
Preservation of layout and formatting of documents (WYSIWYG)
Collaborative editing

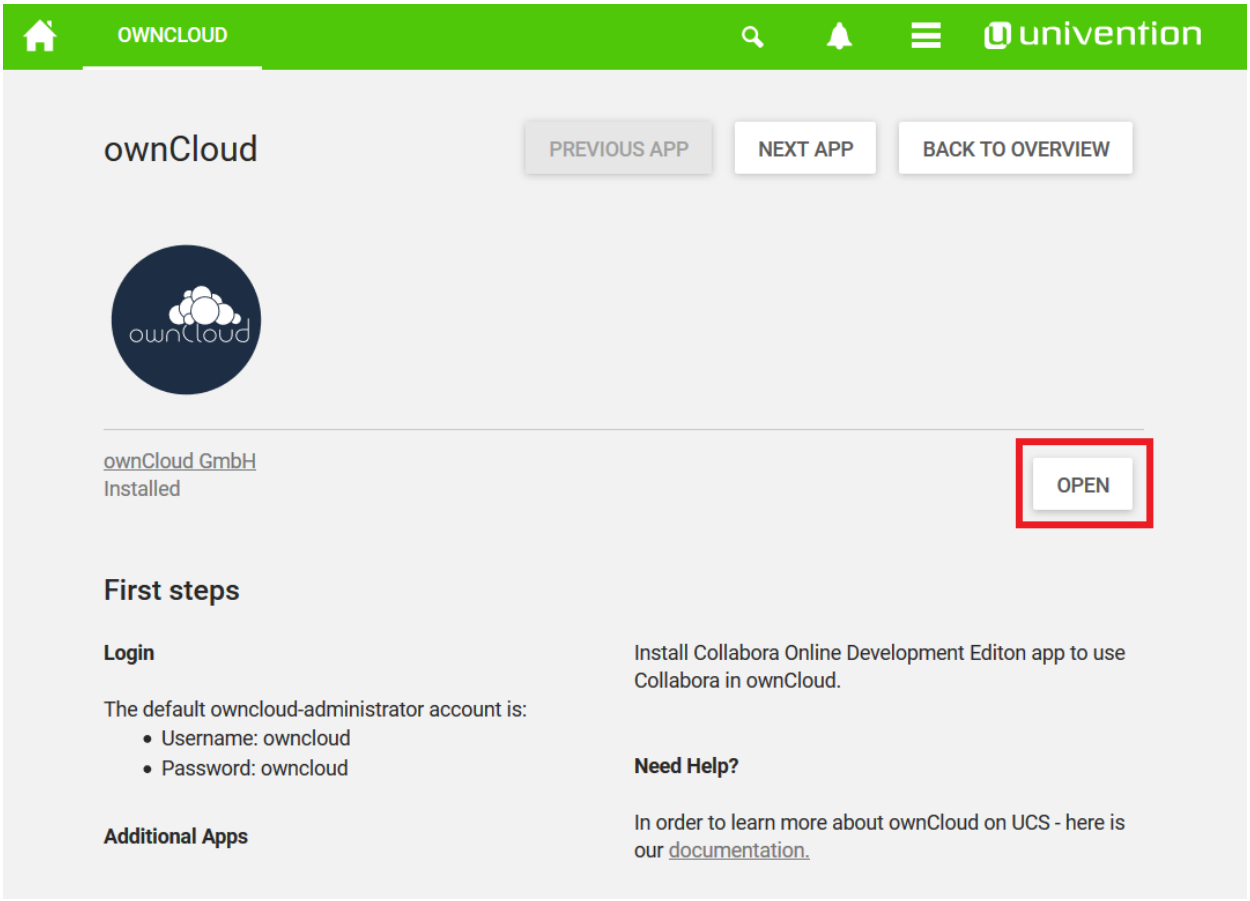




- Enable Collabora in ownCloud.

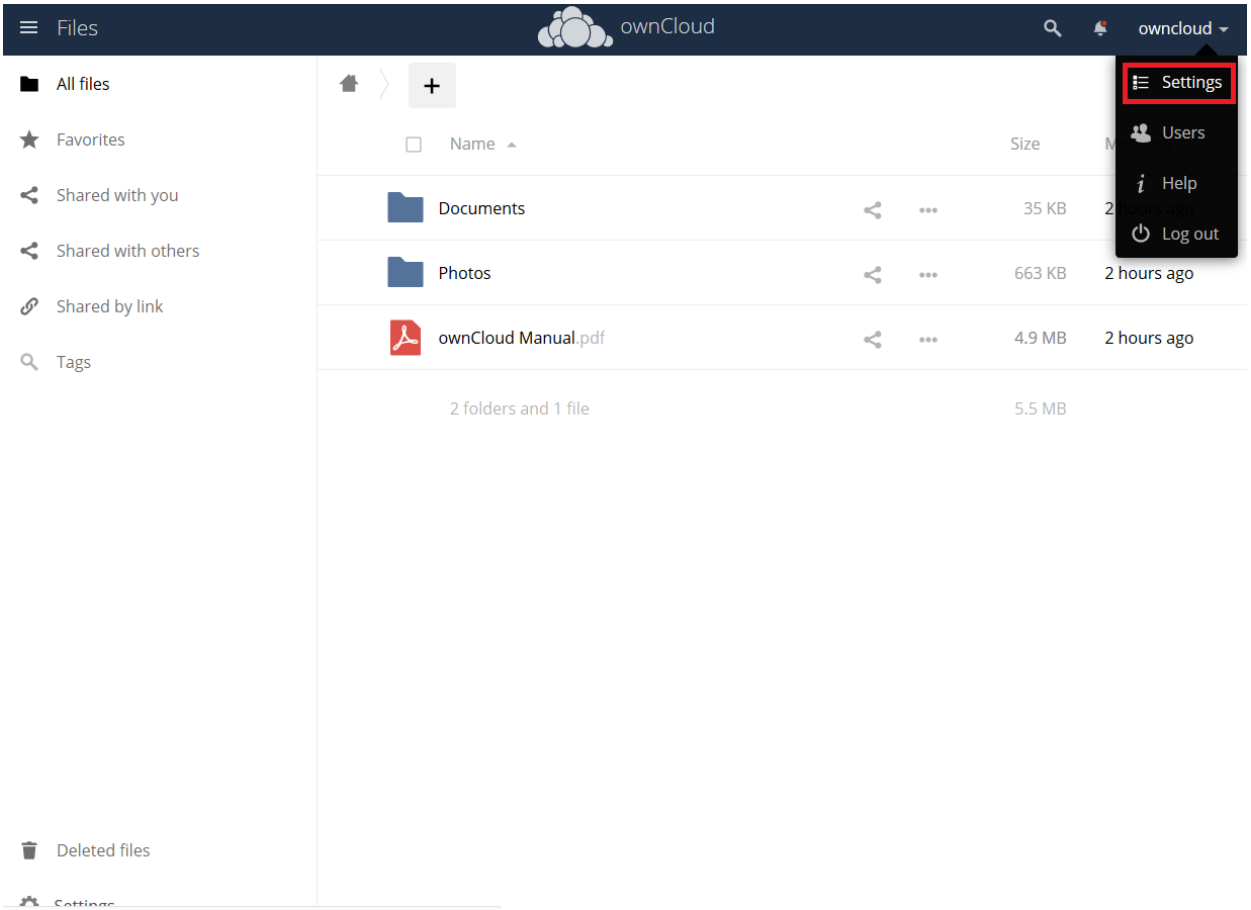

COLLABORA ONLINE DEVELOP...
univention
Collabora Online Development Edition
BACK TO OVERVIEW
Collabora Online
Development Edition
Collabora Ltd
Installed
First steps
1. Completing the Configuration of Collabora Online
First, you need a running File Sync and Share solution like EGroupware, Nextcloud or ownCloud (all are available in Univention App Center).
Next, you need to install the Collabora Plugin in your File Sync and Share solution (see below for more information).
Then you can give https://FQDN_OF_THIS_SERVER without a port
Next step is to set permissions for groups, which should be able to use Collabora Online. Either edit the User group or use context menu on the user group → Access control and add checkmark for Collabora . So Admin can decide who is able to use Collabora in EGroupware.
3.2. Nextcloud
Goto the App Center, select Nextcloud and install it.
Add the UCS root CA to the Nextcloud App. Run the following command as root user on your

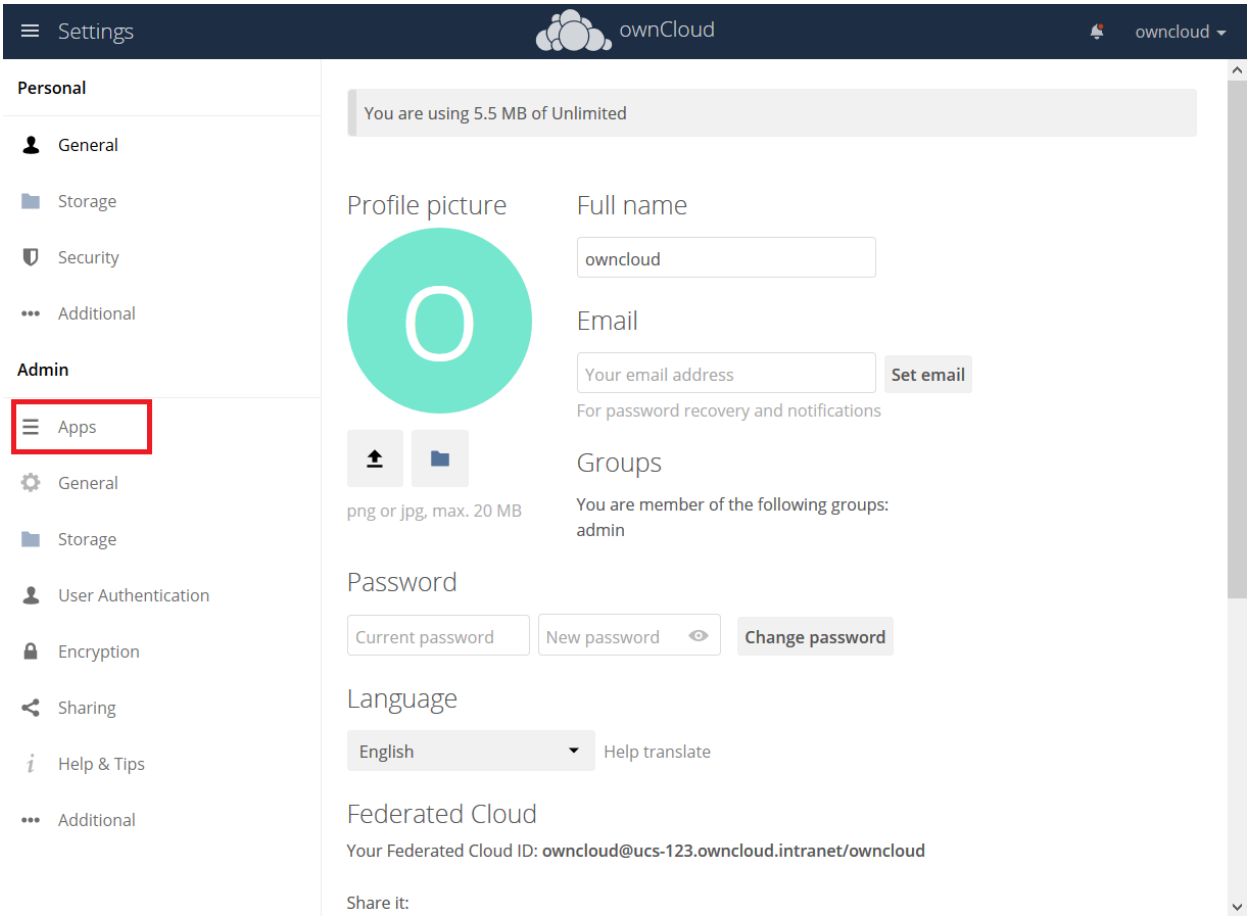

APP CENTER
univention
App Center
CLOSE
Search applications...
All
Installed
ONLYOFFICE Document Server
Ascensio System SIA
ownCloud
ownCloud GmbH
Available

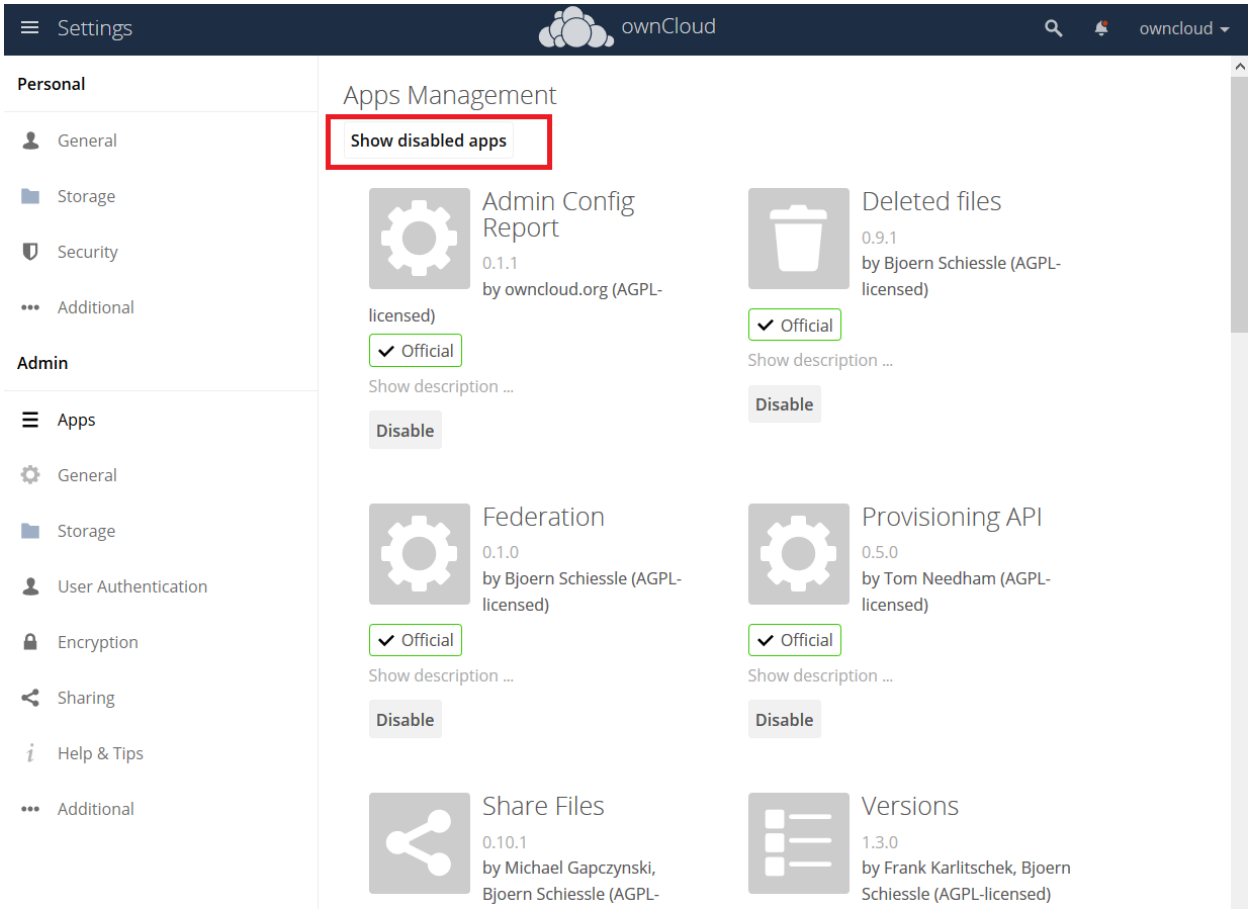


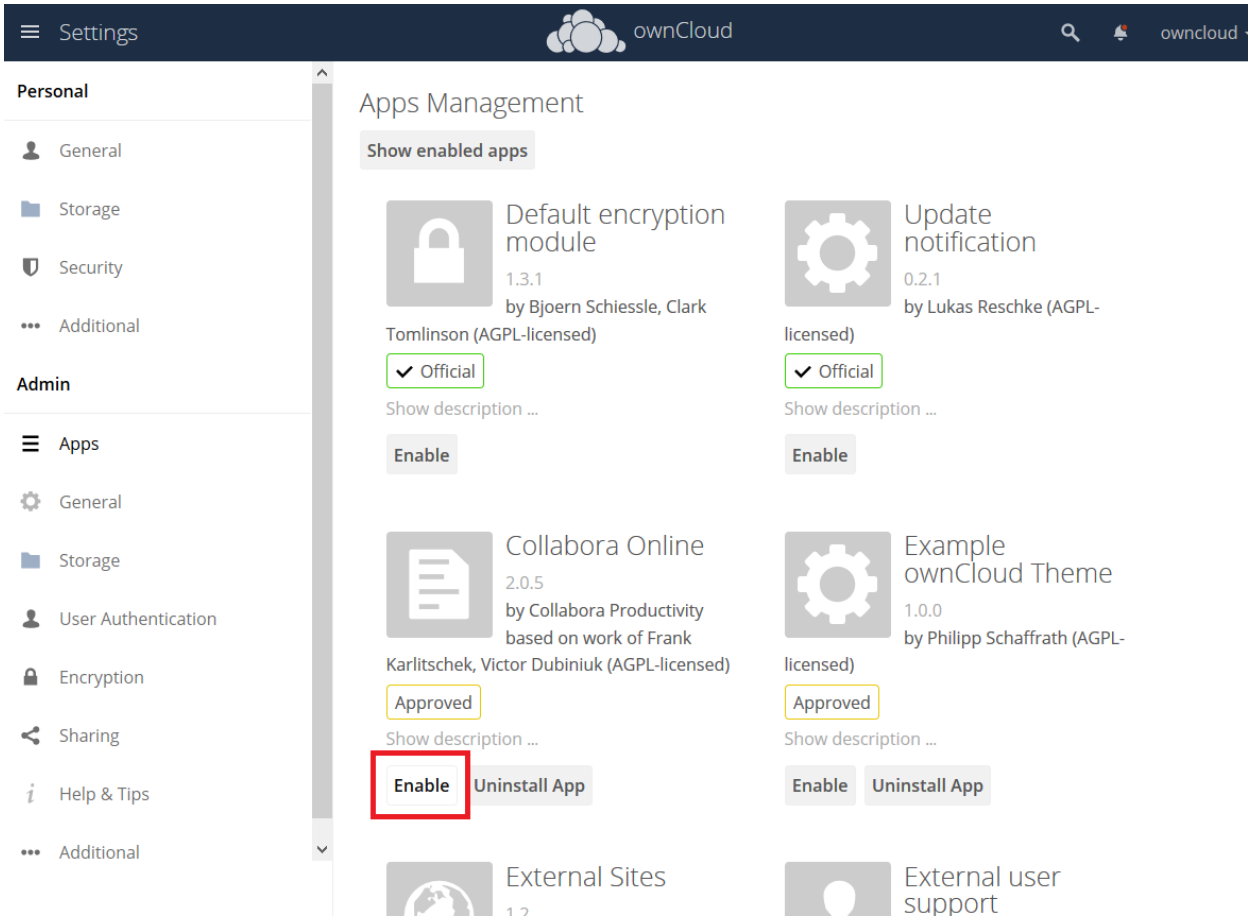


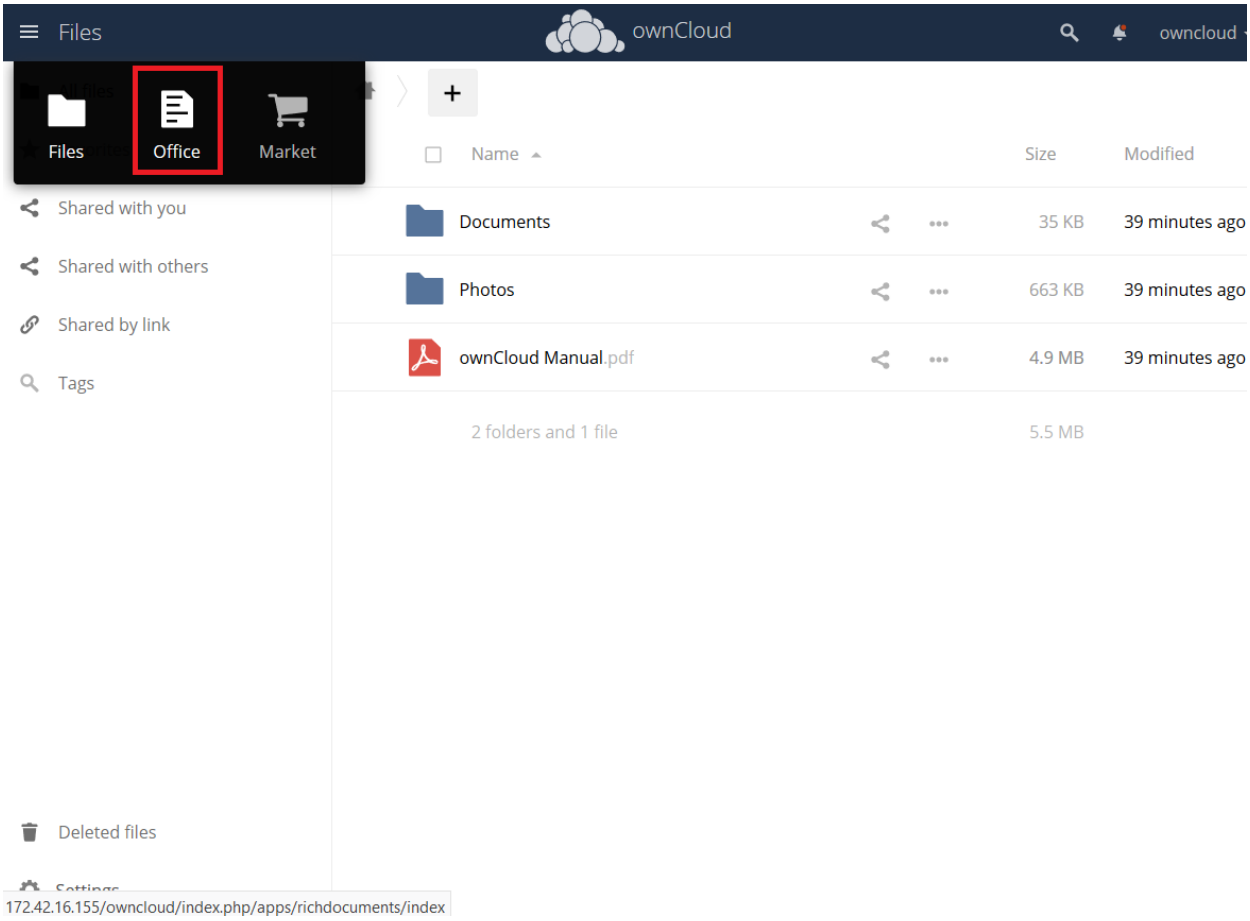
Username and Password are owncloud


Files
ownCloud
owncloud
Settings
Users
Help
Log out
All files
Favorites
Shared with you
Shared with others
Shared by link
Tags
Name
Size
Documents
35 KB
Photos
663 KB
2 hours ago
ownCloud Manual.pdf
4.9 MB
2 hours ago
2 folders and 1 file
5.5 MB
Deleted files

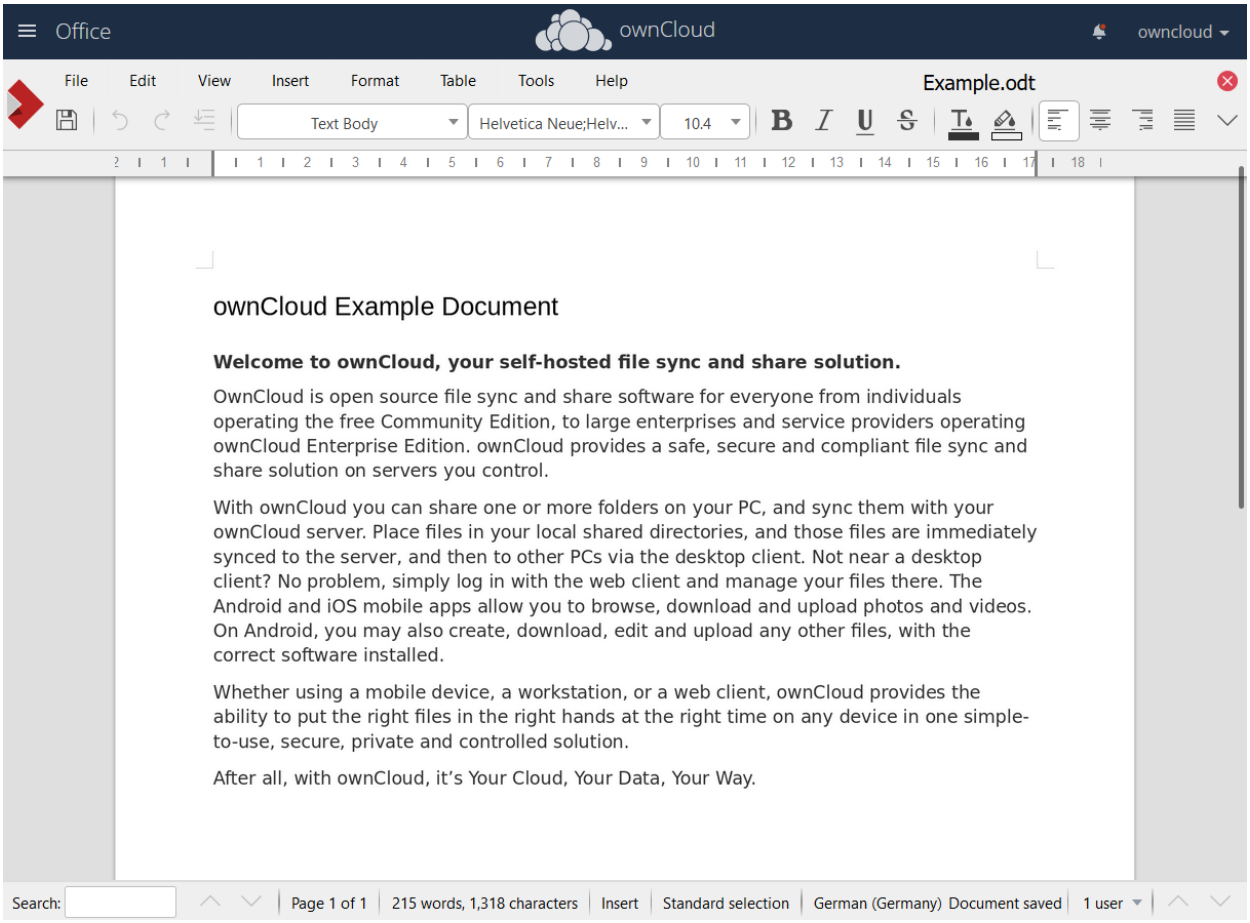

Settings
ownCloud
owncloud
Personal
General
Storage
Security
Additional
Admin
Apps
General
Storage
User Authentication
Encryption
Sharing
Help & Tips
Additional
You are using 5.5 MB of Unlimited
Profile picture
O
png or jpg, max. 20 MB
Full name
owncloud
Email
Your email address
Set email
For password recovery and notifications
Groups
You are member of the following groups:
admin
Password
Current password
New password
Change password
Language
English
Help translate
Federated Cloud
Your Federated Cloud ID: owncloud@ucs-123.owncloud.intranet/owncloud
Share it:


Settings
ownCloud
owncloud
Personal
General
Storage
Security
Additional
Admin
Apps
General
Storage
User Authentication
Encryption
Sharing
Help & Tips
Additional
Apps Management
Show disabled apps
Admin Config Report
0.1.1
by owncloud.org (AGPL-licensed)
Official
Show description ...
Disable
Deleted files
0.9.1
by Bjoern Schiessle (AGPL-licensed)
Official
Show description ...
Disable
Federation
0.1.0
by Bjoern Schiessle (AGPL-licensed)
Official
Show description ...
Disable
Provisioning API
0.5.0
by Tom Needham (AGPL-licensed)
Official
Show description ...
Disable
Share Files
0.10.1
by Michael Gapczynski, Bjoern Schiessle (AGPL-
Versions
1.3.0
by Frank Karlitschek, Bjoern Schiessle (AGPL-licensed)


Settings
ownCloud
owncloud
Personal
General
Storage
Security
Additional
Admin
Apps
General
Storage
User Authentication
Encryption
Sharing
Help & Tips
Additional
Apps Management
Show enabled apps
Default encryption module
1.3.1
by Bjoern Schiessle, Clark Tomlinson (AGPL-licensed)
Official
Show description ...
Enable
Update notification
0.2.1
by Lukas Reschke (AGPL-licensed)
Official
Show description ...
Enable
Collabora Online
2.0.5
by Collabora Productivity based on work of Frank Karlitschek, Victor Dubiniuk (AGPL-licensed)
Approved
Show description ...
Enable
Uninstall App
Example ownCloud Theme
1.0.0
by Philipp Schaffrath (AGPL-licensed)
Approved
Show description ...
Enable
Uninstall App
External Sites
External user support


Files
ownCloud
owncloud
Files
Office
Market
Shared with you
Shared with others
Shared by link
Tags
Name
Size
Modified
Documents
35 KB
39 minutes ago
Photos
663 KB
39 minutes ago
ownCloud Manual.pdf
4.9 MB
39 minutes ago
2 folders and 1 file
5.5 MB
Deleted files
172.42.16.155/owncloud/index.php/apps/richdocuments/index

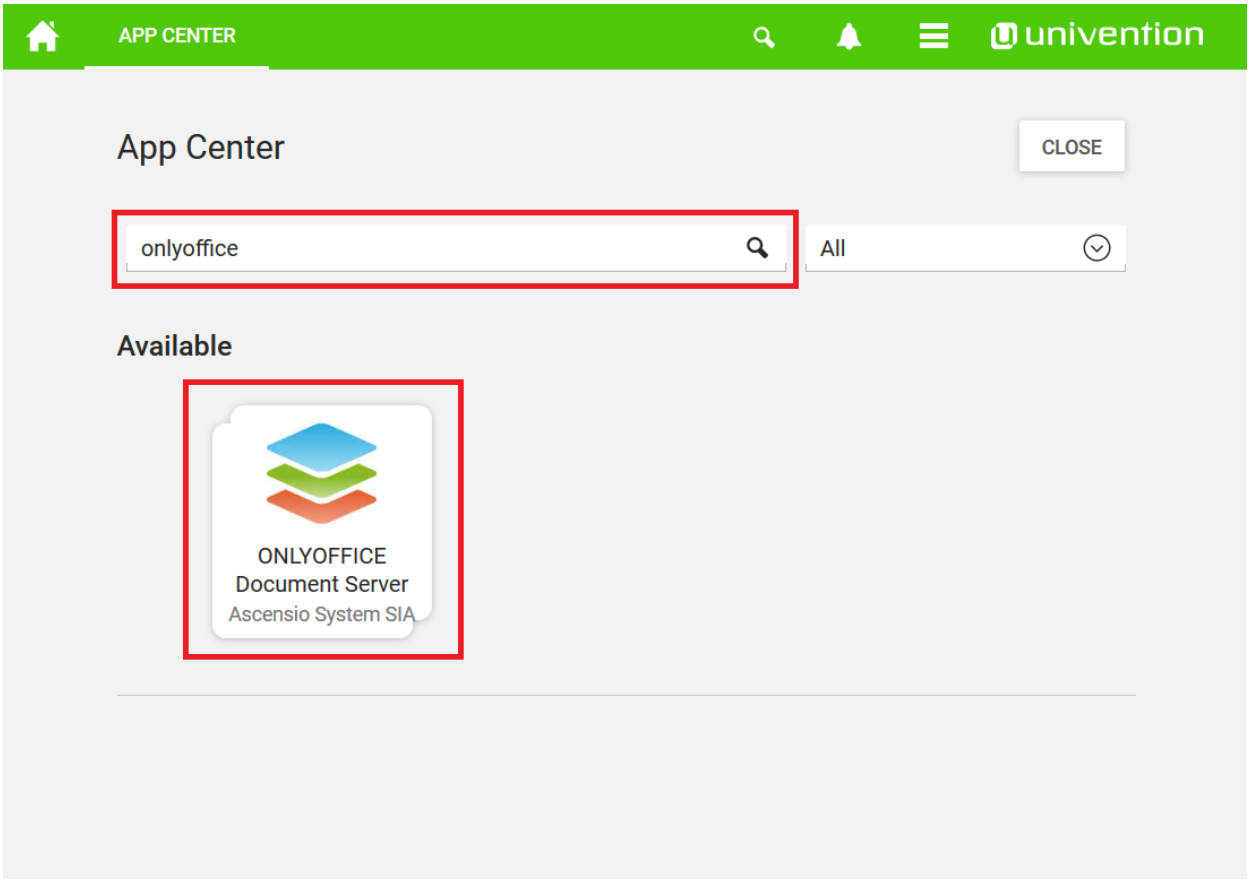

Office
ownCloud
owncloud
New Document
New Spreadsheet
New Presentation
Upload
Example.odt



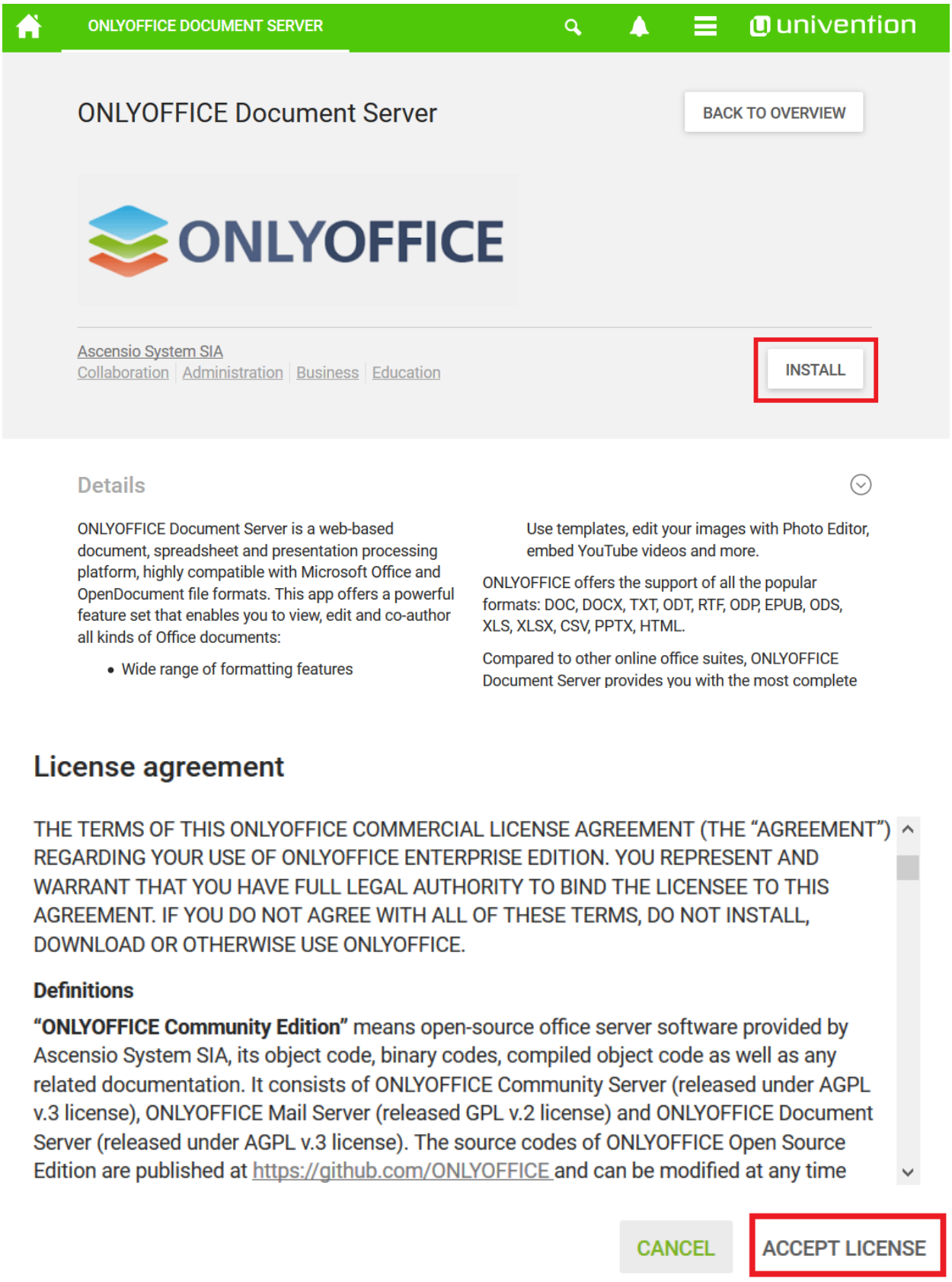
Now you can use Collabora within ownCloud. Start by creating a new Document.

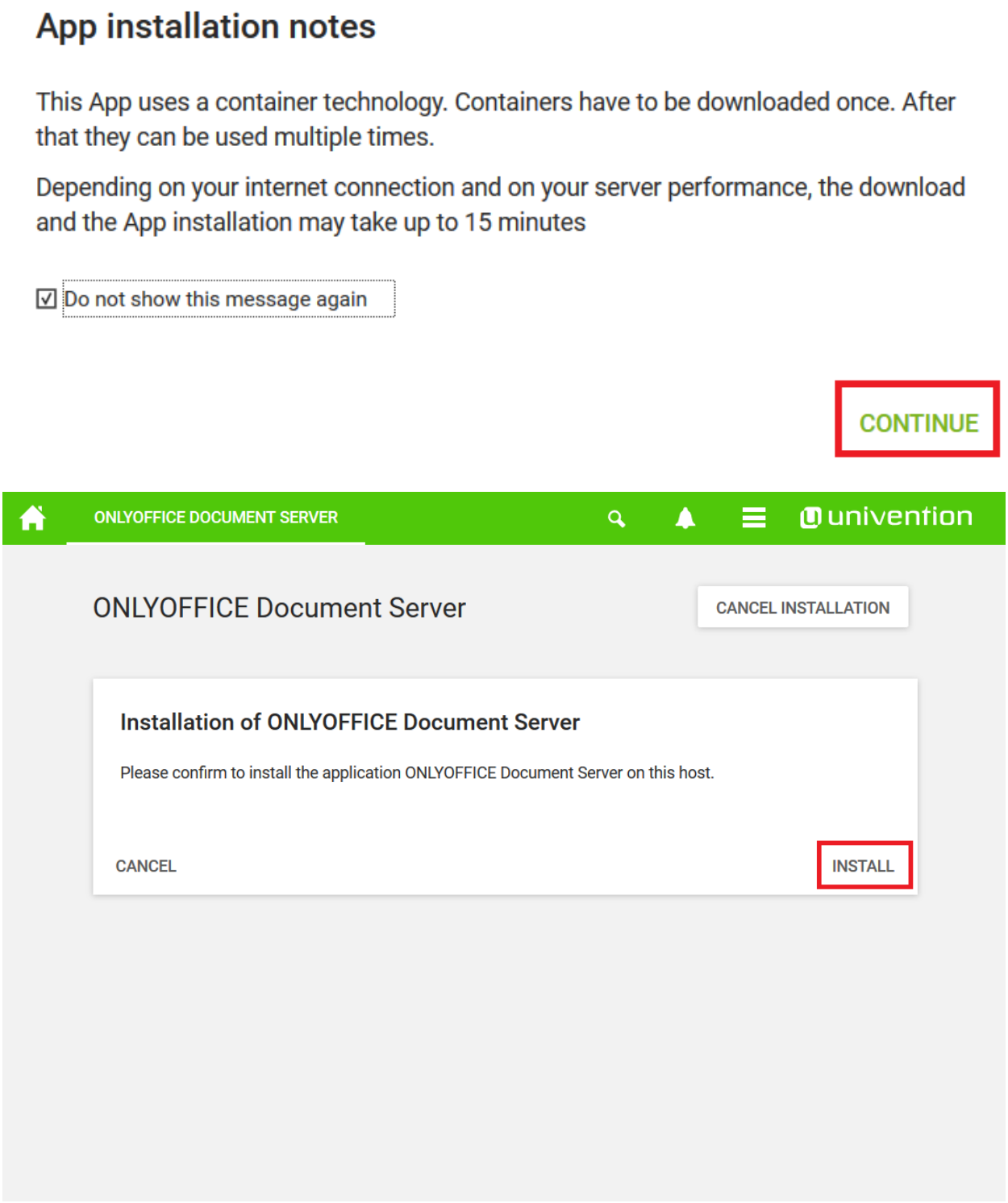
How to Install OnlyOffice

- Search for "**OnlyOffice**" or select it from the application list in the Appcenter.

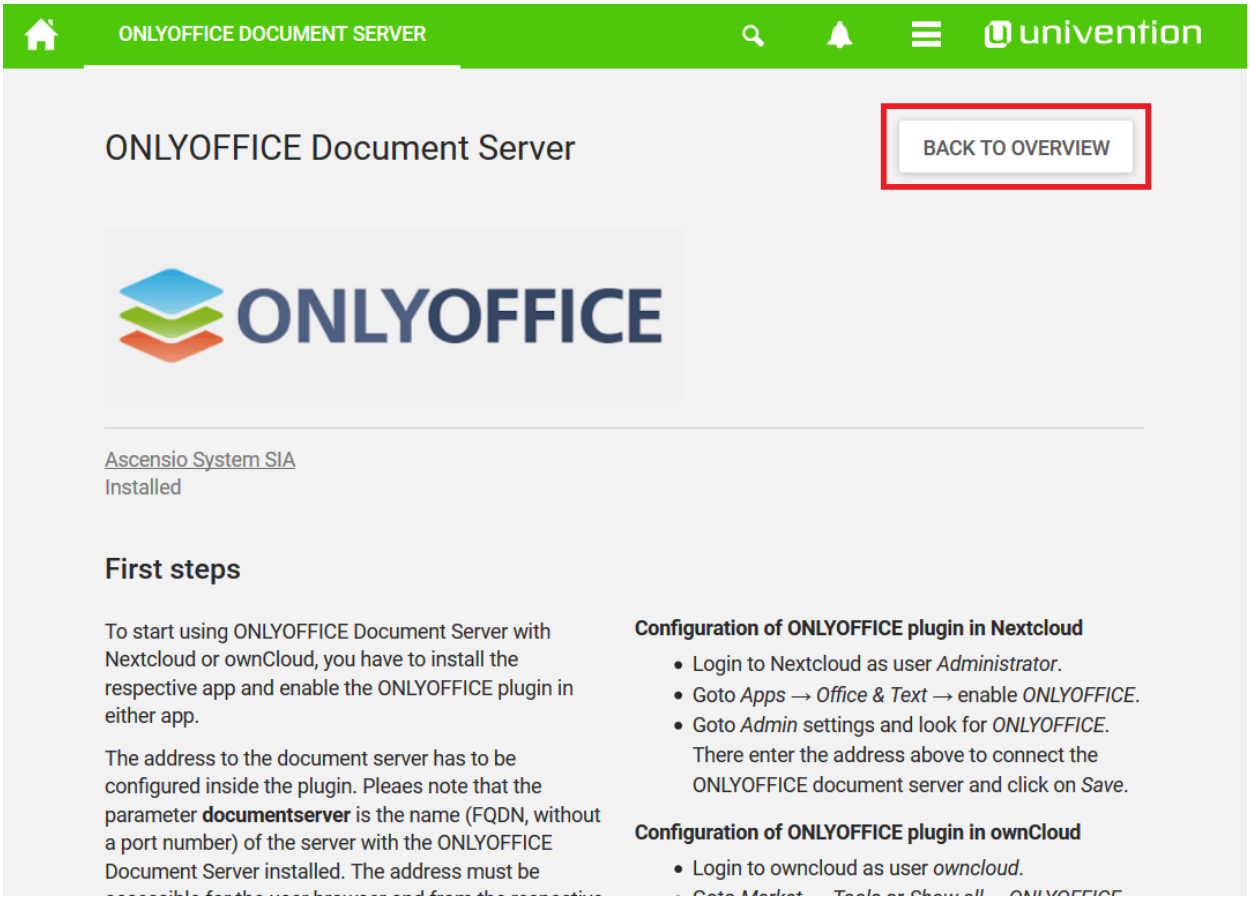


- Install OnlyOffice.

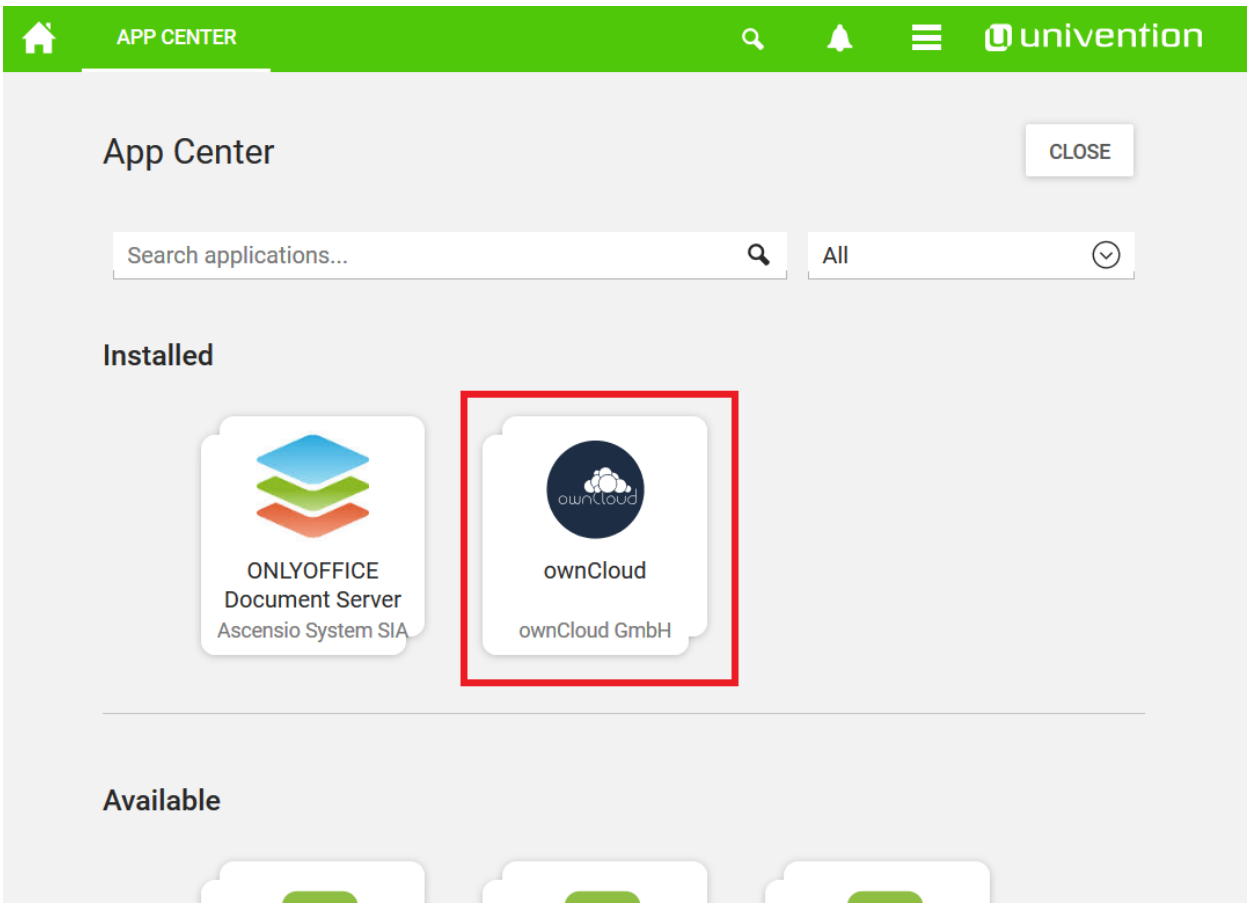


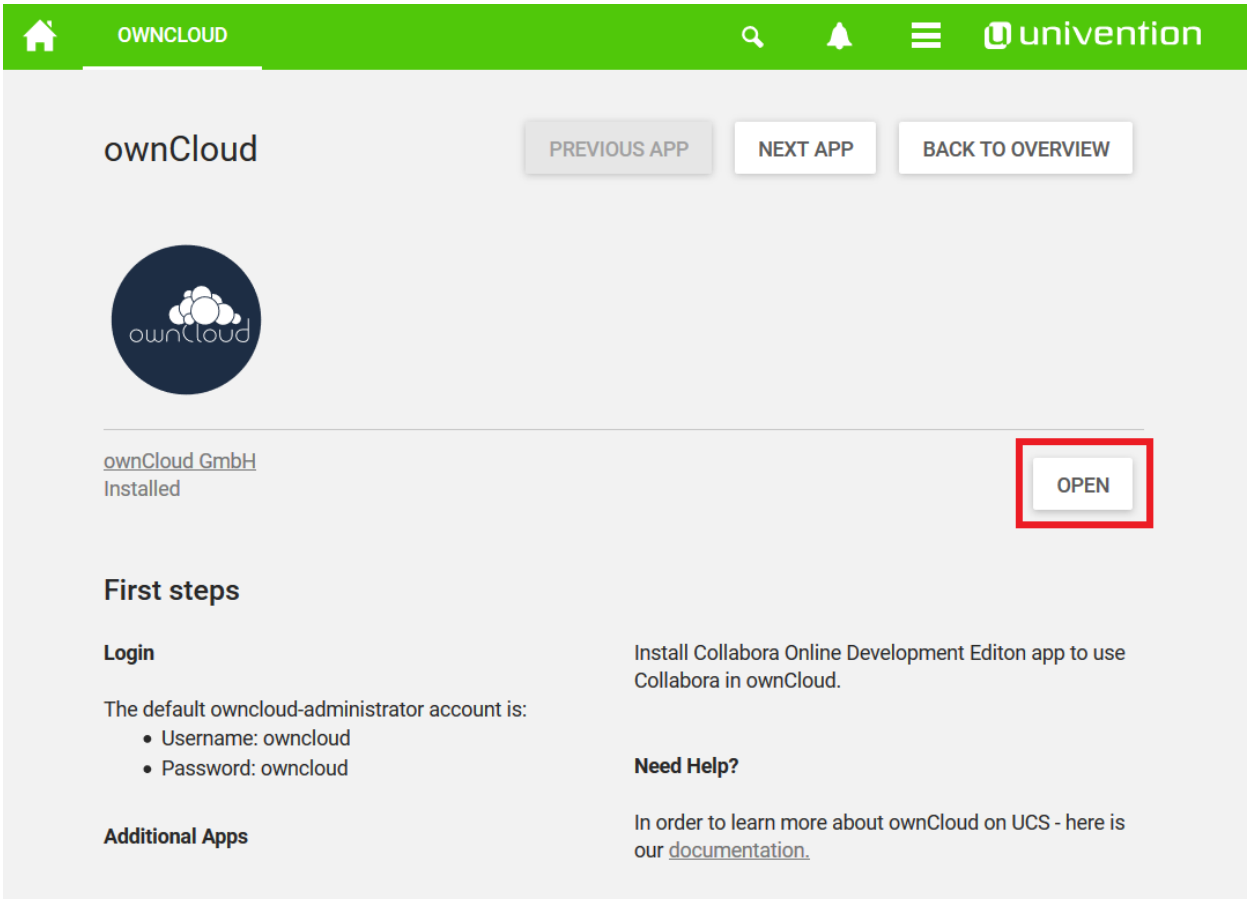


After the installation is complete, return to the Appcenter overview



- Install the ownCloud OnlyOffice connector App
  - Go to ownCloud

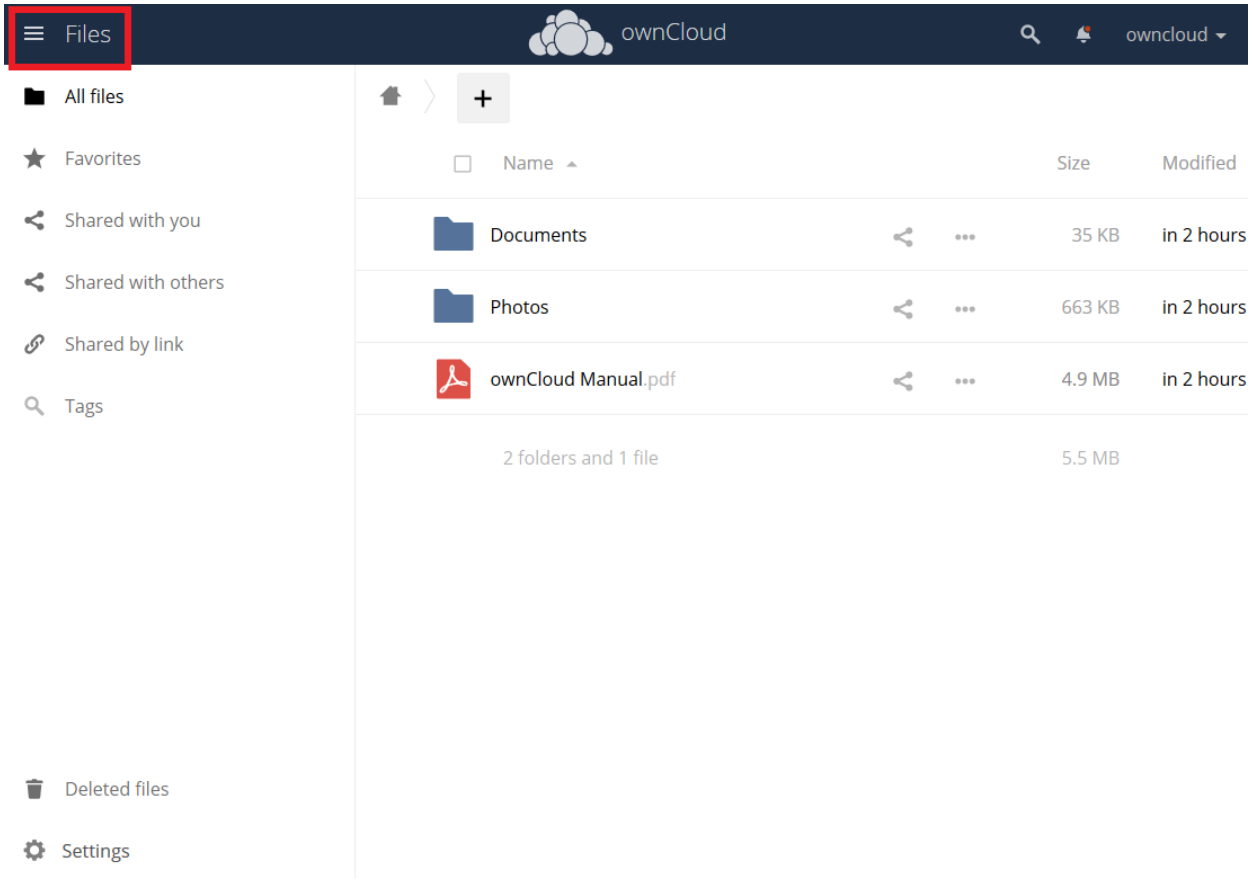


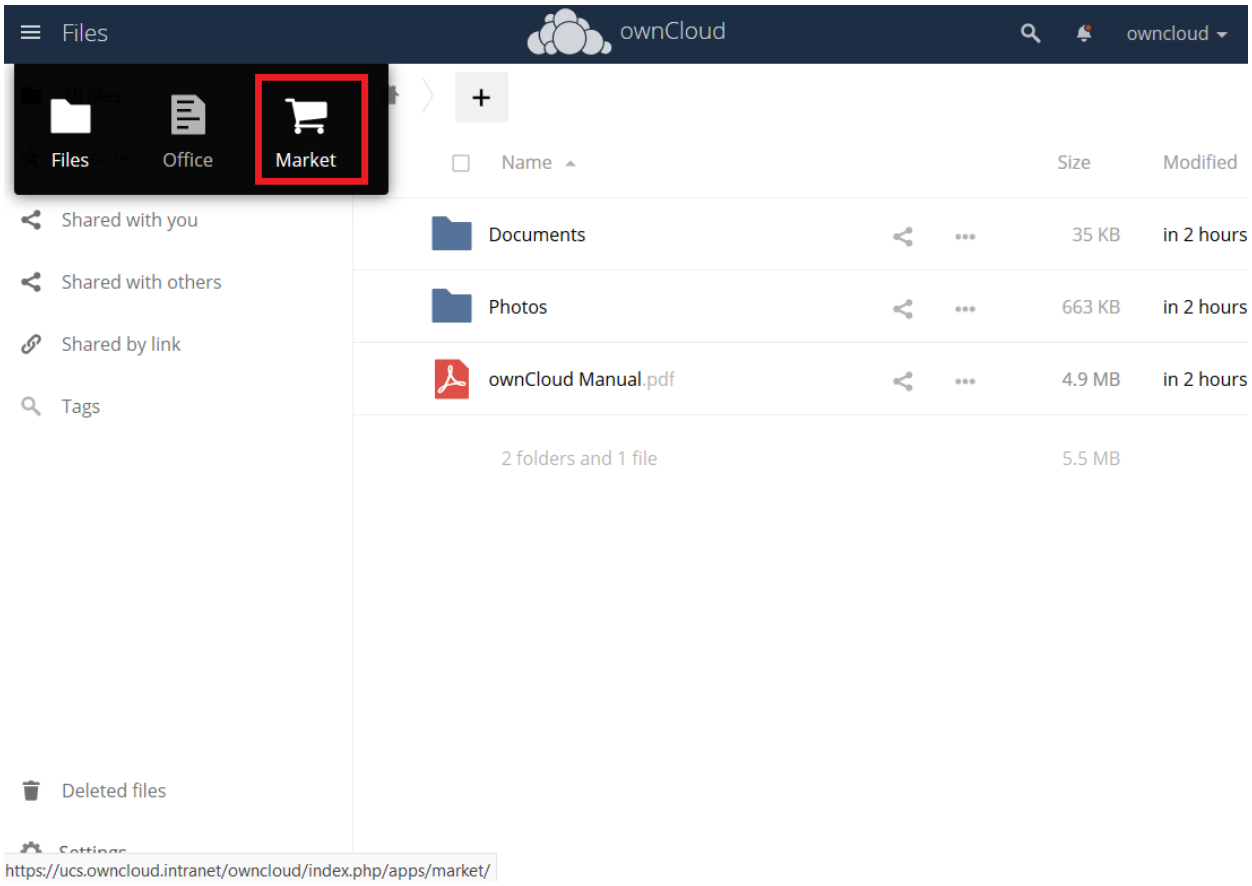




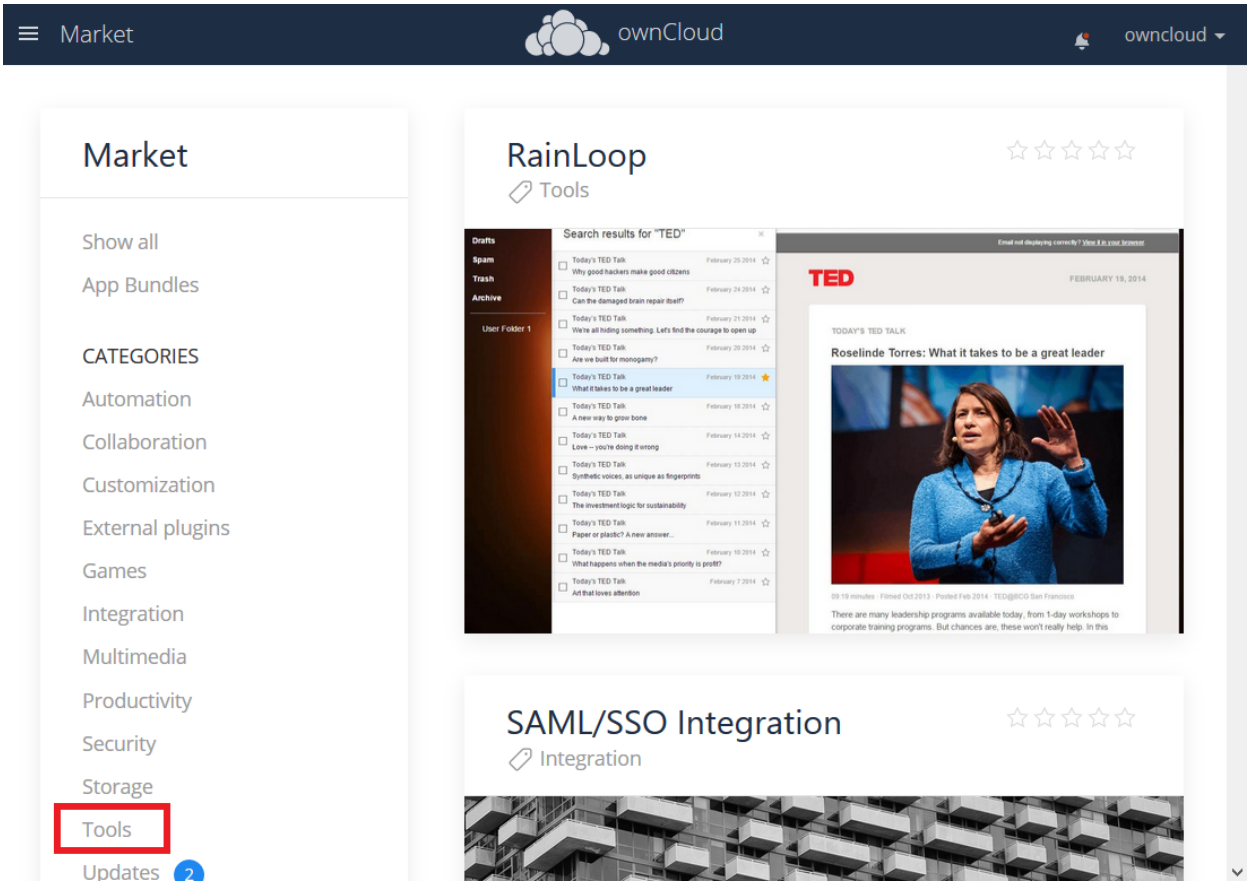
Username and Password are owncloud

- Market

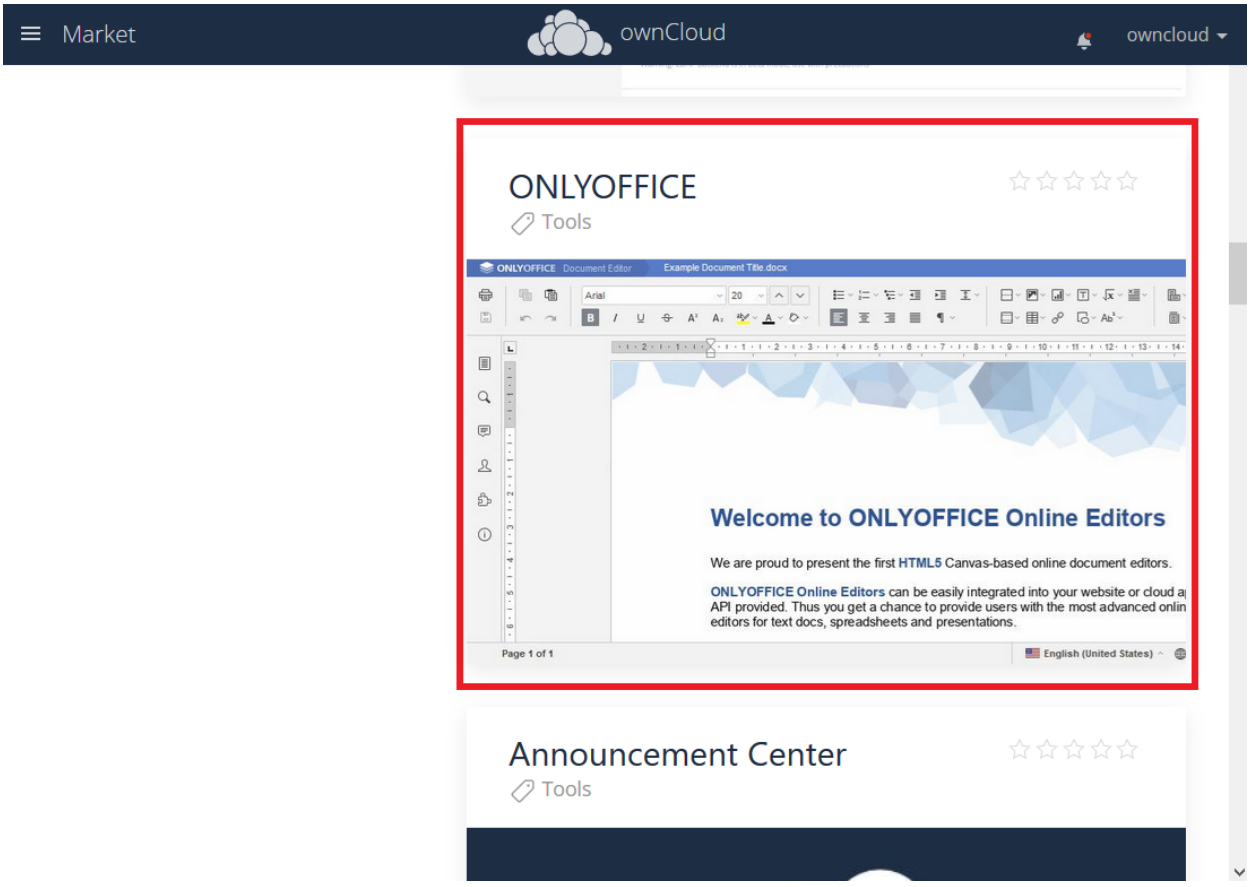

Files
ownCloud
owncloud
All files
Favorites
Shared with you
Shared with others
Shared by link
Tags
Deleted files
Settings
Name
Size
Modified
Documents
35 KB
in 2 hours
Photos
663 KB
in 2 hours
ownCloud Manual.pdf
4.9 MB
in 2 hours
2 folders and 1 file
5.5 MB

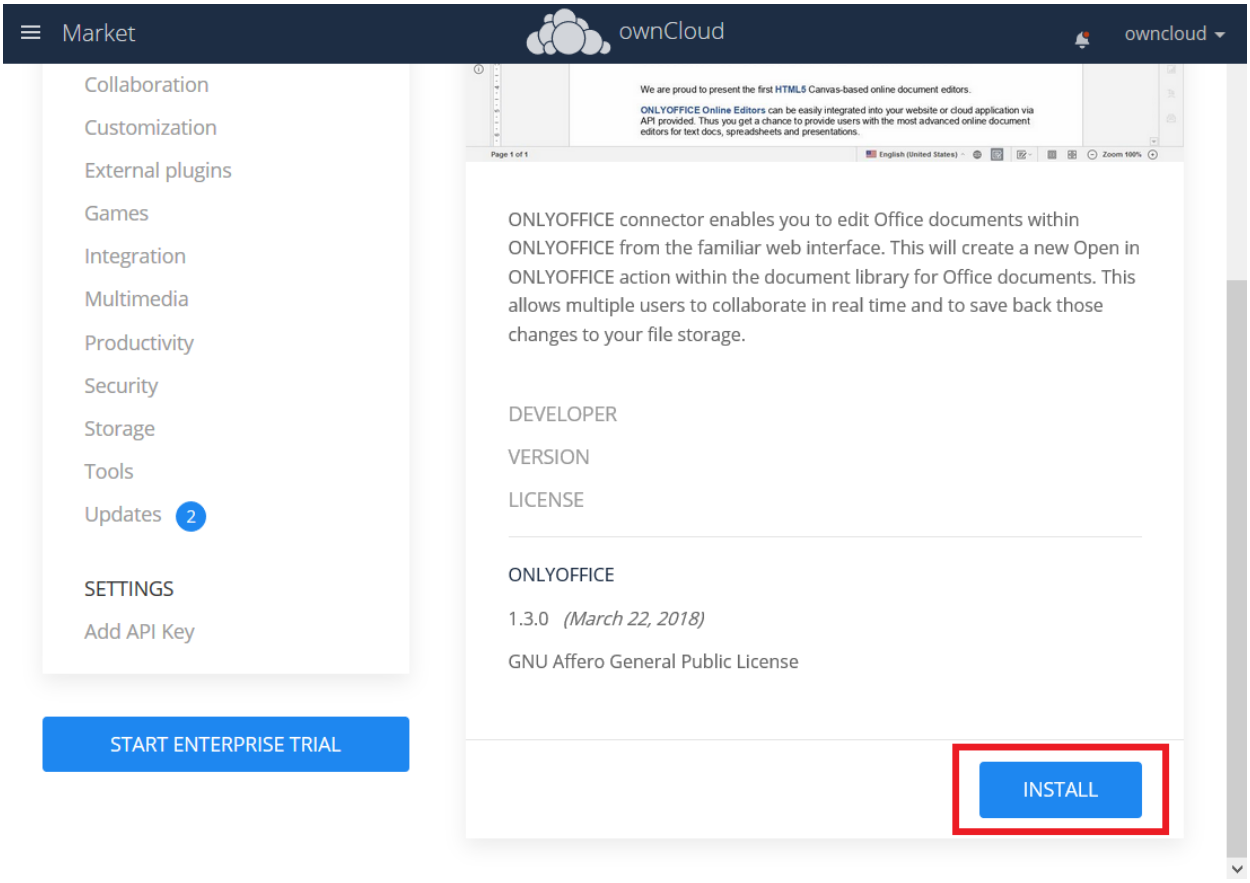

Files
ownCloud
owncloud
Files
Office
Market
Shared with you
Shared with others
Shared by link
Tags
Deleted files
Name
Size
Modified
Documents
35 KB
in 2 hours
Photos
663 KB
in 2 hours
ownCloud Manual.pdf
4.9 MB
in 2 hours
2 folders and 1 file
5.5 MB
https://ucs.owncloud.intranet/owncloud/index.php/apps/market/

- Tools

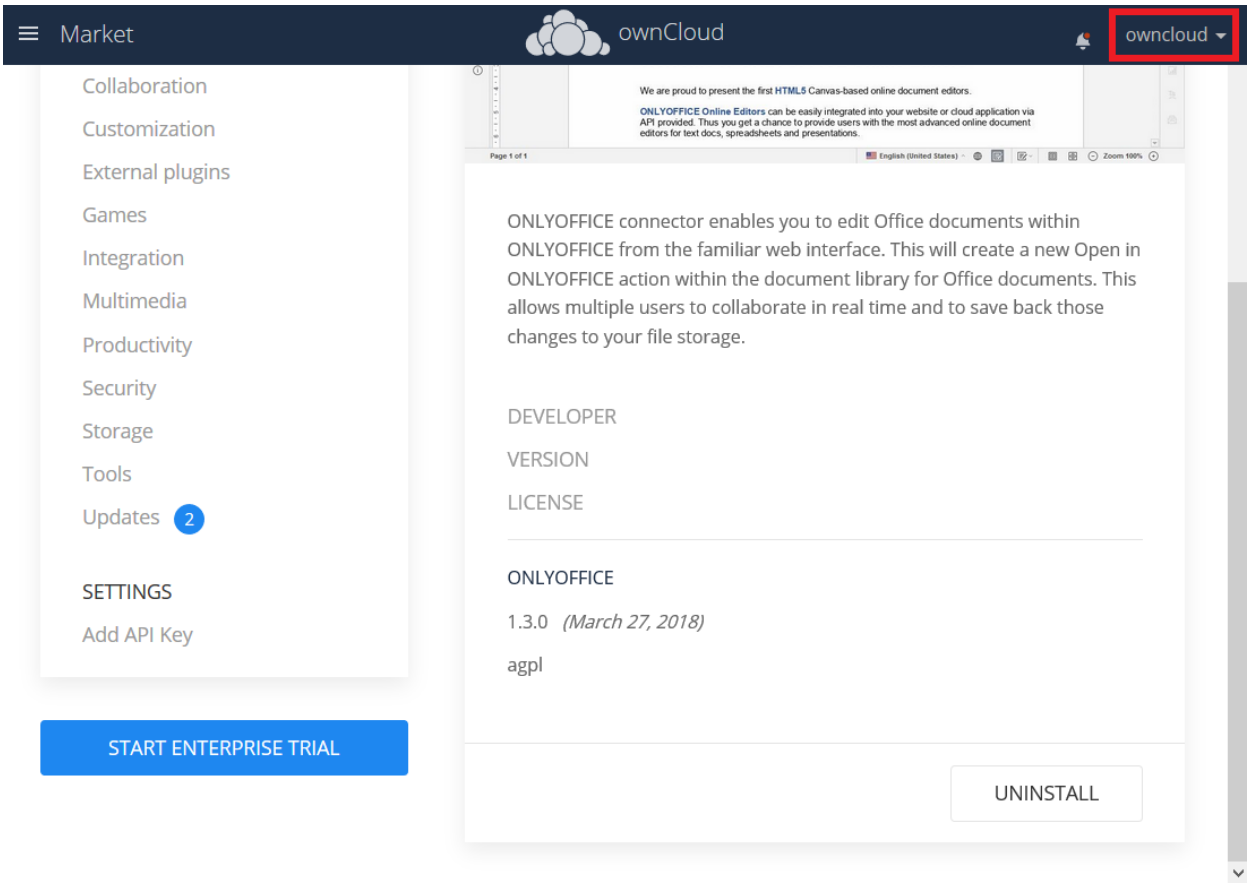


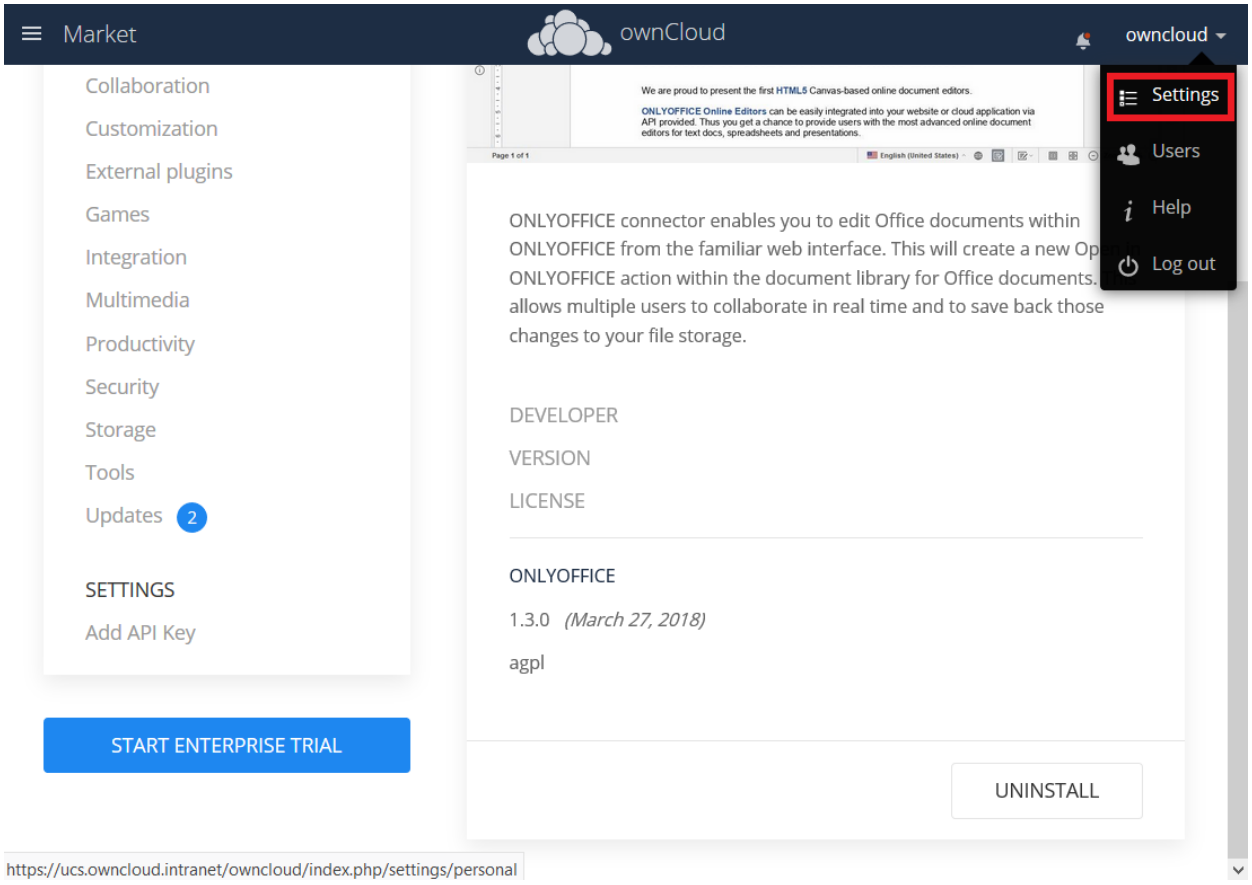
- Install OnlyOffice

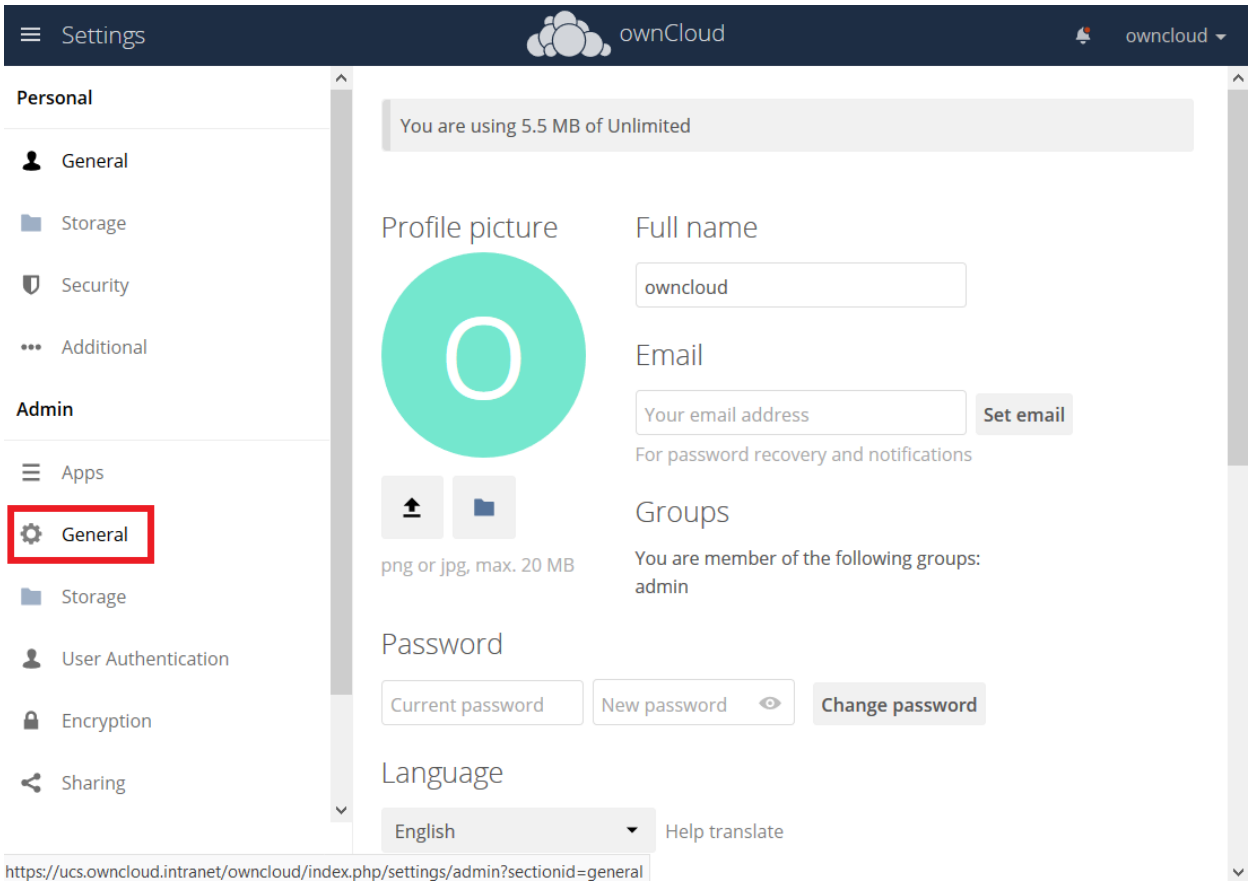




- Go to the OnlyOffice settings inside ownCloud.

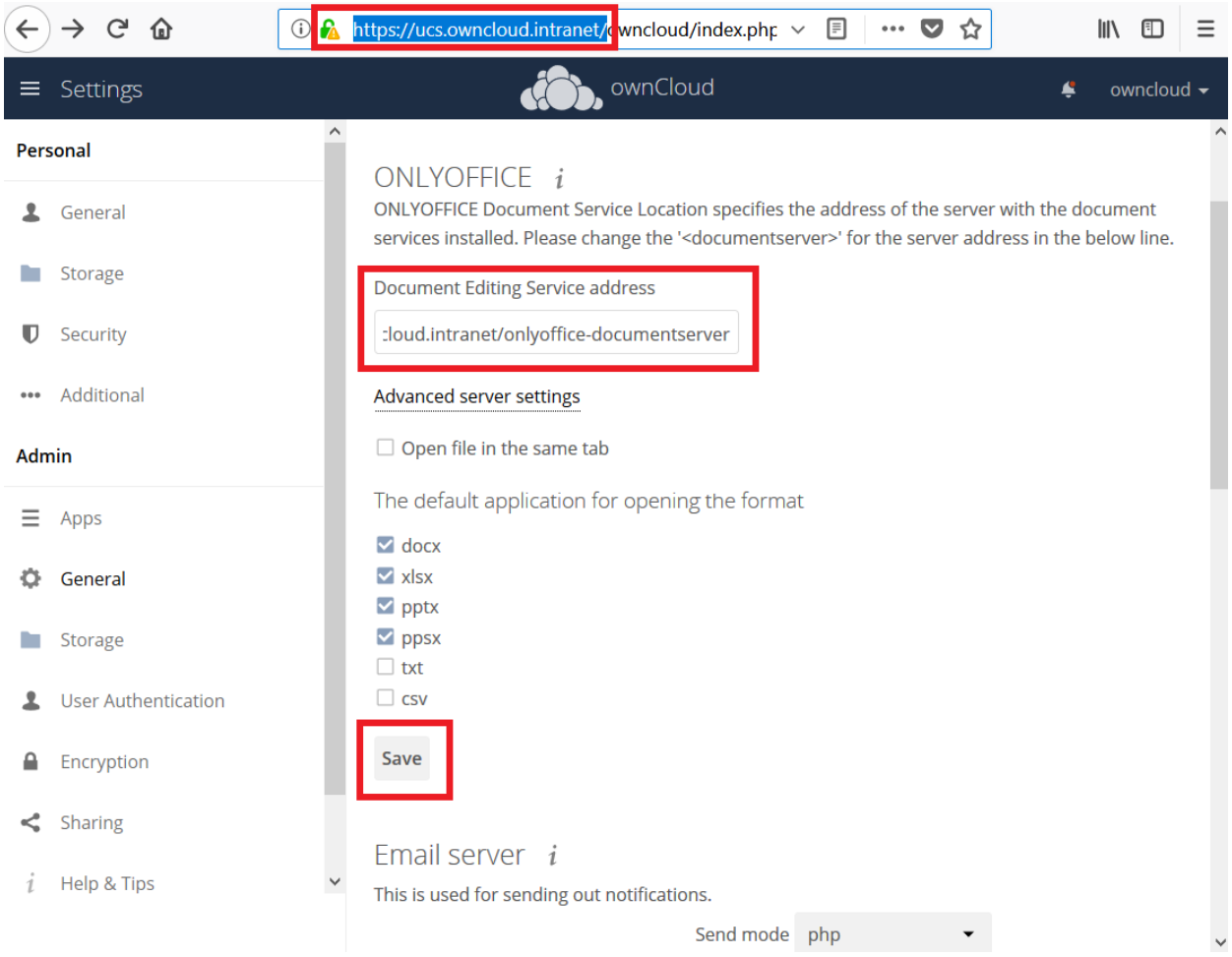




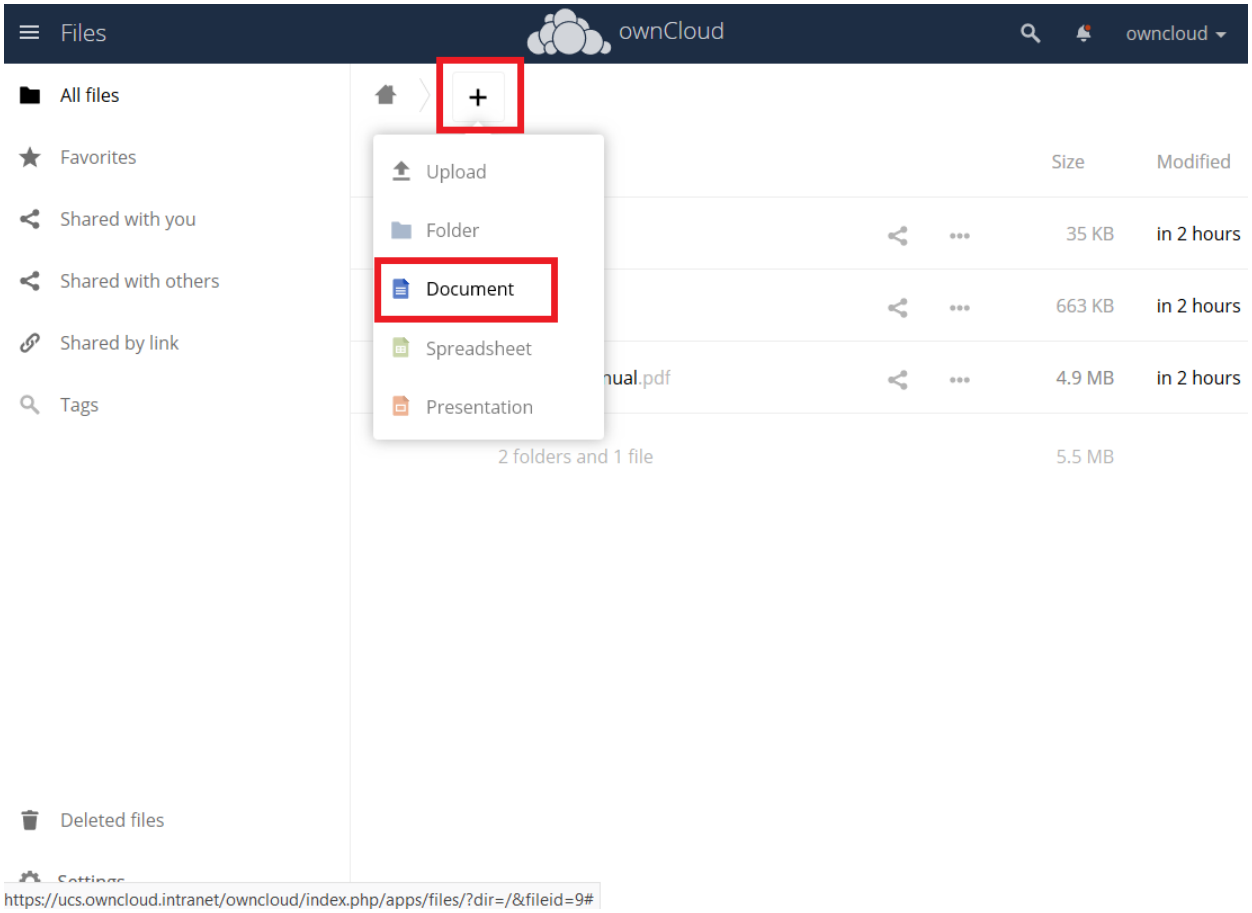


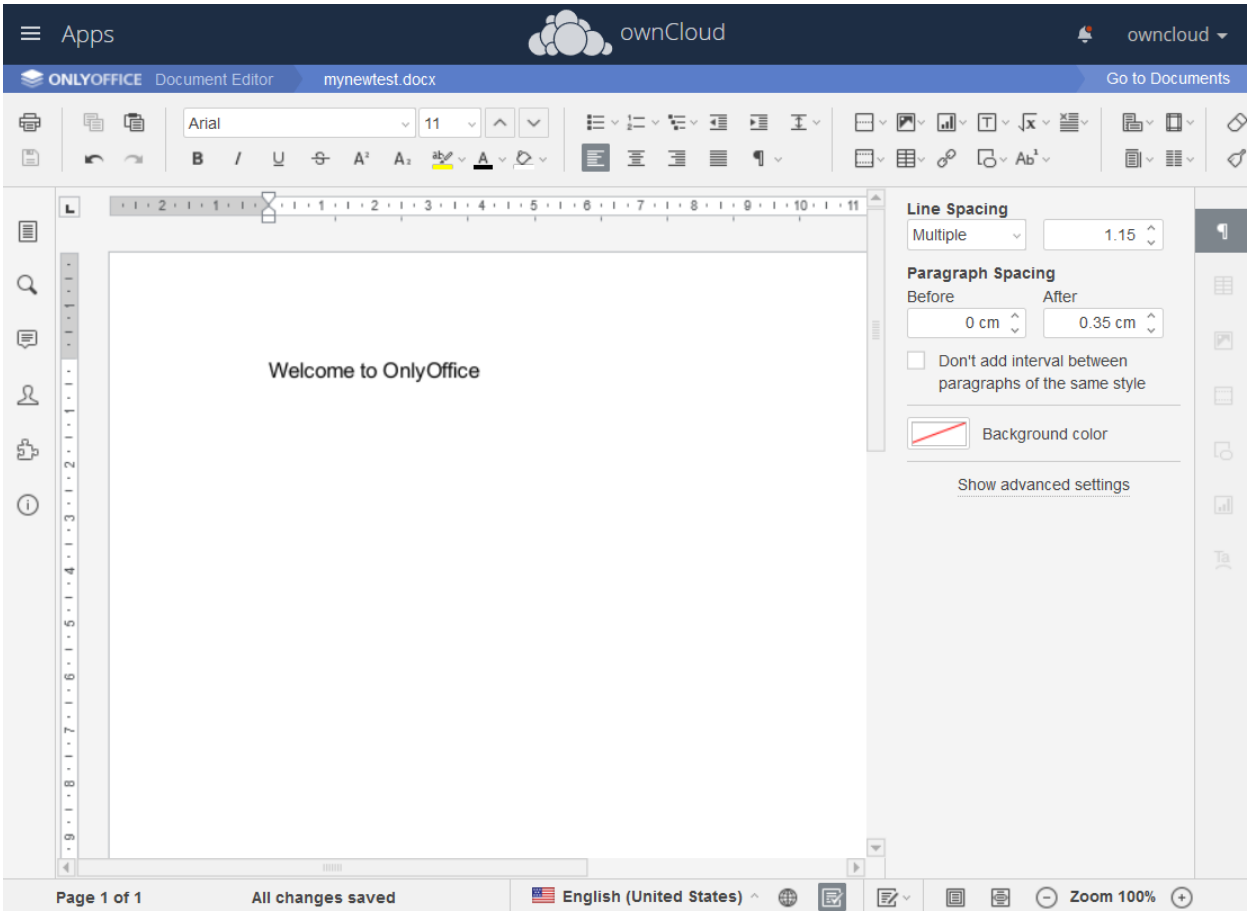
- Enter the OnlyOffice server address in the following format and **save** it:

```
https://<your-domain-name>/onlyoffice-documentserver/
```

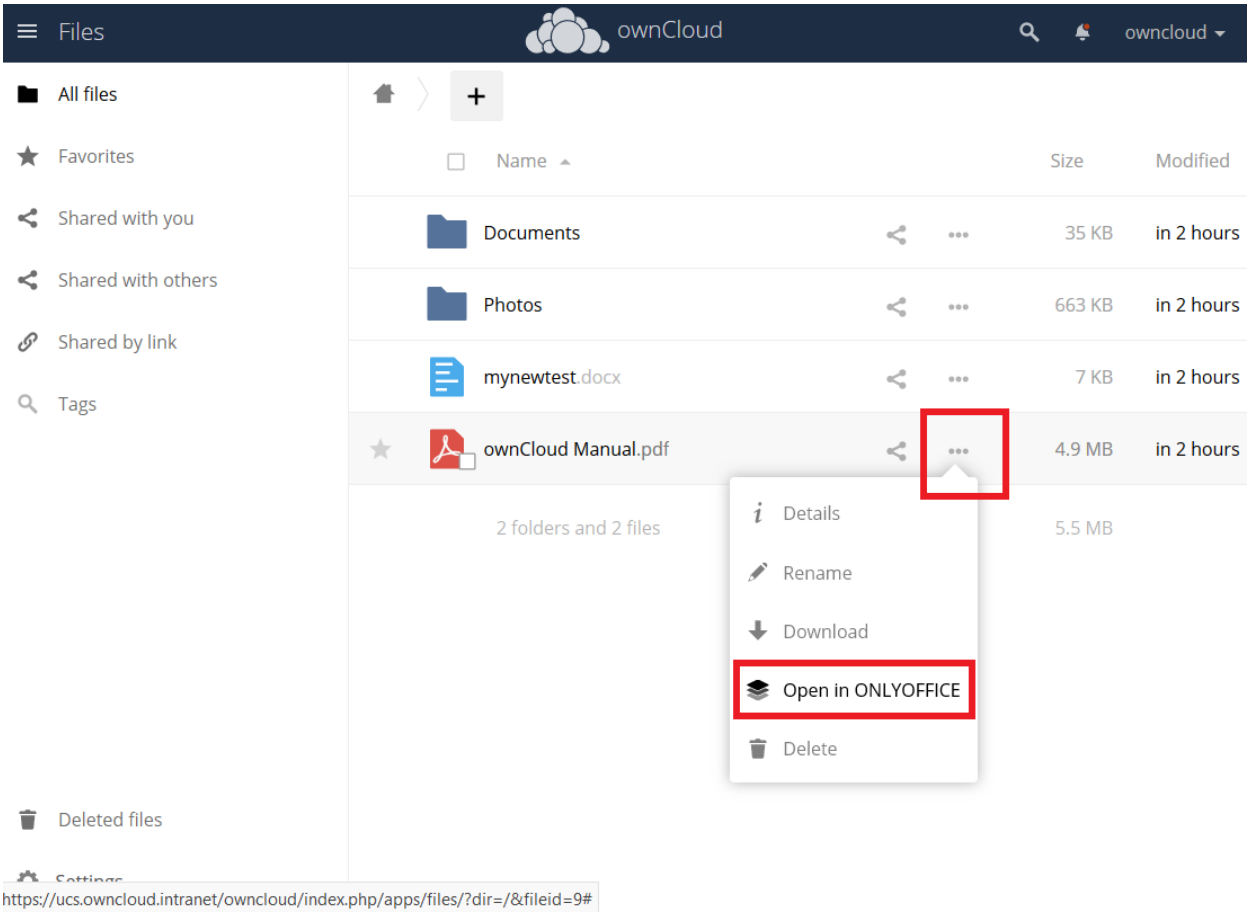


- Now you can create a new document by clicking on the btn:[Plus] button.





PDF documents can also be viewed in OnlyOffice



Updating

When a new App release is available you should update the Office App. Here are the required steps:

- Select **Software update**
- Check if an Update is available
- Select on the App name
- Upgrade the App

Windows Network Drive Configuration (WND)

Introduction

Here are the steps to configure WND in the Appliance.

	Windows Network Drive is avaialbe only in the Enterprise Editon of ownCloud.
	You will need both of the following described steps for each share.
	The steps need to be done in / on the docker host (appliance virtual machine) and not inside the docker container

WND Listener

Create a service following the instructions below that checks the share for changes:

- For each WND mount point distinguished by a SERVER - SHARE pair,
  - place one copy of a file with following content under /etc/systemd/system/owncloud-wnd-listen-SERVER-SHARE.service
  - replacing the all upper case words SERVER, SHARE, USER and PASSWORD
  - in both, the **filename** and in the **contents** below with their respective values. Take care to also adjust the paths in WorkingDirectory and ExecStart according to your installation.

```
[Unit]
Description=ownCloud WND Listener for SERVER SHARE
After=docker-app-owncloud.service
Requires=docker-app-owncloud.service
[Service]
User=root
Group=root
WorkingDirectory=/root
ExecStart=/usr/bin/univention-app shell owncloud occ wnd:listen -vvv SERVER
SHARE USER PASSWORD
Type=simple
StandardOutput=journal
StandardError=journal
SyslogIdentifier=%n
KillMode=process
RestartSec=1
Restart=always
```

- Run once for each created file the following commands:

```
sudo systemctl enable owncloud-wnd-listen-SERVER-SHARE.service
sudo systemctl start  owncloud-wnd-listen-SERVER-SHARE.service
```

WND Process Queue

Create or add a crontab file in /etc/cron.d/oc-wnd-process-queue.

- Make a crontab entry to run a script iterating over all SERVER SHARE pairs with an appropriate occ wnd:process-queue command. The commands must be **strictly sequential**. This can be done by using flock -n and tuning the -c parameter of occ wnd:process-queue

```
0 */15 * * *    root  /usr/bin/univention-app shell owncloud occ wnd:process-queue
-vvv SERVER SHARE
```

Further Reading

Please see also:

- The ownCloud forum and the
- Windows Network Drive Configuration documentation.

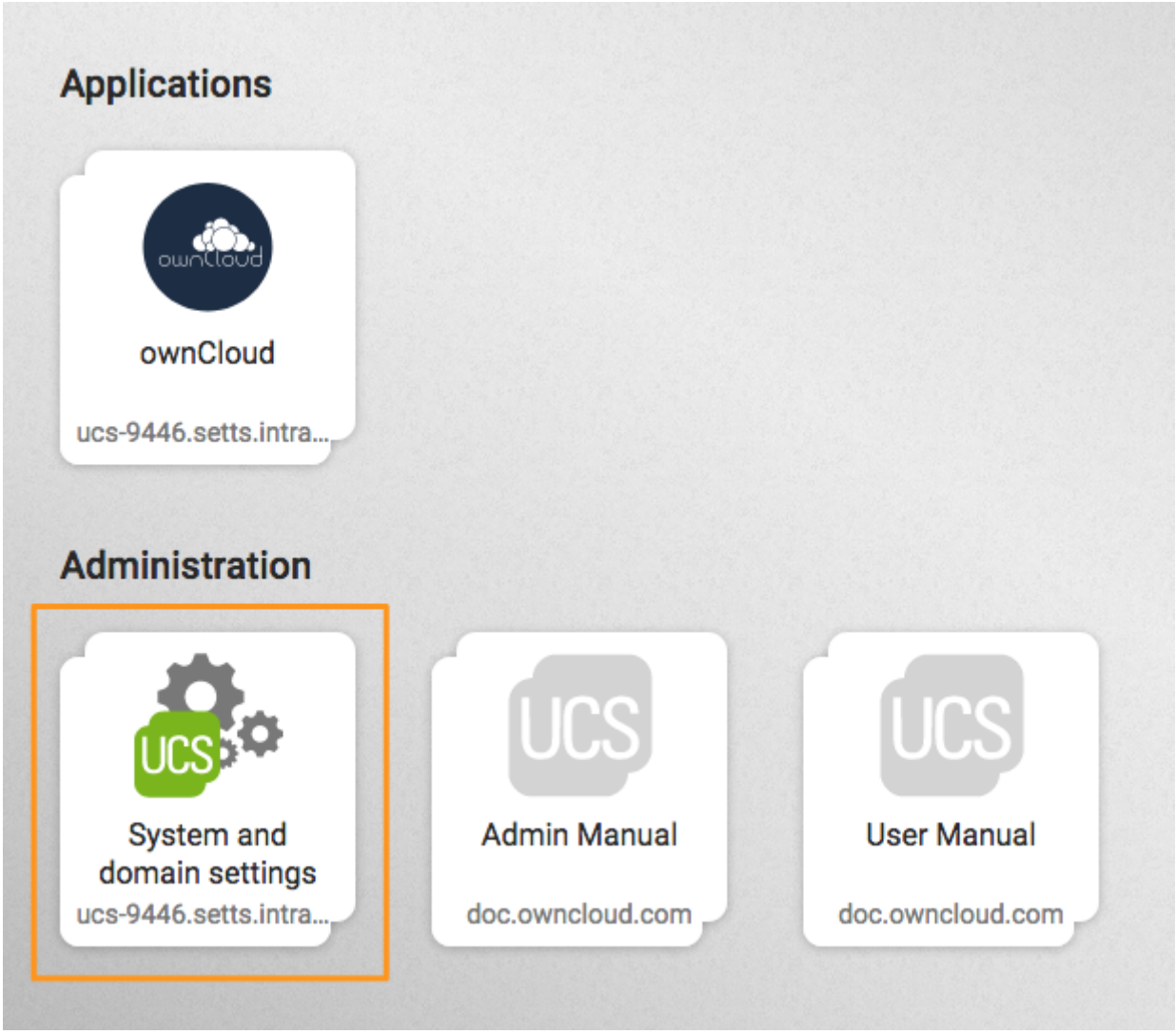
Install Antivirus Software in the ownCloud Appliance

Introduction

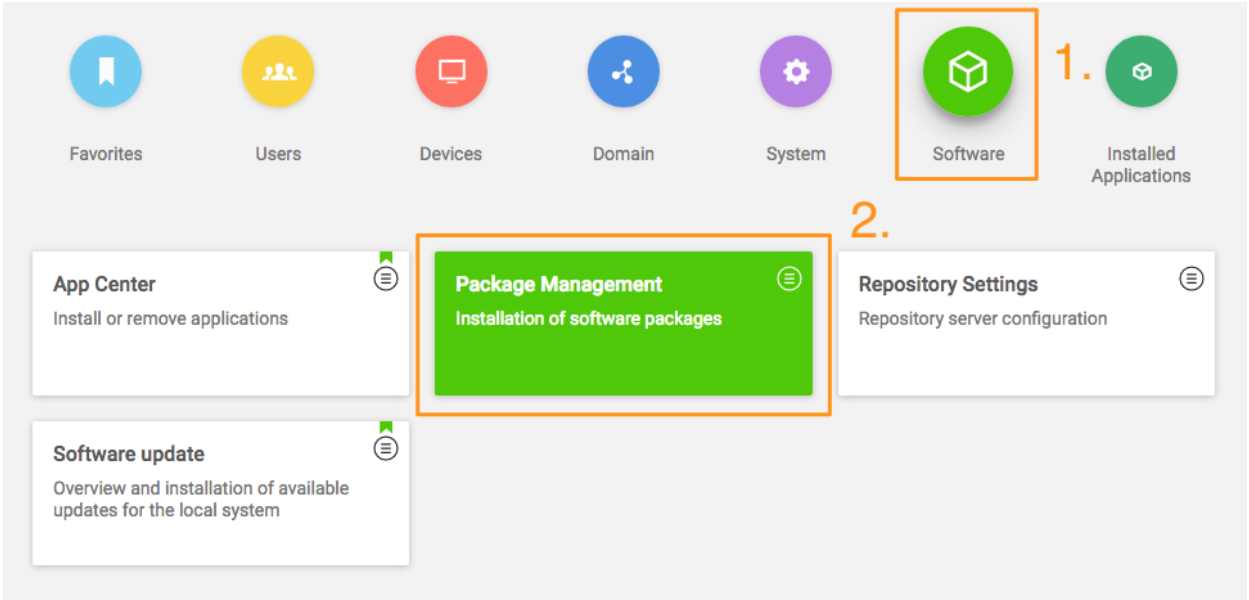
This guide details how to enable a virus scanner in the ownCloud Appliance.

Install ClamAV and Related Components

First, start the appliance and go to "**System and domain settings**".



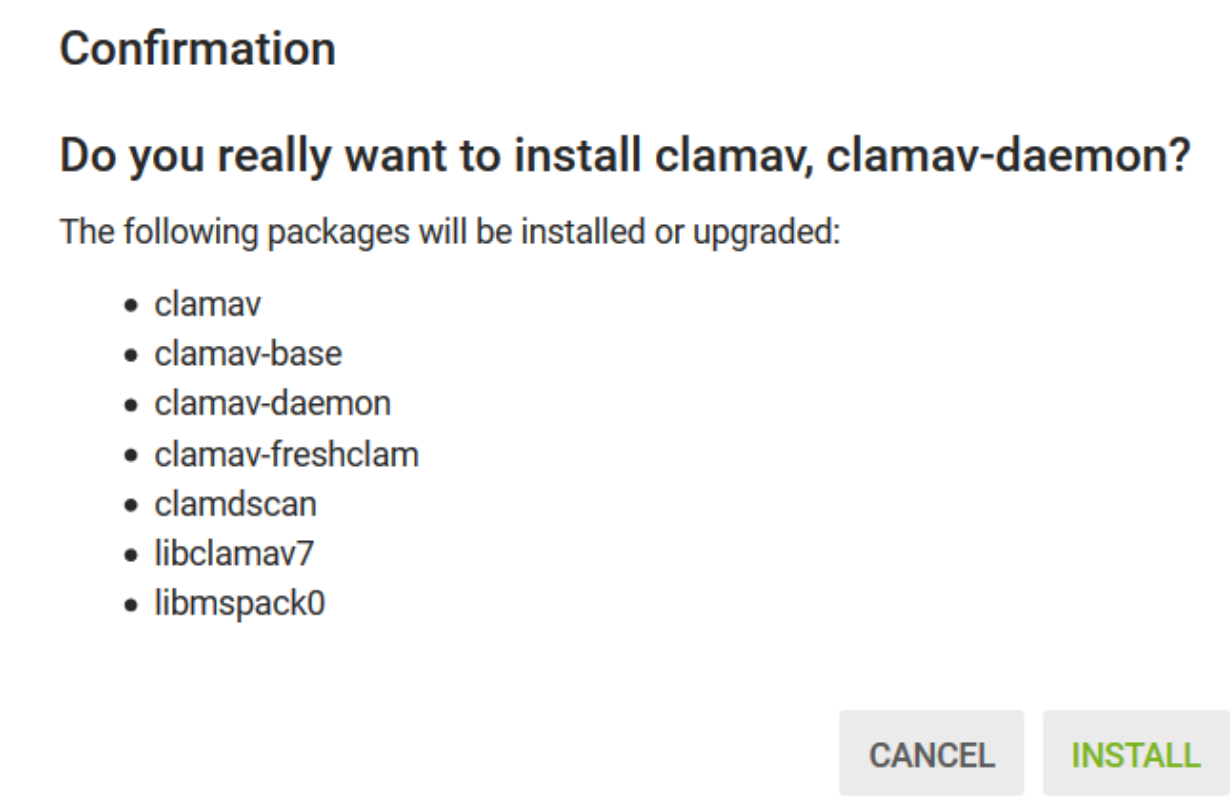
When there, log in with the administrator account. After you have done that, click btn:[Software] and open "**Package Management**", as in the screenshot below.



From there, you first need to install ClamAV. To do this, in the third field, next to the one containing the text "**Package name**", type in the phrase: "**clamav**" (1). Doing so filters the list of packages to only those matching that phrase. In the filtered list of packages, check the checkboxes next to "**clamav**" (2), "**clamav-freshclam**", and "**clamav-daemon**".

After doing that, click btn:[INSTALL] (3) above the listed packages, next to "**SHOW DETAILS**'".

After you do so, a confirmation dialog appears, as in the screenshot below, asking for confirmation to install the packages. Confirm the choice by again clicking btn:[INSTALL].



The installation should only take a few minutes.

Configure ownCloud to Use ClamAV

Start the ClamAV service:

```
systemctl enable clamav-daemon.service
systemctl start clamav-daemon.service
```

Next you need to configure ClamAV in your ownCloud instance. Please refer to the ClamAV documentation for instructions on how to do that.

Troubleshooting

"" If you try to update the ClamAV virus database manually, by entering freshclam, and see the error below, it means that freshclam is already updating the database. ""

```
ERROR: /var/log/clamav/freshclam.log is locked by another process
ERROR: Problem with internal logger (UpdateLogFile =
/var/log/clamav/freshclam.log).
```

Updates are run based on the configured time interval in the applicable Cron job. In

the example below, the update would run every 47 minutes:

```
# m   h  dom mon dow  command
47  *  *   *   *  /usr/bin/freshclam --quiet
```

If there are errors running the freshclam process, check if a process is blocking the log file, by running the following command:

```
lsof /var/log/clamav/freshclam.log
```

If you want to refresh the ClamAV database manually, follow these steps:

```
# Gently end the freshclam process with this command:
sudo pkill -15 -x freshclam

# Start the refresh process again with this command:
sudo freshclam
```



When the app is enabled — but is not configured or has an incorrect configuration — it will reject **all** uploads for the entire instance. To avoid this situation, make sure the ClamAV service is running and you have the execution mode correctly configured in ownCloud.

Configure index.php-less URLs

Introduction

If you want URLs without the trailing "index.php", e.g., https://example.com/apps/files/ instead of https://example.com/index.php/apps/files/, you can enable it by following these steps:

Prerequisites:

Log in to the Docker container running ownCloud, and execute the following command on the host system of the appliance:

```
univention-app shell owncloud
```

Your web server needs to have the following modules enabled: mod_rewrite and mod_env. If you have not yet enabled these modules, or are not sure if you have, execute these commands:

```
a2enmod env rewrite
```

You need an **owncloud.conf** in your /etc/apache2/sites-available/ directory.

Open /etc/apache2/sites-available/owncloud.conf in nano, Vim, or your editor of choice, and paste the following:

```
Alias /owncloud "/var/www/owncloud/"

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

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

 SetEnv HOME /var/www/owncloud
 SetEnv HTTP_HOME /var/www/owncloud

</Directory>
```

Then create a symlink to /etc/apache2/sites-enabled, as follows:

```
ln -s /etc/apache2/sites-available/owncloud.conf /etc/apache2/sites-
enabled/owncloud.conf
```

Enable index.php-less URLs

Adjust your config.php to look like the following:

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

Execute the command command:

```
occ maintenance:update:htaccess
```

Restart or reload your Apache server, by running the following command:

```
service apache2 reload
```

Now you should have index.php-less URLs.

Appliance Maintenance

In this section you will find all the details you need to maintain the ownCloud appliance..

Backup

If you remove the ownCloud app or update it - a backup is created automatically.

The backup remains on the host system and can be restored.

It is stored in :

```
/var/lib/univention-appcenter/backups/
```

The file name is :

```
appcenter-backup-owncloud:date
```

In it, you find your data and conf folders.

Your database backup is in :

```
/var/lib/univention-appcenter/backups/data/backups
```

How to Update ownCloud

Introduction

This page shows how to update an ownCloud installation hosted on an ownCloud X Appliance:



Do not use ownCloud's built in Web Updater!

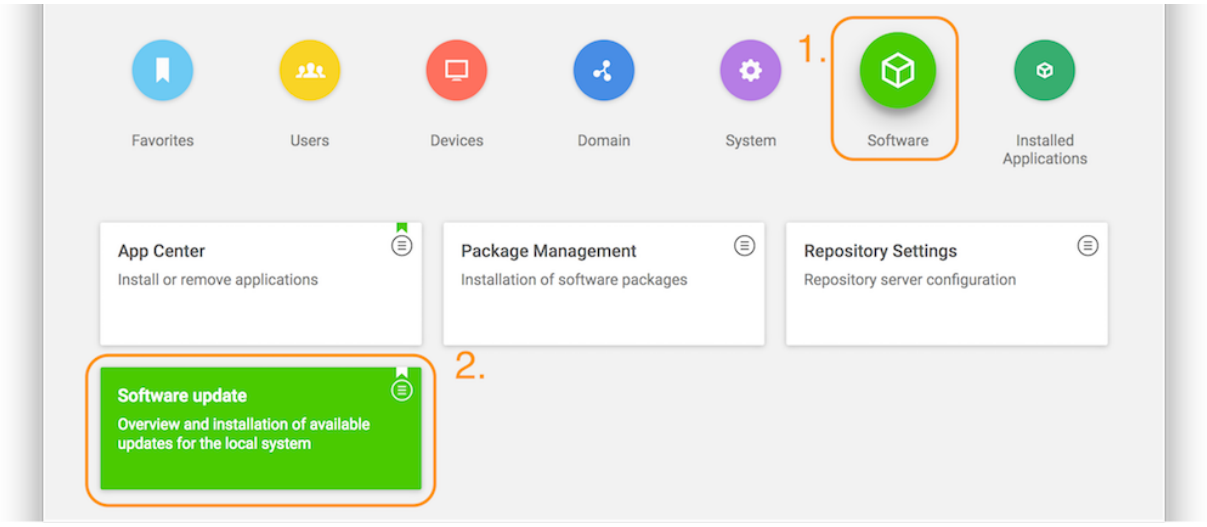
Use the Univention Management Console

Using the Univention Management Console, there are two paths to upgrade an existing ownCloud installation:

- In-place Upgrade (for 10.0 users)
- Uninstall the Existing Version and Install the New Version (for 9.1 users)

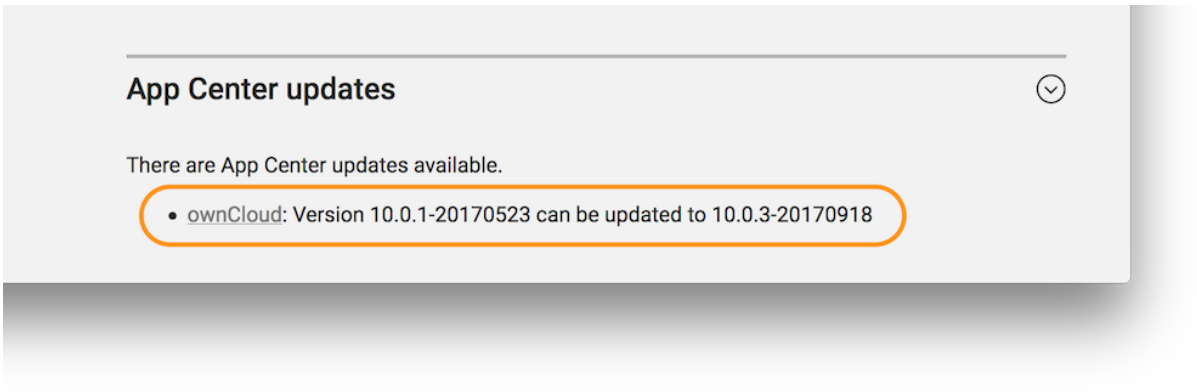
In-place Upgrade (for 10.0 users)

To perform an in-place upgrade, after logging in to the Univention server, under "**Administration**", click the first option labeled btn:[System and domain settings]. This takes you to the Univention Management Console. From there, click the btn:[Software] shortcut (1), and then click btn:[Software update] (2).

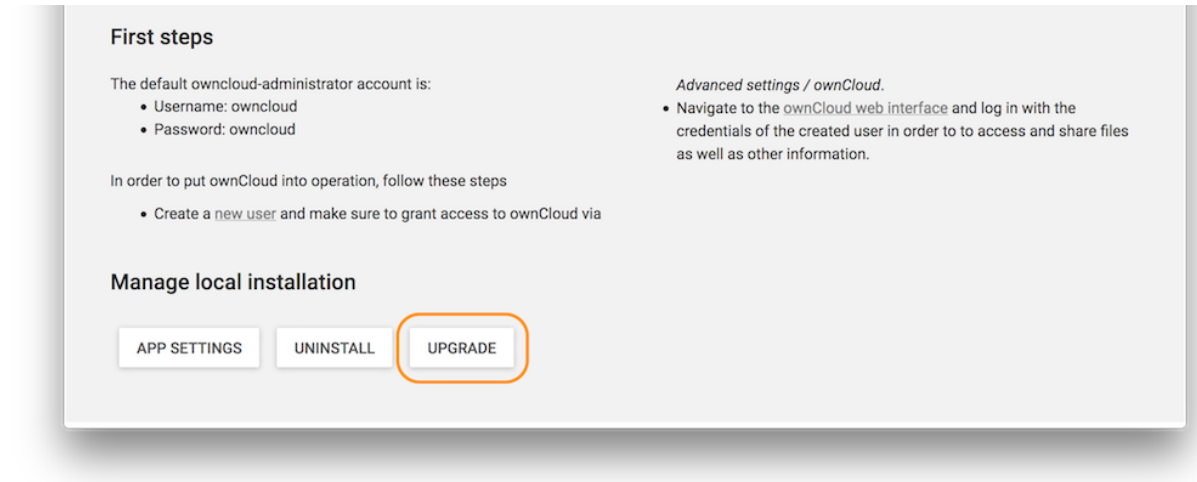


This will load the Software update management panel, after a short time scanning for available updates. If an update is available, under "**App Center updates**" you will see

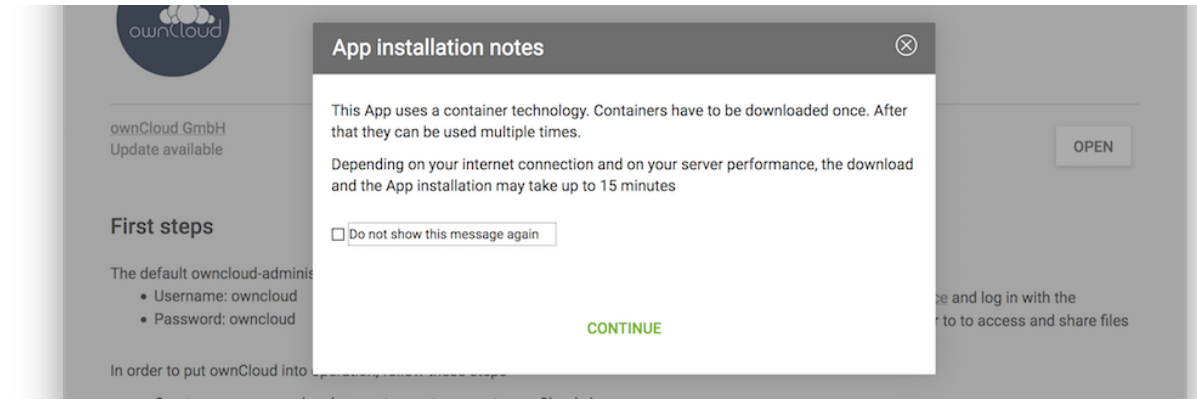
"**There are App Center updates available**". If one is, as in the image below, click btn:[ownCloud] which takes you to the ownCloud application.



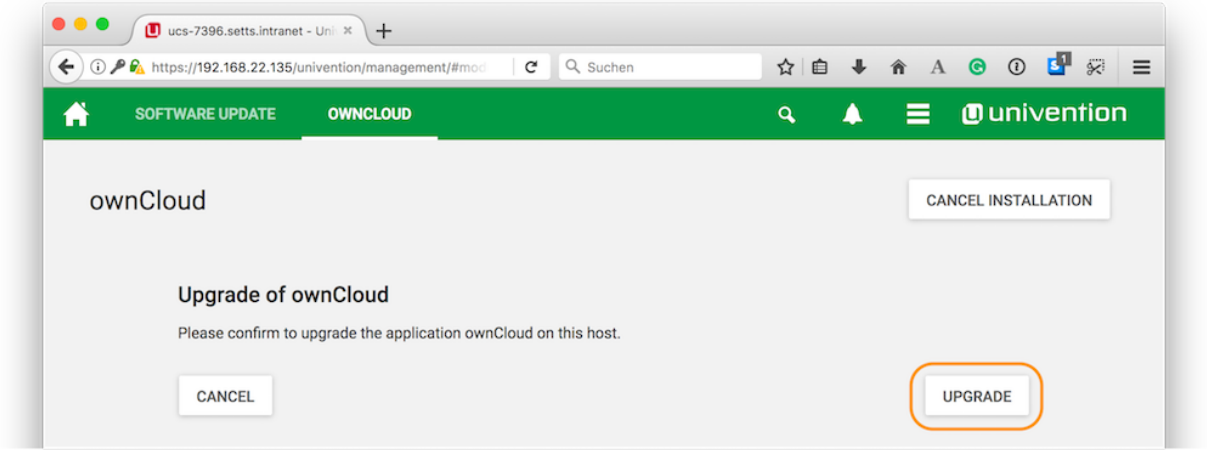
When there, part-way down the page you'll see the "**Manage local installation**" section. Under there, click btn:[UPGRADE].



Before the upgrade starts, a prompt appears titled "**App Installation notes**". This is nothing to be concerned about. So check the checkbox btn:[Do not show this message again]. Then click btn:[CONTINUE].



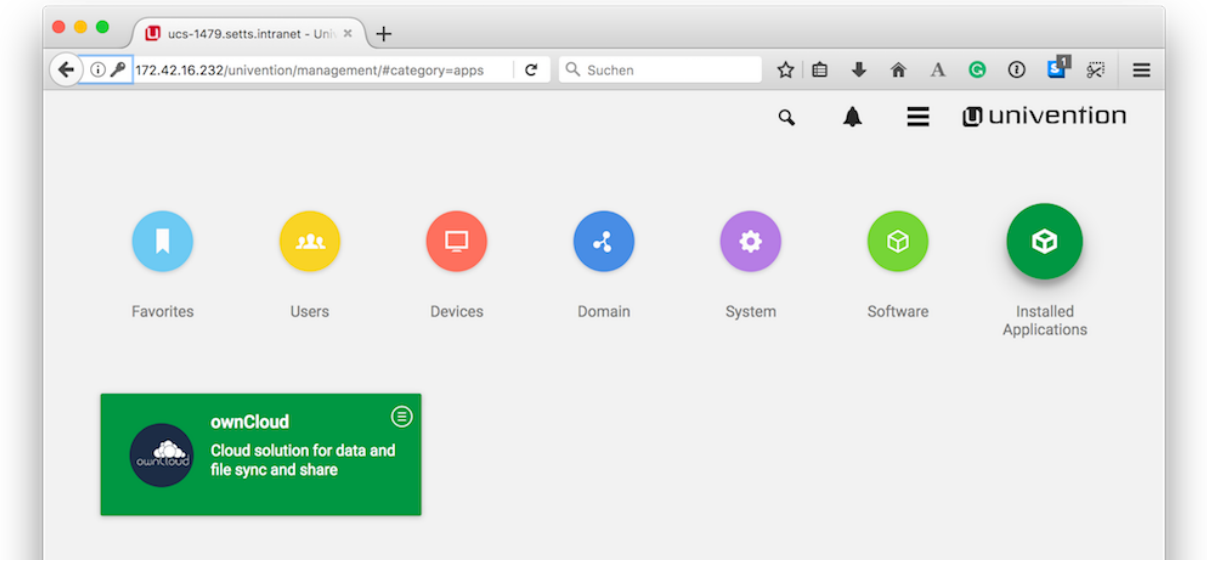
Next an upgrade confirmation page appears. To accept the confirmation, click btn:[UPGRADE] on the far right-hand side of the confirmation page.



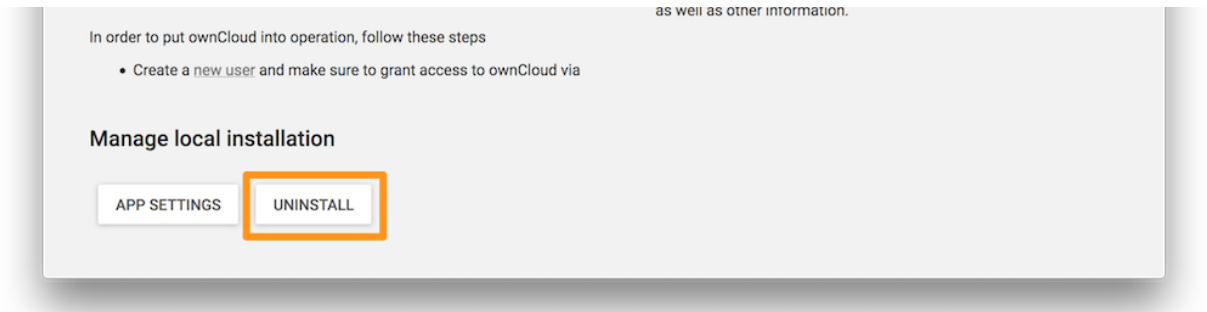
This launches the upgrade process, which requires no manual intervention. When the upgrade completes, the ownCloud app page will be visible again, but without the btn:[UPGRADE] button. Now, login to ownCloud by clicking the btn:[OPEN] button, on the far right-hand side of the page.

Uninstall the Existing Version and Install the New Version (for 9.1 users)

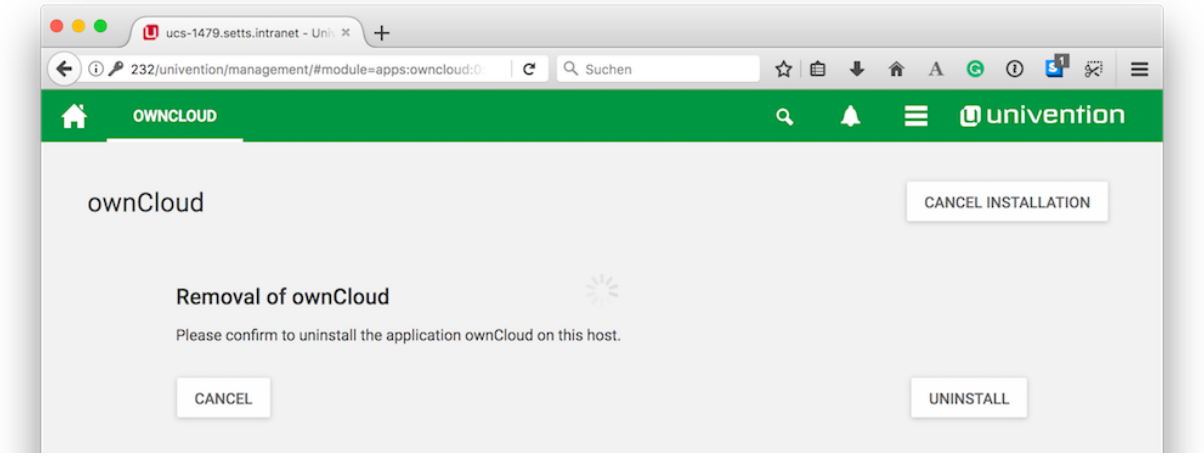
Open your ownCloud X Appliance and go to the "**System and Domain Settings**" dashboard. Then, after logging in, click btn:[Installed Applications], and then click btn:[ownCloud].



This takes you to the ownCloud app settings page. From there, begin uninstalling ownCloud by clicking btn:[UNINSTALL] under "**Manage local installations**"

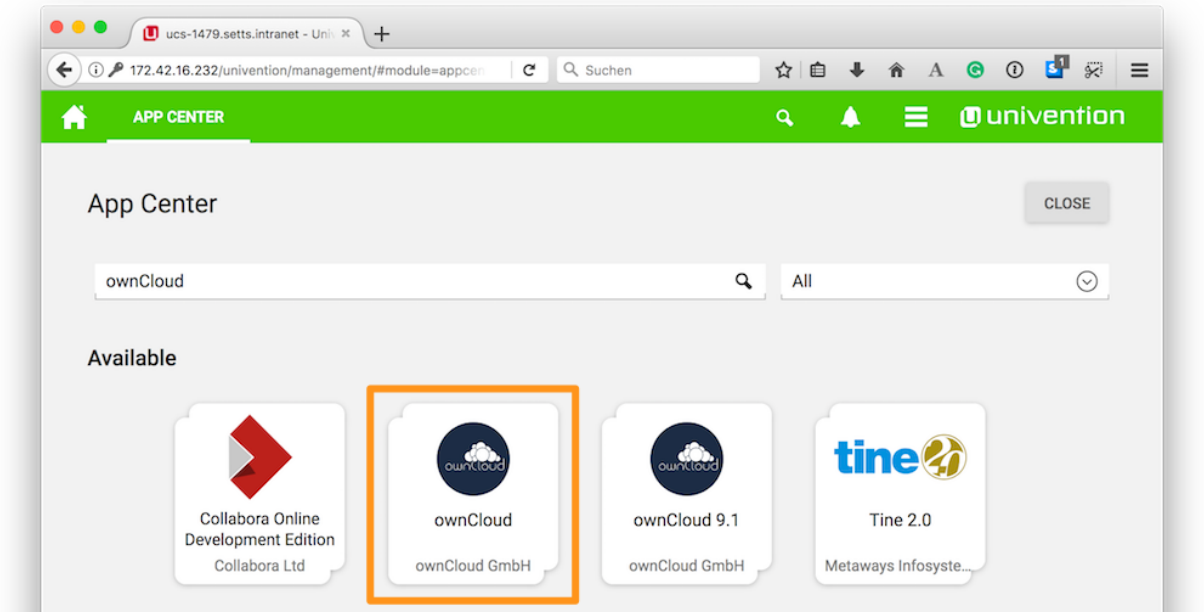


This takes you to an uninstall confirmation page. On that page, click btn:[UNINSTALL] on the lower left-hand side of the page.



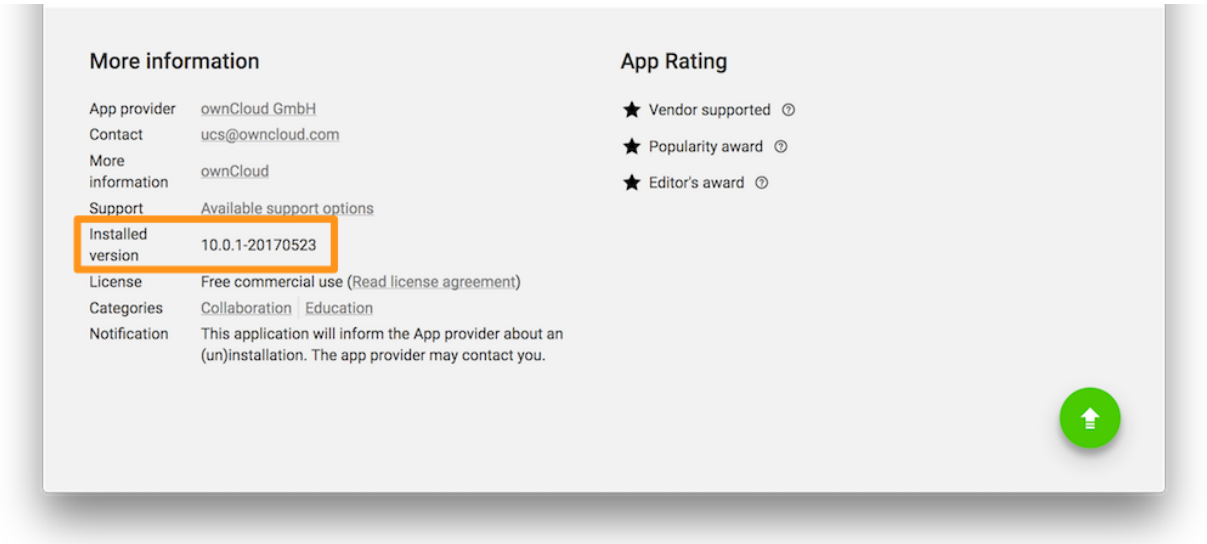
Follow the process until it’s finished. Then, click on btn:[Close] in the upper right corner.

Your data and users will remain.

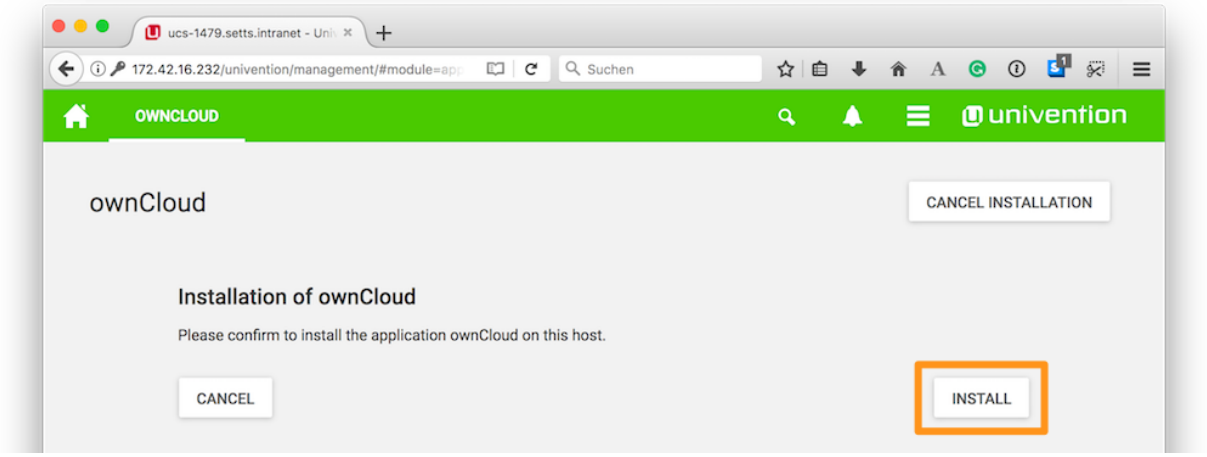


Following that, go to "**Software - Appcenter**", and search for *ownCloud*. At the moment, two matching results will be returned. Pick the one that does not contain a version number.

To confirm the version number, scroll to the bottom of the page, and in the More information section, look for the version string, next to Installed version, as in the screenshot below.



If it is the right version, click btn:[INSTALL]. Then the License Agreement is displayed. If you agree to it, click btn:[ACCEPT LICENSE]. This will display an installation confirmation screen. To confirm the installation, click btn:[INSTALL].



The installation will then be carried out. When it is finished, you will have the latest version of ownCloud installed.

Your data and users will persist.

Use the Command Line

As with the Univention Management Console, there are two paths to upgrade an existing ownCloud installation from the command line:

- Upgrading From Version 10.0.1 to 10.0.3
- Upgrading From Versions Prior to 10.0

Upgrading From Version 10.0.1 to 10.0.3

Upgrading from the command line is also available. To do so, login to your ownCloud X Appliance, either via ssh or directly on the server. Once logged in, check if there is an upgrade available.

You can use the command univention-app info. This command lists information about the current state of every installed App.

```
root@ucs-9446:~# univention-app info
UCS: 4.2-1 errata165
App Center compatibility: 4
Installed: 4.1/owncloud=10.0.1-20170523
Upgradable: owncloud
```

If an upgrade is available, you then need to run the univention-app upgrade, as in the example below.

```
univention-app upgrade owncloud
```

You will have to enter your Administrator password to start the upgrade. This command takes some time to complete, primarily based on the appliance's network connection speed. However, it should not take more than a few minutes.

After the upgrade has completed (if it was successful) as a sanity check, run univention-app info, to confirm the currently installed version of ownCloud. As in the example below, you should see that the installed version is now higher than before, and that ownCloud is no longer upgradable.

```
root@ucs-9446:~# univention-app info
UCS: 4.2-1 errata165
App Center compatibility: 4
Installed: 4.1/owncloud=10.0.3-20170918
Upgradable:
```

Upgrading From Versions Prior to 10.0

If you're running a version of ownCloud prior to 10.0, the above in-place upgrade doesn't work. This is because the earlier versions of ownCloud are installed with a different application to the 10.x version. More specifically, the versions of the ownCloud app, prior to 10, have a version suffix in the name. For example the ownCloud 8.2 app is named owncloud82.

For ownCloud 8.2 users: during the ownCloud App upgrade, user files will be moved to the new Docker data directory, /var/lib/univention-appcenter/apps/owncloud/data/files. Essentially, the following the command will be executed:

```
mv /var/lib/owncloud/* /var/lib/univention-appcenter/apps/owncloud/data/files
```

Please check your filesystems and mountpoints and make sure enough space is available for the operation.

Given that, you first have to uninstall the existing version and then install the 10.x version. To do so, run the following commands:

```
# Assumes that owncloud82 is the currently installed version
univention-app remove owncloud82
univention-app update
univention-app install owncloud
```

And after the upgrade and updates are completed, you can then login to ownCloud and verify the upgrade. Username and Password remain the same as before the upgrade:

- owncloudadmin
- password

Troubleshooting

If you have encountered an issue, here is what support needs in order to get a quick resolution of your issue:

1. Log file located at /var/lib/univention-appcenter/apps/owncloud/data/files/owncloud.log
2. Config Report generated with occ configreport:generate > config_report.json (you have to login to the container with univention-app shell owncloud)
3. The status of your docker containers docker ps > docker.txt
4. The status of your appliance univention-app info > univention.txt
5. Docker Logs: find out your docker ID of the ownCloud container and then execute docker logs <containerID or container name>. Here is an example: docker logs owncloud_owncloud_1

Restore a snapshot to get your appliance to a functional state again.

Enterprise Edition

In this section, you will find all the information you need for managing ownCloud Enterprise Edition.

Enterprise Clients

In this section you will find all the details you need to configure ownCloud enterprise clients.

Creating Branded Client Apps

Overview

ownBrander is an ownCloud build service that is exclusive to Enterprise customers for creating branded Android and iOS ownCloud sync apps, and branded ownCloud desktop sync clients. You build your apps with the ownBrander app on your Customer.owncloud.com account, and within 24-48 hours the completed, customized apps are loaded into your account. You must supply your own artwork, and you'll find all the specifications and required elements in ownBrander.

Building a Branded Desktop Sync Client

See Building Branded ownCloud Clients for instructions on building your own branded desktop sync client, and for setting up an automatic update service.

Your users may run both a branded and un-branded desktop sync client side-by-side. Both clients run independently of each other, and do not share account information or files.

Building a Branded iOS App

Building and distributing your branded iOS ownCloud app involves a large number of interdependent steps. The process is detailed in the Building Branded ownCloud Clients manual. Follow these instructions exactly and in order, and you will have a nice branded iOS app that you can distribute to your users.

Building a Branded Android App

Building and distributing your branded Android ownCloud app is fairly simple, and the process is detailed in Building Branded ownCloud Clients.

Custom Client Download Repositories

See Custom Client Download Repositories to learn how to test and configure custom download repository URLs for your branded clients.

Enterprise Collaboration

In this section you will find all the details you need to configure enterprise collaboration in ownCloud.

Secure View

When Collabora Online is properly setup and integrated into ownCloud Server (10.3 or above) Secure View functionality is available. Secure View is a mode where users can place limitations on files and folders that are shared. These limitations include: * No copying * No downloading * No editing * Watermarking

+ NOTE: Administrators can configure the watermark text. * Optional printing and exporting to PDF with watermarks included



Documents never leave the server when shared with Secure View. Collabora Online Server opens them and streams the files to the user's browser (much like a video stream). Consequently, there's no way to extract the original document from the browser.

Supported File Formats

Secure View only supports a limited number of file formats; these are:

- Microsoft Word (.docx)
- Microsoft Excel (.xlsx)
- Microsoft PowerPoint (.pptx)
- OpenDocument Text Document (.odt)
- OpenDocument Presentation Document (.odp)
- OpenDocument Spreadsheet Document (.ods)
- PDF

If a folder shared with Secure View contains unsupported file types (e.g., JPG), they will not be accessible in any way.

Prerequisites

- ownCloud 10.3 or higher *Enterprise Edition*
- Collabora Online set up and integrated



This functionality is **not** available in the community edition and does not work with Public Links.

How to Enable Secure View

To enable *Secure View*, navigate to menu:Settings[Admin > Additional (Admin) > Collabora Online]. At the bottom of the Collabora Online section, enable btn:[Enable Secure View].

Once enabled, default share permissions for all users can, optionally, be enabled. Currently, these default share permissions are:

- **Secure View (with watermarks)**. When enabled, files are shared in Secure View mode. In this mode , all the Secure View limitations are in-effect. When this mode and "*can edit*" are disabled, the share is a regular, "read-only", share.
- **can print / export PDF**.

  This option is only visible if btn:[Secure View (with watermarks)] is enabled.

  When enabled, this mode allows documents to be can be printed or exported to PDF format — with a watermark — through Collabora Online.



Admins can specify that all shares are "*Secure View*" by default and that the user has to intentionally change this setting, and vice versa.

When "*Secure View (with watermarks)*" is enabled, any attempts to download the file will be blocked, as exemplified in the screenshot below. Additionally, select, copy, and paste are disabled.



Microsoft Office Online / WOPI Integration

About

The WOPI (Web Application Open Platform Interface) app, which is bundled with ownCloud Enterprise Edition, is the connector between ownCloud server and Microsoft Office Online Server.

It allows Microsoft Office Online users to collaboratively work with Office documents in ownCloud in the browser, by connecting ownCloud with your Microsoft Office Online Server via the WOPI protocol. To use it, you need to have a running Microsoft Office Online Server in your data center.



Please bear in mind:

- WOPI is only available for ownCloud enterprise. It *is not available* in the community version.
- Out-of-the box only the on-premise version of Microsoft Office Online Server is supported.
- This app requires ownCloud Version 10.1 and php 7.1.



If you want to integrate the Office 365 (cloud) version of Microsoft Office Online, you need to {oc-contact-url}[get in touch with us].

Preparing the Environment

You need an Office Online Server installed.

All involved servers (OfficeOnline Server and the ownCloud server) need to be accessible by HTTPS with valid certificates.

Configuring the WOPI App in ownCloud

To configure the WOPI app in your ownCloud installation, add the following configuration to config/config.php, and adjust it based on the details of your setup:

```
'wopi.token.key' => 'replace-with-your-own-random-string',
'wopi.office-online.server' => 'https://your.office.online.server.tld',
```

Restrict Usage to Users in a Specific Group

Microsoft Office Online access can be restricted to users in a specific group, by use of the wopi_group configuration key (in config/config.php), as in the following example.

```
'wopi_group' => 'admin'
```

In the example above, only users in the admin group would be able to access Microsoft Office Online.



If the key is not defined, then all users have access to WOPI.

Locking the Document

If you open a document with Microsoft Office Online in ownCloud, it makes use of the WebDAV file locking functionality, available in ownCloud server. The idea is to lock the file so other users, who have access, can’t make changes to the document while you’re editing it.

In other words: the feature ensures that you are the "master editor", your changes will always be the "master state". Other users can make changes, e.g., with the desktop client, but those will create conflict files for them, which can be resolved afterward. When you close the document, Microsoft Office Online unlocks the file, so others can edit it.

You can always click on the lock icon next to your file name and unlock it manually using the button in the sidebar.

Lock Timeout

If a user is editing a file and loses their internet connection the lock will timeout, freeing the lock, after 30 minutes. Please refer to the WOPI documentation for further information.

External Storage

In this section you will find all the details you need to configure enterprise external storage in ownCloud.

Enterprise-Only Authentication Options

In ownCloud 9.0+, there are five authentication backends for external storage mounts:

- Username and password
- Log-in credentials, save in session
- Log-in credentials, save in database
- User entered, store in database
- Global credentials

The first two are common to all editions of ownCloud, and the last three are only in the Enterprise edition. These are available to:

- FTP
- ownCloud
- SFTP
- SMB/CIFS
- WebDAV
- Windows Network Drive

Username and password
: This is the default; a login entered by the admin when the external mount is created. The login is stored in the database, which allows sharing, and background jobs, such as file scanning, to operate.

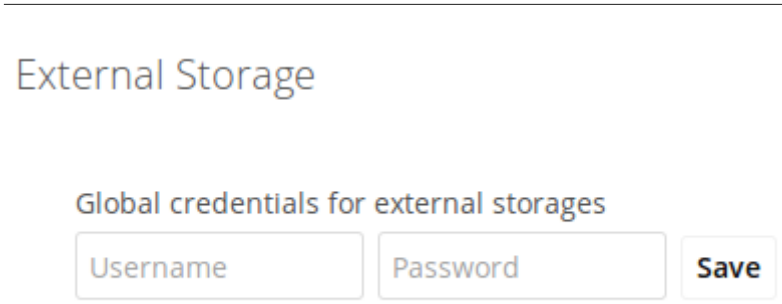
Log-in credentials, save in session
: Credentials are only stored in the session and not captured in the database. Files cannot be shared, as credentials are not stored.

Log-in credentials, save in database
: Credentials are stored in the database, and files can be shared.

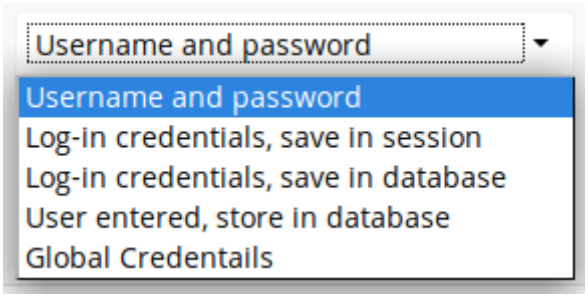
User entered, store in database
: Users provide their own login credentials, rather than using admin-supplied credentials. User credentials are stored in the database, and files can be shared.

Global credentials
: Re-usable credentials entered by the admin, files can be shared.

Global credentials are entered in a separate form.



Use the dropdown selector to choose the authentication backend when you create a new external mount.



LDAP Home Connector

Introduction

The LDAP Home Connector App enables you to configure your ownCloud server to display your users' Windows home directories on their Files pages, just like any other folder. Typically, Windows home directories are stored on a network server in a root folder, such as Users, which then contains individual folders for each user.

You must already have the LDAP app enabled and a working LDAP/Active Directory configuration in ownCloud.

Next, configure the root Windows home directory to be mounted on your ownCloud server. Then use the LDAP Home Connector and LDAP app to connect it to ownCloud.

Mount Home Directory

Create an entry in /etc/fstab for the remote Windows root home directory mount. Store the credentials to access the home directory in a separate file, for example /etc/credentials, with the username and password on separate lines, like this:

```
username=winhomeuser
password=winhomepassword
```

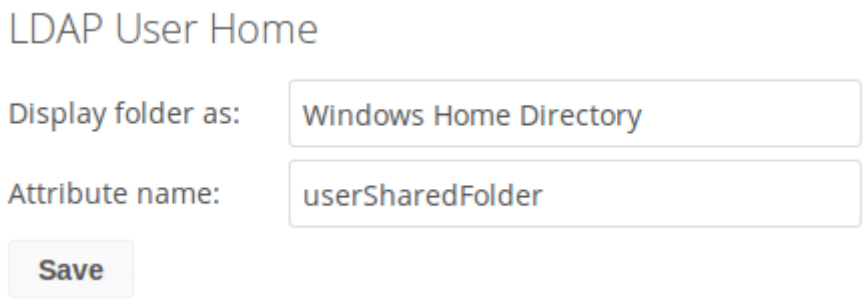
Then add a line like this to /etc/fstab, substituting your own server address and filenames:

```
//192.168.1.58/share /mnt/share cifs credentials=/etc/credentials,uid=33,gid=33
```

Configure the LDAP Home Connector

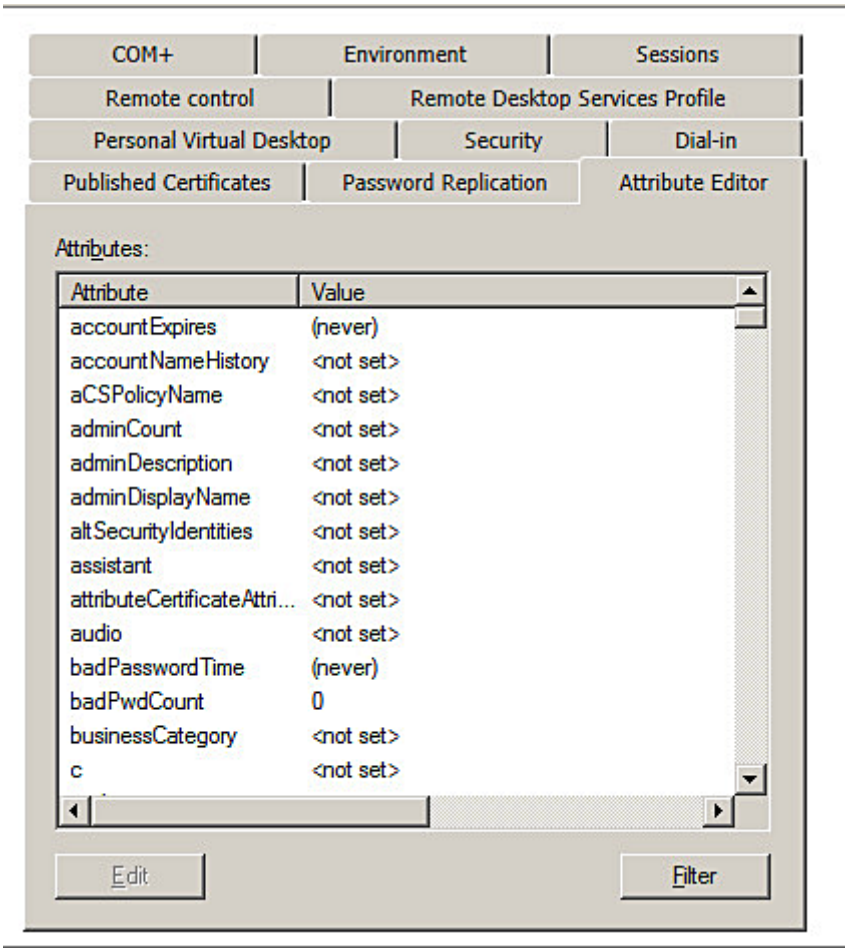
Enable the LDAP Home Connector app. Then go to the LDAP Home Connector form on your ownCloud admin page. In the **Display folder as:** field enter the name as you want it to appear on your users' File pages.

Then in the **Attribute name:** field enter the LDAP attribute name that will contain the home directory. Use any LDAP attribute that is not already in use, then save your changes.

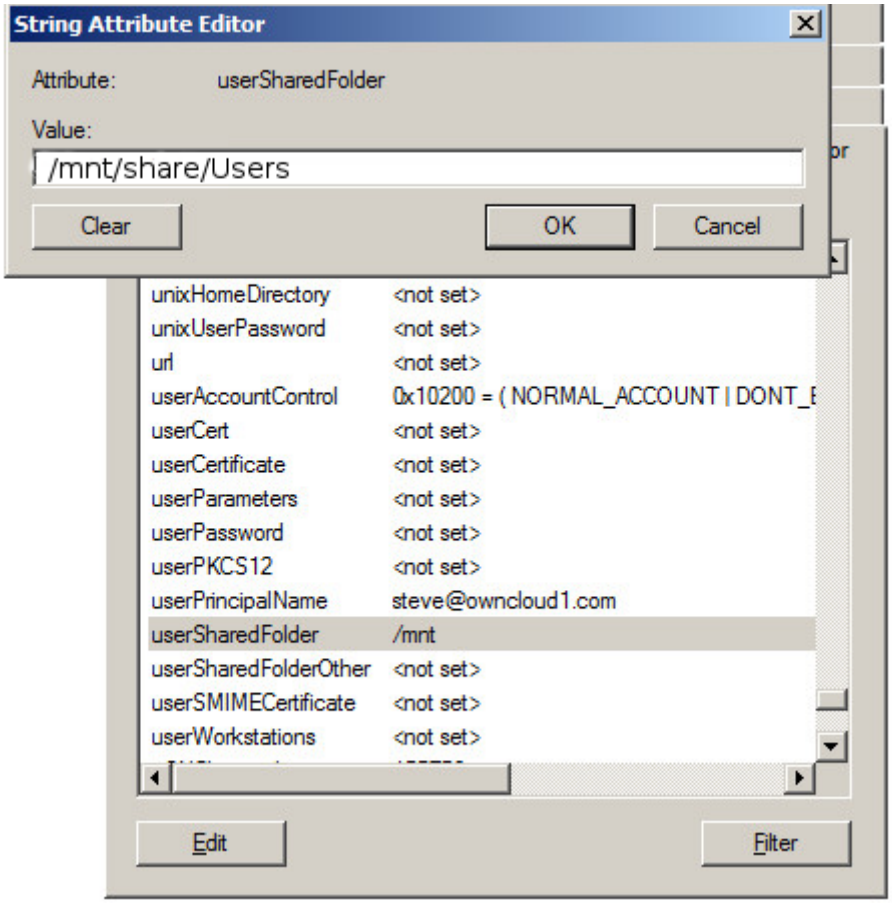


Configure the LDAP Server

In Active Directory, open the user profile. Scroll to the **Extensions** section and open the **Attribute Editor** tab



Scroll to the attribute being used (UserSharedFolder in this instance), and click **Edit**. Enter the users home directory.



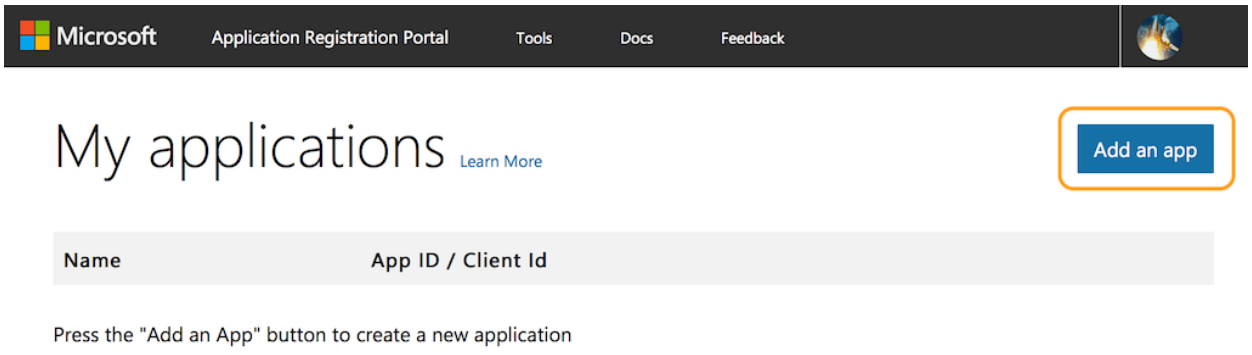
Save your changes, and you are finished.

How to Create and Configure Microsoft OneDrive

Introduction

Follow this guide to use Microsoft OneDrive as an external storage option in ownCloud.

Create an Application Configuration



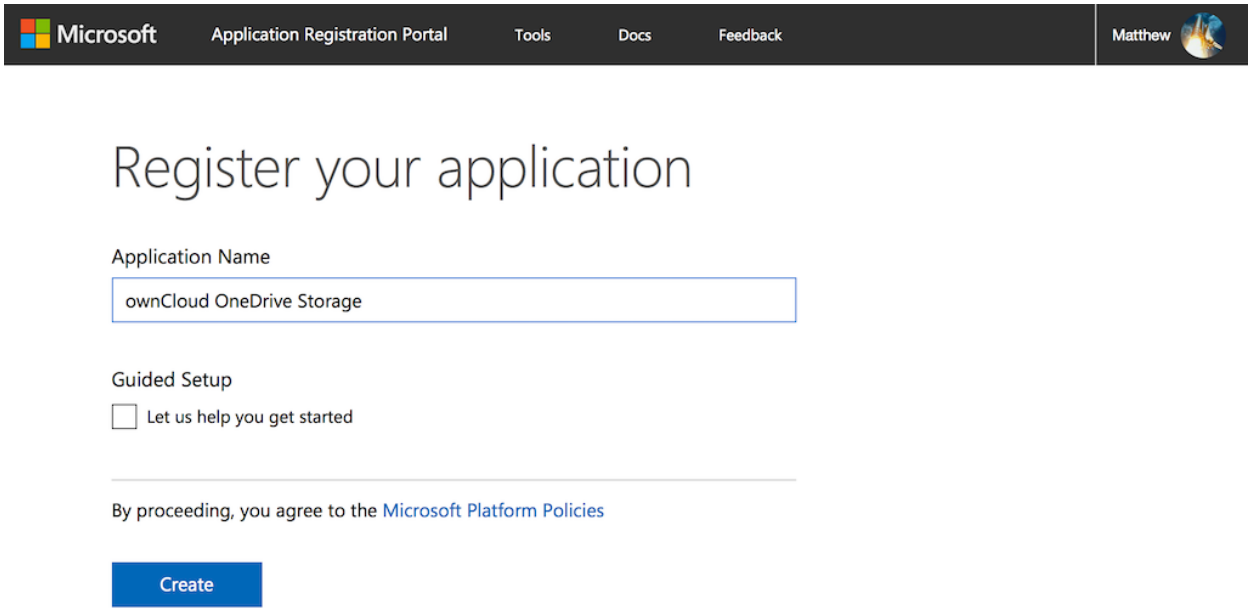
To create a new application:

- Open https://apps.dev.microsoft.com/ in your browser of choice and click "*Create App*".
- Under "*Properties*", set the application's name.
- Click "*Create*".

With the application created, you can then add a range of further settings. However,

only a few of them are required for use with ownCloud.

Application Password



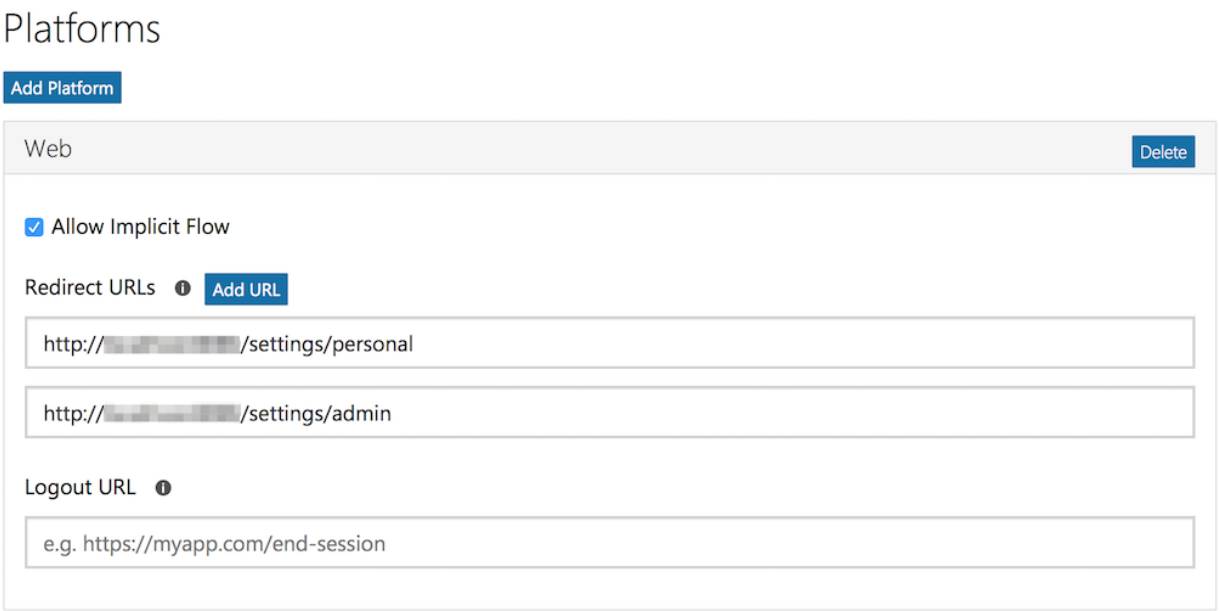
Under "*Application Secrets*", click "*Generate New Password*", which generates a password and displays it in a popup window. It is required later during when configuring a mount point.

Copy the password to your preferred password manager, as it is only displayed **once**.

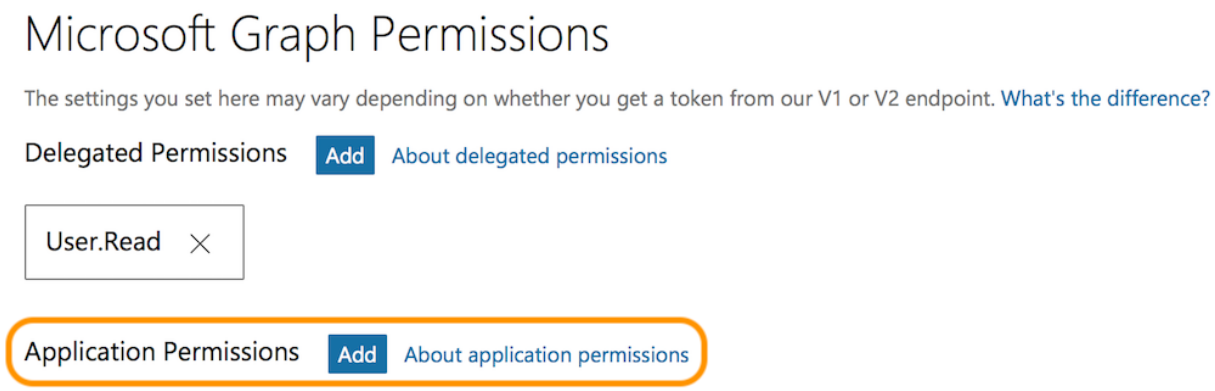
Redirect URLs

Under "*Platforms*", click "*Add Platform*" and choose "*Web*" in the popup window which appears. Only one redirect URL field is visible at first, so click "*Add URL*" to add another one.

With two fields available, add two redirect URLs; one for settings/admin and one for settings/personal, as you can see in the image below.



Application Permissions



Under "*Microsoft Graph Permissions*", click "*Add*" next to "*Application Permissions*". This opens a popup window where you can choose the required permissions. Add a least the following four:

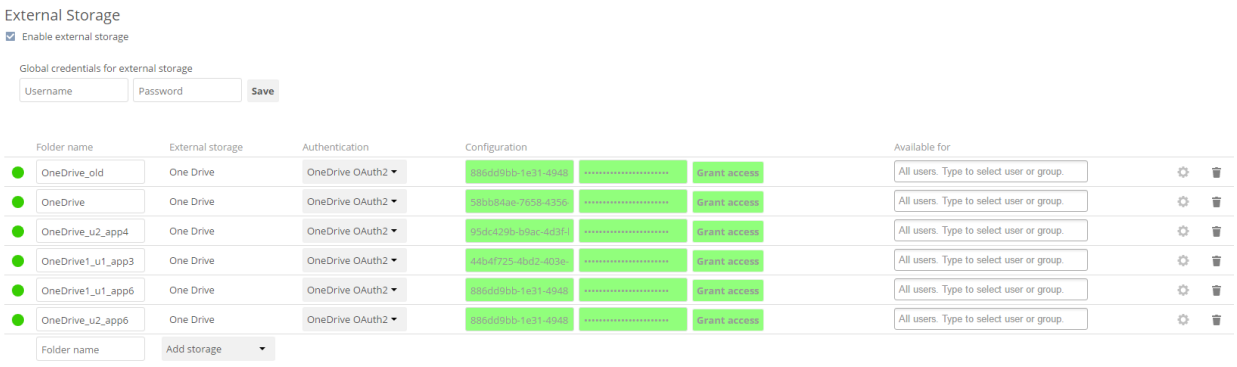
- Files.Read.All
- Files.ReadWrite.All
- IdentityRiskEvent.Read.All
- User.Read.All

With those settings added, click "*Save*", located right at the bottom of the page.

Configure a Mount Point in ownCloud

You can add as many OneDrive mount points as you want. To do so:

1. Add a new storage, selecting "One Drive" for external storage.
2. Set the credentials of your OneDrive application, and then accept the permissions.
3. If everything is accepted, the mount points should appear, with a green status icon on the far left-hand side.



To be able to use the occ command files_onedrive:subscribe, you need to have the variable overwrite.cli.url set in config/config.php, as in this example:

```
'overwrite.cli.url' => 'https://example.org:63984/index.php',
```

The HTTPS prefix, port, and /index.php suffix are mandatory.

Configuring SharePoint Integration

Introduction

Native SharePoint support has been added to the ownCloud Enterprise edition as a secondary storage location for SharePoint 2007, 2010 and 2013. When this is enabled, users can access and sync all of their SharePoint content via ownCloud, whether in the desktop sync, mobile or Web interfaces. Updated files are bi-directionally synced automatically. SharePoint shares are created by the ownCloud admin, and optionally by any users who have SharePoint credentials.

The ownCloud SharePoint plugin uses SharePoint document lists as remote storage folders. ownCloud respects SharePoint access control lists (ACLs), so ownCloud sharing is intentionally disabled for SharePoint mountpoints. This is to preserve SharePoint ACLs and ensure content is properly accessed as per SharePoint rules.

The plugin uses the Simple Object Access Protocol (SOAP) and WebDAV for the uploads and downloads to talk to SharePoint servers. Your ownCloud server must have php-soap or php5-soap installed. Linux packages and ownCloud appliances will install php5-soap as a required dependency.

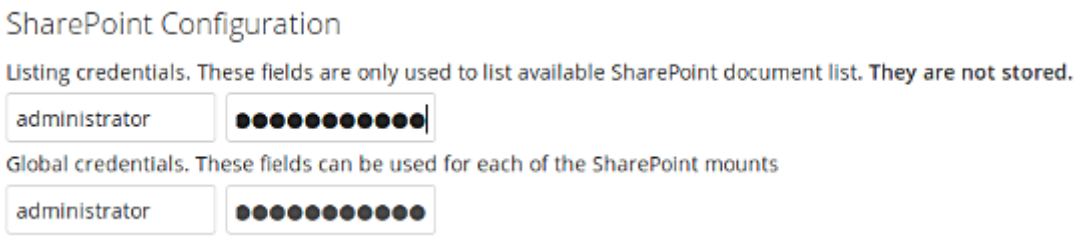
The supported authentication methods are:

- Basic Auth
- NTLM (Recommended)

Creating a SharePoint Mount

Enable the SharePoint app, and then enter the Admin panel to set up SharePoint connections in the SharePoint Drive Configuration section.

Enter your SharePoint Listing credentials. These credentials are not stored in the database, but are used only during plugin setup to list the Document Libraries available per SharePoint site.



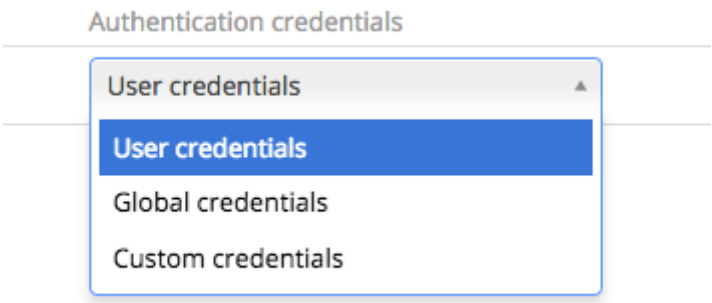
Global credentials is optional. If you fill in these fields, these credentials will be used on on all SharePoint mounts where you select: **Use global credentials** as the authentication credentials.



Enter your ownCloud mountpoint in the Local Folder Name column. This is the name of the folder that each user will see on the ownCloud filesystem. You may use an existing folder, or enter a name to create a new mount point

Select who will have access to this mountpoint, by default **All users**, or a user or a group.

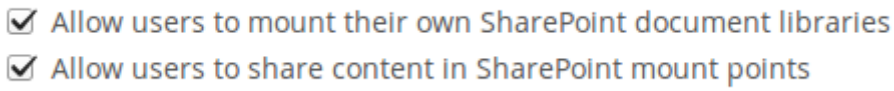
Enter your SharePoint server URL, then click the little refresh icon to the left of the Document Library field. If your credentials and URL are correct you'll get a dropdown list of available SharePoint libraries. Select the document library you want to mount.



Select which kind of Authentication credentials you want to use for this mountpoint. If you select **Custom credentials** you will have to enter the the credentials on this line. Otherwise, the global credentials or the user's own credentials will be used. Click Save, and you're done

Enabling Users

You may allow your users to create their own SharePoint mounts on their Personal pages, and allow sharing on these mounts.



Note

Speed up load times by disabling file previews in config.php, because the previews are generated by downloading the remote files to a temp file. This means ownCloud will spend a lot of time creating previews for all of your SharePoint content. To disable file previews, add the following line to the ownCloud config file found in /owncloud/config/config.php:

```
'enable_previews' => false,
```

Troubleshooting

Unsharing

SharePoint unsharing is handled in the background via Cron. If you remove the sharing option from a SharePoint mount, it will take a little time for the share to be removed, until the Cron job runs.

Logging

Turn on SharePoint app logging by modifying config/config.php, setting sharepoint.logging.enable to true, as in the example below.

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

Mount Points

Global mount points can't be accessed: You have to fill out your SharePoint credentials as User on the personal settings page, or in the popup menu. These credentials are used to mount all global mount points.

Personal mount points can't be accessed: You have to fill your SharePoint credentials as User on the personal settings page in case your personal mount point doesn't have its own credentials.

A user can't update the credentials: Verify that the correct credentials are configured, and the correct type, either global or custom.

Installing and Configuring the External Storage: Windows Network Drives App

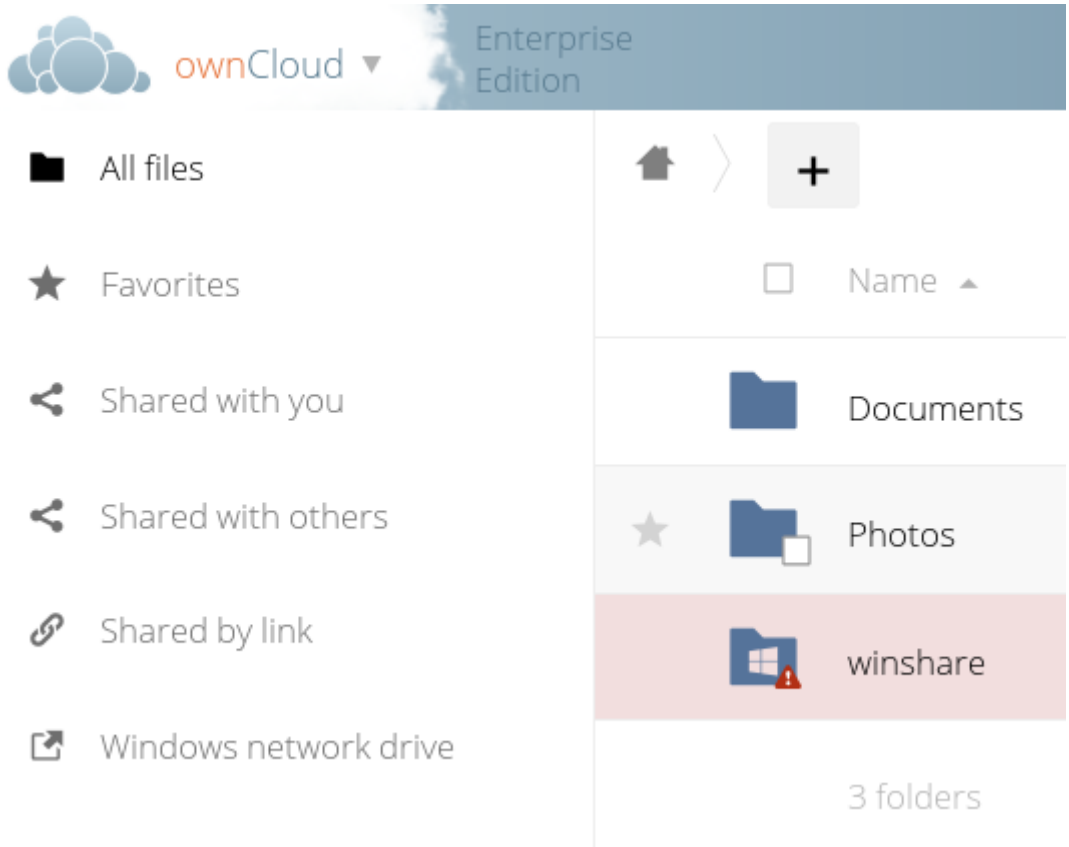
Introduction

The External Storage: Windows Network Drives app creates a control panel in your Admin page for seamlessly integrating Windows and Samba/CIFS shared network drives as external storages.

Any Windows file share and Samba servers on Linux and other Unix-type operating systems use the SMB/CIFS file-sharing protocol. The files and directories on the SMB/CIFS server will be visible on your Files page just like your other ownCloud files and folders.

They are labeled with a little four-pane Windows-style icon, and the left pane of your Files page includes a Windows Network Drive filter. Figure 1 shows a new Windows Network Drive share marked with red warnings.

These indicate that ownCloud cannot connect to the share because it requires the user to login, it is not available, or there is an error in the configuration.





For more information on SMB/CIFS in ownCloud, refer to the Samba file server configuration documentation.



If you encounter errors with NT_STATUS_REVISION_MISMATCH, please get in touch with support@owncloud.com.

Files are synchronized bi-directionally, and you can create, upload, and delete files and folders. ownCloud server admins can create Windows Network Drive mounts and optionally allow users to set up their own personal Windows Network Drive mounts.

Depending on the authentication method, passwords for each mount are encrypted and stored in the ownCloud database, using a long random secret key stored in config.php, which allows ownCloud to access the shares when the users who own the mounts are not logged in. Or, passwords are not stored and available only for the current session, which adds security.

Installation

Install the External Storage: Windows Network Drives app from the ownCloud Market App or ownCloud Marketplace. To make it work, a few dependencies have to be installed.

- A Samba client. This is included in all Linux distributions. On Debian, Ubuntu, and other Debian derivatives it is called smbclient. On SUSE, Red Hat, CentOS, and other Red Hat derivatives it is samba-client.
- php-smbclient (version 0.8.0+). It should be included in most Linux distributions. You can use eduardok/libsmbclient-php, if your distribution does not provide it.
- which and stdbuf. These should be included in most Linux distributions.

Example

Assuming that your ownCloud installation is on Ubuntu, then the following commands will install the required dependencies:

```
# Install core packages
sudo apt-get update -y
sudo apt-get install -y smbclient php-smbclient coreutils
```

Other method using PECL is:

```
# Install php-smbclient using PECL
pecl install smbclient

# Install it from source
git clone git://github.com/eduardok/libsmbclient-php.git
cd libsmbclient-php ; phpize
./configure
make
sudo make install
```

Regardless of the method you use, remember to check if an smbclient.ini file exists in /etc/php/<your php version>/mods-available and contains the following line:

```
extension="smbclient.so"
```

If so, then make it available via by running the following command:

```
sudo phpenmod -v ALL smbclient
```

Configuration

Enabling External Storage

To enable external storage go to menu:Settings[Storage (in the admin section)], as the ownCloud administrator. Tick the checkbox to enable external storage.

Creating a New Share

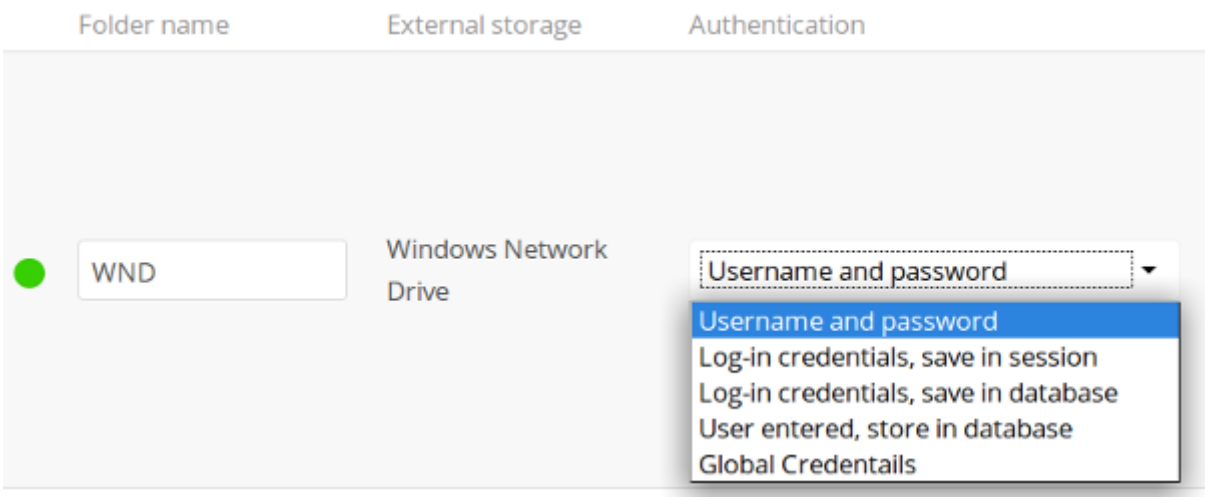
When you create a new WND share you need three things:

- The login credentials for the share
- The server address, the share name; and
- The folder you want to connect to

Treat all the parameters as being case-sensitive.

Although some parts of the app might work properly, regardless of case, other parts might have problems if case isn't respected.

1. Enter the ownCloud mount point for your new WND share. This must not be an existing folder.
2. Then select your authentication method; See enterprise_only_auth for complete information on the five available authentication methods.



1. Enter the address of the server that contains the WND share.
2. The Windows share name.
3. The root folder of the share. This is the folder name, or the $user variable for user's home directories. Note that the LDAP Internal Username Attribute must be set to

the samaccountname for either the share or the root to work, and the user's home directory needs to match the samaccountname. (See User Authentication with LDAP.)

4. Login credentials.
5. Select users or groups with access to the share. The default is all users.
6. Click the gear icon for additional mount options. Note that previews are enabled by default, while sharing is not (see figure 2). Sharing is not available for all authorization methods; see enterprise_only_auth. For large storages with many files, you may want to disable previews, because this can significantly increase performance.

Your changes are saved automatically.

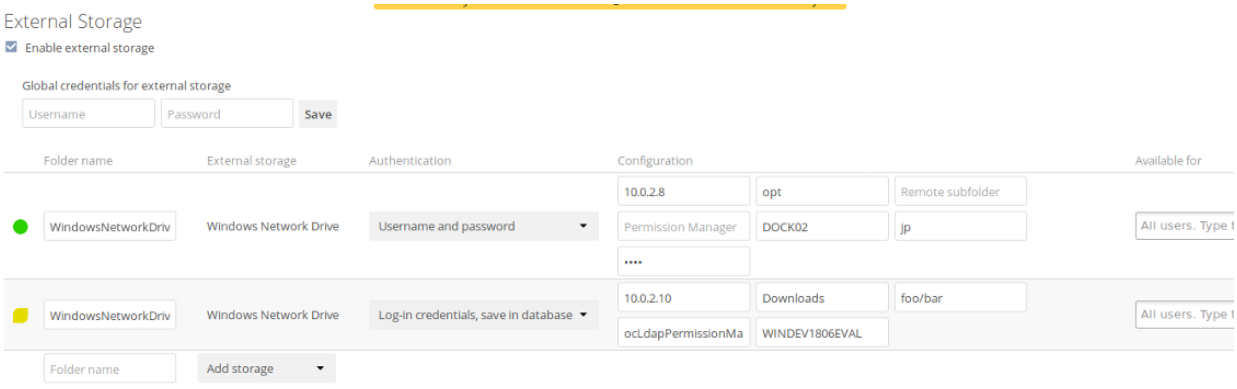


When you create a new mountpoint using Login credentials, you must log out of ownCloud and then log back in so you can access the share. You only have to do this the first time.

Access Control Lists

From version 1.0.1 the Windows Network Drives App supports Access Control Lists (ACLs). To obtain ACL information, it supports two ACL providers:

- The Null Permission Manager
- The ownCloud LDAP Permission Manager



Regardless of which provider you choose, an ownCloud administrator should run a files:scan, manually, after changing the configuration, to update the permissions correctly. Otherwise, the permissions shown by ownCloud might be incorrect.



Permissions are only auto-updated if there has been a change in the files.

The Null Permission Manager

The Null Permission Manager is the default permission manager and is used if no other ACL manager is specified. If you want to retain ownCloud's current behaviour, then use this permission manager. When in effect, the Windows Network Drive app uses a file's attributes (e.g., read-only, and hidden), to determine how the user can interact with the file. There are no usage restrictions.

The value to use for this provider is: nullPermissionManager.

The ownCloud LDAP Permission Manager

The ownCloud LDAP Permission Manager evaluates ACLs in files along with file attributes to determine the permissions. In order to evaluate the ACLs, it needs access to the user and group membership information of the target Windows or Samba server, so it uses the ownCloud's user_ldap app for this.



Both the Windows (or Samba) server and ownCloud's user_ldap app must connect to the same Active Directory server so that ownCloud can retrieve the same user and group information.

To use this provider requires two key things:

- An Active Directory server which contains the standard user and group information that can be used by ownCloud's LDAP app.
- ownCloud's LDAP app to be correctly configured to retrieve user and group information from the same Active Directory server as the one that the Windows (or Samba) server uses.



The ownCloud LDAP apps must configure the sAMAccountName to be the ownCloud server's username.



Some groups, such as "everyone" might not be handled properly. This is because such groups don't exist in the LDAP server, or might not be found if the domain is different, such as nt authority\system or builtin\domain-users.

The value to use for this provider is: ocLdapPermissionManager.

Windows Network Drive Listener



ownCloud requires at least Samba version 4.7.8 or Samba 4.8.1 on the ownCloud server, when:

1. The Windows Network Drive Listener is used; **and**
2. The remote Windows/Samba file server requires at least version 2.0 of the SMB protocol.

The Windows Network Drive Listener only supports version 1 of the SMB protocol with earlier Samba versions.

Here's Why

A Samba server, often a Microsoft Windows Server, can enforce the minimum and maximum protocol versions used by connecting clients. However, in light of the WannaCry ransomware attack, Microsoft patched Windows Server to only allow SMB2 protocol by default (as SMB1 is insecure).

The ownCloud windows network drive listener utilizes the SMB notification feature which works well with SMB1 in conjunction with most Samba versions. However, when the minimum protocol a server accepts is SMB2, ownCloud require Samba 4.7.8+ (4.8+ etc.) to be able to properly work, as prior versions of Samba had a bug that break this feature.

The SMB protocol supports registering for notifications of file changes on remote Windows SMB storage servers. Notifications are more efficient than polling for

changes, as polling requires scanning the whole SMB storage. ownCloud supports SMB notifications with an occ command, occ wnd:listen.



The notifier only works with remote storage on Windows servers. It does not work reliably with Linux servers due to technical limitations.

Your smbclient version needs to be 4.x, as older versions do not support notifications. The ownCloud server needs to know about changes to files on integrated storage so that the changed files will be synced to the ownCloud server, and to desktop sync clients.

Files changed through the ownCloud Web Interface, or sync clients are automatically updated in the ownCloud file cache, but this is not possible when files are changed directly on remote SMB storage mounts.

To create a new SMB notification, start a listener on your ownCloud server with occ wnd:listen. The listener marks changed files, and a background job updates the file metadata.

Setting Up the WND Listener

The WND listener for ownCloud 10 includes two different commands that need to be executed:

- wnd:listen
- wnd:process-queue

wnd:listen

This command listens and stores notifications in the database coming from one specific host and share. It is intended to be run as a service. The command requires the host and share, which the listener will listen to, and the Windows/Samba account that will listen. The command does not produce any output by default, unless errors happen.



You can increase the command's verbosity by using -vvv. Doing so displays what the listener is doing, including a timestamp and the notifications received. Although the exact permissions required for the Windows account are unknown, read-only should be enough.

The simplest way to start the wnd:listen process manually, perhaps for initial testing, is as follows

```
{occ-command-example-prefix} wnd:listen <host> <share> <username>
```

The password is an optional parameter and you'll be asked for it if you didn't provide it, as in the example above. In order to start the wnd:listen without any user interaction, provide the password as the user's 4th parameter, as in the following example:

```
{occ-command-example-prefix} wnd:listen <host> <share> <username>
<password>
```

For additional options to provide the password, check Password Options

Note that in any case there won't be any processing of the password by default. This

means that spaces or newline chars won't be removed unless explicitly told. Use the --password-trim option in those cases.

You should be able to run any of those commands, and/or wrap them into a systemd service or any other startup service, so that the wnd:listen command is automatically started during boot, if you need it.

wnd:process-queue

This command processes the stored notifications for a given host and share. This process is intended to be run periodically as a Cron job, or via a similar mechanism. The command will process the notifications stored by the wnd:listen process, showing only errors by default. If you need more information, increase the verbosity by calling wnd:process-queue -vvv.

As a simple example, you can check the following:

```
{occ-command-example-prefix} wnd:process-queue <host> <share>
```

You can run that command, even if there are no notifications to be processed.

As said, you can wrap that command in a Cron job so it's run every 5 minutes for example.

WND Listener Configuration

Create a service for systemd following the instructions below that checks the share for changes:

- For each WND mount point distinguished by a SERVER - SHARE pair:
  - Place one copy of a file with following content under /etc/systemd/system/owncloud-wnd-listen-SERVER-SHARE.service
  - Replace the all upper case words SERVER, SHARE, USER and PASSWORD in both, the **filename** and in the **contents** below with their respective values.
  - Take care to also adjust the paths in WorkingDirectory and ExecStart according to your installation.
  - Password: Create a file readable only by the www-data and outside the directories handled by apache (let's suppose in /tmp/mypass). The file must contain only the password for the share. In this example our file is: "/tmp/mypass". The listener will read the contents of the file and use them as the password for the account. This way, only root and the apache user should have access to the password.
  - "--password-trim" removes blank characters from the password file added by 3rdparty software or other services.

```
[Unit]
Description=ownCloud WND Listener for SERVER SHARE
After=syslog.target
After=network.target
Requires=apache2.service
[Service]
User=www-data
Group=www-data
WorkingDirectory=/var/www/owncloud
ExecStart=/usr/bin/php ./occ wnd:listen -vvv SERVER SHARE USER --password
-file=/tmp/mypass --password-trim
Type=simple
StandardOutput=journal
StandardError=journal
SyslogIdentifier=%n
KillMode=process
RestartSec=1
Restart=always
```

- Run the following command, once for each created file:

```
sudo systemctl enable owncloud-wnd-listen-SERVER-SHARE.service
sudo systemctl start  owncloud-wnd-listen-SERVER-SHARE.service
```

WND Process Queue Configuration

Create or add a crontab file in /etc/cron.d/oc-wnd-process-queue.

- Make a crontab entry to run a script iterating over all SERVER SHARE pairs with an appropriate occ wnd:process-queue command. The commands must be **strictly sequential**. This can be done by using flock -n and tuning the -c parameter of occ wnd:process-queue

```
* * * * *  sudo -u www-data /usr/bin/php /var/www/owncloud/occ wnd:process-queue
<HOST> <SHARE>
```

Execution Serialization

Parallel runs of wnd:process-queue might lead to a user lockout. The reason for this is that several wnd:process-queue might use the same wrong password because it hasn't been updated by the time they fetch it.

It's recommended to force the execution serialization of the wnd:process-queue command. You might want to use Anacron, which seems to have an option for this scenario, or wrap the command with flock.

If you need to serialize the execution of the wnd:process-queue, check the following example with flock

```
flock -n /my/lock/file {occ-command-example-prefix} wnd:process-queue <host>
<share>
```

In that case, flock will try get the lock of that file and won't run the command if it isn't possible. For our case, and considering that file isn't being used by any other process, it will run only one wnd:process-queue at a time. If someone tries to run the same command a second time while the previous one is running, the second will fail and won't be executed. Check flock's documentation for details and other options.

Troubleshooting

If you encounter issues using Windows network drive, then try the following troubleshooting steps:

First check the connection to the share by using smbclient on the command line of the ownCloud server. Here is an example:

```
smbclient -U Username -L //Servername
```

Take the example of attempting to connect to the share named MyData using occ wnd:listen. Running the following command would work:

```
{occ-command-example-prefix} wnd:listen MyHost MyData svc_owncloud password
```



The command is case sensitive, and that it must match the information from the mount point configuration.

libsmbclient Issues

If your Linux distribution ships with libsmbclient 3.x, which is included in the Samba client, you may need to set up the HOME variable in Apache to prevent a segmentation fault. If you have libsmbclient 4.1.6 and higher it doesn't seem to be an issue, so you won't have to change your HOME variable. To set up the HOME variable on Ubuntu, modify the /etc/apache2/envvars file:

```
unset HOME
export HOME=/var/www
```

In Red Hat/CentOS, modify the /etc/sysconfig/httpd file and add the following line to set the HOME variable in Apache:

```
export HOME=/usr/share/httpd
```

By default, CentOS has activated SELinux, and the httpd process can not make outgoing network connections. This will cause problems with the curl, ldap and samba libraries. You'll need to get around this to make this work. First, check the status:

```
getsebool -a | grep httpd
httpd_can_network_connect --> off
```

Then enable support for network connections:

```
setsebool -P httpd_can_network_connect 1
```

In openSUSE, modify the /usr/sbin/start_apache2 file:

```
export HOME=/var/lib/apache2
```

Restart Apache, open your ownCloud Admin page and start creating SMB/CIFS mounts.

Basic Setup for One ownCloud Server

First, go to the admin settings and set up the required WND mounts. Be aware though, that there are some limitations. These are:

- We need access to the Windows account password for the mounts to update the file cache properly. This means that "*login credentials, saved in session*" won't work with the listener. "*login credentials, saved in DB*" should work and could be the best replacement.
- The $user placeholder in the share, such as //host/$user/path/to/root, for providing a share which is accessible per/user won't work with the listener. This is because the listener won't scale, as you'll need to setup one listener per/share. As a result, you'll end up with too many listeners. An alternative is to provide a common share for the users and use the $user placeholder in the root, such as //host/share/$user/folder.

Second, start the wnd:listen process if it's not already started, ideally running it as a service. If it isn't running, no notification are stored. The listener stores the notifications. Any change in the mount point configuration, such as adding or removing new mounts, and logins by new users, won't affect the behavior, so there is no need to restart the listener in those cases.

In case you have several mount point configurations, note that each listener attaches to one host and share. If there are several mount configurations targeting different shares, you'll need to spawn one listener for each. For example, if you have one configuration with 10.0.0.2/share1 and another with 10.0.0.2/share2, you'll need to spawn 2 listeners, one for the first configuration and another for the second.

Third, run the wnd:process-queue periodically, usually via a Cron job. The command processes all the stored notifications for a specific host and share. If you have several, you could set up several Cron jobs, one for each host and share with different intervals, depending on the load or update urgency. As a simple example, you could run the command every 2 minutes for one server and every 5 minutes for another.

As said, the command processes all the stored notifications, squeeze them and scan the resulting folders. The process might crash if there are too many notifications, or if it has too many storages to update. The --chunk-size option will help by making the command process all the notifications in buckets of that size.

On the one hand the memory usage is reduced, on the other hand there is more network activity. We recommend using the option with a value high enough to process a large number of notifications, but not so large to crash the process. Between 200 and 500 should be fine, and we'll likely process all the notifications in one go.

Password Options

There are several ways to supply a password:

1. Interactively in response to a password prompt.

```
{occ-command-example-prefix} wnd:listen <host> <share> <username>
```

2. Sent as a parameter to the command.

```
{occ-command-example-prefix} wnd:listen <host> <share> <username>
<password>
```

3. Read from a file, using the --password-file switch to specify the file to read from. Note that the password must be in plain text inside the file, and neither spaces nor newline characters will be removed from the file by default, unless the --pasword -trim option is added. The password file must be readable by the apache user (or www-data)

```
{occ-command-example-prefix} wnd:listen <host> <share> <username> \
 --password-file=/my/secret/password/file
```

```
{occ-command-example-prefix} wnd:listen <host> <share> <username> \
 --password-file=/my/secret/password/file --password-trim
```



If you use the --password-file switch, the entire contents of the file will be used for the password, so please be careful with newlines.



If using --password-file make sure that the file is only readable by the apache / www-data user and inaccessible from the web. This prevents tampering or leaking of the information. The password won't be leaked to any other user using ps.

1. Using 3rd party software to store and fetch the password. When using this option, the 3rd party app needs to show the password as plaintext on standard output.

Reduce WND Notifier Memory Usage

The WND in-memory notifier for password changes provides the ability to notify all *affected* WND storages to reset their passwords. This feature is intended to prevent a password lockout for the user in the backend. However, this functionality *can* consume a significant amount of memory. To disable it, in one of the config/config.php. files, add the following configuration:

```
'wnd.in_memory_notifier.enable' => false,
```



The password will be reset on the next request, regardless of the flag setting.

3rd Party Software Examples

```
cat /tmp/plainpass | {occ-command-example-prefix} wnd:listen <host> <share>
<username> --password-file=-
```

This provides a bit more security because the /tmp/plainpass password should be owned by root and only root should be able to read the file (0400 permissions); Apache, particularly, shouldn't be able to read it. It's expected that root will be the one to run this command.

```
base64 -d /tmp/encodedpass | sudo -u www-data \
  ./occ wnd:listen <host> <share> <username> --password-file=-
```

Similar to the previous example, but this time the contents are encoded in Base64 format (there's not much security, but it has additional obfuscation).

Third party password managers can also be integrated. The only requirement is that they have to provide the password in plain text somehow. If not, additional operations might be required to get the password as plain text and inject it in the listener.

As an example:

You can use "pass" as a password manager. You can go through http://xmodulo.com/manage-passwords-command-line-linux.html to setup the keyring for whoever will fetch the password (probably root) and then use something like the following

```
pass the-password-name | {occ-command-example-prefix} wnd:listen <host>
<share> <username> --password-file=-
```

Password Option Precedence

If both the argument and the option are passed, e.g., occ wnd:listen <host> <share> <username> <password> --password-file=/tmp/pass, then the --password-file option will take precedence.

Optimizing wnd:process-queue



Do not use this option if the process-queue is fast enough. The option has some drawbacks, specifically regarding password changes in the backend.

wnd:process-queue creates all the storages that need to be updated from scratch. To do so, we need to fetch all the users from all the backends (currently only the ones that have logged in at least once because the others won't have the storages that we'll need updates).

To optimize this, wnd:process-queue make use of two switches: –serializer-type and –serializer-params. These serialize storages for later use, so that future executions don't need to fetch the users, saving precious time — especially for large organizations.

Switch	Allowed Values
--serializer-type	file. Other valid values may be added in the future, as more implementations are requested.
--serializer-params	Depends on --serializer-type, because those will be the parameters that the chosen serializer will use. For the file serializer, you need to provide a file location in the host FS where the storages will be serialized. You can use --serializer -params file=/tmp/file as an example.

While the specific behavior will depend on the serializer implementation, the overall behavior can be simplified as follows:

If the serializer's data source (such as *a file*, *a database table*, or some *Redis keys*) has storage data, it uses that data to create the storages; otherwise, it creates the storages from scratch.

After the storages are created, notifications are processed for the storages. If the storages have been created from scratch, those storages are written in the data source so that they can be read on the next run.



It's imperative to periodically clean up the data source to fetch fresh data, such as for new storages and updated passwords. There isn't a generic command to do this from ownCloud, because it depends on the specific serializer type. Though this option could be provided at some point if requested.

The File Serializer

The file serializer is a serializer implementation that can be used with the wnd:process-queue command. It requires an additional parameter where you can specify the location of the file containing the serialized storages.

There are several things you should know about this serializer:

- The generated file contains the encrypted passwords for accessing the backend. This is necessary in order to avoid re-fetching the user information, when next accessing the storages.
- The generated file is intended to be readable and writable **only** for the web server user. Other users shouldn't have access to this file. Do not manually edit the file. You can remove the file if it contains obsolete information.

Usage Recommendations

Number of Serializers

Only one file serializer should be used per server and share, as the serialized file has to be per server and share. Consider the following usage scenario:

- If you have three shares: 10.0.2.2/share1, 10.0.2.2/share2, and 10.0.10.20/share2, then you should use three different calls to wnd:process-queue, changing the target file for the serializer for each one.

Since the serialized file has to be per server and share, the serialized file has some checks to prevent misuse. Specifically, if we detect you're trying to read the storages

for another server and share from the file, the contents of the file won't be read and will fallback to creating the storage from scratch. At this point, we'll then update the contents of that file with the new storage.

Doing so, though, creates unneeded competition, where several process-queue will compete for the serializer file. For example, let's say that you have two process-queues targeting the same serializer file. After the first process creates the file the second process will notice that the file is no longer available. As a result, it will recreate the file with new content.

At this point the first process runs again and notices that the file isn't available and recreate the file again. When this happens, the serializer file's purpose isn't fulfilled As a result, we recommend the use of a different file per server and share.

File Clean Up

The file will need to cleaned up from time to time. The easiest way to do this is to remove the file when it is no longer needed. The file will be regenerated with fresh data the next execution if the serializer option is set.

Interaction Between Listener and Windows Password Lockout

Windows supports password lockout policies. If one is enabled on the server where an ownCloud share is located, and a user fails to enter their password correctly several times, they may be locked out and unable to access the share.

This is a known issue that prevents these two inter-operating correctly. Currently, the only viable solution is to ignore that feature and use the wnd:listen and wnd:process-queue, without the serializer options.

Multiple Server Setup

Setups with several servers might have some difficulties in some scenarios:

- The wnd:listen component *might* be duplicated among several servers. This shouldn't cause a problem, depending on the limitations of the underlying database engine. The supported database engines should be able to handle concurrent access and de-duplication.
- The wnd:process-queue *should* also be able to run from any server, however limitations for concurrent executions still apply. As a result, you might need to serialized command execution of the wnd:process-queue among the servers (to avoid for the password lockout), which might not be possible or difficult to achieve. You might want to execute the command from just one specific server in this case.
- wnd:process-queue + serializer. First, check the above section to know the interactions with the password lockout. Right now, the only option you have to set it up is to store the target file in a common location for all the server. We might need to provide a specific serializer for this scenario (based on Redis or DB)

Basic Command Execution Examples

```
{occ-command-example-prefix} `wnd:listen` host share username password

{occ-command-example-prefix} `wnd:process-queue` host share

{occ-command-example-prefix} `wnd:process-queue` host share -c 500

{occ-command-example-prefix} `wnd:process-queue` host share -c 500 \
   --serializer-type file \
   --serializer-params file=/opt/oc/store

{occ-command-example-prefix} `wnd:process-queue` host2 share2 -c 500 \
   --serializer-type File \
   --serializer-params file=/opt/oc/store2
```

To set it up, make sure the listener is running as a system service:

```
{occ-command-example-prefix} `wnd:listen` host share username password
```

Setup a Cron job or similar with something like the following two commands:

```
{occ-command-example-prefix} wnd:process-queue host share -c 500 \
   --serializer-type file \
   --serializer-params file=/opt/oc/store1

rm -f /opt/oc/store1 # With a different schedule
```

The first run will create the /opt/oc/store1 with the serialized storages, the rest of the executions will use that file. The second Cron job, the one removing the file, will force the wnd:process-queue to refresh the data.

It's intended to be run in a different schedule, so there are several executions of the wnd:process-queue fetching the data from the file. Note that the file can be removed manually at any time if it's needed (for example, the admin has reset some passwords, or has been notified about password changing).

Enterprise File Management

In this section you will find all the details you need to configure enterprise file management in ownCloud..

Advanced File Tagging With the Workflow App

Introduction

The Workflow App provides advanced management of file tagging. The app has three parts: Tag Manager, Automatic Tagging, and Retention.

The Workflow App should be enabled by default (Apps page), and the three configuration modules visible on your ownCloud Admin page.

See Tagging Files in the ownCloud User manual to learn how to apply and filter tags on files.

To use this functionality, administrators need to install and enable the **Collaborative Tag Management** app from Market.

The Tag Manager is for creating new tags, editing existing tags, and deleting tags. Tags may be marked as **Visible**, **Static**, **Restricted**, or **Invisible**.

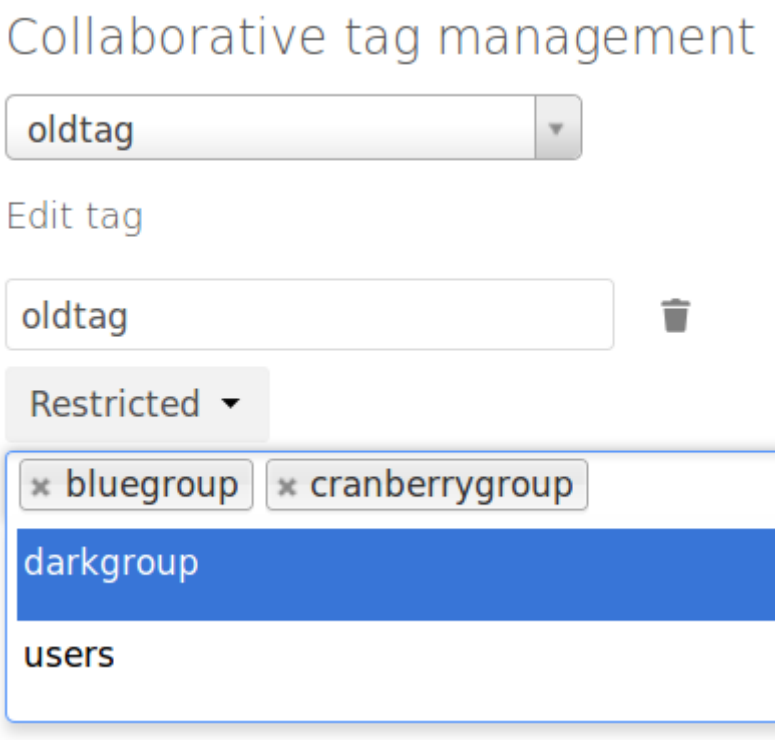
Visible
: All users may see, rename, and apply these tags to files and folders.

Static
: Only users in the specified groups can assign and un-assign the tag to a file. However, only admins can rename and edit the tag.

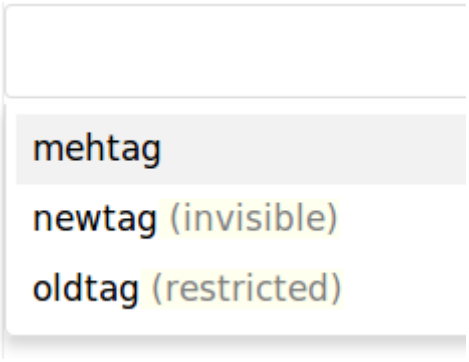
Restricted
: Tags are assignable and editable only to the user groups that you select. Other users can filter files by restricted tags, but cannot tag files with them or rename them. The tags are marked (restricted).

Invisible
: Tags are visible only to ownCloud admins.

To access this functionality, select menu:Settings[Admin > Workflow & Tags].



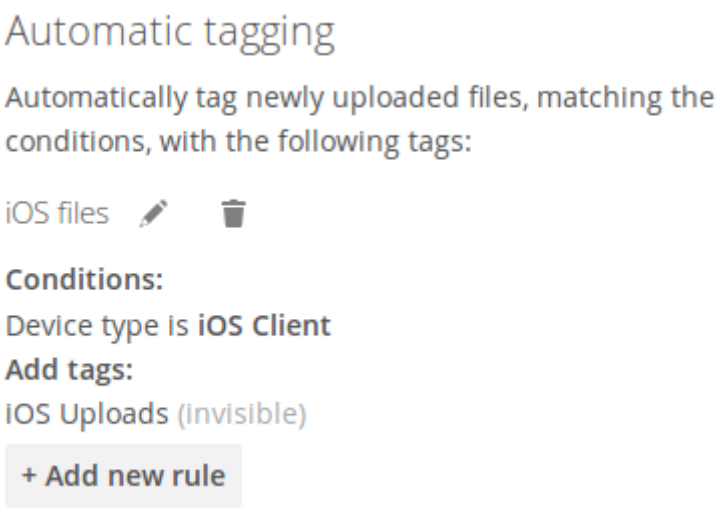
This is an example of what your tags look like in the **Tags** view on your files page. Non-admin users will not see invisible tags, but they will see visible and restricted tags.



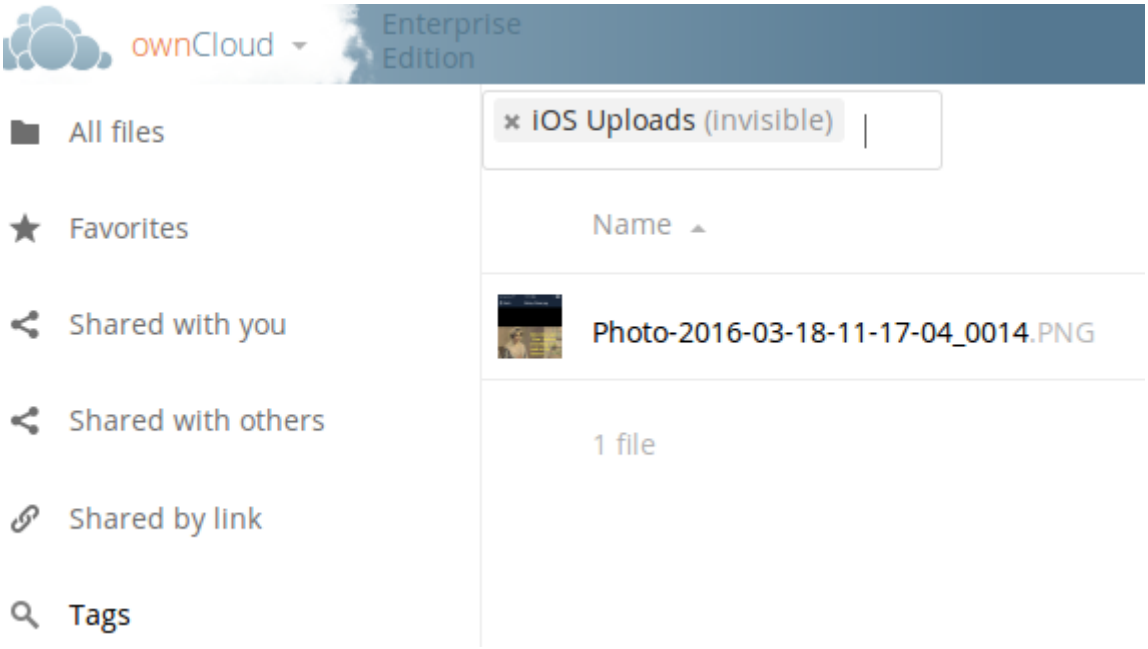
Automatic Tagging

The Automatic Tagging module operates on newly-uploaded files. Create a set of conditions, and then when a file or folder matches those conditions it is automatically tagged. The tag must already have been created with the Tag Manager.

For example, you can assign the invisible tag **iOS Uploads** to all files uploaded from iOS devices. This tag is visible only to admins.



When files with this tag are shared with you, you can view them with the Tags filter on the Files page.

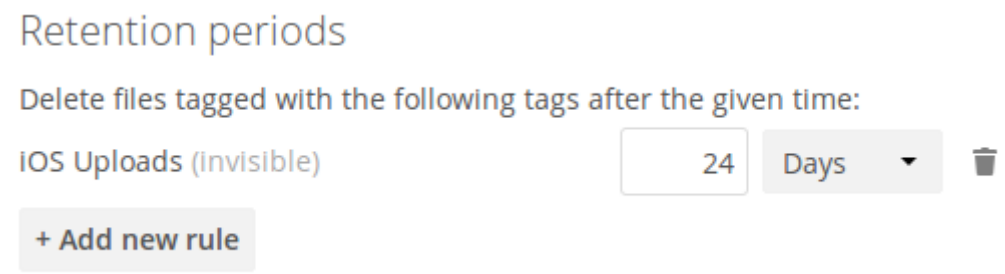


Automatic Tagging is especially useful with the Retention module.

Retention

The Retention module is your housecleaning power tool, because it automatically deletes files after a time period that you specify. Select which tag to set a time limit on, and then set your time limit. File age is calculated from the file mtime (modification time).

	ownCloud does not preserve directory mtimes (modification time), though it does update file mtimes.



For best performance, retention tags should be applied high in your file hierarchy. If subfolders have the same tags as their parent folders, their tags must also be processed, so it will take a little longer.

Retention Engines

There are two retention engines that further allow you to fine-tune your retention settings: **TagBasedRetention** and **UserBasedRetention**. **TagBasedRetention** is the default.

TagBasedRetention: This checks files that have a particular tag assigned. Then it checks (depth-first) the children of the tagged item, before continuing with the other tagged items. Children that have already been checked will not be checked a second time.

This is optimised for processing smaller numbers of files that have multiple retention tags.

UserBasedRetention: Examines files per user. It first iterates over all files and folders (siblings first), then examines the tags for those items and checks their respective retention periods. This is optimised for many files with few retention tags.

To select UserBasedRetention, add this line to your ee.config.php:

```
'workflow.retention_engine' => userbased,
```

Enterprise Firewall Configuration

In this section you will find all the details you need to configure enterprise firewall configuration in ownCloud appliance..

File Firewall

Introduction

The File Firewall GUI enables you to manage firewall rule sets. You can find it in your ownCloud admin page, under menu:Admin[Security]. The File Firewall lets you control access and sharing in fine detail, by creating rules for allowing or denying access restrictions based on: *group*, *upload size*, *client devices*, *IP address*, *time of day*, as well as many more criteria. In addition to these restriction options, the File Firewall app also supports rules based on regular expressions.

How the File Firewall Works

Each firewall rule set consists of one or more conditions. If a request matches all of the conditions, in at least one rule set, then the request is blocked by the firewall. Otherwise, the request is allowed by the firewall.



The File Firewall app cannot lock out administrators from the web interface when rules are misconfigured.

Using the File Firewall

Figure 1 shows an empty firewall configuration panel. Set your logging level to **Blocked Requests Only** for debugging, and create a new rule set by clicking btn:[Add Group]. After setting up your rules you must click btn:[Save Rules].

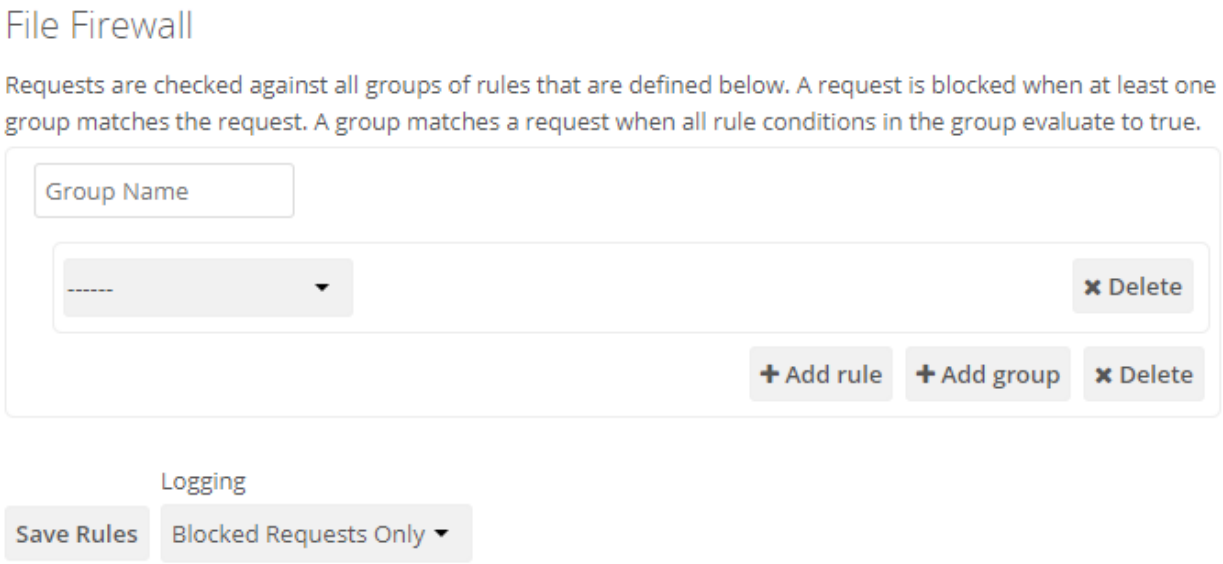
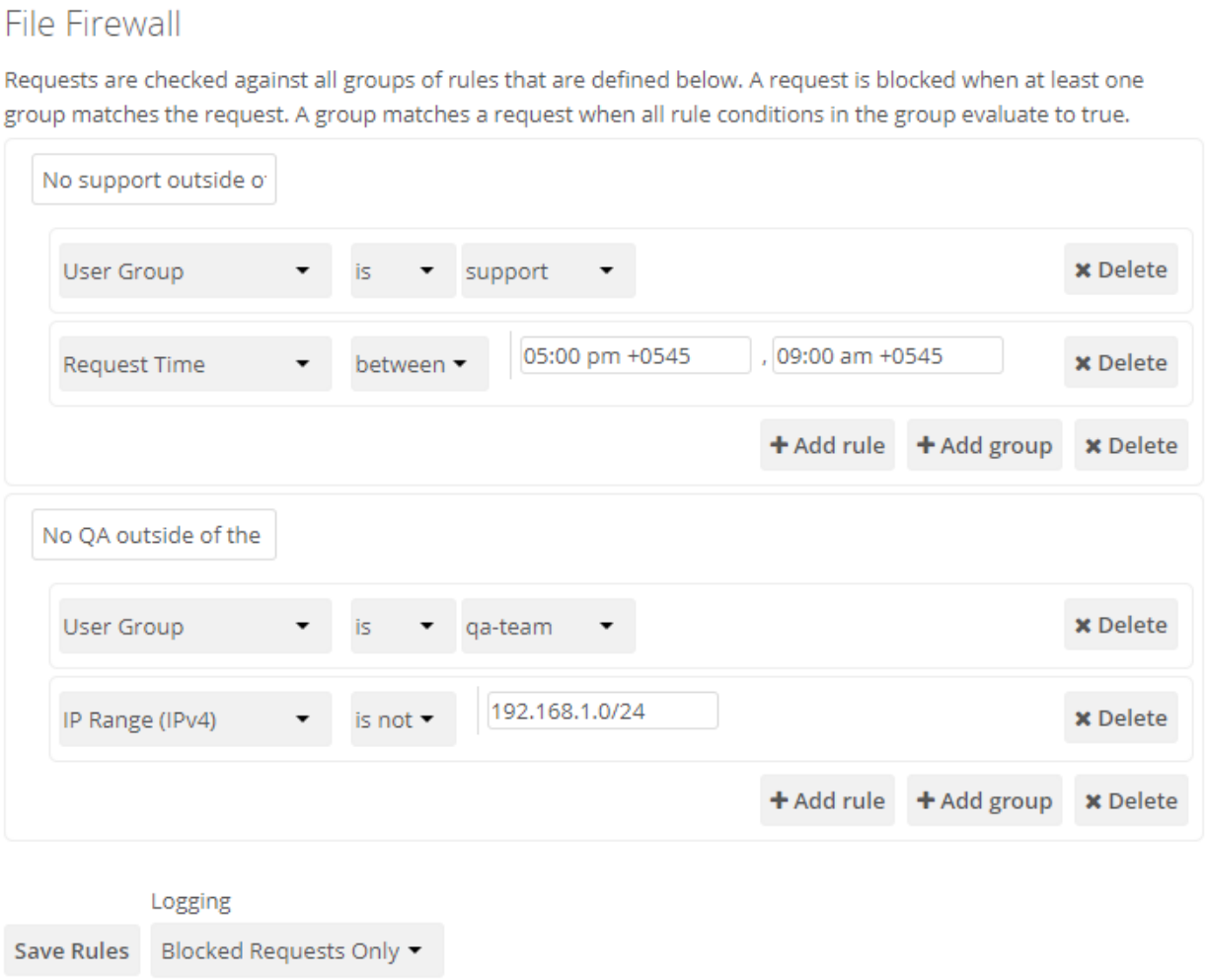


Figure 2 shows two rules. The first rule, **No Support outside office hours**, prevents

members of the support group from logging into the ownCloud Web interface from 5pm-9am, and also blocks client syncing. The second rule prevents members of the "qa-team" group from accessing the Web UI from IP addresses that are outside of the local network.



All other users are not affected, and can log in anytime from anywhere.

Available Conditions

User Group

The user (is|is not) a member of the selected group.

User Agent

The User-Agent of the request (matches|does not match) the given string.

User Device

A shortcut for matching all known (android | ios | desktop) sync clients by their User Agent string.

Request Time

The time of the request (has to|must not) be in a single range from beginning time to end time.

Request URL

The **full page URL** (has to contain) with a given string.

Request Type

The request (is a public link share|other) request.

Request IP Range (IPv4) and IP Range (IPv6)

The request's REMOTE_ADDR header (is|is not) matching the given IP range.

File Size Upload

When a file is uploaded the size has to be (less|greater or equal) to the given size.

File Mimetype Upload

Block a request based on the mimetype of a file being uploaded. The match can be the complete mimetype, part of the mimetype from the start or end of the mimetype. Negative matches are also supported; i.e., all mimetypes that don't match the supplied mimetype, or all mimetypes that don't start or end with the partial mimetype supplied.

The full list of conditions is (File mimetype upload):

- is
- is not
- begins with
- doesn't begin with
- ends with
- doesn't end with



The complete list of available mimetypes which ownCloud supports is available in the ownCloud core source.

Common Mimetypes

suffix	mimetype
avi	video/x-msvideo
exe	application/x-ms-dos-executable
flv	video/x-flv
mp4	video/mp4
mkv	video/x-matroska
msi	application/x-msi
php	application/x-php

System File Tag

One of the parent folders or the file itself (is|is not) tagged with a System tag.

Regular Expression

The File Firewall supports regular expressions, allowing you to create custom rules using the following conditions:

- IP Range (IPv4)
- IP Range (IPv6)
- User agent
- User group
- Request URL

You can combine multiple rules into one rule, e.g., if a rule applies to both the support and the qa-team you could write your rule like this:

```
Regular Expression > ^(support|qa-team)$ > is > User group
```



We do not recommend modifying the configuration values directly in your config.php. These use JSON encoding, so the values are difficult to read and a single typo will break all of your rules.

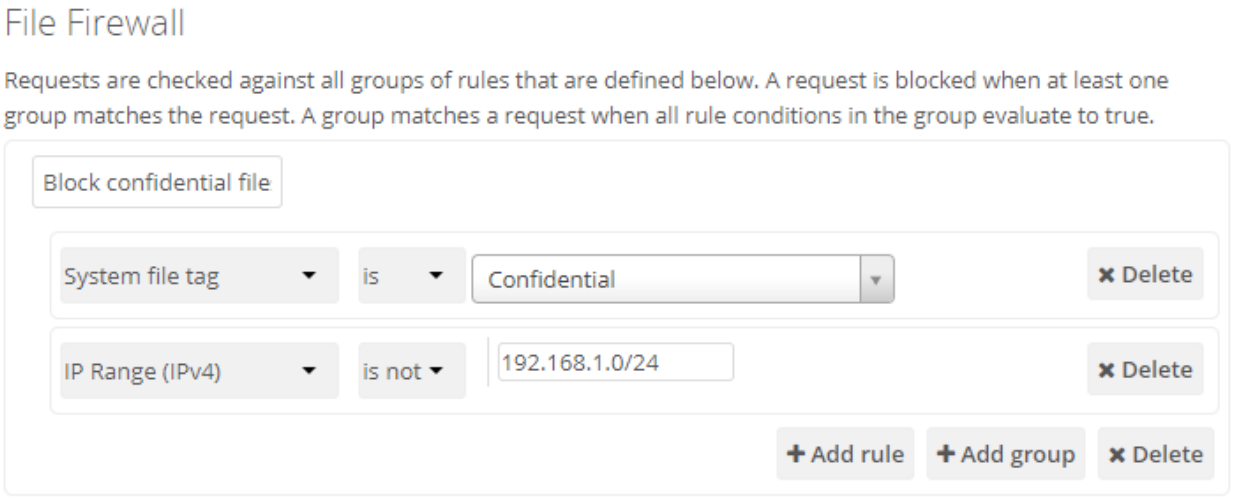
Controlling Access to Folders

The easiest way to block access to a folder, starting with ownCloud 9.0, is to use a system tag. A new rule type was added which allows you to block access to files and folders, where at least one of the parents has a given tag.

Now you just need to add the tag to the folder or file, and then block the tag with the File Firewall. This example blocks access to any folder with the tag "Confidential" from outside access.

Block by System Tag:

```
System file tag:   is      "Confidential"
IP Range (IPv4):   is not   "192.168.1.0/24"
```



Logging

Firewall logging can be set to **Off**, **Blocked Requests Only** or **All Requests**

Off

The firewall blocks requests according to the defined rules but does not log any of its actions.

Blocked Requests Only

The firewall logs blocked requests to the system log at **warning** level. To see these logs, the system log level must be set to a minimum level of **warning**.

All Requests

The firewall logs blocked and successful requests to the system log at **warning** and **info** levels respectively. To see all these logs, the system log level must be set to a minimum level of **info**.



Logging all requests can generate a large amount of log data. It is recommended to only select all requests for short-term checking of rule settings.

Custom Configuration for Branded Clients

If you are using branded ownCloud clients, you may define firewall.branded_clients in your config.php to identify your branded clients in the firewall **"User Device"** rule.

The configuration is a User-Agent ⇒ Device map. Device must be one of the following:

- android
- android_branded
- ios
- ios_branded
- desktop
- desktop_branded

The User-Agent is always compared all lowercase. By default the agent is compared with equals. When a trailing or leading asterisk, , is found, the agent is compared with starts with or ends with. If the agent has both a leading and a trailing , the string must appear anywhere. For technical reasons the User-Agent string must be at least 4 characters, including wildcards. When you build your branded client you have the option to create a custom User Agent.

In this example configuration you need to replace the example User Agent strings, for example 'android_branded', with your own User Agent strings:

```
// config.php

'firewall.branded_clients' => array(
  'my ownbrander android user agent string' => 'android_branded',
  'my ownbrander second android user agent string' => 'android_branded',
  'my ownbrander ios user agent string' => 'ios_branded',
  'my ownbrander second ios user agent string' => 'ios_branded',
  'my ownbrander desktop user agent string' => 'desktop_branded',
  'my ownbrander second desktop user agent string' => 'desktop_branded',
),
```

The Web UI dropdown then expands to the following options:

- Android Client - always visible
- iOS Client - always visible
- Desktop Client - always visible
- Android Client (Branded) - visible when at least one android_branded is defined
- iOS Client (Branded) - visible when at least one ios_branded is defined
- Desktop Client (Branded) - visible when at least one desktop_branded is defined
- All branded clients - visible when at least one of android_branded, ios_branded or desktop_branded is defined
- All non-branded clients - visible when at least one of android_branded, ios_branded or desktop_branded is defined
- Others (Browsers, etc.) - always visible

Then these options operate this way:

- The * Client options only match android, ios and desktop respectively.
- The * Client (Branded) options match the *_branded agents equivalent.
- All branded clients matches: android_branded, ios_branded and desktop_branded
- All non-branded clients matches: android, ios and desktop

Installing & Upgrading ownCloud Enterprise Edition

Introduction

After you have completed your initial installation of ownCloud as detailed in the README, follow the instructions in The Installation Wizard to finish setting up ownCloud. To upgrade your Enterprise server, refer to How to Upgrade Your ownCloud Server.

Manual Installation

Download the ownCloud archive from your account at https://customer.owncloud.com/owncloud, then follow the instructions at Manual Installation on Linux.

SELinux

Linux distributions that use SELinux need to take some extra steps so that ownCloud will operate correctly under SELinux. Please see SELinux Configuration for some recommended configurations.

License Keys

Introduction

You'll need to install a license key to use ownCloud Enterprise Edition. There are two types of license keys: one is a free 30-day trial key. The other is a full license key for Enterprise customers.

You can download and try ownCloud Enterprise for 30 days for free, which auto-generates a free 30-day key. When this key expires your ownCloud installation is not removed, so when you become an Enterprise customer you can enter your new key to regain access. See How to Buy ownCloud for sales and contact information.

Configuration

Once you get your Enterprise license key, it needs to be copied to your ownCloud configuration file, config/config.php file like this example:

```
'license-key' => 'test-20150101-
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX-YYYYYY',
```

Each running instance of ownCloud requires a license key. Keys will work across upgrades without issue, so new keys will not be required when you upgrade your ownCloud Enterprise to a new version.

Supported ownCloud Enterprise Edition Apps

See Supported Apps in ownCloud for a list of supported apps.



3rd party and unsupported apps must be disabled before performing a system upgrade. Then install the upgraded versions, and after the upgrade is complete re-enable them.

Oracle Database Setup & Configuration

Introduction

This document will cover the setup and preparation of the ownCloud server to support the use of Oracle as a backend database.

Outline of Steps

This document will cover the following steps:

- Setup of the ownCloud user in Oracle: This involves setting up a user space in Oracle for setting up the ownCloud database.
- Installing the Oracle Instant Client on the Web server (facilitating the connection to the Oracle Database).
- Compiling and installing the Oracle PHP Plugin oci8 module
- Pointing ownCloud at the Oracle database in the initial setup process

The document assumes that you already have your Oracle instance running, and have provisioned the needed resources. It also assumes that you have installed ownCloud with all of the prerequisites.

Configuring Oracle

Setting up the User Space for ownCloud

Step one, if it has not already been completed by your DBA (DataBase Administrator), provision a user space on the Oracle instance for ownCloud. This can be done by logging in as a DBA and running the script below:

```
CREATE USER owncloud IDENTIFIED BY password;
ALTER USER owncloud DEFAULT TABLESPACE users TEMPORARY TABLESPACE
temp QUOTA unlimited ON users;
GRANT create session, create table, create procedure, create sequence,
create trigger, create view, create synonym, alter session TO owncloud;
```

Substitute an actual password for password. Items like *TableSpace*, *Quota* etc., will be determined by your DBA (database administrator).

Add OCI8 Client Packages

Installation of the OCI8 client is dependent on your distribution. Given that, please use the relevant section below to find the relevant instructions to install the client.

Ubuntu

If you're using Ubuntu, we recommend that you use this very thorough guide from the Ubuntu Community Wiki to install the OCI8 extension.



This *should* work for other Debian-based distributions, however your mileage may vary.

RedHat / Centos / Fedora

To install the OCI8 extension on a RedHat-based distribution, you first need to download two Oracle Instant Client packages:

- Instant Client Package - Basic (oracle-instantclient12.2-basic-12.2.0.1.0-1.x86_64.rpm)
- Instant Client Package - SDK (oracle-instantclient12.2-devel-12.2.0.1.0-1.x86_64.rpm)

Then, to install them, use the following commands:

```
rpm --install oracle-instantclient12.2-basic-12.2.0.1.0-1.x86_64.rpm \
  oracle-instantclient12.2-devel-12.2.0.1.0-1.x86_64.rpm
```

Install the OCI8 PHP Extension

With the Oracle packages installed you're now ready to install PHP's OCI8 extension.



Provide: instantclient,/usr/lib/oracle/12.2/client64/lib when requested, or let it auto-detect the location (if possible).

```
pecl install oci8
```

With the extension installed, you now need to configure it, by creating a configuration file for it. You can do so using the command below, substituting FILE_PATH with one from the list below the command.

```
cat << EOF > FILE_PATH
; Oracle Instant Client Shared Object extension
extension=oci8.so
EOF
```

Configuration File Paths

Debian & Ubuntu

PHP Version	Filename
{minimum-php-version}	/etc/php/{minimum-php-version}/apache2/conf.d/20-oci.ini

RedHat, Centos, & Fedora

PHP Version	Filename
{minimum-php-version}	/etc/opt/rh/rh-php{minimum-php-version-short-code}/php.d/20-oci8.ini

Validating the Extension

With all that done, confirm that it's been installed and available in your PHP distribution, run the following command:

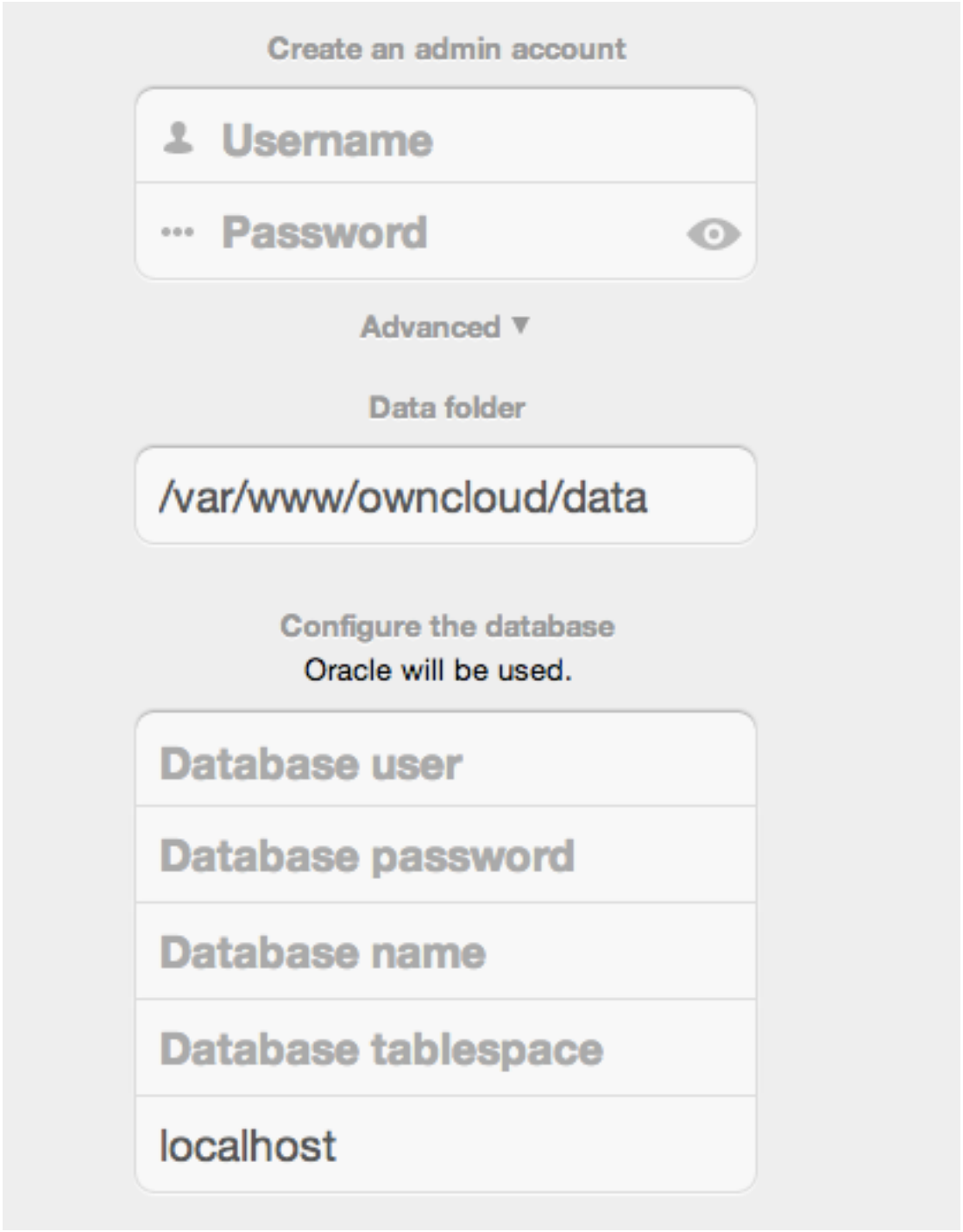
```
php -m | grep -i oci8
```

When the process has completed, assuming that you don't encounter any errors, restart Apache and the extension is ready to use.

Configure ownCloud

The next step is to configure the ownCloud instance to point to the Oracle Database, again this document assumes that ownCloud has previously been installed.

Configuration Wizard



Database user

This is the user space created in step 2.1. In our Example this would be owncloud.

Database password

Again this is defined in the script from section 2.1 above, or pre-configured and provided to you by your DBA.

Database Name

Represents the database or the service that has been pre-configured on the TSN Listener on the Database Server. This should also be provided by the DBA. In this example, the default setup in the Oracle install was orcl (there is a TSN Listener entry for orcl on our database server).

This is not like setting up with MySQL or SQL Server, where a database based on the

name you give is created. The oci8 code will call this specific service and it must be active on the TSN Listener on your Oracle Database server.

Database Table Space

Provided by the DBA. In this example the users table space (as is seen in the user creation script above), was used.

Configuration File

Assuming all of the steps have been followed to completion, the first run wizard should complete successfully, and an operating instance of ownCloud should appear.

The configuration file should look something like this:

Useful SQL Commands

Is my Database Reachable?

On the machine where your Oracle database is installed, type:

```
sqlplus username
```

```
SQL> select * from v$version;

BANNER
--------------------------------------------------------------------------------
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
PL/SQL Release 11.2.0.2.0 - Production
CORE  11.2.0.2.0  Production
TNS for Linux: Version 11.2.0.2.0 - Production
NLSRTL Version 11.2.0.2.0 - Production

SQL> exit
```

Show Database Users:

```
Oracle    : SELECT * FROM all_users;
```

Show available Databases:

```
Oracle    : SELECT name FROM v$database; (requires DBA privileges)
```

Show ownCloud Tables in Database:

```
Oracle    : SELECT table_name FROM user_tables;
```

Quit Database:

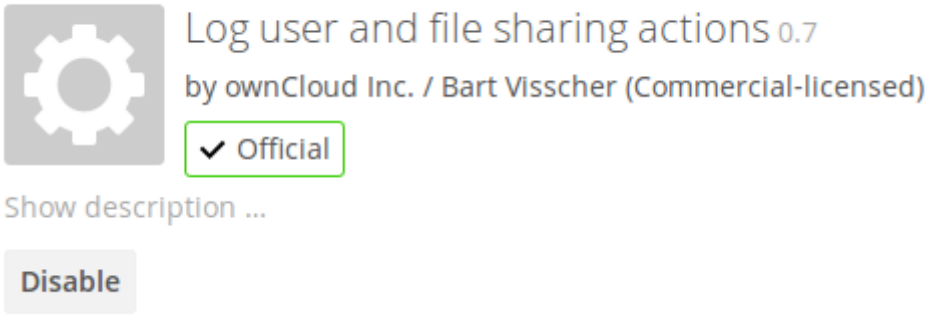
```
Oracle    : quit
```

Enterprise Logging Configuration

In this section you will find all the details you need to configure enterprise logging configuration in ownCloud.

Enterprise Logging Apps

The **Log user and file sharing actions** app (apps/admin_audit) records the file sharing activity of your users, file tagging, and user logins and logouts.



Your logging level must be set to at least **Info, warnings, errors, and fatal issues** on your ownCloud admin page, or 'loglevel' ⇒ 1 in config.php.

View your logfiles on your admin page. Click the **Download logfile** button to dump the plain text log, or open the logfile directly in a text editor. The default location is owncloud/data/owncloud.log.



See Logging Configuration and File Tagging for more information on logging and tagging.

Enterprise Security

In this section you will find all the details you need to configure enterprise security in ownCloud.

Ransomware Protection

Introduction

Ransomware is an ever-present threat, both for large enterprises as well as for individuals. Once infected, a whole hard disk (or just parts of it) can become encrypted, leading to unrecoverable data loss.

Once this happens, attackers usually ask victims to pay a ransom, often via cryptocurrencies such as Bitcoin, in exchange for the decryption key required to decrypt their data.

While paying the ransom works in some cases, it is not recommended, as there is no guarantee that the attackers will supply the key after payment is made. To help mitigate such threats and ensure ongoing access to user data, ownCloud provides the Ransomware Protection app.



It is essential to be aware that user data needs to be synchronized with you ownCloud Server using the ownCloud Desktop synchronization client. Data that is not synchronized and stored in ownCloud cannot be protected.

About Ransomware Protection

The app is tasked with *detecting*, *preventing*, and *reverting* anomalies. Anomalies are file operations (including *create*, *update*, *delete*, and *move*) not intentionally conducted by the user. It aims to do so in two ways: prevention, and protection.

Prevention: Blocking Common Ransomware File Extensions

Like other forms of cyberattack, ransomware has a range of diverse characteristics. On the one hand it makes them hard to detect and on the other it makes them even harder to prevent. Recent ransomware attacks either encrypt a user's files and add a specific file extension to them (e.g., .crypt), or they replace the original files with an encrypted copy and add a particular file extension.

File Extension Blacklist

The first line of defense against such threats is a blacklist that blocks write access to file extensions known to originate from ransomware.

Ransomware Protection ships with a static extension list of more than 3,000 file extensions. As new extensions are regularly created, this list also needs to be regularly reviewed and updated. Future releases of Ransomware Protection will include an updated list and the ability to update the list via syncing with FSRM's API by using an occ command



Please check the provided ransomware blacklist! It is **strongly recommended** to check the provided ransomware blacklist to ensure that it fits your needs. In some cases, the patterns might be too generic and result in false positives.

File Blocking

The second line of defense is file blocking. As files are uploaded, they are compared against the file extension blacklist. If a match is found, the upload is denied.



File blocking is always enabled.

Account Locking

The third line of defense is account locking. If a client uploads a file matching a pattern in the ransomware blacklist, the account is locked (set as read-only) for client access (*create*, *change*, *move*, and *delete* operations). Doing this prevents further, malicious, changes.

Following this, clients receive an error (403 Access Forbidden) which notifies the user that the account is locked by Ransomware Protection.



Write access (e.g., moving and deleting files) is still possible for users when they log in with their web browser.

When an account is locked, administrators can unlock the account using the occ ransomguard:unlock command. Administrators can also manually lock user accounts, using the occ ransomguard:lock command.



When an account is locked, it will still be fully usable from the ownCloud web UI. However, ownCloud clients (as well as other WebDAV clients) will see the account as set to read-only mode.

Users will see a yellow notification banner in the ownCloud web UI directing them to menu:Settings[Personal > Security] (*Ransomware detected: Your account is locked (read-only) for client access to protect your data. Click here to unlock.*), where additional information is displayed and users can unlock their account when ransomware issues are resolved locally.



Locking is enabled by default. If this is not desired, an administrator can disable it in the menu:Settings[Admin > Security] panel.

Protection: Data Retention and Rollback

While Ransomware Prevention mitigates risks of a range of ransomware attacks, it is not a future-proof solution, because ransomware is becoming ever-more sophisticated. There are known attacks that change file extensions randomly or keep them unchanged which makes them harder to detect.

Ultimately there is a consensus that only one solution can provide future-proof protection from ransomware attacks: retaining data and providing the means to roll back to a particular point in time.

ownCloud Ransomware Protection will, therefore, record all changes on an ownCloud Server and allow administrators to rollback user data to a particular point in time, making use of ownCloud's integrated Versioning and Trash bin features.

Doing so allows all user data that is synchronized with the server to be rolled back to its state before the attack occurred. A combination of Ransomware prevention and protection reduces risks to a minimum acceptable level.

Other Elements of Ransomware Protection

Name	Command (if applicable)	Description
Ransomware Prevention (Blocker)		First line of defense against ransomware attacks. Ransomware Protection uses a file name pattern blacklist to prevent uploading files that have file extensions associated with ransomware (e.g. .crypt) thereby preserving the original files on the ownCloud Server.
Ransomguard Scanner	occ ransomguard:scan <timestamp> <user>	A command to scan the ownCloud database for changes in order to discover anomalies in a user's account and their origin. It enables an administrator to determine the point in time where undesired actions happened as a prerequisite for restoration.

Name	Command (if applicable)	Description
Ransomguard Restorer	occ ransomguard:restore <timestamp> <user>	A command for administrators to revert all operations in a user account that occurred after a certain point in time.
Ransomguard Lock	occ ransomguard:lock <user>	Set a user account as read-only for ownCloud and other WebDAV clients. This prevents any further changes to the account.
Ransomguard Unlock	occ ransomguard:unlock <user>	Unlock a user account which was set to read-only.

<timestamp> must be in the Linux timestamp format.

Requirements

Mandatory

1. **File Firewall rule (previous approach for ransomware protection).** If you have configured the File Firewall rule which was provided as a preliminary protection mechanism, please remove it. The functionality (Blocking) is covered by Ransomware Protection in an improved way.
2. **Ransomware Protection.** Ransomware protection needs to be in operation before an attack occurs, as it needs to record file operations to be able to revert them, in case of an attack.
3. **ownCloud Versions App.** Required to restore older file versions. The capabilities of Ransomware Protection depend on its configuration regarding version retention.
4. **ownCloud Trash Bin App.** Required to restore deleted files. The capabilities of Ransomware Protection depend on its configuration regarding trash bin retention.

Optional

1. **Activity app.** For viewing activity logs.

Limitations

- Ransomware Protection works with master-key based storage encryption. With credential-based storage encryption, only Ransomware Prevention (Blocking) works.
- Rollback is not based on snapshots:
  - The trash bin retention policy may delete files, making them unrecoverable. To avoid this, set trashbin_retention_obligation to disabled, or choose a conservative policy for trash bin retention. However, please be aware that this may increase storage requirements.
  - Trash bin items may be deleted by the user making them unrecoverable by Ransomware Protection ⇒ Users need to know this.
  - Versions have a built-in thin-out policy which makes it possible that required file versions are unrecoverable by Ransomware Protection. To help avoid this, set versions_retention_obligation to disabled or choose a conservative policy for version retention. Please be aware that this might increase your storage needs.
- A specific version of a file that is needed for rollback might have been manually restored, making this version potentially unrecoverable by Ransomware Protection.

Currently, after restoration the restored version is not a version anymore, e.g., the version is not present in versioning.

- Recovery capabilities in received shared folders are currently limited. Changed file contents and deletions can be restored but MOVE operations can't. The case when a ransomware attack renames files in a received shared folder is therefore not yet covered.
- Contents in secondary storages, such as *Windows network drives*, *Dropbox*, and *Google Drive*, are unrecoverable by Ransomware Protection, because they do not have versioning or trash bin enabled in ownCloud.
- Rolling files forward is not *currently* supported or tested. Therefore it is vital to:
  - Carefully decide the point in time to rollback to.
  - To have proper backups to be able to conduct the rollback again, if necessary.

Enterprise Server Branding

In this section you will find all the details you need to configure enterprise server branding in ownCloud.

Enterprise Server Branding

ownBrander is an ownCloud build service that is exclusive to Enterprise edition customers for creating branded ownCloud clients and servers. You may brand your ownCloud server using ownBrander to easily build a custom theme, using your own logo and artwork. ownCloud has always been theme-able, but it was a manual process that required editing CSS and PHP files. Now Enterprise customers can use ownBrander, which provides an easy graphical wizard.

You need an Enterprise subscription, an account on customer.owncloud.com, and the ownBrander app enabled on your account. When you complete the steps in the wizard the ownBrander service builds your new branded theme, and in 24-48 hours you'll see it in your account.

When you open the ownBrander app, go to the Web tab. You will see an introduction and the wizard, which starts with uploading your logo. You will need a number of images in specific sizes and formats, and the wizard tells you what you need. Example images are on the right, and you can click to enlarge them.

If you see errors when you upload SVG files, such as "Incorrect extension. File type image/svg+xml is not correct", "This SVG is invalid", or "Error uploading file: Incorrect size", try opening the file in Inkscape then save as "Plain SVG" and upload your SVG image again.

The wizard has two sections. The first section contains all the required elements: logos and other artwork, colors, naming, and your enterprise URL. The Suggested section contains optional items such as additional logo placements and custom URLs.

When you are finished, click the **Generate Web Server** button. If you want to change anything, go ahead and change it and click the **Generate Web Server** button. This will override your previous version, if it has not been created yet.In 24-48 hours you'll find your new branded theme in the **Web** folder in your Customer.owncloud.com account.

Inside the **Web** folder you'll find a **themes** folder. Copy this to your owncloud/themes directory. You may name your **themes** folder anything you want, for example

myBrandedTheme. Then configure your ownCloud server to use your branded theme by entering it in your config.php file:

```
"theme" => "myBrandedTheme"
```

If anything goes wrong with your new theme, comment out this line to re-enable the default theme until you fix your branded theme. The branded theme follows the same file structure as the default theme, and you may further customize it by editing the source files.

Always edit only your custom theme files. Never edit the default theme files.

Enterprise User Management

In this section you will find all the details you need to configure enterprise user management in ownCloud.

- Shibboleth Integration
- SAML 2.0 Based SSO

Shibboleth Integration

Introduction

The ownCloud Shibboleth user backend application integrates ownCloud with a Shibboleth Service Provider (SP) and allows operations in federated and single-sign-on (SSO) infrastructures. Setting up Shibboleth has two big steps:

1. Enable and configure the Apache Shibboleth module.
2. Enable and configure the ownCloud Shibboleth app.

The Apache Shibboleth module

Currently supported installations are based on the native Apache integration. The individual configuration of the service provider is highly dependent on the operating system, as well as on the integration with the Identity Providers (IdP), and require case-by-case analysis and installation.

A good starting point for the service provider installation can be found in the official Shibboleth Wiki.

A successful installation and configuration will populate Apache environment variables with at least a unique user id which is then used by the ownCloud Shibboleth app to login a user.

Apache Configuration

This is an example configuration as installed and operated on a Linux server running the Apache 2.4 Web server. These configurations are highly operating system specific and require a high degree of customization.

The ownCloud instance itself is installed in /var/www/owncloud/. Further Shibboleth specific configuration as defined in /etc/apache2/conf.d/shib.conf.

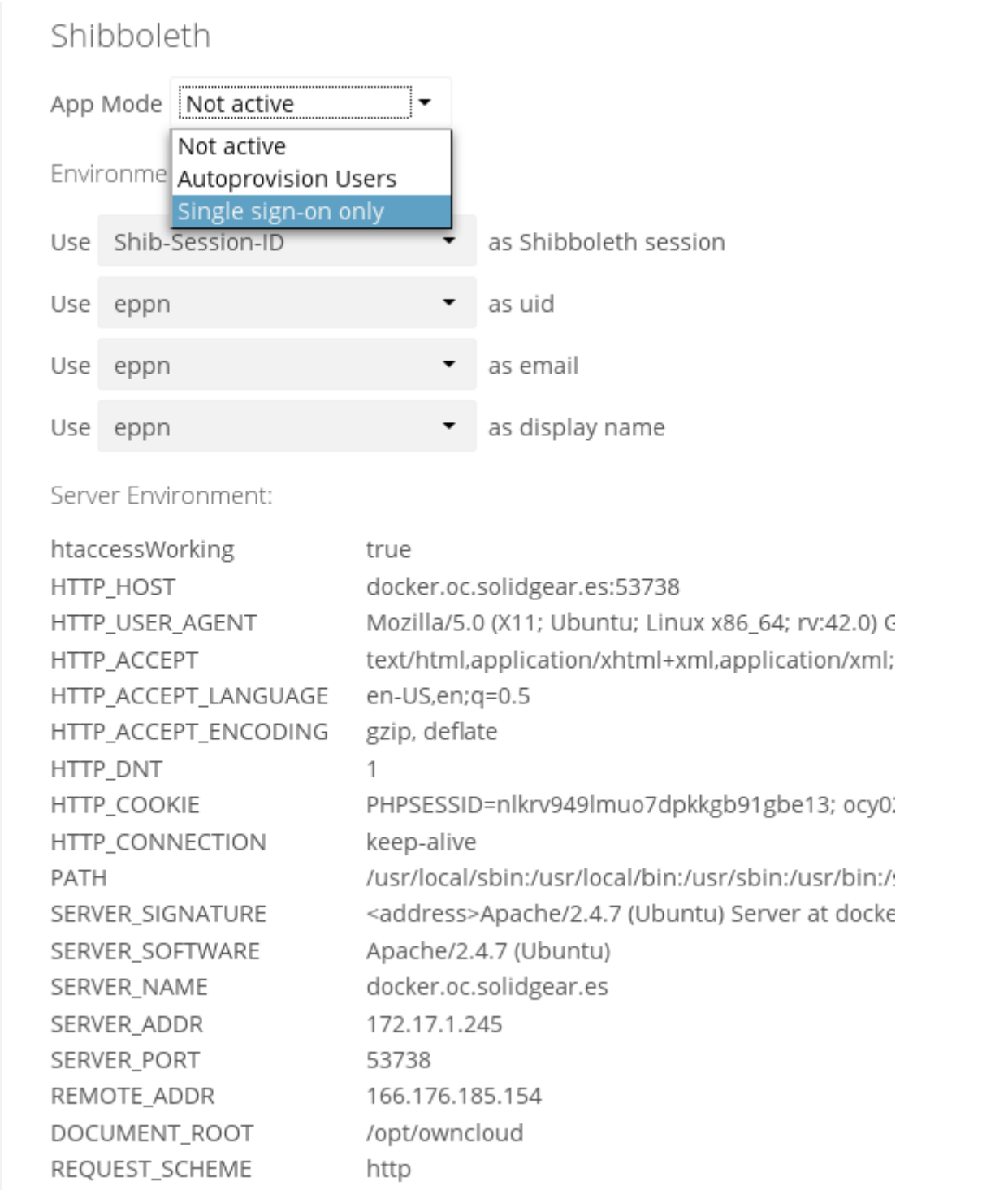
```
# Load the Shibboleth module.
LoadModule mod_shib /usr/lib64/shibboleth/mod_shib_24.so

# Ensure handler will be accessible
<Location /Shibboleth.sso>
   AuthType None
   Require all granted
</Location>

# always fill env with shib variable for logout url
<Location />
   AuthType shibboleth
   ShibRequestSetting requireSession false
   Require shibboleth
</Location>

# authenticate only on the login page
<Location ~ "^(/index.php)?/login">
   # force internal users to use the IdP
   <If "-R '192.168.1.0/24'">
      AuthType shibboleth
      ShibRequestSetting requireSession true
      require valid-user
   </If>
   # allow basic auth for eg. guest accounts
   <Else>
      AuthType shibboleth
      ShibRequestSetting requireSession false
      require shibboleth
   </Else>
</Location>

# shib session for css, js and woff not needed
#
# WARNING!!!: The following lines could potentially override other location
statements
# made in other Apache config-files depending on include-order.
# Please double-check your Apache config by consulting the Apache debug-log.
<Location ~ "/.*\.(css|js|woff)">
   AuthType None
   Require all granted
</Location>
```

To allow users to login via the IdP, add a login alternative with the login.alternatives option in config/config.php. Depending on the ownCloud Shibboleth app mode, you may need to revisit this configuration.

The ownCloud Shibboleth App

After enabling the Shibboleth app on your Apps page, you need to choose the app

mode and map the necessary Shibboleth environment variables to ownCloud user attributes on your Admin page.



Choosing the App Mode

After enabling the app it will be in **Not active** mode, which ignores a Shibboleth session and allows you to login as an administrator and inspect the currently available Apache environment variables. Use this mode to set up the environment mapping for the other modes, and in case you locked yourself out of the system. You can also change the app mode and environment mappings by using the occ command, like this example on Ubuntu Linux:

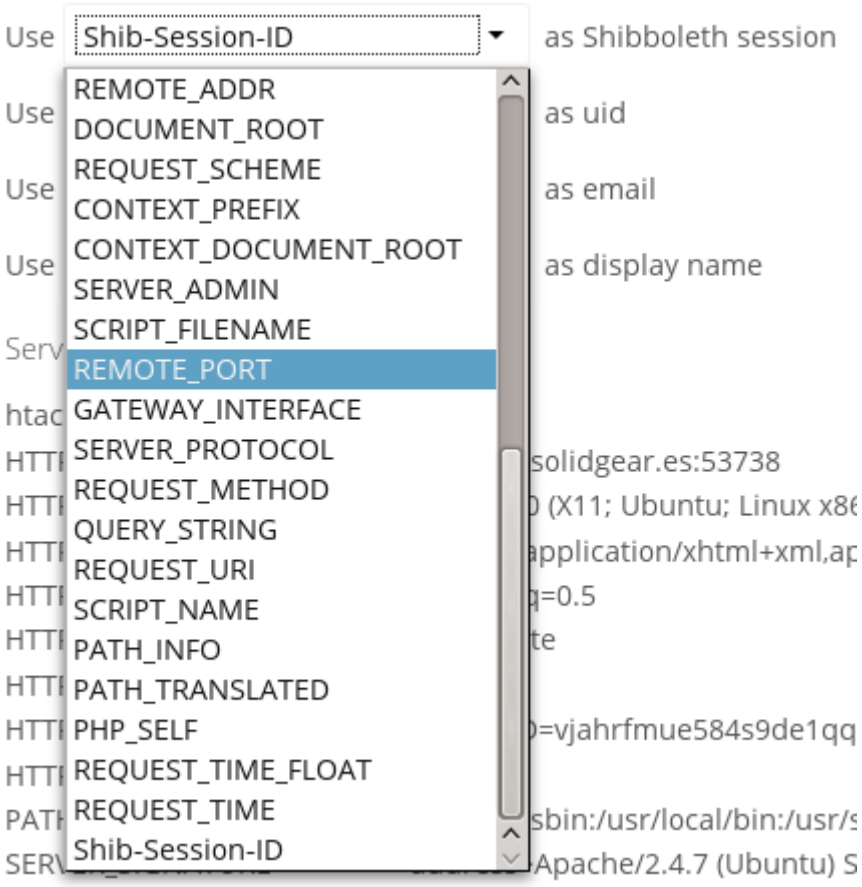
```
{occ-command-example-prefix} shibboleth:mode notactive
{occ-command-example-prefix} shibboleth:mapping --uid login
```

In **Single sign-on only** mode the app checks if the environment variable for the Shibboleth session, by default **Shib-Session-Id**, is set. If that is the case it will take the value of the environment variable as the uid, by default eppn, and check if a user is known by that uid. In effect, this allows another user backend, e.g., the LDAP app, to provide the displayname, email and avatar.

As an example the IdP can send the userPrincipalName which the Apache Shibboleth module writes to a custom Apache environment variable called login. The ownCloud Shibboleth app reads that login environment variable and tries to find an LDAP user with that username. For this to work userPrincipalName needs to be added to the **Additional Search Attributes** in the LDAP directory settings on the advanced tab. We recommend using a scoped login attribute like userPrincipalName or mail because otherwise the search might find multiple users and prevent login.

In many scenarios Shibboleth is not intended to hide the user's password from the service provider, but only to implement SSO. If that is the case it is sufficient to protect the ownCloud base URL with Shibboleth. This will send Web users to the IdP but allow desktop and mobile clients to continue using username and password, preventing popups due to an expired Shibboleth session lifetime.

In **Autoprovision Users** mode the app will not ask another user backend, but instead provision users on the fly by reading the two additional environment variables for display name and email address.



In ownCloud 8.1 the Shibboleth environment variable mapping was stored in apps/user_shibboleth/config.php. This file was overwritten on upgrades, preventing a seamless upgrade procedure. In ownCloud 8.2+ the variables are stored in the ownCloud database, making Shibboleth automatically upgradeable.

Mapping ownCloud User IDs

From 3.1.2 you can now specify a mapper that is used on inbound ownCloud user IDs, to adjust them before usage in ownCloud. You can set the mapper using occ:

```
{occ-command-example-prefix} config:app:set user_shibboleth \
    uid_mapper --value="OCA\User_Shibboleth\Mapper\ADFSMapper"
```

You may view the currently configured mapper using:

```
{occ-command-example-prefix} shibboleth:mapping
```

The following mappers are provided with the app:

Class	Description
OCA\User_Shibboleth\Mapper\NoOpMapper	The default, does not alter the UID
OCA\User_Shibboleth\Mapper\ADFSMapper	Splits the UID around a ; character and takes the first piece
OCA\User_Shibboleth\Mapper\GUIDInMemoryMapper	Maps in binary GUIDs to strings

Shibboleth with Desktop and Mobile Clients

The ownCloud Desktop Client can interact with an ownCloud instance running inside a Shibboleth Service Provider by using OAuth2 tokens to authenticate. The ownCloud Android and iOS mobile apps also work with OAuth2 tokens.

WebDAV Support

Users of standard WebDAV clients can generated an App Password on the Personal settings page. Use of App Passwords may be enforced with the token_auth_enforced option in config/config.php.

Known Limitations

Encryption

File encryption can only be used together with Shibboleth when master key-based encryption is used because the per-user encryption requires the user's password to unlock the private encryption key. Due to the nature of Shibboleth the user's password is not known to the service provider.

Other Login Mechanisms

You can allow other login mechanisms (e.g., LDAP or ownCloud native) by creating a second Apache virtual host configuration; such as in the below example.

```
<VirtualHost *:80>
  DocumentRoot /var/www/owncloud
  ServerName   https://www.myowncloud.com
  ServerAlias  myowncloud.com

  <Directory "/var/www/owncloud">
     Options       FollowSymlinks MultiViews
     AllowOverride All
     Order         Allow,Deny
     Allow         from All
  </Directory>

  <Location />
     AuthType shibboleth
     ShibRequestSetting requireSession false
     Require shibboleth
  </Location>

  # Path for shibboleth
  Alias "/index.php/login-shib" "/var/www/owncloud/index.php/login"
  <Location ~ "/index.php/login-shib">
     AuthType shibboleth
     ShibRequestSetting requireSession 1
     ShibRequestSetting REMOTE_ADDR X-Forwarded-For
     require valid-user
  </Location>

  RewriteEngine On
  RewriteCond %{HTTP_HOST} !myowncloud.com$ [NC]
  RewriteRule ^(.*)$ https://myowncloud.com/$1 [L,R=301]
</VirtualHost>
```



The second location in the above configuration is **not** protected by Shibboleth, and you can use your other ownCloud login mechanisms.



The above configuration can be used with multi-factor authentication as well.

If you use the above configuration, after it's enabled, configure the alternative logins option with a button to point to /login-shib. This will trigger the Shibboleth session and redirect the user back to /login. At this point, the existing session will be picked up, continuing with the authentication process.

Session Timeout

Session timeout on Shibboleth is controlled by the IdP. It is not possible to have a session length longer than the length controlled by the IdP. In extreme cases this could result in re-login on mobile clients and desktop clients every hour.

UID Considerations and Windows Network Drive Compatibility

To log in LDAP users via SAML for Single Sign On the user in LDAP must be uniquely resolvable by searching for the username that was sent in the SAML token. For this to work the LDAP attribute containing the username needs to be added to the **Additional Search Attributes** in the LDAP directory settings on the advanced tab. We recommend using a scoped login attribute like userPrincipalName or mail because otherwise the search might find multiple users and prevent login.

user_shibboleth will do the authentication, and user_ldap will provide user details such as email and displayname.

SAML 2.0 Based SSO with Active Directory Federation Services (ADFS) and mod_shib

Preparation

Before you can setup SAML 2.0 based Single Sign-On with Active Directory Federation Services (ADFS) and mod_shib, ask your ADFS admin for the relevant server URLs. These are:

- The SAML 2.0 single sign-on service URL, e.g., https://<ADFS server FQDN>/ADFS/ls
- The IdP metadata URL, e.g., https://<ADFS server FQDN>/FederationMetadata/2007-06/FederationMetadata.xml

Then, make sure that the web server is accessible with a trusted certificate:

```
sudo a2enmod ssl
sudo a2ensite default-ssl
sudo service apache2 restart
```

Installation

Firstly, install mod_shib. You can do this using the following command:

```
sudo apt-get install libapache2-mod_shib2
```

This will install packages needed for mod_shib, including shibd. Then, generate certificates for the shibd daemon by running the following command:

```
sudo shib-keygen
```

Download and Filter the ADFS Metadata

The metadata provided by ADFS cannot be automatically imported, and must be cleaned up before using it with the file based MetadataProvider. To do so, use adfs2fed.php, as in the following command:

```
php apps/user_shibboleth/tools/adfs2fed.php \
  https://<ADFS server FQDN>/FederationMetadata/2007-
06/FederationMetadata.xml \
  <AD-Domain> > /etc/shibboleth/filtered-metadata.xml
```

Configure shibd

Next, you need to configure shibd. To do this, in /etc/shibboleth/shibboleth2.xml:

Define the ownCloud Instance

Use the URL of the ownCloud instance as the entityID in the ApplicationDefaults

```
<ApplicationDefaults entityID="https://<owncloud server FQDN>/login/saml"
REMOTE_USER="eppn upn">
```



https://<owncloud server FQDN>/login/saml is just an example. Adjust <owncloud server FQDN> to the full qualified domain name of your server.

Configure SSO

Configure the SSO to use the entityID from the filtered-metadata.xml

```
<SSO entityID="https://<ADFS server FQDN>/<URI>/">
  SAML2
</SSO>
```



Grab <ADFS server FQDN>/<URI>/ from the filtered-metadata.xml.

Configure XML

Configure an XML MetadataProvider with the local filtered-metadata.xml file

```
<MetadataProvider type="XML" file="/etc/shibboleth/filtered-metadata.xml"/>
```

Metadata Available

Under https://<owncloud server FQDN>/Shibboleth.sso/Metadata shibd exposes the metadata that is needed by ADFS to add the SP as a Relying party.

Active Directory Federation Services (ADFS)

This part needs to be done by an ADFS administrator. Let him do his job while you continue with the Apache configuration below.

Add a Relying Party Using Metadata

See step 2 in AD FS 2.0 Step-by-Step Guide.

Configure ADFS to Send the userPrincipalName in the SAML Token

If you have control over ADFS make it send the UPN and Group by adding the following LDAP claim rule:

- Map User Principal Name to UPN
- Map Token Groups - Unqualified Names and map it to Group

Change shibd attribute-map.xml to:

```
<Attributes xmlns="urn:mace:shibboleth:2.0:attribute-map"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Attribute name="http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn"
id="upn"/>
</Attributes>
```

That will make the userPrincipalName available as the environment variable upn.

Apache2

To protect ownCloud with shibboleth you need to protect the URL with a mod_shib based auth. Currently, we recommend protecting only the login page.

user_shibboleth

When the app is enabled and ownCloud is protected by mod_shib, due to the Apache 2 configuration, you should be forced to authenticate against an ADFS. After a successful authentication you will be redirected to the ownCloud login page, where you can login as the administrator. Double check you have a valid SAML session by browsing to https://<owncloud server FQDN>/Shibboleth.sso/Session.

In the "User Authentication" settings for Shibboleth the upn environment variables will be filled with the authenticated user's userPrincipalName in the "Server Environment" section.

Use upn as uid and set the app mode to 'SSO Only' by running:

```
occ shibboleth:mode ssoonly
occ shibboleth:mapping -u upn
```

displayName and email are only relevant for autoprovisioning mode. Add Claims in ADFS and map them in the attribute-map.xml if needed.

Testing

- Close the browser tab to kill the session.
- Then visit https://<owncloud server FQDN> again.
- You should be logged in automatically.
- Close the tab or delete the cookies to log out.
- To make the logout work see the Logout section in this document.

Configuring SSO

- On the ADFS Server:
- Add "Windows Authentication" to the "Service" → "Authentication Methods" for "Intranet"
- Run the following Powershell script for Firefox:

```
# Save the list of currently supported browser user-agents to a variable
$browsers=Get-ADFSProperties | Select -ExpandProperty
WIASupportedUseragents

# Add Mozilla/5.0 user-agent to the list
$browsers+="Mozilla/5.0"

# Apply the new list
Set-ADFSProperties -WIASupportedUseragents $browsers

# Turn off Extended Protection
#Set-ADFSProperties –ExtendedProtectionTokenCheck None

# Restart the AD FS service
Restart-Service ADFSsrv
```

- On the Windows client:
- For Internet Explorer, Edge, and Chrome
- In the "Internet Settings" → "Security" → "Local Intranet"
- Click on "Sites"
- Click on "Advanced"
- Add your ADFS machine with https://<ADFS server FQDN>/ and click OK.
- Click on "customize level"
- Find "User Authentication"
- Check "Automatic login only for Intranet zone"
- For Firefox
- Open "about:config"
- Accept the warning
- Search for network.negotiate-auth.trusted-uris and set it to the FQDN of your ADFS server
- Search for network.automatic-ntlm-auth.trusted-uris and set it to the FQDN of your ADFS server

Now if you logged into the domain and open your ownCloud server in the browser of your choice you should get directly to your ownCloud files without a login.

Debugging

In /etc/shibboleth/shibd.logger, set the overall behavior to debug:

```
# set overall behavior
log4j.rootCategory=DEBUG, shibd_log, warn_log
[...]
```

After a restart /var/log/shibbloeth/shibd.log will show the parsed SAML requests and also which claims / attributes were found and mapped, or why not.

Browsers

- For Chrome there is a SAML Chrome Panel that allows checking the SAML messages in the developer tools reachable via F12.
- For Firefox there is SAML tracer
- In the Network tab of the developer extension make sure that "preserve logs" is enabled in order to see the redirects without wiping the existing network requests

Logout

In SAML scenarios the session is held on the SP as well as the IdP. Killing the SP session will redirect you to the IdP where you are still logged in, causing another redirect that creates a new SP session, making logout impossible. Killing only the IdP session will allow you to use the SP session until it expires.

There are multiple ways to deal with this:

1. By default ownCloud shows a popup telling the user to close the browser tab. That kills the SP session. If the whole browser is closed the IdP may still use a Kerberos-based authentication to provide SSO in effect making logout impossible.
2. Hide the logout action in the personal menu via CSS. This forces users to log out at the IdP.

OAuth2

In upcoming versions the clients will use OAuth2 to obtain a device specific token to prevent session expiry, making the old /oc-shib/remote.php/nonshib-webdav obsolete

Further Reading

- ADFS 2.0 Step-by-Step Guide: Federation with Shibboleth 2 and the InCommon Federation
- ADFS: How to Invoke a WS-Federation Sign-Out
- Shibboleth Service Provider Integration with ADFS
- adfs2fed Python Script
- AD FS 2.0 Step-by-Step Guide: Federation with Shibboleth 2 and the InCommon Federation
- Shibboleth Basic Configuration (Version 2.4 and Above)
- Shibboleth XML MetadataProvider
- Shibboleth NativeSPServiceSSO

Document Classification and Policy Enforcement

Introduction

When dealing with large amounts of data in an enterprise, it is essential to have mechanisms in place that allow you to stay in control of data flows. To implement such mechanisms the first step to take is to define guidelines that describe how the content of different security levels have to be treated.

Depending on the industry, such information security guidelines can originate from regulatory requirements, from recommendations of industry associations, or they can be self-imposed if there’s no external factor but internal risk management requirements that demand special treatment for specific information.

The leading information security standard ISO 27001 defines guidelines for managing

information security which can be certified. More specifically:

1. Information should enter an asset inventory (A.8.1.1)
2. Information should be classified (A.8.2.1)
3. Information should be labeled (A.8.2.2)
4. Information should be handled in a secure way (A.8.2.3)

As the leading international standard and certification for information security, ISO 27001 covers 75-80% of the GDPR. This makes it the ideal framework choice to support GDPR compliance requirements. Please see the GDPR to ISO-27001 Mapping Guide as an example to match the mentioned ISO Controls to the relevant *General Data Protection Regulation* (GDPR) articles.

Once the guidelines are set up, they need to be put into practice. First of all, highly sensitive data needs to be separated from less sensitive data. This is, usually, done by outlining the security levels present in the enterprise, and defining the criteria for information to qualify for each of these security levels.

Typically used security levels are "*Public*", "*Internal*", "*Confidential*", and "*Strictly Confidential*", but the requirements are usually determined individually. For example, if you are seeking GDPR compliance, then administrators can add additional ones, such as "*No PID (Personally Identifiable Information)*", "*PID*", and "*Special PID*".

The actual separation of information can then be done by requiring users to classify documents according to the security levels before they leave their workstation, or by using other criteria to assign classification levels to data during further processing.

Based on the classification level, information can then be labeled and policies can be enforced to ensure that information is handled in a secure way - and in compliance with corporate guidelines.

ownCloud can boost productivity with unique collaboration features. Firstly, there's "*Document Classification and Policy Enforcement*". This adds the capability to ensure that sensitive data is handled as required by information security guidelines.

Specifically, it enables ownCloud providers to:

- Comply with information security standards, such as ISO 27001/2 as recommended by the German Association of the Automotive Industry (VDA) and get certified to work securely within your value chain.
- Handle data in compliance with GDPR
- Manage risks effectively and cover potential data breaches.
- Separate information based on metadata.
- Display the data classification levels to raise user awareness.
- Prevent human mistakes when dealing with sensitive information.
- Fulfil corporate data protection requirements.

Classification

Employing document classification and respective policies in ownCloud generally involves three steps, which are outlined in detail below.

1. Create tags for classification
2. Configure rules for classification (tagging)
3. Associate policies to the classification rules

Tags for Classification

Document classification levels in ownCloud are represented via Collaborative Tags. Different categories of tags can be used to achieve different behaviors for users; these are detailed in the table below.

Table 2. Tag Categories Available in ownCloud

Tag Name	Description
Visible	These tags are not available for classification based on metadata and feature policies because users can edit and delete them, which is undesirable in many cases
Restricted	These tags can be created by administrators using Collaborative Tags Management. This category is recommended as it enables users to recognize the classification level of files and to be able to filter accordingly. Additionally, certain groups of users can have the privilege to edit and assign or unassign these tags.
Static	These tags can be created by administrators using Collaborative Tags Management. This category is recommended as it enables users to recognize the classification level of files and to be able to filter accordingly. Additionally this tag category should be used for manual classification as users in specified groups can only assign and unassign them but only administrators can edit or delete them. This way administrators can provide a tag linked to a classification policy that specified users can then impose on files.
Invisible	These tags can be created by administrators using Collaborative Tags Management. This category is recommended when users should not be able to recognize the classification level of files or to be able to filter accordingly.

For setting up each classification rule, create a separate tag using Collaborative Tags Management, which you can later assign to classification rules and/or policies.

Automated Classification Based on Document Metadata

Automated classification based on document metadata consists of two parts:

1. The actual classification metadata is embedded in documents using Office suite features
2. Document metadata is evaluated on file upload via the web interface and all ownCloud Clients. Automated classification in ownCloud therefore takes place on file upload. Existing files containing classification metadata currently can't be classified subsequently, except via manual user interaction.

Office Suite Features for Document Classification

Microsoft Office can be extended with the NovaPath addon, to provide classification capabilities. Currently Microsoft Office formats (*docx*, *dotx*, *xlsx*, *xltx*, *pptx*, *ppsx* and *potx*) are supported LibreOffice provides an integrated classification manager (TSCP).

To use automated classification based on document metadata, install and enable the Document Classification extension. The configuration depends on the tools and the classification framework in use.

Administrators can find examples and generalized configuration instructions below.

Microsoft Office with Add-Ons

Microsoft Office does *not* provide classification capabilities out-of-the-box. To extend it, we recommend the Microsoft Azure Information Protection or NovaPath add-ons. These extensions come with easy-to-use default classification categories, and provide the flexibility to set up custom classification schemes as desired.

Let's assume you want to use the default classification framework provided by NovaPath. In addition, let's assume that you take the classification level for documents classified as *Confidential* over to ownCloud to set up a policy that prevents said documents from being accessed by users in the group "**Trainees**".

This is how you set up an automated classification and the access policy in ownCloud:

- As an ownCloud administrator, navigate to menu:Settings[Workflows & Tags]. Adding a group with special privileges for the tag is optional.
- Within "User Management", create the group "*Trainees*" and add some users.
- Set up the classification rule in the panel "*Document Classification and Feature Policies*" in the same section, and set the following two properties:
  - **Property XPath** = //property[@name='Klassifizierung']/vt:lpwstr
  - **Property Value** = Confidential



Take care, the property and value fields are case-sensitive!

- For "*Tag*", choose btn:[Class: Confidential].
- Don't tick a policy checkbox as you don't want to set up a feature policy but an access policy.
- Hit btn:[Save].
- Set up the access policy in menu:Settings[Security].
- In the panel "*File Firewall*" enter a name for the group of rules, e.g., Confidential (optional). Hint: first click btn:[Add group] if you already have other rules configured.
- From the drop-down menu, choose btn:[System file] tag. In the tag picker, choose btn:[Class: Confidential]. Now you should have [System file tag] [is] [Class: Confidential].
- To add the group restriction, click btn:[Add rule], choose btn:[User group] from the drop-down menu. In the group picker drop-down, choose btn:[Trainees]. Now you should have [User group] [is] [Trainees].
- Hit btn:[Save Rules] to put the rules in place.
- To verify that the rule is in place, upload a classified file and check for the tag. Then share it with a member of the group "Trainees" (or with the whole group) and try to access it from a user account that is a member of said group.

LibreOffice

LibreOffice implemented the open standards produced by TSCP (*Transglobal Secure Collaboration Participation, Inc.*):

- The Business Authentication Framework (BAF) specifies how to describe the existing policy in a machine-readable format
- The Business Authorization Identification and Labeling Scheme (BAILS) defines how to refer to such a BAF policy in a document

There are three default BAF categories that come with different classification levels, which can be used out-of-the-box:

- Intellectual Property
- National Security
- Export Control

Assume you want to use the BAF category "*Intellectual Property*" and take the classification level for documents classified as "*Confidential*" over to ownCloud, to set up a policy that prevents said documents from being shared via a public link. This is how you set up an automated classification and the feature policy in ownCloud:

- As an ownCloud administrator, navigate to menu:Settings[Workflows & Tags]. Adding a group with special privileges for the tag is optional.
- Set up the classification rule and feature policy in the panel "*Document Classification and Feature Policies*" of the same section:
  - **Property XPath** = //property[@name='urn:bails:IntellectualProperty:BusinessAuthorizationCategory:Name']/vt:lpwstr
  - **Property Value** = Confidential (Take care, the property and value fields are case-sensitive!)
  - For "*Tag*" choose btn:[Class: Confidential].
  - Tick the checkbox btn:[Prevent link sharing].
  - Hit btn:[Save].
- To verify that the rule is in place, upload a classified file, check for the tag and try to create a public link share.

General Approach

Apart from the concrete examples above, a generalized method to employ document classification is available below.

Find the Metadata Properties and Values

- Classify a document in LibreOffice/MS Office and save it in an MS Office format.
- Rename the document's file extension to "*.zip*" and open it.
- Find the file docProps/custom.xml in the archive and open it with a text editor.
- Within custom.xml, find the property that contains the classification level value.
- Note down the classification property and value.
- Repeat the steps for all classification properties and values you want to set up classification rules for in ownCloud.

Set Up Classification Rules

- As an ownCloud administrator, navigate to menu:Settings[Workflows & Tags]
- In the panel ***Document Classification and Feature Policies*** set up the rules:
  - **Property XPath**: Enter the XPath that identifies the classification property. Below you find a generalized example where classification-property is a placeholder for the property to evaluate.

```
//property[@name='classification-property']/vt:lpwstr
```

- **Property Value**: Enter the value that triggers the classification rule when it matches with the metadata of an uploaded document, e.g., Confidential. Take care, the property and value fields are case-sensitive.
- **Tag**: Choose the tag to apply to files when a match occurs.

- Repeat the steps to create classification rules for all desired properties and values

Automated Classification Based on File or User Properties

Apart from automated classification based on document metadata, uploaded files may also be classified according to criteria inherent to files or to the users uploading them, making use of the Workflow extension.

- Administrators may add rules for automated classification of files according to a file's size or file type.
- File uploads by specific users, devices, or source networks can be used as indicators for classification.
- Furthermore, administrators can define shared folders to automatically classify files uploaded to such folders, by tagging the respective folder and creating a *Workflow* rule based on the chosen *System file tag*.
- Additionally, the rules may be linked to achieving a more granular classification behavior (e.g., PDF files uploaded by a specific group of users should be classified as *Confidential*).

Assume you want to automatically classify all PDF documents uploaded by users that are members of the "**Management**" group. You can construct a workflow rule using the following steps:

- Within user management create the group "*Management*" and add some users.
- Navigate to menu:Settings[Workflows & Tags].
- In the Collaborative Tags Management panel, create a tag of type "*Static*" and call it Class: Confidential. Adding a group with special privileges for the tag is optional.
- In the panel "*Workflow*" you can now set up the classification rules. Hit btn:[Add new workflow] and specify a useful name. Now configure the conditions that trigger the classification once they are met. For that choose "*User group*" from the drop-down menu, click btn:[+], then choose "*File mimetype*" and click btn:[+] again. Then you have to provide the group "*Management*" and the MIME type for PDF (application/pdf) in the respective fields.
- Select the tag btn:[Class: Confidential] to be added when the rules match.
- Click btn:[Add workflow] to save and enable it.



For more information, please check the options available for auto-tagging and consult the Workflow Extension documentation. For files classified with the *Workflow* extension, administrators can impose feature and access policies as described in the next section.

Manual Classification

As a further measure, it is possible to supply tags for users to autonomously classify all types of files in their own or shared spaces.

- As an ownCloud administrator, create a group within user management and add the users that should be able to classify files.
- Then navigate to menu:Settings[Workflows & Tags].
- In the Collaborative Tags Management panel, create a tag of type "*Static*" and give

it a meaningful name. Then assign the group you created, in the beginning, to give it's users special privileges for the tag.

- Users that are not a member of the specified group(s) will only be able to see the respective tag but can't alter or assign/un-assign it.

For files that are classified manually, administrators can impose feature and access policies as described in the next section.

Policy Enforcement

ownCloud currently provides two types of policies that can be enforced based on classification, *Feature* and *Access* policies. These policies can be imposed independently of the classification mechanism. The following sections illustrate the available policies and explain how they can be applied to classified contents.

Feature Policies

Feature policies are restrictions that prevent users from using a feature or force them to use it in a certain way. They are provided by the Document Classification extension, which currently supports the following policies:

- Prevent Upload
- Prevent Link Sharing
- Unprotected Links Expire After X Days

Prevent Upload

To follow guidelines that prevent data of certain classification levels (e.g., "*strictly confidential*") from being used in ownCloud at all, the "*Prevent upload*" policy is the right instrument to use. To impose such policies, tick the checkbox associated with the classification rule for the respective classification level.

When trying to upload documents caught by the policy, users will get an error message: A policy prohibits uploading files classified as '<tag>', where <tag> is the tag chosen for the classification rule.



Even though the server won't accept the uploaded files, in the end, it is mandatory to configure a tag for the classification rule to work.

Prevent Link Sharing

The prevent link sharing policy is tasked to ensure that classified data of certain confidentiality levels can't be shared publicly. This way, users can collaborate on the data internally, but it can't leave the company via ownCloud. To enable such policies, tick the checkbox associated with the classification rule for the respective classification level.

Documents with the associated classification level:

- Can't be shared via link (*public links on single files and folders containing classified files*); and
- Can't be moved to a publicly shared folder.

In all cases the user will see an error message containing the reasoning and the respective file(s): The file(s) "**<file1>, <file2>**" can't be shared via public link (classified as <tag>), where <tag> is the tag chosen for the classification rule.

Unprotected Links Expire After X Days

The policy *Unprotected links expire after X days* enables administrators to define public link expiration policies depending on the classification levels of the data that is shared via public links without password protection.

This makes it possible, for instance, to allow documents classified as *public* to be shared via public links for 30 days while documents classified as *internal* require public links to expire after seven days. To enable such policies, just define an expiration period associated with the classification rule for the respective classification level.



The Password Policy extension also provides options to enforce public link expiration depending on whether the user sets a password or not.

The option "*X days until link expires if password is not set*" is mutually exclusive with this policy. When you enable the Password Policy option, it will always be dominant and effectively override the policy discussed in this section. In contrast, the Password Policy option "*X days until link expires if password is set*" can be used in parallel.



The Sharing settings option provides the means to define a general public link expiration policy. This option currently is also mutually exclusive and will always override the policy discussed in this section.

Setting Up Policies Without Automated Classification Based on Document Metadata

All policies can also be enforced when using Manual Classification or Automated Classification based on File or User Properties. For this, specify the tag that determines the files that the policy should apply to and leave the fields for "*Property XPath*" and "*Property Value*" empty. Then choose the desired policy and click btn:[Save].

Access Policies

Access policies are restrictions that prevent users or groups of users from accessing specific resources even though they appear in their file list, e.g., via a share from another user. They are provided by the File Firewall extension which currently supports policies to prevent access to classified documents.

To link access policies with classification levels, the bottom line of such policies is the associated classification tag ([System file tag] [is] [<tag>]). It can, for instance, be combined with the following conditions to realize exclusive ([is]) or inclusive ([is not]) policies:

Documents with the respective classification tag can’t be accessed:

- *User group*: by users that are a member of the configured group (or can only be accessed by users that are a member of the configured group when using the [is not] operator).
- *User device*: from the configured device(s) (or only from the configured devices when using the [is not] operator)
- *Request time*: within the configured time frame (or only within the configured time frame when using the [is not] operator)
- *IP Range (Source network)*: from the configured IP range (or only from the configured IP range when using the [is not] operator)

Logging

When classified documents are uploaded, log entries will be written to ownCloud's log file, (data/owncloud.log). For this, it is possible to additionally specify another metadata property that will be used to add it's value to the log entries in the form of a "**Document ID**".

With this, it is possible to filter the log according to a document identifier or to forward classification events for certain documents to external log analyzers. To set it up, add the desired property XPath to the "*Document ID XPath*" field of the respective rule as you did for the classification property.

Each uploaded file will generate three entries with different log levels. See some exemplary entries below:

```
INFO: `"Checking classified file 'confidential.xlsx' with document id '2'"`
INFO: `"Alice uploaded a classified file 'confidential.xlsx' with document class
'Confidential'"`
DEBUG: `"Assigning tag 'Class: Confidential' to 'confidential.xlsx'"`
```

Limitations

Automated Classification Based on Document Metadata: Handling Classification Changes for Existing Files

- When a formerly classified document is replaced with a new version that does not contain classification metadata, the classification tag will remain assigned, and configured policies will still apply. In this case, it is recommended to either delete the original or upload the new version with a different name.
- When a formerly unclassified document is replaced with a new version that does contain classification metadata, the classification tag will be assigned. However, when the policy "**Prevent upload**" is set up in addition, the original file will be deleted, and the new version will be rejected due to the policy.

Troubleshooting

In this section you will find all the details you need to troubleshoot ownCloud.

Retrieve Log Files and Configuration Settings

Introduction

When you report a problem to ownCloud Support or our Forum (ownCloud Central) you will be asked to provide certain log files or configurations for our engineers (or other users). These are essential in better understanding your issue, your specific configuration, and the cause of the problem.

Here are instructions for how to collect them.

Generate a Config Report

You can use the webUI or the command line to generate a config report. Please note that you have to have the configreport app enabled. Check if it's already enabled by going to the apps section of the admin settings. You can enable this app using the following commands:

```
# Install it, if it's not already installed
{occ-command-example-prefix} market:install configreport

# Or enable it, if it's already installed
{occ-command-example-prefix}:app enable configreport
```

Generate via webUI

To generate a config report using the webUI, navigate to:
menu:Settings[Admin > General > "Generate Config report" > "Download ownCloud config report"].

Generate via Command Line

To generate a config report from the command line, run the following command from the root directory of your ownCloud installation:

```
{occ-command-example-prefix} configreport:generate > config_report.txt
```

ownCloud Server Log File

Generate via webUI

You can use the webUI to download your ownCloud Server log file. To do so, navigate to:
menu:Settings[Admin > General > Log > "Download logfile"].

Generate via Command Line

If the log file is too big, you will need to transfer it from the command line. The location of the log file can be found in your config.php. It's in your data directory.

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

You also can specify a different location of the log file.

```
'logfile' => '/home/www-data/owncloud.log',
```

Note that the web server user has to have rights to write in that directory.

LDAP Config

Assuming that LDAP is used, viewing the LDAP configuration is important when checking for errors between your ownCloud instance and your LDAP server. To get the output file, execute this command:

```
{occ-command-example-prefix} ldap:show-config > ldap_config.txt
```

Have You Found a Mistake In The Documentation?

If you have found a mistake in the documentation, no matter how large or small,

please let us know by creating a new issue in the docs repository.