ownCloud Server Administration Manual

Release 8.0

The ownCloud developers

April 26, 2016

CONTENTS

CHAPTER

ONE

OWNCLOUD 8.0 SERVER ADMINISTRATION MANUAL INTRODUCTION

1.1 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 replaces the Enterprise Edition. This includes the core Server, Enterprise apps and support.

See *What's New for Admins in ownCloud 8* for more information on the different ownCloud editions.

1.2 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.

1.3 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

CHAPTER

TWO

OWNCLOUD 8.0 RELEASE NOTES

2.1 ownCloud Server 8.0

Home folder rule is enforced in the user_ldap application in new ownCloud installations; see *User Authentication with LDAP*. This affects ownCloud 8.0.10, 8.1.5 and 8.2.0 and up.

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

2.1.1 APCu 4.0.6 and Higher Only

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.

2.1.2 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.

2.1.3 No Preview Icon on Text Files

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

2.1.4 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 *Configuring Federated Cloud Sharing* to learn more about federated cloud sharing)

2.1.5 Manually Migrate Encryption Keys after Upgrade

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

2.1.6 Windows Server Not Supported

Windows Server is not supported in ownCloud 8.

2.1.7 PHP 5.3 Support Dropped

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

2.1.8 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.

2.1.9 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.

2.1.10 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.

2.1.11 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.

2.1.12 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.

2.1.13 Encrypting Large Numbers of Files

When you activate the Encryption app 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.

2.2 Enterprise 8.0 Only

2.2.1 No Federated Cloud Sharing with Shibboleth

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

2.2.2 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.

2.2.3 SWIFT Objectstore Incompatible with Encryption App

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

2.2.4 App Store is Back

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

2.3 ownCloud 7 Release Notes

2.3.1 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.

2.3.2 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 your search attributes in the `User Search Attributes` form 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.

2.3.3 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.

2.3.4 Antivirus App Modes

The Antivirus App 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.

2.3.5 "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.

2.3.6 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.

2.3.7 4GB Limit on SFTP Transfers

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

2.3.8 "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`.

2.3.9 No More Expiration Date On Local Shares

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

2.3.10 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.

2.4 Enterprise 7 Only

2.4.1 No Federated Cloud Sharing with Shibboleth

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

2.4.2 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 App 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 App in the Enterprise Admin manual for instructions.

2.4.3 Sharepoint Drive SSL

The SharePoint Drive app 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.

2.4.4 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

2.4.5 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.

2.4.6 No App Store

The App Store is disabled for the Enterprise Edition.

2.4.7 LDAP Home Connector Linux Only

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

CHAPTER

THREE

WHAT'S NEW FOR ADMINS IN OWNCLOUD 8

3.1 New Packaging

We are introducing a new packaging system. Instead of two editions, Community and Enterprise, we now offer Server to replace the old Community edition, and Enterprise Subscription replaces the old Enterprise Edition.

Server includes core file share and synchronization features plus community apps, and is community-supported and free of cost. See owncloud.org for current information and links to downloads.

ownCloud now offers two levels of paid support:

- ownCloud Standard
- ownCloud Enterprise

The ownCloud Standard Subscription is for customers who want paid support for the core Server, and do not need Enterprise apps. This includes:

- ownCloud Server
- ownCloud desktop and mobile apps (without the custom branding build service)
- ownCloud open-source apps licensed as AGPL (Share Files, Federated Cloud, Versions, Deleted files, LDAP/AD, Antivirus, Encryption, External Storage. etc.), 8x5 support hours

Note: This does not include support for Contacts, Calendar, Tasks, Chat, Documents, or any other community-only apps, and it does not include support for Enterprise-only apps or services.

The ownCloud Enterprise Subscription replaces the Enterprise Edition. This includes the core Server plus Enterprise apps. The Enterprise Subscription includes:

- ownCloud Standard Subscription
- ownCloud Enterprise apps (Logging modules, SAML, File Firewall, Sharepoint, Windows Network Drive, Home Directory Mounts, etc)
- ownBrander for mobile app branding
- ownCloud Commercial License for closed-source customizations
- Up to 24x7 support
- Deployment assistance for the rollout

New customers, or customers upgrading from the old Enterprise Edition will install ownCloud 8 from the existing `owncloud-enterprise` repository. The `owncloud-server` dependency will be pulled in automatically. Customers using the community edition will upgrade via package manager by adding the `owncloud-enterprise repository`, and installing ownCloud Enterprise Edition apps on top of their `owncloud-server`.

Visit owncloud.com for more information on the Enterprise Subscription.

3.2 No More PHP 5.3

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

3.3 Improved Admin and Settings UI

The ownCloud Admin and Settings pages have been streamlined and re-organized, and include a new sidebar for accessing configuration options more quickly.

3.4 Simplified App Management

Installing, upgrading, and removing apps is easier, and includes a new dependency feature to automatically resolve installation dependendencies.

3.5 New Updater Layout

The Updater app (Server only) is more intelligent, and detects errors before they happen and rolls the update back. Status and error messages are more informative.

3.6 More Powerful User Management

In addition to the existing filter and text string searchable user management, an entirely new set of features has been added to ownCloud. Admins can now edit email addresses for system users and send email notification to newly created system users.

3.7 LDAP Improvements, Including LDAP User Cleanup

ownCloud has significantly improved LDAP and AD performance, provided additional configuration options and expert modes, and added a new utility for verifying and removing users from ownCloud that are no longer active in LDAP or AD.

3.8 Improved and Open-Sourced Provisioning API

Remote management of ownCloud users is enabled with the Provisioning API. Previously limited to the Enterprise edition, it is now open source and available to the community.

3.9 Favorites

Users can now assign a favorite icon to files and folders. Look for improvements in this feature in future ownCloud editions to make finding and managing files even easier.

3.10 Improved Server-to-Server Sharing

Server-to-Server Sharing, introduced in ownCloud 7, allows you to mount file shares from remote ownCloud servers, and create a "cloud of ownClouds". In ownCloud 8 the process for creating a new Server-to-Server link is easier and more streamlined.

3.11 Improved Search

The search interface has been streamlined and simplified, with more features including enhanced result set reporting and additional search parameters.

3.12 Web Interface Enhancements

The ownCloud Web interface has been improved to make it easier for all users to access, edit, sync and share their files.

3.13 Download Broker Improves Performance

When ownCloud delivers universal file access to end users, files from many different document sources are aggregated into a single interface and served to end users. In some cases, passing all of the files aggregated in this interface through a single server, ownCloud, slows down data access. ownCloud now supports direct downloads of files from select storage back-ends, reducing the load on the ownCloud server without sacrificing control over the files that are stored in the various back end systems.

CHAPTER

FOUR

INSTALLATION

4.1 System Requirements

4.1.1 Memory

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

4.1.2 Recommended Setup for Running ownCloud

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

- Red Hat Enterprise Linux 7
- MySQL/MariaDB
- PHP 5.4 +
- Apache 2.4

4.1.3 Supported Platforms

- Server: Linux (Debian 7, SUSE Linux Enterprise Server 11 SP3 & 12, Red Hat Enterprise Linux/Centos 6.5 and 7 (7 is 64-bit only), Ubuntu 12.04 LTS, 14.04 LTS, 14.10)
- Webserver: Apache 2
- Databases: MySQL/MariaDB 5.x; Oracle 11g; PostgreSQL
- PHP 5.4 + required
- Hypervisors: Hyper-V, VMware ESX, Xen, KVM
- Desktop: Windows XP SP3 (EoL Q2 2015), Windows 7+, Mac OS X 10.7+ (64-bit only), Linux (CentOS 6.5, 7 (7 is 64-bit only), Ubuntu 12.04 LTS, 14.04 LTS, 14.10, Fedora 20, 21, openSUSE 12.3, 13, Debian 7 & 8).
- Mobile apps: iOS 7+, Android 4+
- Web browser: IE8+ (except Compatibility Mode), Firefox 14+, Chrome 18+, Safari 5+

4.2 ownCloud Deployment Recommendations

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

4.2.1 General Recommendations

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

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

- Operating system: Linux.
- Webserver: Apache 2.4.
- Database: MySQL/MariaDB.
- PHP 5.5+. PHP 5.4 is the minimum supported version; note that it reached end-of-life in September 2015 and is no longer supported by the PHP team. Some Linux vendors, such as Red Hat, still support PHP 5.4. 5.6+ is recommended. `mod_php` is the recommended Apache module because it provides the best performance.

4.2.2 Small Workgroups or Departments

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

Recommended System Requirements

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

Authentication via an existing LDAP or Active Directory server.

- **Components** One server with at least 2 CPU cores, 16GB RAM, local storage as needed.
- **Operating system** Enterprise-grade Linux distribution with full support from OS vendor. We recommend Red Hat Enterprise Linux or SUSE Linux Enterprise Server 12.
- **SSL Configuration** The SSL termination is done in Apache. A standard SSL certificate is needed, installed according to the Apache documentation.

- **Load Balancer** None.
- **Database** MySQL, MariaDB or PostgreSQL. We currently recommend MySQL / MariaDB, as our customers have had good experiences when moving to a Galera cluster to scale the DB.
- **Backup** Install owncloud, ownCloud data directory and database on Btrfs filesystem. Make regular snapshots at desired intervals for zero downtime backups. Mount DB partitions with the "nodatacow" option to prevent fragmentation.

  Alternatively, make nightly backups with service interruption:

  – Shut down Apache.

  – Create database dump.

  – Push data directory to backup.

  – Push database dump to backup.

  – Start Apache.

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

4.2.3 Mid-sized Enterprises

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

Recommended System Requirements

2 to 4 application servers.

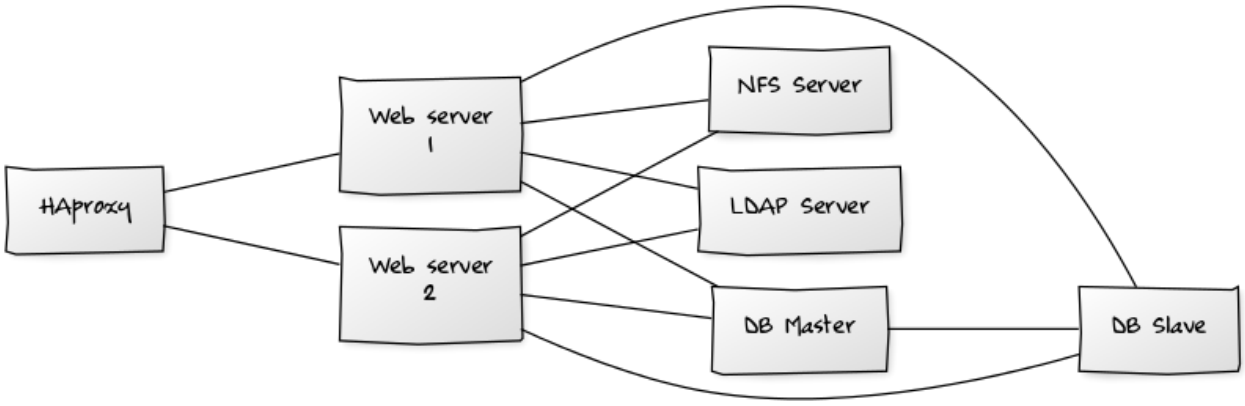
A cluster of two database servers.

Storage on an NFS server.

Authentication via an existing LDAP or Active Directory server.

- **Components**

  – 2 to 4 application servers with 4 sockets and 32GB RAM.



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

- **Operating system** Enterprise grade Linux distribution with full support from OS vendor. Red Hat Enterprise Linux or SUSE Linux Enterprise Server 12 are recommended.
- **SSL Configuration** The SSL termination is done in the HAProxy load balancer. A standard SSL certificate is needed, installed according to the HAProxy documentation.
- **Load Balancer** HAProxy running on a dedicated server in front of the application servers. Sticky session needs to be used because of local session management on the application servers.
- **Database** MySQL/MariaDB Galera cluster with master-master replication.
- **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.)
- **LDAP** Read-only slaves should be deployed on every application server for optimal scalability
- **Session Management** Session management on the application server. PHP sessions are stored in a tmpfs mounted at the operating system-specific session storage location. You can find out where that is by running `grep -R 'session.save_path' /etc/php5` and then add it to the `/etc/fstab` file, for example: `echo "tmpfs /var/lib/php5/pool-www tmpfs defaults,noatime,mode=1777 0 0" >> /etc/fstab`.
- **Memory Caching** A memcache speeds up server performance, and ownCloud supports four memcaches; refer to Configuring Memory Caching for information on selecting and configuring a memcache.
- **Storage** Use an off-the-shelf NFS solution, such as IBM Elastic Storage or RedHat Ceph.

- **ownCloud Edition** Enterprise Edition. (See ownCloud Server or Enterprise Edition for comparisons of the ownCloud editions.)

4.2.4 Large Enterprises and Service Providers

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

Recommended System Requirements

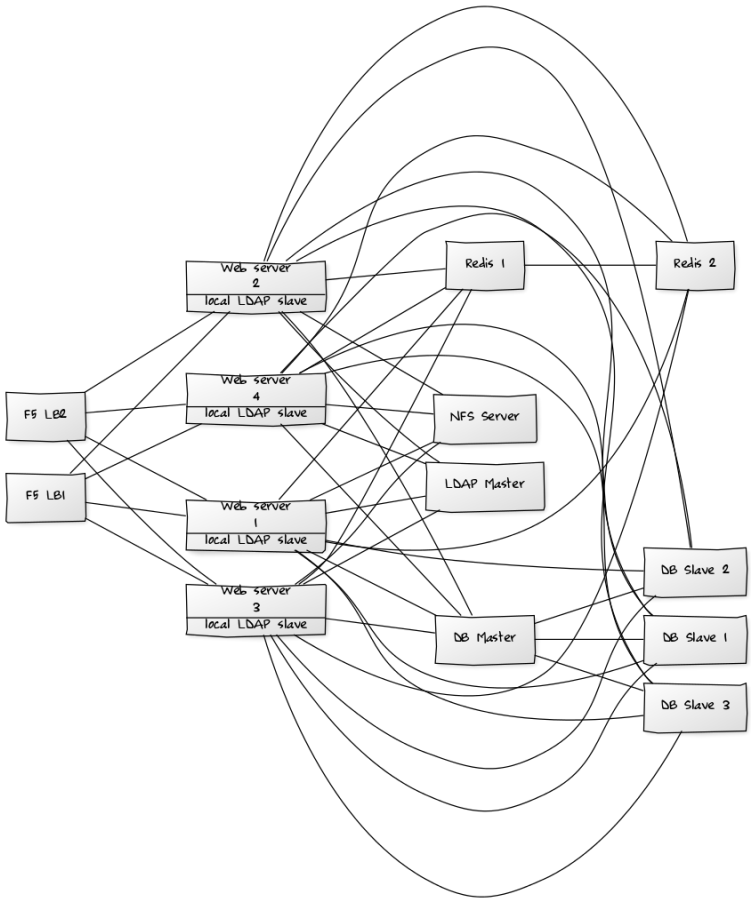
4 to 20 application/Webservers.

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 4 sockets and 64GB RAM.
  - 4 DB servers with 4 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 4x master – master replication.
- **Backup** Minimum daily backup without downtime. All MySQL/MariaDB statements should be replicated to a backup MySQL/MariaDB slave instance.
  - Create a snapshot on the NFS storage server.
  - At the same time stop the MySQL replication.
  - Create a MySQL dump of the backup slave.
  - Push the NFS snapshot to the backup.
  - Push the MySQL dump to the backup.
  - Delete the NFS snapshot.
  - Restart MySQL replication.
- **Authentication** User authentication via one or several LDAP or Active Directory servers, or SAML/Shibboleth. (See User Authentication with LDAP 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 (see Configuring Memory Caching).
- **Storage** An off-the-shelf NFS solution should be used. Examples are IBM Elastic Storage or RedHAT Ceph. Optionally, an S3 compatible object store can also be used.
- **ownCloud Edition** Enterprise Edition. (See ownCloud Server or Enterprise Edition for comparisons of the ownCloud editions.)

4.2.5 Hardware Considerations

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

Single Machine / Scale-Up Deployment

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

Pros:

- Easy setup: no session storage daemon, use tmpfs and memory caching to enhance performance, local storage.

- No network latency to consider.
- To scale buy a bigger CPU, more memory, larger hard drive, or additional hard drives.

Cons:

- Fewer high availability options.
- The amount of data in 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)

Pros:

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

Cons:

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

What About Nginx / PHP-FPM?

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

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

When master fails another slave can become master. 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.

4.2.6 Software Considerations

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.

Webserver

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

Relational Database

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

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.

4.2.7 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 (supported in Enterprise Edition only) or Swift. S3 also allows access to Ceph Storage.

4.2.8 Session Storage

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

4.2.9 References

Database High Availability

Performance enhancements for Apache and PHP

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

4.3 Preferred Linux Installation Method

4.3.1 Installation Quick Start

See the *System Requirements* for the recommended ownCloud setup and supported platforms.

Installing ownCloud Server (the free community edition) on Linux from the openSUSE Build Service packages is the preferred method. These are maintained by ownCloud engineers, and you can use your package manager to keep your ownCloud server up-to-date. Follow the instructions for your distro to add the oBS repository, download and install the repository signing key, and install ownCloud. Then run the Installation Wizard to complete your installation. (see *Installation Wizard*).

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

4.3.2 Installing ownCloud Enterprise Subscription

See *Installing ownCloud Enterprise Subscription on Linux* for instructions on installing ownCloud Enterprise Subscription.

4.3.3 Downgrading Not Supported

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.

4.3.4 Additional Installation Guides and Notes

See *Installation Wizard* for important steps such as choosing the best database and setting correct directory permissions.

See *SELinux Configuration* 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 code (see *Manual Installation on Linux*).

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

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

Debian/Ubuntu: The package installs an additional Apache config file to `/etc/apache2/conf-available/owncloud.conf` which contains an `Alias` to the owncloud installation directory as well as some more needed configuration options.

Running ownCloud in a subdir: If you're running ownCloud in a subdir and want to use CalDAV or CardDAV clients make sure you have configured the correct *Service discovery* URLs.

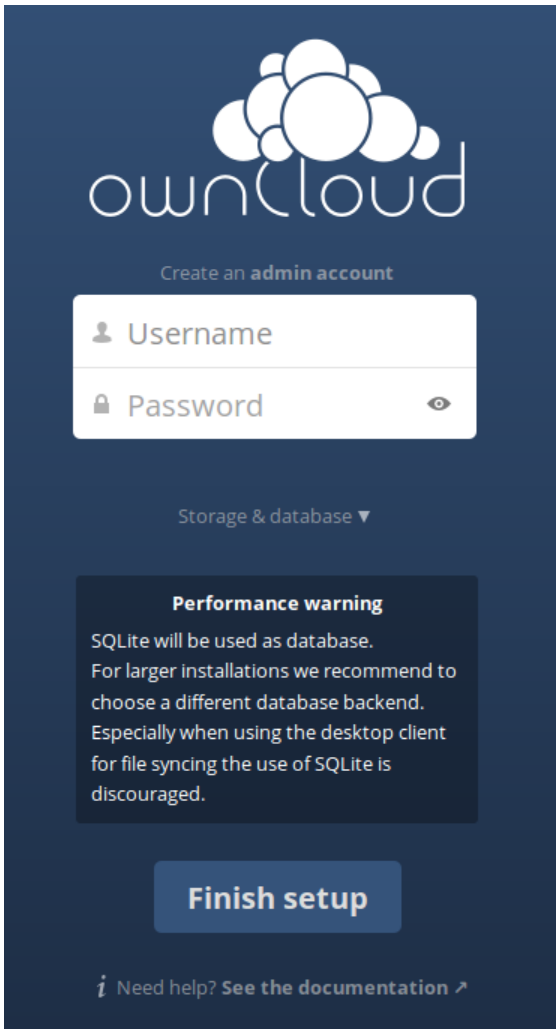
Note for MySQL/MariaDB environments: Please refer to *MySQL / MariaDB with Binary Logging Enabled* on how to correctly configure your environment if you have binary logging enabled.

4.4 Installation Wizard

4.4.1 Quick Start

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

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



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

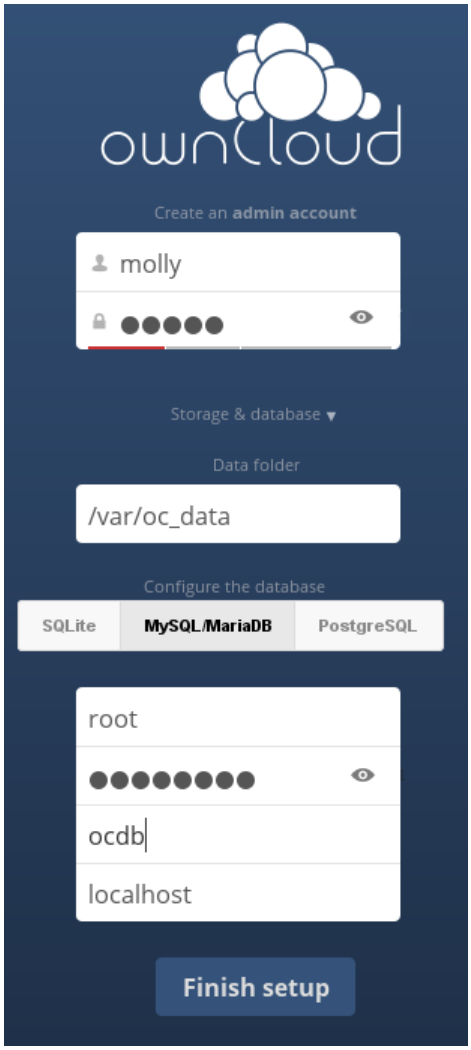
Of course, there is much more that you can do to set up your ownCloud server for best performance and security. In the following sections we will cover important installation and post-installation steps. Note that you must follow the instructions in *Setting Strong Permissions* in order to use the *occ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

- *Data Directory Location*
- *Database Choice*
- *Trusted Domains*

- *Setting Strong Permissions*

4.4.2 Data Directory Location

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



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). 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 (see *Setting Strong Directory Permissions*).

4.4.3 Database Choice

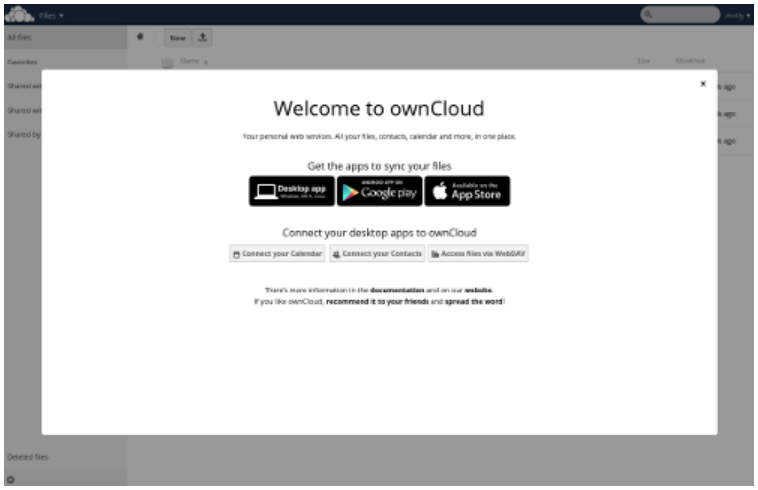
SQLite is the default database for ownCloud Server (it is not available and not supported in the Enterprise edition), and it is good only for testing and lightweight single-user setups without client synchronization. Supported databases are MySQL, MariaDB, Oracle 11g, and PostgreSQL, and we recommend *MySQL/MariaDB*. Your database and PHP

connectors must be installed before you run the Installation Wizard. When you install ownCloud from packages all the necessary dependencies will be satisfied (see *Manual Installation on Linux* for a detailed listing of required and optional PHP modules). You will need the root database login, or any administrator login that has permissions to create and modify databases, and then enter any name you want for your ownCloud database.

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

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

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



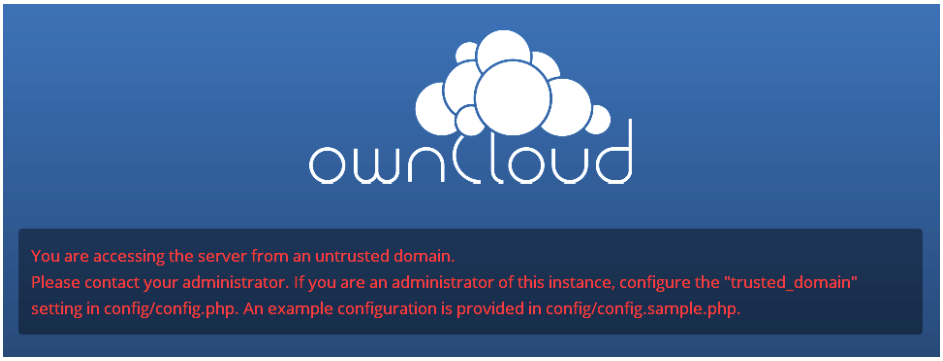
Now we will look at some important post-installation steps.

4.4.4 Trusted Domains

All URLs used to access your ownCloud server must be whitelisted 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. You may use IP addresses and domain names. A typical configuration looks like this:

```
'trusted_domains' =>
  array (
    0 => 'localhost',
    1 => 'server1.example.com',
    2 => '192.168.1.50',
),
```

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



4.4.5 Setting Strong Directory Permissions

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. Your HTTP user must own the `config/`, `data/` and `apps/` 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 *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`.

Note: 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 this script. 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
ocpath='/var/www/owncloud'
htuser='www-data'
htgroup='www-data'
rootuser='root'


find ${ocpath}/ -type f -print0 | xargs -0 chmod 0640
find ${ocpath}/ -type d -print0 | xargs -0 chmod 0750

chown -R ${rootuser}:${htgroup} ${ocpath}/
chown -R ${htuser}:${htgroup} ${ocpath}/apps/
chown -R ${htuser}:${htgroup} ${ocpath}/config/
chown -R ${htuser}:${htgroup} ${ocpath}/data/
chown -R ${htuser}:${htgroup} ${ocpath}/themes/

chown ${rootuser}:${htgroup} ${ocpath}/.htaccess
chown ${rootuser}:${htgroup} ${ocpath}/data/.htaccess

chmod 0644 ${ocpath}/.htaccess
```

```
chmod 0644 ${ocpath}/data/.htaccess
```

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

This 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 `config/` directory should be owned by `[HTTP user]:[HTTP group]`
- The `themes/` directory should be owned by `[HTTP user]:[HTTP group]`
- The `data/` 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

4.5 Installing and Managing Apps

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

4.5.1 Enterprise Subscription Supported Apps

See *Supported ownCloud Enterprise Subscription Apps* for a list of supported Enterprise Subscription apps.

4.5.2 Viewing Enabled Apps

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

You will see which apps are enabled, not enabled, and recommended. You'll also see additional filters, such as Multimedia and Productivity for finding apps quickly. The `More apps` link takes you to the ownCloud Apps Store, and the `Add your app` link takes you to the ownCloud Developer Manual.

4.5.3 Re-enabling Contacts and Calendar Apps

The Contacts and Calendar apps are unsupported community apps, and by default are not enabled or installed in ownCloud 8. You may easily install and enable them by clicking on the Productivity filter, and then clicking the **Enable** buttons for both apps. This will download and enable them.

If you were using Contacts and Calendar in previous versions of ownCloud, and you upgraded to ownCloud 8, your Contacts and Calendar data are still in your ownCloud database. Installing and enabling them in ownCloud 8 will automatically restore your data.

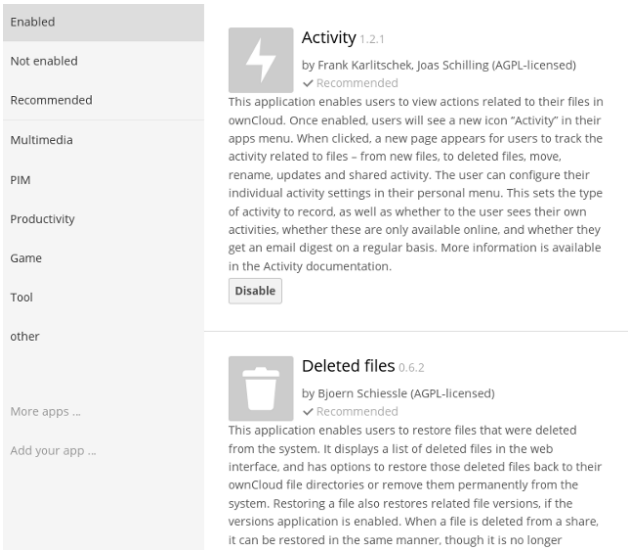


Figure 4.1: *Click to enlarge*

4.5.4 Managing Apps

In the Apps page you can enable or disable applications. Some apps have configurable options on the Apps page, such as **Enable only for specific groups**, but mainly they are enabled or disabled on the Apps page, and are configured on your ownCloud Admin page, Personal page, or in `config.php`.

4.5.5 Adding Third Party Apps

Some apps are developed and supported by ownCloud directly, while other apps are created by third parties and available for your ownCloud server installation. Apps developed by the ownCloud community show a *recommended* designation. Any apps that are not developed by ownCloud but have been reviewed by the ownCloud security team show a *3rd party* designation. Install unsupported apps at your own risk.

To understand what an application does, you can click the app name to view a description of the app and any of the app settings in the Application View field. Clicking the *Enable* button will enable the app. If the app is not part of the ownCloud installation, it will be downloaded from the app store, installed and enabled.

You can view new, unreviewed and unstable applications in the ownCloud Apps Store.

To view or install apps from the ownCloud Apps Store:

1. Scroll to the bottom of the Apps Information Field.
2. Click *More apps*.

   The ownCloud Apps Store launches.
3. Read about any of the apps in the ownCloud Apps Store and download any that you like.
4. Extract a downloaded compressed file and place the contents (which should themselves be contained in a folder with the app name) in the apps folder in your ownCloud installation, typically `owncloud/apps`.
5. Ensure the permissions and ownership are similar to the other ownCloud apps. Typically, access rights are **rwxr-x—**, or **0750** in octal notation, and the owner and group are your HTTP user. On CentOS this is `apache`, Debian/Ubuntu is `www-data`, and on openSUSE is it `wwwrun:www`.

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

Note: If you would like to create or add your own ownCloud app, please use the *Add your App...* button on the same page. This button redirects you to the App Development documentation, where you can find information about creating and adding your own apps.

4.5.6 Using Custom App Directories

Use the **apps_paths** array in *config.php* to set any custom apps directory locations. The key **path** defines the absolute file system path to the app folder. The key **url** defines the HTTP web path to that folder, starting at the ownCloud web root. The key **writable** indicates if a user can install apps in that folder.

Note: To ensure that the default **/apps/** folder only contains apps shipped with ownCloud, follow this example to setup an **/apps2/** folder which will be used to store all other apps.

```
<?php

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

4.5.7 Using Your Own Appstore

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

To enable installation from your own apps store:

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

   This parameter is used to enable your apps store in ownCloud.

2. Set the **appstoreurl** to the URL of your ownCloud apps store.

   This parameter is used to set the HTTP path to the ownCloud apps store. The appstore server must use OCS (Open Collaboration Services).

```
<?php

  "appstoreenabled" => true,
  "appstoreurl" => "http://api.apps.owncloud.com/v1",
```

4.6 Manual Installation on Linux

Installing ownCloud on Linux from our Open Build Service packages is the preferred method (see *Preferred Linux Installation Method*). These are maintained by ownCloud engineers, and you can use your package manager to keep your ownCloud server up-to-date.

Note: Enterprise Subscription customers should refer to *Installing ownCloud Enterprise Subscription on Linux*

If there are no packages for your Linux distribution, or you prefer installing from the source tarball, you can setup ownCloud from scratch using a classic LAMP stack (Linux, Apache, MySQL/MariaDB, PHP). This document provides a complete walk-through for installing ownCloud on Ubuntu 14.04 LTS Server with Apache and MariaDB, using the ownCloud .tar archive.

- *Prerequisites*
- *Example Installation on Ubuntu 14.04 LTS Server*
- *Apache Web Server Configuration*
- *Enabling SSL*
- *Installation Wizard*
- *Setting Strong Directory Permissions*
- *SELinux Configuration Tips*
- *php.ini Configuration Notes*
- *php-fpm Configuration Notes*
- *Other Web Servers*

Note: 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 Configuration Tips* for a suggested configuration.

4.6.1 Prerequisites

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

Required:

- php5 (>= 5.4)
- PHP module ctype
- PHP module dom
- PHP module GD
- PHP module iconv
- PHP module JSON
- PHP module libxml
- PHP module mb multibyte
- PHP module posix

- PHP module SimpleXML
- PHP module XMLWriter
- PHP module zip
- PHP module zlib

Database connectors (pick the one for your database:)

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

Recommended packages:

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

Required for specific apps:

- PHP module ldap (for LDAP integration)
- php5-libsmbclient (SMB/CIFS integration)
- PHP module ftp (for FTP storage / external user authentication)
- PHP module imap (for external user authentication)

Recommended for specific apps (*optional*):

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

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

- PHP module apc
- PHP module apcu
- PHP module memcached
- PHP module redis (required for Transactional File Locking)

See *Configuring Memory Caching* to learn how to select and configure a memcache.

For preview generation (*optional*):

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

For command line processing (*optional*):

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

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

4.6.2 Example Installation on Ubuntu 14.04 LTS Server

On a machine running a pristine Ubuntu 14.04 LTS server, install the required and recommended modules for a typical ownCloud installation, using Apache and MariaDB, by issuing the following commands in a terminal:

```
apt-get install apache2 mariadb-server libapache2-mod-php5
apt-get install php5-gd php5-json php5-mysql php5-curl
apt-get install php5-intl php5-mcrypt php5-imagick
```

- This installs the packages for the ownCloud core system. If you are planning on running additional apps, keep in mind that they might require additional packages. See *Prerequisites* for details.
- At the installation of the MySQL/MariaDB server, you will be prompted to create a root password. Be sure to remember your password as you will need it during ownCloud database setup.

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.

4.6.3 Apache Web Server Configuration

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

```
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`:

```
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
  ```

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

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

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

  ```
  a2enmod setenvif
  ```

- You must disable any server-configured authentication for 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 turn off 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 subdirectory and want to use CalDAV or CardDAV clients make sure you have configured the correct *Service discovery* URLs.

4.6.4 Enabling SSL

Note: 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
```

Note: 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.

4.6.5 Installation Wizard

After restarting Apache you must complete your installation by running either the graphical Installation Wizard, or on the command line with the `occ` command. To enable this, temporarily change the ownership on your ownCloud directories to your HTTP user (see *Setting Strong Directory Permissions* to learn how to find your HTTP user):

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

Note: Admins of SELinux-enabled distributions may need to write new SELinux rules to complete their ownCloud installation; see *SELinux Configuration Tips*.

To use `occ` see *Using the occ Command*.

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

4.6.6 Setting Strong Directory Permissions

After completing installation, you must immediately set the directory permissions in your ownCloud installation as strictly as possible for stronger security. Please refer to *Setting Strong Directory Permissions*.

Now your ownCloud server is ready to use.

4.6.7 SELinux Configuration Tips

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

4.6.8 php.ini Configuration Notes

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

php.ini - used by the Web server:

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

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

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

4.6.9 php-fpm Configuration Notes

Security: Use at least PHP => 5.5.22 or >= 5.6.6

Due to a bug with security implications in older PHP releases with the handling of XML data you are highly encouraged to run at least PHP 5.5.22 or 5.6.6 when in a threaded environment.

System environment variables

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

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

Ubuntu/Mint	CentOS/Red Hat/Fedora
`/etc/php5/fpm/`	`/etc/php-fpm.d/`

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

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

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

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

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

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

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

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

4.6.10 Other Web Servers

Nginx Configuration

See *Nginx Configuration*

Yaws Configuration

See *Yaws Configuration*

Hiawatha Configuration

See *Hiawatha Configuration*

4.7 ownCloud Community Appliance

ownCloud has a publicly developed community appliance on GitHub. Download the latest release from the Appliances tab on the ownCloud server installation page. The easiest way to get the VM up and running is by using VirtualBox and downloading the OVA image from the installation page.

4.7.1 Instructions for VirtualBox and OVA

Follow these steps to get the appliance working:

1. Download the Virtual Machine image zip file and unpack it.
2. Start VirtualBox and click on *File ...* > *Import Appliance* and import your new ownCloud image.
3. Click the green Start arrow. After a minute you should see the console greeting message.
4. Note the username and password here. It is a random password that we generate for you on first boot. If you log in at the console, you'll be prompted to change the password. This is optional.
5. With your Web browser try `http://localhost:8888` or `http://localhost:80` or the address printed on the console. One of them should work. If not, please review and adjust the network setup of Virtualbox to bridged mode.
6. You should see a Web page with login credentials (if you haven't changed them already) and a list of URLs to try to reach the ownCloud web service. Which one works, again depends on the network setup of your hypervisor.

Note: You should write down your admin password, and make sure the login credentials are no longer displayed. Click the *[Hide Credentials]* button. When using the ownCloud Proxy app, this Web page may be publicly visible.

Note: Inside the VM, ownCloud runs with a default disk size of 40 GB and its own MySql database. The ownCloud admin user is also a valid account on the Ubuntu system that runs inside the VM. You can administer the VM via SSH.

For VMware

You can follow most of the steps above, however, after opening the VMX file, you will have to configure `Bridged Network` as *Network Adapter*

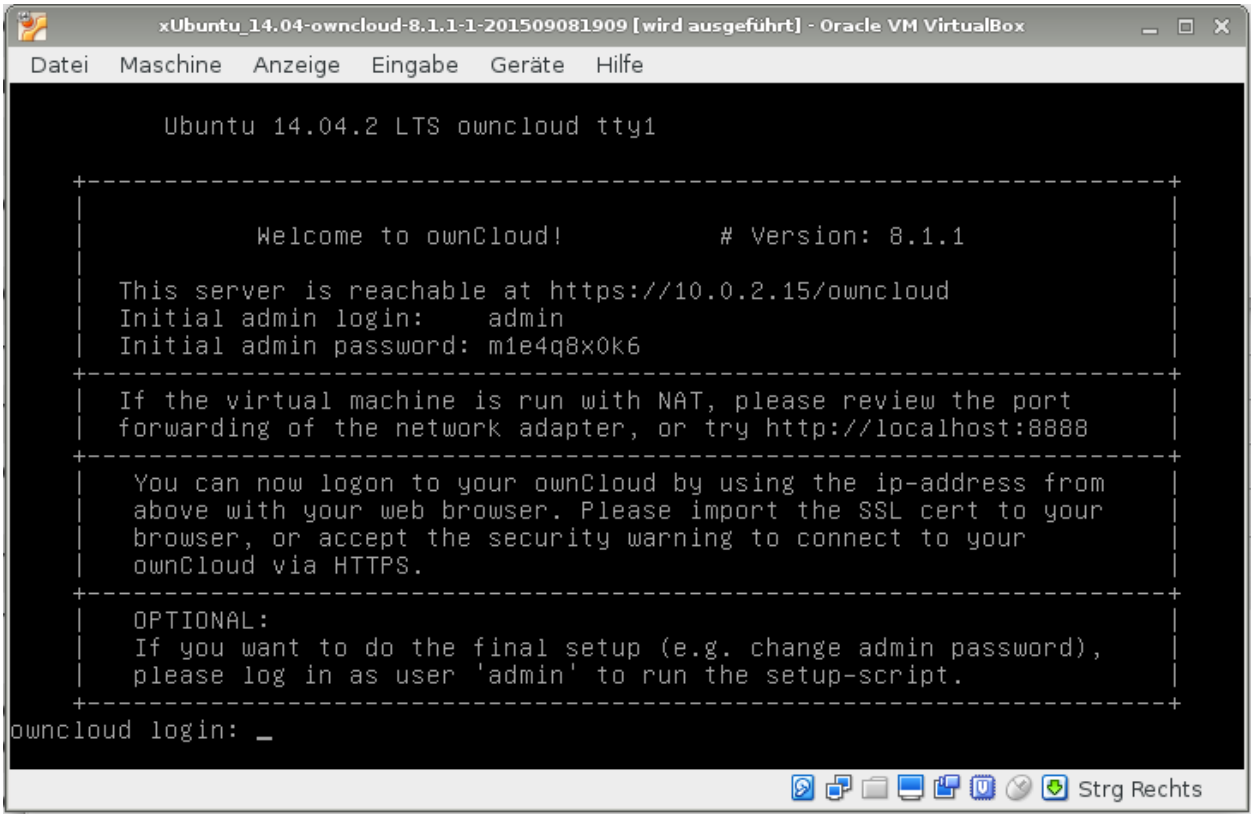


Figure 4.2: *Click to enlarge*

4.7.2 Software Appliances

There are a number of unofficial pre-made virtual machine-based appliances:

- SUSE Studio, ownCloud on openSuSE, which runs directly from an USB stick.
- Amahi home server
- ownCloud VM on Ubuntu 14.04 with MySQL and Apache, fully configured environment.

4.8 SELinux Configuration

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'
restorecon '/var/www/html/owncloud/data'
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/html/owncloud/config'
restorecon '/var/www/html/owncloud/config'
semanage fcontext -a -t httpd_sys_rw_content_t '/var/www/html/owncloud/apps'
restorecon '/var/www/html/owncloud/apps'
```

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 -t httpd_sys_rw_content_t '/var/www/html/owncloud/data'
restorecon '/var/www/html/owncloud/data'
semanage fcontext -d -t httpd_sys_rw_content_t '/var/www/html/owncloud/config'
restorecon '/var/www/html/owncloud/config'
semanage fcontext -d -t httpd_sys_rw_content_t '/var/www/html/owncloud/apps'
restorecon '/var/www/html/owncloud/apps'
```

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
```

4.8.1 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
```

4.8.2 Allow access to LDAP server

Use this setting to allow LDAP connections:

```
setsebool -P httpd_can_connect_ldap on
```

4.8.3 Allow access to remote network

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

```
setsebool -P httpd_can_network_connect on
```

4.8.4 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
```

4.8.5 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
```

4.8.6 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.

4.9 Nginx Configuration

- You need to insert the following code into **your nginx config file.**
- The config assumes that ownCloud is installed in /var/www/owncloud and that it is accessed via http(s)://cloud.example.com.
- Adjust **server_name**, **root**, **ssl_certificate** and **ssl_certificate_key** to suit your needs.
- Make sure your SSL certificates are readable by the server (see Nginx HTTP SSL Module documentation).
- `add_header` statements are only taken from the current level and are not cascaded from or to a different level. All necessary `add_header` statements must be defined in each level needed. For better readability it is possible to move *common* add header statements into a separate file and include that file wherever necessary. However, each `add_header` statement must be written in a single line to prevent connection problems with sync clients.

Note: The following example assumes that your ownCloud is installed in your webroot. If you're using a subfolder you need to adjust the configuration accordingly.

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

server {
```

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

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

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

  # Path to the root of your installation
  root /var/www/owncloud/;
  # set max upload size
  client_max_body_size 10G;
  fastcgi_buffers 64 4K;

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

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

  rewrite ^/caldav(.*)$ /remote.php/caldav$1 redirect;
  rewrite ^/carddav(.*)$ /remote.php/carddav$1 redirect;
  rewrite ^/webdav(.*)$ /remote.php/webdav$1 redirect;

  index index.php;
  error_page 403 /core/templates/403.php;
  error_page 404 /core/templates/404.php;

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

  location ~ ^/(?:\.htaccess|data|config|db_structure\.xml|README){
    deny all;
  }

  location / {
    # The following 2 rules are only needed with webfinger
    rewrite ^/.well-known/host-meta /public.php?service=host-meta last;
    rewrite ^/.well-known/host-meta.json /public.php?service=host-meta-json last;

    rewrite ^/.well-known/carddav /remote.php/carddav/ redirect;
    rewrite ^/.well-known/caldav /remote.php/caldav/ redirect;

    rewrite ^(/core/doc/[^\/]+/)$ $1/index.html;

    try_files $uri $uri/ =404;
  }

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

```
    fastcgi_split_path_info ^(.+\.php)(/.+)$;
    include fastcgi_params;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
    fastcgi_param PATH_INFO $fastcgi_path_info;
    fastcgi_param HTTPS on;
    fastcgi_pass php-handler;
    fastcgi_intercept_errors on;
  }

  # Adding the cache control header for js and css files
  # Make sure it is BELOW the location ~ \.php(?:$|/) { block
  location ~* \.(?:css|js)$ {
    add_header Cache-Control "public, max-age=7200";
    # Add headers to serve security related headers
    add_header Strict-Transport-Security "max-age=15768000; includeSubDomains; preload;";
    add_header X-Content-Type-Options nosniff;
    add_header X-Frame-Options "SAMEORIGIN";
    add_header X-XSS-Protection "1; mode=block";
    add_header X-Robots-Tag none;
    # Optional: Don't log access to assets
    access_log off;
  }

  # Optional: Don't log access to other assets
  location ~* \.(?:jpg|jpeg|gif|bmp|ico|png|swf)$ {
    access_log off;
  }
}
```

Note: 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.

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

Note: If you want to effectively increase maximum upload size you will also have to modify your **php-fpm configuration** (**usually at /etc/php5/fpm/php.ini**) and increase **upload_max_filesize** and **post_max_size** values. You'll need to restart php5-fpm and nginx services in order these changes to be applied.

Note: ownCloud comes with its own `owncloud/.htaccess` file. If PHP-FPM is used, it can't read `.htaccess` PHP settings unless the `htscanner` PECL extension is installed. If PHP-FPM is used without this PECL extension installed, settings and permissions must be set in the `owncloud/.user.ini` file.

Note: If you are using php-fpm please read *php-fpm Configuration Notes*

4.9.1 Suppressing Log Messages

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

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

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

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

This could be caused by the:

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

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

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

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

4.10 Univention Corporate Server

Subscribers to the ownCloud Enterprise edition can also integrate with UCS (Univention Corporate Server).

4.10.1 Pre configuration

ownCloud makes use of the UCR, the Univention Configuration Registry. The values are read during installation, most of them can be changed later, too. Changes done directly via ownCloud are not taken over to UCR. We think we found sane defaults, nevertheless you might have your own requirements. The installation script will listen to the UCR keys listed below. If want to override any default setting, simply add the key in question to the UCR and assign your required value.

Key	Default	Description	Introduced
owncloud/directory/data	/var/lib/owncloud	Specifies where the file storage will be placed	2012.0.1
owncloud/db/name	owncloud	Name of the MySQL database. ownCloud will create an own user for it.	2012.0.1
owncloud/user/quota	(empty)	The default quota, when a user is being added. Assign values in human readable strings, e.g. "2 GB". Unlimited if empty.	2012.0.1
owncloud/user/enabled	0	Whether a new user is allowed to use ownCloud by default.	2012.0.1
owncloud/group/enabled	0	Whether a new group is allowed to be used in ownCloud by default.	2012.4.0.4
owncloud/ldap/base/users	cn=users,$ldap_base	The users-subtree in the LDAP directory. If left blank it will fall back to the LDAP base.	2012.4.0.4

Continued on next page

Table 4.1 – continued from previous page

Key	Default	Description	Introduced
owncloud/ldap/base/groups	cn=groups,$ldap_base	The groups-subtree in the LDAP directory. If left blank it will fall back to the LDAP base.	2012.4.0.4
owncloud/ldap/groupMemberAssoc	uniqueMember	The LDAP attribute showing the group-member relationship. Possible values: uniqueMember, memberUid and member	2012.4.0.4
owncloud/ldap/tls	1	Whether to talk to the LDAP server via TLS.	2012.0.1
owncloud/ldap/disableMainServer	0	Deactivates the (first) LDAP Configuration	5.0.9
owncloud/ldap/cacheTTL	600	Lifetime of the ownCloud LDAP Cache in seconds	5.0.9
owncloud/ldap/UUIDAttribute	(empty)	Attribute that provides a unique value for each user and group entry. Empty value for autodetection.	5.0.9
owncloud/ldap/loginFilter	(&(|(&(objectClass=posixAccount) (objectClass=shadowAccount)) (objectClass=univentionMail) (objectClass=sambaSamAccount) (objectClass=simpleSecurityObject) (&(objectClass=person) (objectClass=organizationalPerson) (objectClass=inetOrgPerson))) (!(uidNumber=0)) (!(uid=*$)) (&(uid=%uid) (ownCloudEnabled=1)))	The LDAP filter that shall be used when a user tries to log in.	2012.0.1
owncloud/ldap/userlistFilter	(&(|(&(objectClass=posixAccount) (objectClass=shadowAccount)) (objectClass=univentionMail) (objectClass=sambaSamAccount) (objectClass=simpleSecurityObject) (&(objectClass=person) (objectClass=organizationalPerson) (objectClass=inetOrgPerson))) (!(uidNumber=0))(!(uid=*$)) (&(ownCloudEnabled=1)))	The LDAP filter that shall be used when the user list is being retrieved (e.g. for sharing)	2012.0.1
owncloud/ldap/groupFilter	(&(objectClass=posixGroup) (ownCloudEnabled=1))	The LDAP filter that shall be used when the group list is being retrieved (e.g. for sharing)	2012.4.0.4
owncloud/ldap/internalNameAttribute	uid	Attribute that should be used to create the user's owncloud internal name	5.0.9
owncloud/ldap/displayName	uid	The LDAP attribute that should be displayed as name in ownCloud	2012.0.1
owncloud/ldap/user/searchAttributes	uid,givenName,sn,description,employeeNumber,mail,primaryAddress	Attributes used in addition when searching for users (comma separated)	5.0.9

Continued on next page

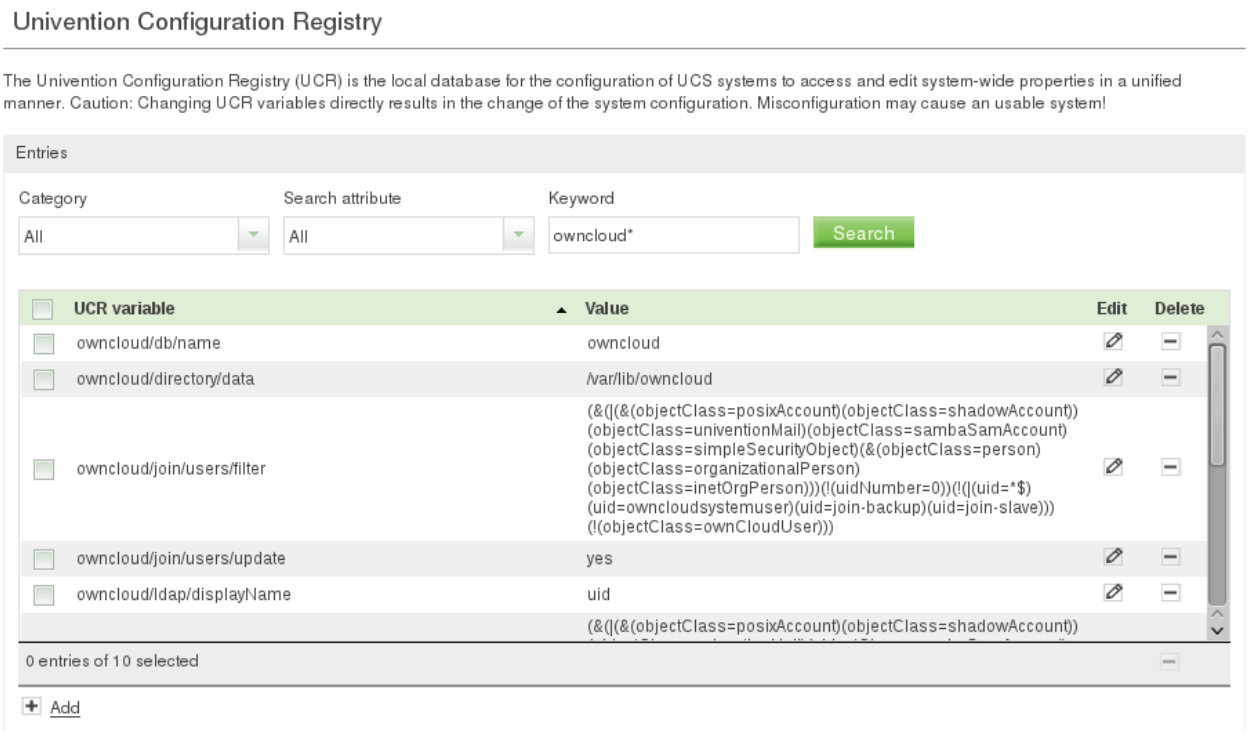
Table 4.1 – continued from previous page

Key	Default	Description	Introduced
owncloud/ldap/user/quotaAttribute	ownCloudQuota	Name of the quota attribute. The default attribute is provided by owncloud-schema.	5.0.9
owncloud/ldap/user/homeAttribute	(empty)	Attribute that should be used to create the user's owncloud internal home folder	5.0.9
owncloud/ldap/group/displayName	cn	The LDAP attribute that should be used as groupname in ownCloud	2012.4.0.4
owncloud/ldap/group/searchAttributes	cn,description, mailPrimaryAddress	Attributes taken into consideration when searching for groups (comma separated)	5.0.9
owncloud/join/users/update	yes	Whether ownCloud LDAP schema should be applied to existing users	2012.0.1
owncloud/group/enableDomainUsers	1	Whether the group "Domain Users" shall be enabled for ownCloud on install	2012.4.0.4
owncloud/join/users/filter	(&(|(&(objectClass=posixAccount) (objectClass=shadowAccount)) (objectClass=univentionMail) (objectClass=sambaSamAccount) (objectClass=simpleSecurityObject) (&(objectClass=person) (objectClass=organizationalPerson) (objectClass=inetOrgPerson))) (!(uidNumber=0)) (!(|(uid=*$) (uid=owncloudsystemuser) (uid=join-backup) (uid=join-slave))) (!(objectClass=ownCloudUser)))	Filters, on which LDAP users the ownCloud schema should be applied to. The default excludes system users and existing ownCloudUsers.	2012.0.1
owncloud/join/groups/filter	(empty)	Filters which LDAP groups will be en/disabled for ownCloud when running the script /usr/share/owncloud/update-groups.sh	2012.4.0.4

If you want to override the default settings, simply create the key in question in the UCR and assign your required value, for example:

```
ucr set owncloud/user/enabled=1
```

or via UMC:



4.10.2 Installation

Now, we are ready to install ownCloud. The recommended method is by using the UCS App Center.

UCS App Center

Open the Univention Management Console and choose the App Center module. You will see a variety of available applications, including ownCloud. You can install and upgrade ownCloud from the App Center.

4.10.3 Postconfiguration (optional)

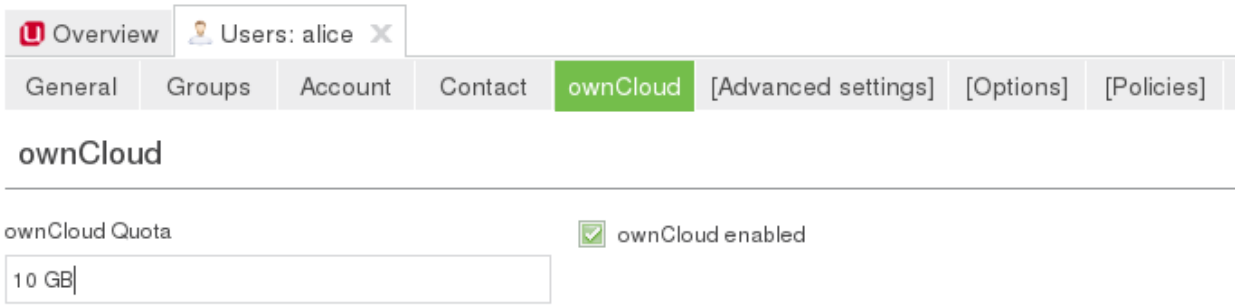
There is only one local admin user "owncloudadmin", you can find the password in */etc/owncloudadmin.secret*. Use this account, if you want to change basic ownCloud settings.

In the installation process a virtual host is set up (Apache is required therefore). If you want to modify the settings, edit `/etc/apache2/sites-available/owncloud` and restart the web server. You might want to do it to enable HTTPS connections. Besides that, you can edit the **.htaccess-File in /var/www/owncloud/**. In the latter file the PHP limits for file transfer are also specified.

4.10.4 Using ownCloud

If you decided to enable every user by default to use ownCloud, simply open up http://myserver.com/owncloud/ and log in with your LDAP credentials and enjoy.

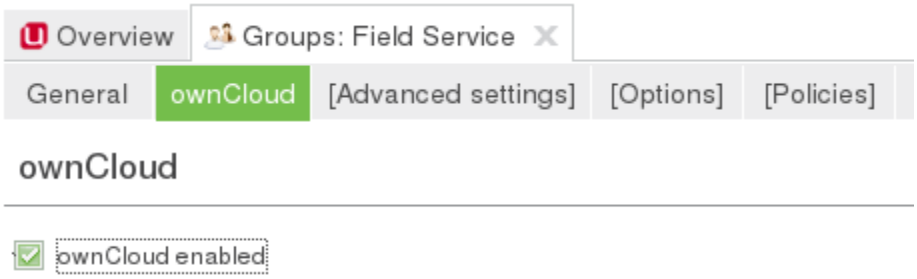
If you did not, go to the UMC and enable the users who shall have access (see picture below). Then, login at http://myserver.com/owncloud/ with your LDAP credentials.



Updating users can also be done by the script `/usr/share/owncloud/update-users.sh` . It takes the following UCR variables as parameters: **owncloud/user/enabled** for enabling or disabling, **owncloud/user/quota** as the Quota value and **owncloud/join/users/filter** as LDAP filter to select the users to update.

Groups

Groups can be enabled and disabled via UCM as shown in the screen shot below.



Another way to enable or disable groups is to use the script `/usr/share/owncloud/update-groups.sh`. Currently, it takes an argument which can be 1=enable groups or 0=disable groups. The filter applied is taken from the UCR variable **owncloud/join/groups/filter**. If it is empty, a message will be displayed.

4.11 Hiawatha Configuration

Add `WebDAVapp = yes` to the ownCloud virtual host. Users accessing WebDAV from MacOS will also need to add `AllowDotFiles = yes`.

Disable access to data folder:

```
UrlToolkit {
    ToolkitID = denyData
    Match ^/data DenyAccess
}
```

4.12 Yaws Configuration

This should be in your **yaws_server.conf**. In the configuration file, the **dir_listings = false** is important and also the redirect from **data/** to somewhere else, because files will be saved in this directory and it should not be accessible from the outside. A configuration file would look like this

```
<server owncloud.myserver.com/>
        port = 80
        listen = 0.0.0.0
        docroot = /var/www/owncloud/src
        allowed_scripts = php
        php_handler = <cgi, /usr/local/bin/php-cgi>
        errormod_404 = yaws_404_to_index_php
        access_log = false
        dir_listings = false
        <redirect>
                /data == /
        </redirect>
</server>
```

The Apache `.htaccess` that comes with ownCloud is configured to redirect requests to non-existent pages. To emulate that behaviour, you need a custom error handler for yaws. See this github gist for further instructions on how to create and compile that error handler.

4.13 Mac OS X

Note: Due to an issue with Mac OS Unicode support, installing ownCloud Server 8.0 on Mac OS is currently not supported.

CHAPTER

FIVE

USER MANAGEMENT

5.1 User Management

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

- Create new users
- View all of your users in a single scrolling window
- Filter users by group
- See what groups they belong to
- Edit their full names and passwords
- See their data storage locations
- View and set quotas
- Create and edit their email addresses
- Send an automatic email notification to new users
- Delete them with a single click

The default view displays basic information about your users.

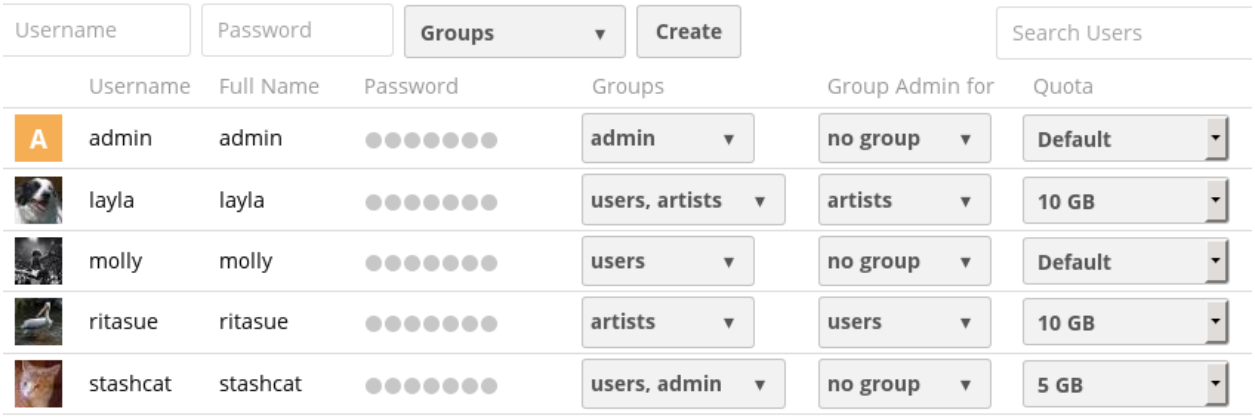
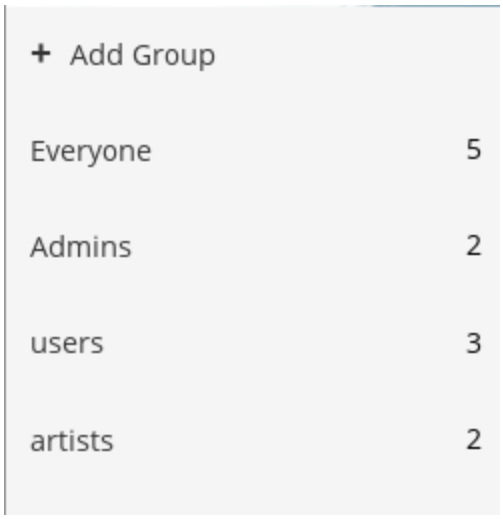
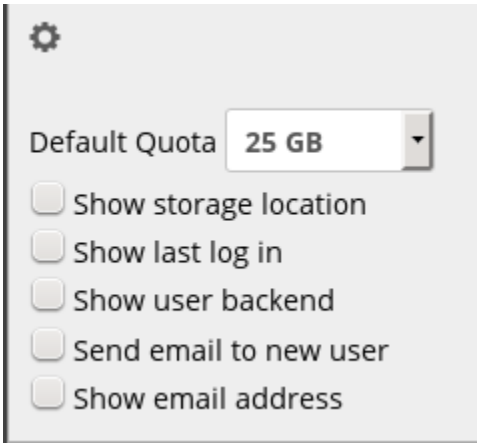


Figure 5.1: *The user administration page displays all users in a table format.*

The Group filters on the left sidebar lets you quickly filter users by their group memberships, and create new groups.



Click the gear icon on the lower left sidebar to set a default storage quota, and to display additional fields: **Show storage location, Show last log in, Show user backend, Send email to new users,** and **Show email address**.



User accounts have the following properties:

Login Name (Username) The unique ID of an 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.

Groups You may create groups, and assign group memberships to users. By default new users are not assigned to any groups.

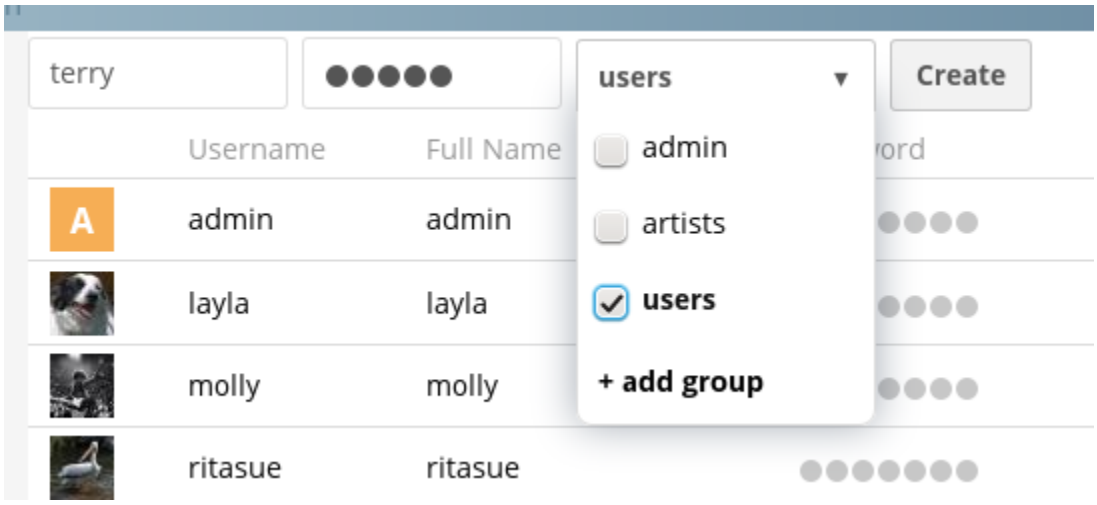
Group Admin Group admins are granted administrative privileges on specific groups, and can add and remove users from their groups.

Quota The maximum disk space assigned to each user. Any user that exceeds the quota cannot upload or sync data. You have the the option to include external storage in user quotas.

5.1.1 Creating a New User

To create a user account:

- Enter the new user's **Login Name** and their initial **Password**
- Optionally, assign **Groups** memberships
- Click the **Create** button



Login names may contain letters (a-z, A-Z), numbers (0-9), dashes (-), underscores (_), periods (.) and at signs (@). After creating the user, you may fill in their **Full Name** if it is different than the login name, or leave it for the user to complete.

If you have checked **Send email to new user** in the control panel on the lower left sidebar, you may also enter the new user's email address, and ownCloud will automatically send them a notification with their new login information. You may edit this email using the email template editor on your Admin page (see *Email Configuration*).

5.1.2 Reset a User's Password

You cannot recover a user's password, but you can set a new one:

- Hover your cursor over the user's **Password** field
- Click on the **pencil icon**
- Enter the user's new password in the password field, and remember to provide the user with their password

If you have encryption enabled, there are special considerations for user password resets. Please see *Encryption Configuration*.

5.1.3 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.

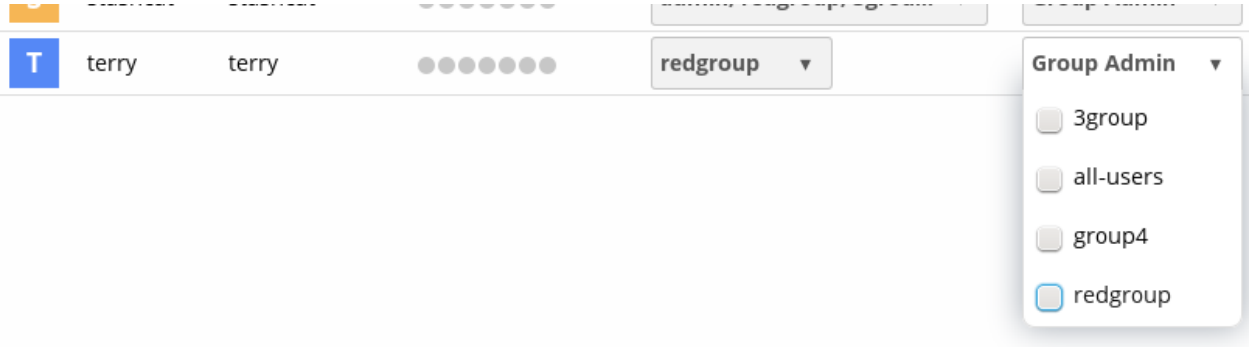
To set or change a user's display name:

- Hover your cursor over the user's **Full Name** field
- Click on the **Pencil icon**

- Enter the user's new display name

5.1.4 Granting Administrator Privileges to a User

ownCloud has two types of administrators: **Super Administrators** and **Group Administrators**. Group administrators have the rights to create, edit and delete users in their assigned groups. Group administrators cannot access system settings, or add or modify users in the groups that they are not **Group Administrators** for. Use the dropdown menus in the **Group Admin** column to assign group admin privileges.



Super Administrators have full rights on your ownCloud server, and can access and modify all settings. To assign the **Super Administrators** role to a user, simply add them to the `admin` group.

5.1.5 Managing Groups

You can assign new users to groups when you create them, and create new groups when you create new users. You may also use the **Add Group** button at the top of the left pane to create new groups. New group members will immediately have access to file shares that belong to their new groups.

5.1.6 Setting Storage Quotas

Click the gear on the lower left pane to set a default storage quota. This is automatically applied to new users. You may assign a different quota to any user by selecting from the **Quota** dropdown, selecting either a preset value or entering a custom value. When you create custom quotas, use the normal abbreviations for your storage values such as 500 MB, 5 GB, 5 TB, and so on.

You now have a configurable option in `config.php` that controls whether external storage is counted against user's quotas. This is still experimental, and may not work as expected. The default is to not count external storage as part of user storage quotas. If you prefer to include it, then change the default `false` to `true`.:

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

Metadata (such as thumbnails, temporary files, and encryption keys) takes up about 10% of disk space, but is not counted against user quotas. Users can check their used and available space on their Personal pages. Only files that originate with users count against their quotas, and not files shared with them that originate from other users. For example, if you upload files to a different user's share, those files count against your quota. If you re-share a file that another user shared with you, that file does not count against your quota, but the originating user's.

Encrypted files are a little larger than unencrypted files; the unencrypted size is calculated against the user's quota.

Deleted files that are still in the trash bin do not count against quotas. The trash bin is set at 50% of quota. Deleted file aging is set at 30 days. When deleted files exceed 50% of quota then the oldest files are removed until the total is below 50%.

When version control is enabled, the older file versions are not counted against quotas.

When a user creates a public share via URL, and allows uploads, any uploaded files count against that user's quota.

5.1.7 Deleting users

Deleting a user is easy: hover your cursor over their name on the **Users** page until a trashcan icon appears at the far right. Click the trashcan, and they're gone. You'll see an undo button at the top of the page, which remains until you refresh the page. When the undo button is gone you cannot recover the deleted user.

All of the files owned by the user are deleted as well, including all files they have shared. If you need to preserve the user's files and shares, you must first download them from your ownCloud Files page, which compresses them into a zip file, or use a sync client to copy them to your local computer. See *File Sharing* to learn how to create persistent file shares that survive user deletions.

5.2 Resetting a Lost Admin Password

The normal ways to recover a lost password are:

1. Click the password reset link on the login screen; this appears after a failed login attempt. This works only if you have entered your email address on your Personal page in the 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:

```
$ sudo -u www-data php /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.

5.3 User Authentication with IMAP, SMB, and FTP

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

```
<?php

"user_backends" => array (
    0 => array (
```

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

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

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

5.3.1 IMAP

Provides authentication against IMAP servers

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

```
<?php

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

5.3.2 SMB

Provides authentication against Samba servers

- **Class:** OC_User_SMB
- **Arguments:** the samba server to authenticate against
- **Dependency:** smbclient (See *Manual Installation on Linux*)
- **Example:**

```
<?php

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

5.3.3 FTP

Provides authentication against FTP servers

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

```
<?php

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

5.4 User Authentication with LDAP

ownCloud ships with an LDAP application to allow LDAP users (including Active Directory) to appear in your ownCloud user listings. These users will authenticate to ownCloud with their LDAP credentials, so you don't have to create separate ownCloud user accounts for them. You will manage their ownCloud group memberships, quotas, and sharing permissions just like any other ownCloud user.

Note: The PHP LDAP module is required; this is supplied by php5-ldap on Debian/Ubuntu, and php-ldap on CentOS/Red Hat/Fedora. PHP 5.4+ is required in ownCloud 8.

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
- Auto-detection of LDAP attributes such as base DN, email, and the LDAP server port number
- Only read access to your LDAP (edit or delete of users on your LDAP is not supported)

Warning: The LDAP app is not compatible with the `WebDAV user backend` app. You cannot use both of them at the same time.

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

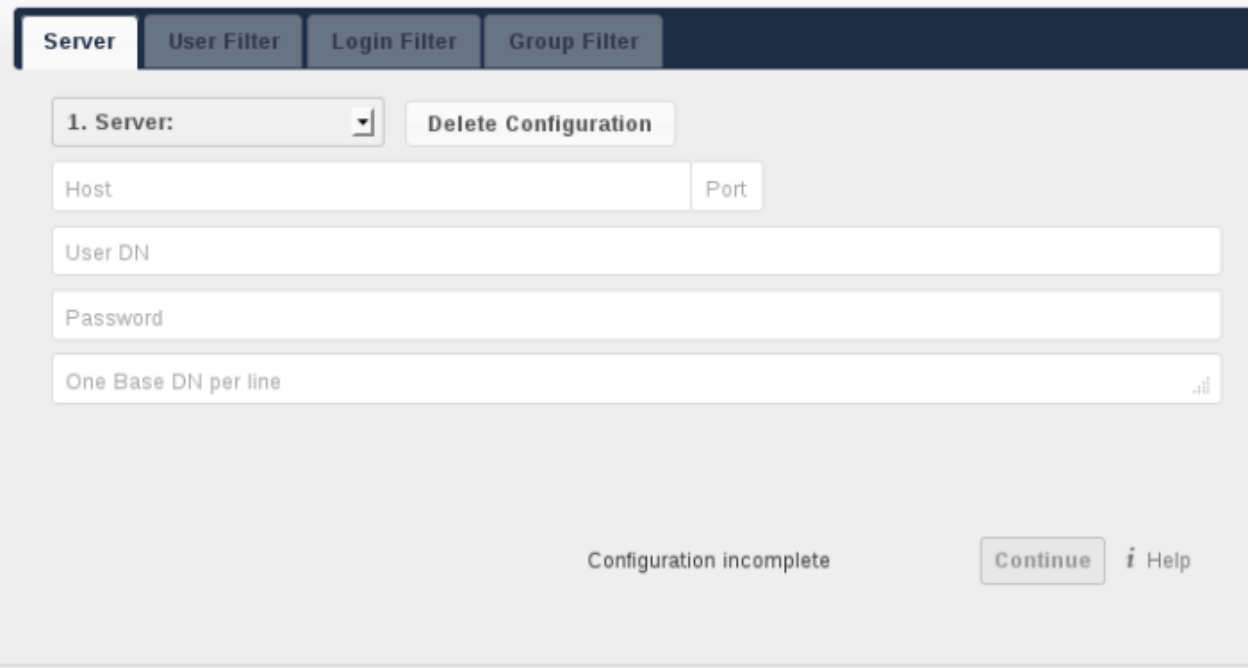
5.4.1 Configuration

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 lights when the configuration is correct. Hover your cursor over the fields to see some pop-up tooltips.

Server Tab

Start with the Server tab. You may configure multiple servers if you have them. At a minimum you must supply the LDAP server's hostname. If your server requires authentication, enter your credentials on this tab. ownCloud will then attempt to auto-detect the server's port and base DN. The base DN and port are mandatory, so if ownCloud cannot detect them you must enter them manually.



Server configuration: Configure one or more LDAP servers. Click the **Delete Configuration** button to remove the active configuration.

Host: The host name or IP address of the LDAP server. It can also be a **ldaps://** URI. If you enter the port number, it speeds up server detection.

Examples:

- *directory.my-company.com*
- *ldaps://directory.my-company.com*
- *directory.my-company.com:9876*

Port: The port on which to connect to the LDAP server. The field is disabled in the beginning of a new configuration. If the LDAP server is running on a standard port, the port will be detected automatically. If you are using a non-standard port, ownCloud will attempt to detect it. If this fails you must enter the port number manually.

Example:

- *389*

User DN: The name as DN of a user who has permissions to do searches in the LDAP directory. Leave it empty for anonymous access. We recommend that you have a special LDAP system user for this.

Example:

- *uid=owncloudsystemuser,cn=sysusers,dc=my-company,dc=com*

Password: The password for the user given above. Empty for anonymous access.

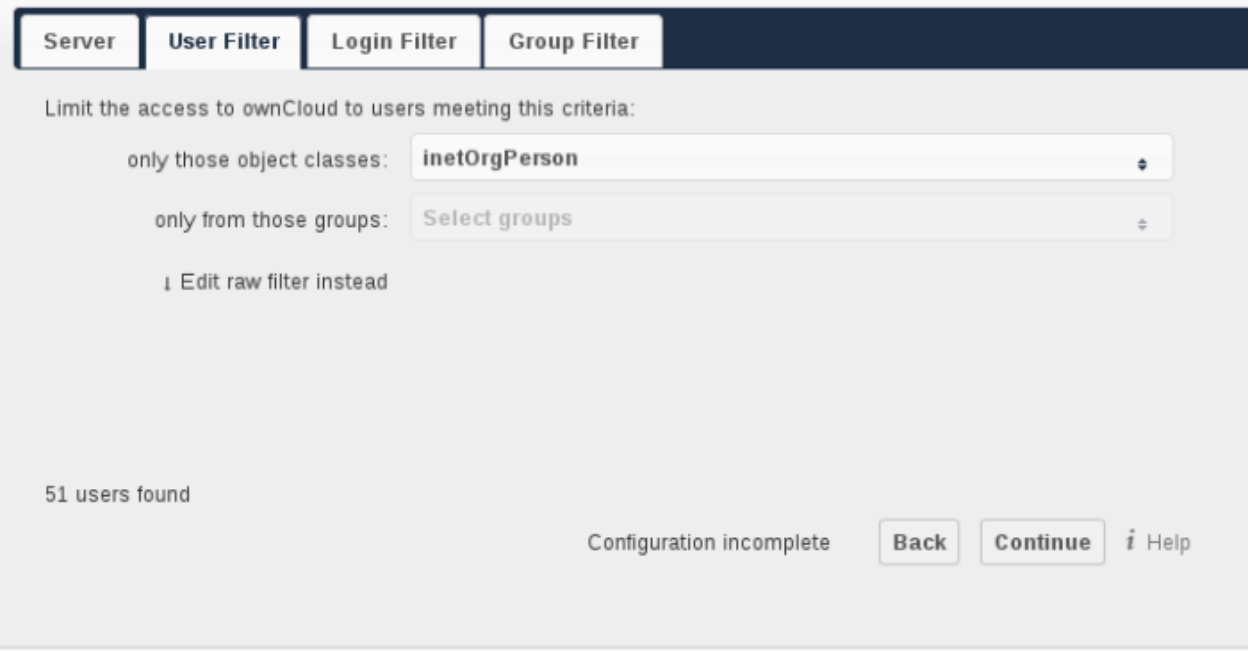
Base DN: The base DN of LDAP, from where all users and groups can be reached. You may enter multiple base DNs, one per line. (Base DNs for users and groups can be set in the Advanced tab.) This field is mandatory. ownCloud attempts to determine the Base DN according to the provided User DN or the provided Host, and you must enter it manually if ownCloud does not detect it.

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 login to your ownCloud server use the Login filter. Those LDAP users who have access but are not listed as users (if there are any) will be hidden users. You may bypass the form fields and enter a raw LDAP filter if you prefer.



only those object classes: ownCloud will determine the object classes that are typically available for user objects in your LDAP. ownCloud will automatically select the object class that returns the highest amount of users. You may select multiple object classes.

only from those groups: If your LDAP server supports the `member-of-overlay` in LDAP filters, you can define that only users from one or more certain groups are allowed to appear in user listings in ownCloud. By default, no value will be selected.

You may select multiple groups.

If your LDAP server does not support the member-of-overlay in LDAP filters, the input field is disabled. Please contact your LDAP administrator.

Edit raw filter instead: Clicking on this text toggles the filter mode and you can enter the raw LDAP filter directly.

Example:

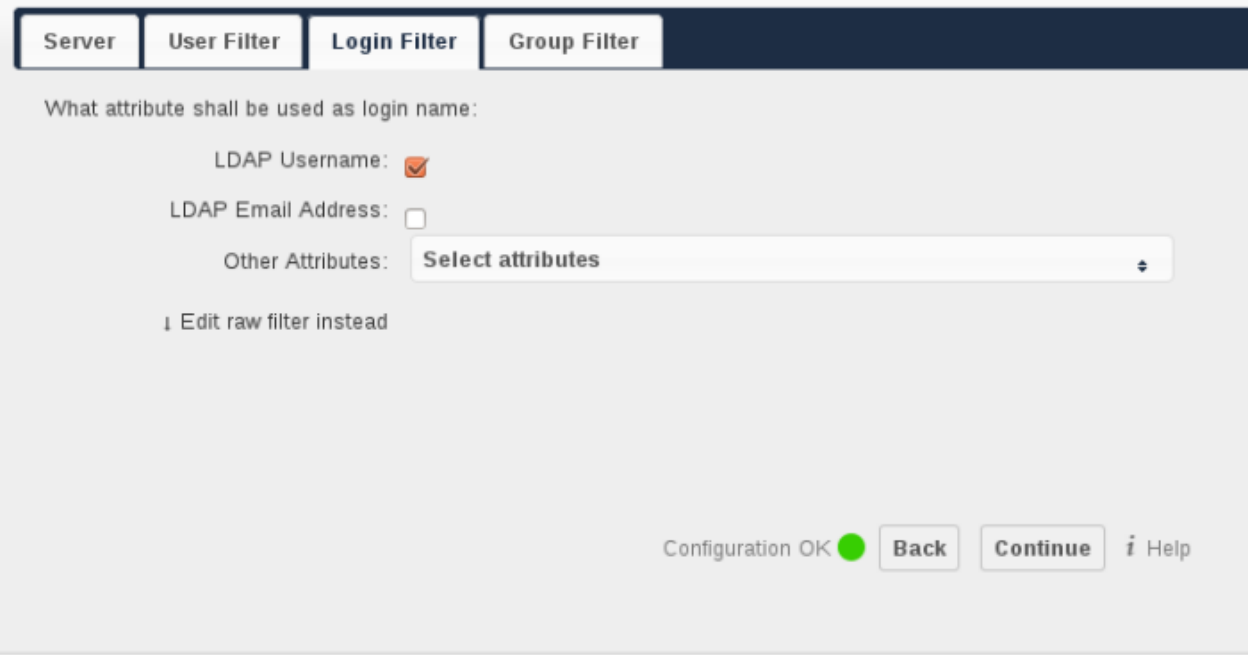
- *(&(objectClass=inetOrgPerson)(memberOf=cn=owncloudusers,ou=groups,dc=example,dc=com))*

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.

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.

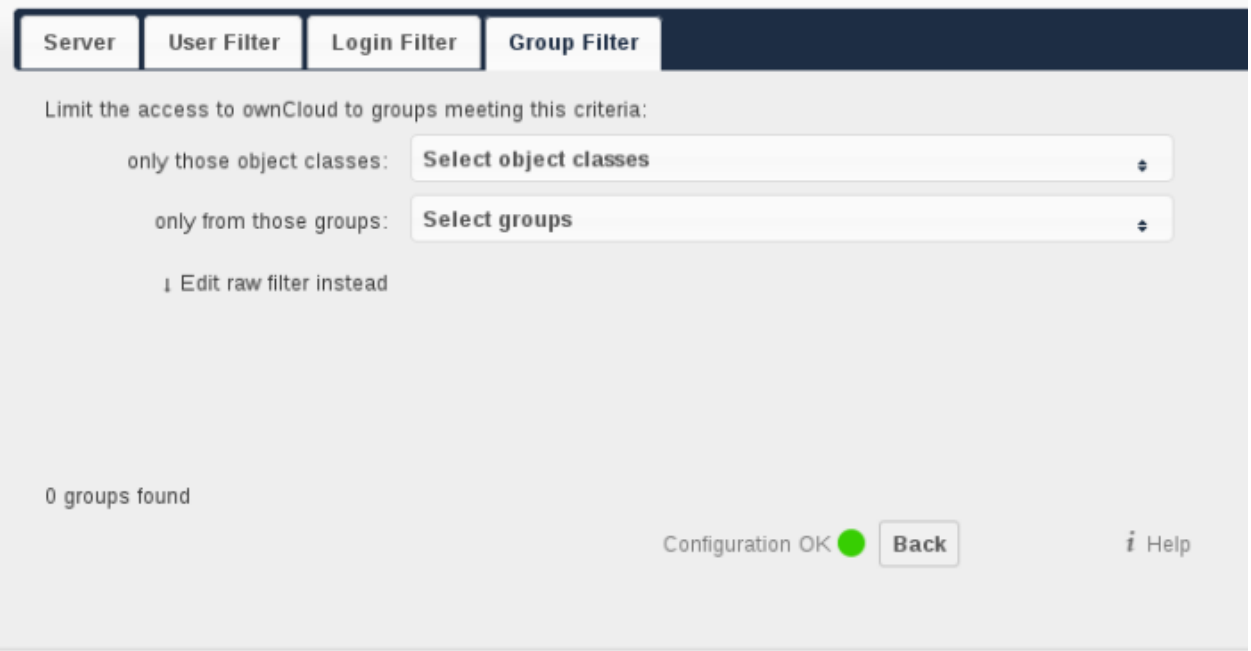
The **%uid** placeholder is replaced with the login name entered by the user upon login.

Examples:

- only username: *(&(objectClass=inetOrgPerson)(memberOf=cn=owncloudusers,ou=groups,dc=example,dc=com)(uid=%u*
- username or email address: *((&(objectClass=inetOrgPerson)(memberOf=cn=owncloudusers,ou=groups,dc=example,dc=co*

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. and then you 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.

5.4.2 Advanced Settings

The LDAP Advanced Setting section contains options that are not needed for a working connection. This provides controls to disable the current configuration, configure replica hosts, and various performance-enhancing options.

The Advanced Settings are structured into three parts:

- Connection Settings
- Directory Settings
- Special Attributes

Connection Settings

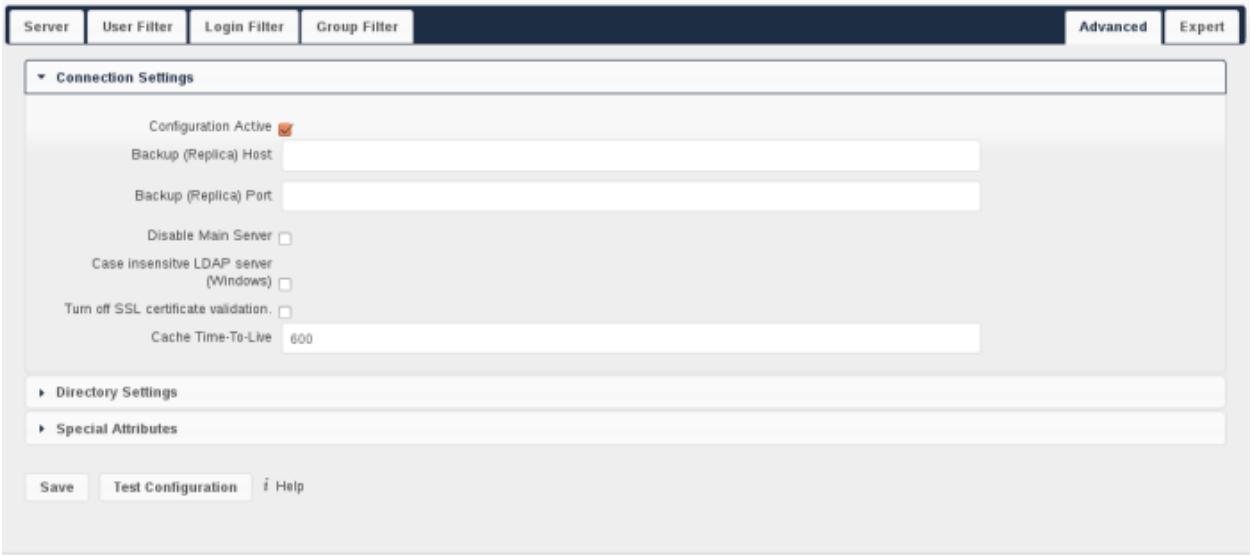


Figure 5.2: LDAP Advanced Settings, section Connection Settings

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 given, but only a host, then the main port (as specified above) will be used.

Example:

- *389*

Disable Main Server: You can manually override the main server and make ownCloud only connect to the backup server. This is useful for planned downtimes.

Case insensitive LDAP server (Windows): When the LDAP server is running on a Windows Host.

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

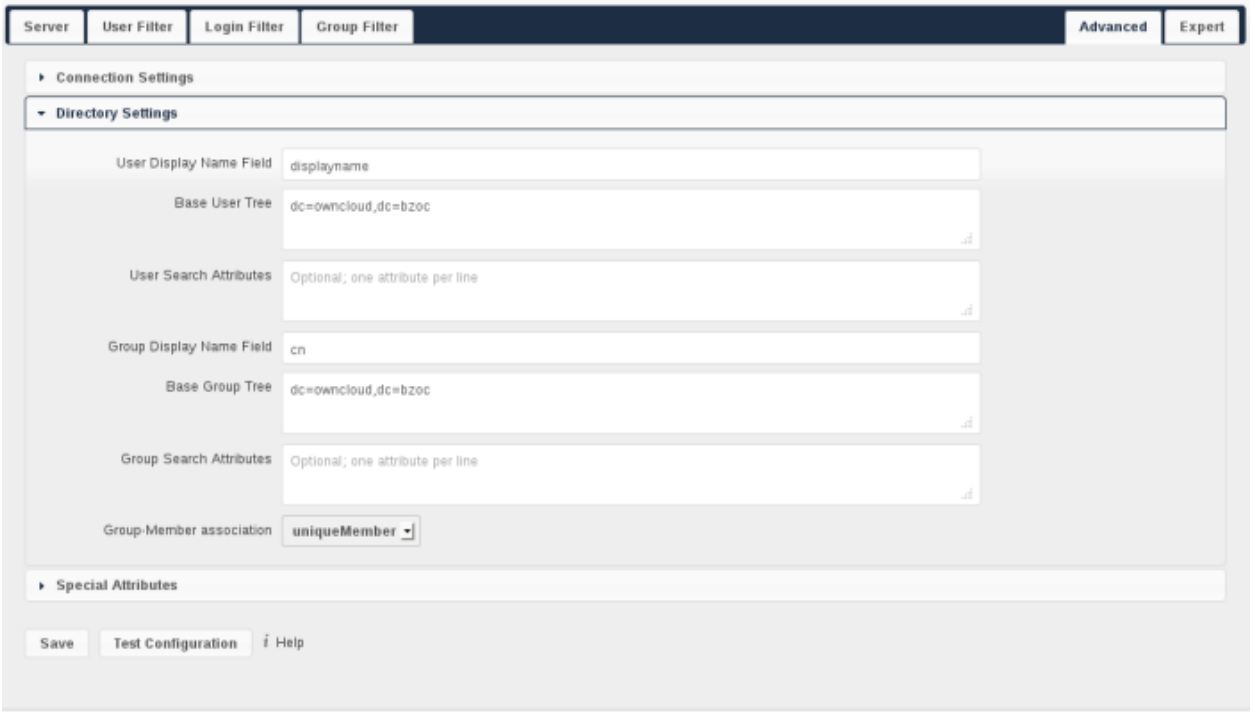


Figure 5.3: LDAP Advanced Settings, section Directory Settings

User Display Name Field: The attribute that should be used as display name in ownCloud.

- Example: *displayName*

Base User Tree: The base DN of LDAP, from where all users can be reached. This must be a complete DN, regardless of what you have entered for your Base DN in the Basic setting. You can specify multiple base trees, one on each line.

- Example:

cn=programmers,dc=my-company,dc=com
cn=designers,dc=my-company,dc=com

User Search Attributes: These attributes are used when searches for users are performed, for example in the share dialogue. The user display name attribute is the default. You may list multiple attributes, one per line.

If an attribute is not available on a user object, the user will not be listed, and will be unable to login. This also affects the display name attribute. If you override the default you must specify the display name attribute here.

- Example:

  displayName
  mail

Group Display Name Field: The attribute that should be used as ownCloud group name. ownCloud allows a limited set of characters (a-zA-Z0-9.-_@). Once a group name is assigned it cannot be changed.

- Example: *cn*

Base Group Tree: The base DN of LDAP, from where all groups can be reached. This must be a complete DN, regardless of what you have entered for your Base DN in the Basic setting. You can specify multiple base trees, one in each line.

- Example:

  cn=barcelona,dc=my-company,dc=com
  cn=madrid,dc=my-company,dc=com

Group Search Attributes: These attributes are used when a search for groups is done, for example in the share dialogue. By default the group display name attribute as specified above is used. Multiple attributes can be given, one in each line.

If you override the default, the group display name attribute will not be taken into account, unless you specify it as well.

- Example:

  cn
  description

Group Member association: The attribute that is used to indicate group memberships, i.e. the attribute used by LDAP groups to refer to their users.

ownCloud detects the value automatically. You should only change it if you have a very valid reason and know what you are doing.

- Example: *uniquemember*

Special Attributes

Quota Field: ownCloud can read an LDAP attribute and set the user quota according to its value. Specify the attribute here, and it will return human-readable values, e.g. "2 GB". Any quota set in LDAP overrides quotas set on the ownCloud user management page.

- Example: *ownCloudQuota*

Quota Default: Override ownCloud default quota for LDAP users who do not have a quota set in the Quota Field.

- Example: *15 GB*

Email Field: Set the user's email from their LDAP attribute. Leave it empty for default behavior.

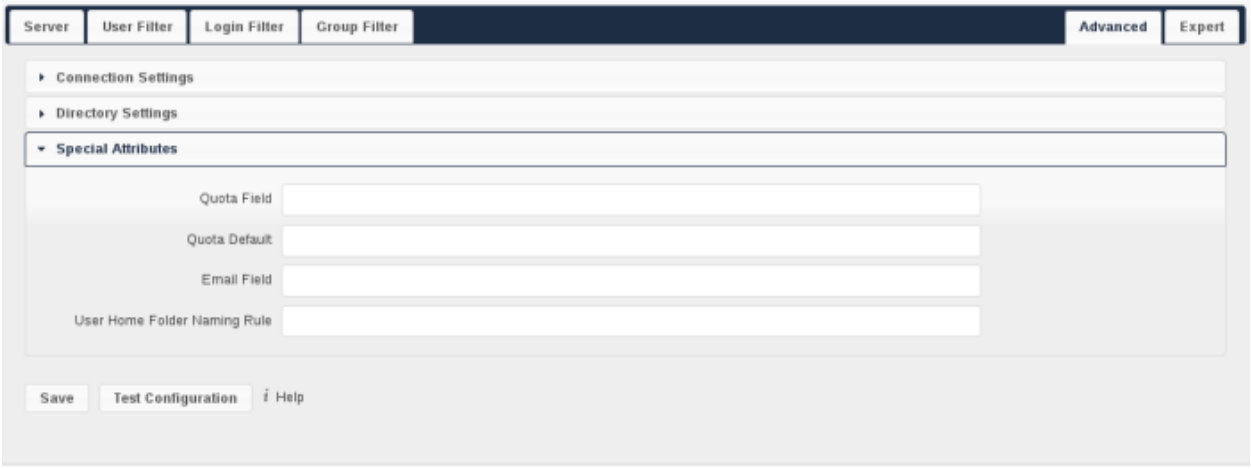


Figure 5.4: LDAP Advanced Settings, section Special Attributes

- Example: *mail*

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/alice`. You may want to override this setting and name it after an LDAP attribute value. The attribute can also return an absolute path, e.g. `/mnt/storage43/alice`. Leave it empty for default behavior.

- Example: *cn*

In new ownCloud installations (8.0.10, 8.1.5, 8.2.0 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.

In existing ownCloud installations the old behavior still applies, which is using the ownCloud username as the home folder when an LDAP attribute is not set.

5.4.3 Expert Settings

In the Expert Settings fundamental behavior can be adjusted to your needs. The configuration should be well-tested before starting production use.

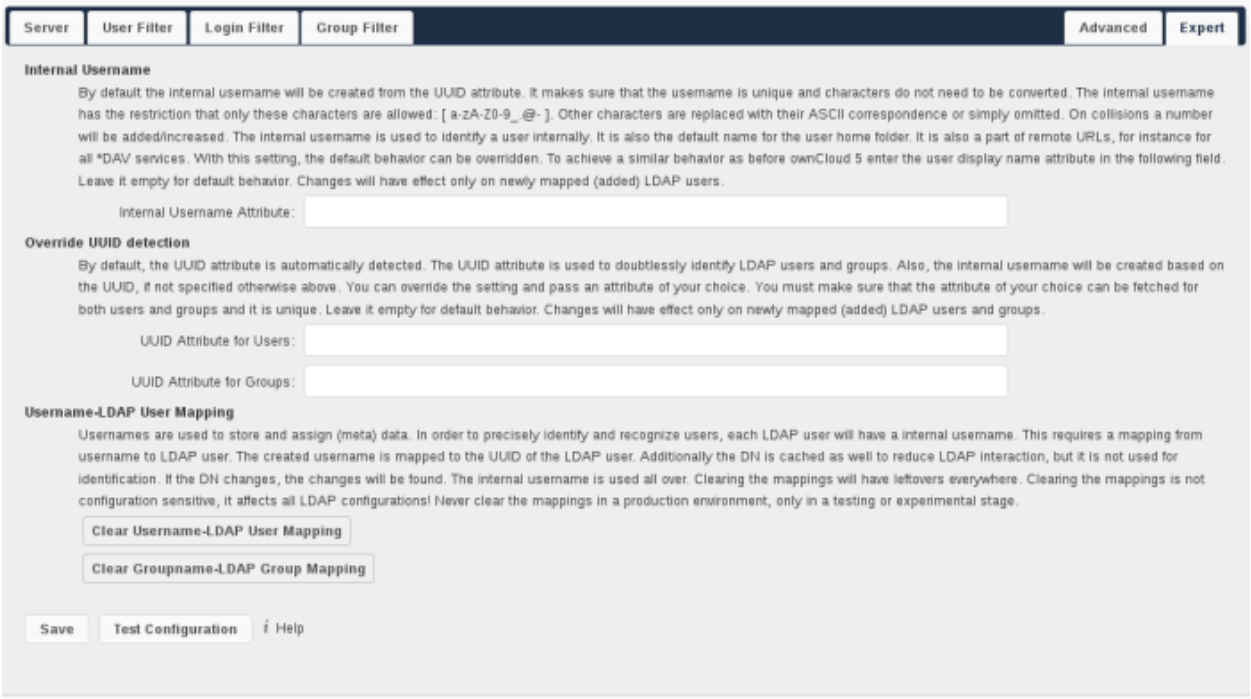
Internal Username: The internal username is the identifier in ownCloud for LDAP users. By default it will be created from the UUID attribute. The UUID attribute ensures that the username is unique, and that characters do not need to be converted. Only these characters are allowed: [a-zA-Z0-9_.@-]. Other 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 behaviour. Changes will affect only newly mapped LDAP users.

- Example: *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 it is unique. Leave it empty for default behaviour. Changes will have effect only on newly mapped LDAP users and groups. It also will have effect when a user's or group's DN changes and an old UUID was cached, which will result in a new user. Because of this, the setting should be applied before putting ownCloud in production use and clearing the bindings (see the `User and Group Mapping` section below).

- Example: *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 ownCloud username to LDAP user. The created username is mapped to the UUID of the LDAP user. Additionally the DN is cached as well to reduce LDAP interaction, but it is not used for identification. If the DN changes, the change will be detected by 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!

5.4.4 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. The response will look like this:

Figure 5.5: Failure

In case the configuration fails, you can see details in ownCloud's log, which is in the data directory and called **owncloud.log** or on the bottom the **Settings – Admin page**. You must refresh the Admin page to see the new log entries.

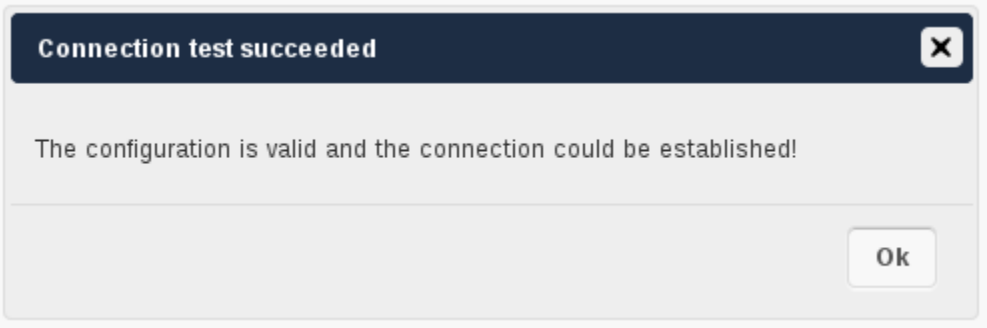


Figure 5.6: Success

In this case, Save the settings. You can check if the users and groups are fetched correctly on the Users page.

5.4.5 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.

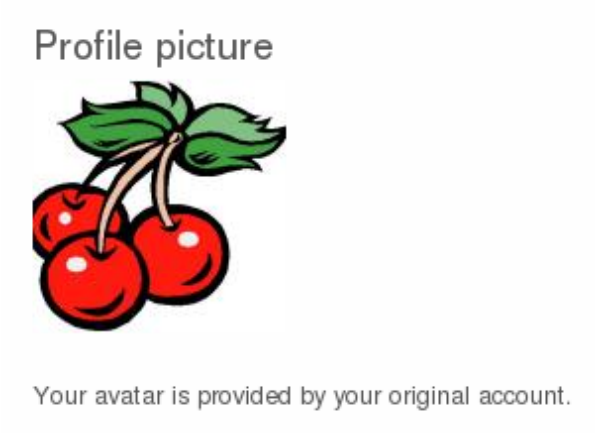


Figure 5.7: Profile picture fetched from LDAP

5.4.6 Troubleshooting, Tips and Tricks

5.4.7 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 announced in the system's LDAP configuration file (usually */etc/ldap/ldap.conf* on Linux, *C:\openldap\sysconf\ldap.conf* or *C:\ldap.conf* on Windows) using a **TLS_CACERT /path/to/cert** line.
- Using LDAPS, also make sure that the port is correctly configured (by default 636)

5.4.8 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.

5.4.9 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.

5.4.10 Duplicating Server Configurations

In case you have a working configuration and want to create a similar one or "snapshot" configurations before modifying them you can do the following:

1. Go to the **Server** tab
2. On **Server Configuration** choose *Add Server Configuration*
3. Answer the question *Take over settings from recent server configuration?* with *yes*.
4. (optional) Switch to **Advanced** tab and uncheck **Configuration Active** in the *Connection Settings*, so the new configuration is not used on Save

5. Click on **Save**

Now you can modify and enable the configuration.

5.4.11 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 is created and mapped to the LDAP DN and UUID. If the DN changes in LDAP it will be detected, and there will be no conflicts.

Those mappings are done in the database table `ldap_user_mapping` and `ldap_group_mapping`. The user name is also used for the user's folder (except if something else is specified in *User Home Folder Naming Rule*), which contains files and meta data.

As of 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.

Caching

The ownCloud **Cache** helps to speed up user interactions and sharing. It is populated on demand, and remains populated until the **Cache Time-To-Live** for each unique request expires. User logins are not cached, so if you need to improve login times set up a slave LDAP server to share the load.

Another significant performance enhancement is to install the Alternative PHP Cache (APC). APC is an OPcache, which is several times faster than a file cache. APC improves PHP performance by storing precompiled script bytecode in shared memory, which reduces the overhead of loading and parsing scripts on each request. (See http://php.net/manual/en/book.apc.php for more information.)

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 presumes PHP >= 5.4.

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.

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 **Disable Main Server** to avoid unnecessary connection attempts.

5.5 LDAP User Cleanup

LDAP User Cleanup is a new feature in the `LDAP user and group backend` application. LDAP User Cleanup is a background process that automatically searches the ownCloud LDAP mappings table, and verifies if the LDAP users are still available. Any users that are not available are marked as `deleted` in the `oc_preferences` database table. Then you can run a command to display this table, displaying only the users marked as `deleted`, and then you have the option of removing their data from your ownCloud data directory.

These items are removed upon cleanup:

- Local ownCloud group assignments
- User preferences (DB table `oc_preferences`)
- User's ownCloud home folder
- User's corresponding entry in `oc_storages`

There are two prerequisites for LDAP User Cleanup to operate:

1. Set `ldapUserCleanupInterval` in `config.php` to your desired check interval in minutes. The default is 51 minutes.
2. All configured LDAP connections are enabled and operating correctly. As users can exist on multiple LDAP servers, you want to be sure that all of your LDAP servers are available so that a user on a temporarily disconnected LDAP server is not marked as `deleted`.

The background process examines 50 users at a time, and runs at the interval you configured with `ldapUserCleanupInterval`. For example, if you have 200 LDAP users and your `ldapUserCleanupInterval` is 20 minutes, the process will examine the first 50 users, then 20 minutes later the next 50 users, and 20 minutes later the next 50, and so on.

There are two `occ` commands to use for examining a table of users marked as deleted, and then manually deleting them. The `occ` command is in your ownCloud directory, for example `/var/www/owncloud/occ`, and it must be run as your HTTP user. To learn more about `occ`, see *Using the occ Command*.

These examples are for Ubuntu Linux:

1. `sudo -u www-data php occ ldap:show-remnants` displays a table with all users that have been marked as deleted, and their LDAP data.
2. `sudo -u www-data php occ user:delete [user]` removes the user's data from the ownCloud data directory.

This example shows what the table of users marked as `deleted` looks like:

```
$ sudo -u www-data php occ ldap:show-remnants
+-----------------+-----------------+------------------+-------------------------------------+
| ownCloud name   | Display Name    | LDAP UID         | LDAP DN                             |
+-----------------+-----------------+------------------+-------------------------------------+
| aaliyah_brown   | aaliyah brown   | aaliyah_brown    | uid=aaliyah_brown,ou=people,dc=com  |
| aaliyah_hammes  | aaliyah hammes  | aaliyah_hammes   | uid=aaliyah_hammes,ou=people,dc=com |
| aaliyah_johnston| aaliyah johnston| aaliyah_johnston | uid=aaliyah_johnston,ou=people,dc=com|
```

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

Then you can run `sudo -u www-data php occ user:delete aaliyah_brown` to delete user aaliyah_brown. You must use the user's ownCloud name.

5.5.1 Deleting Local ownCloud Users

You may also use `occ user:delete [user]` to remove a local ownCloud user; this removes their user account and their data.

5.6 User Provisioning API

The Provisioning API application enables a set of APIs that external systems can use to create, edit, delete and query user attributes, query, set and remove groups, set quota and query total storage used in ownCloud. Group admin users can also query ownCloud and perform the same functions as an admin for groups they manage. The API also enables an admin to query for active ownCloud applications, application info, and to enable or disable an app remotely. HTTP requests can be used via a Basic Auth header to perform any of the functions listed above. The Provisioning API app is enabled by default.

The base URL for all calls to the share API is **owncloud_base_url/ocs/v1.php/cloud**.

5.6.1 Instruction Set For Users

users / adduser

Create a new user on the ownCloud server. Authentication is done by sending a basic HTTP authentication header.

Syntax: ocs/v1.php/cloud/users

- HTTP method: POST
- POST argument: userid - string, the required username for the new user
- POST argument: password - string, the required password for the new user

Status codes:

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

Example

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

XML Output

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

users / getusers

Retrieves a list of users from the ownCloud server. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users

- HTTP method: GET
- url arguments: search - string, optional search string
- url arguments: limit - int, optional limit value
- url arguments: offset - int, optional offset value

Status codes:

- 100 - successful

Example

- GET `http://admin:secret@example.com/ocs/v1.php/cloud/users?search=Frank`
- Returns list of users matching the search string.

XML Output

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

users / getuser

Retrieves information about a single user. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}

- HTTP method: GET

Status codes:

- 100 - successful

Example

- GET `http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank`
- Returns information on the user `Frank`

XML Output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>100</statuscode>
    <status>ok</status>
  </meta>
  <data>
    <email>frank@example.org</email>
    <quota>0</quota>
    <enabled>true</enabled>
  </data>
</ocs>
```

users / edituser

Edits attributes related to a user. Users are able to edit email, displayname and password; admins can also edit the quota value. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}

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

Status codes:

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

Examples

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

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

XML Output

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

users / deleteuser

Deletes a user from the ownCloud server. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}

- HTTP method: DELETE

Statuscodes:

- 100 - successful
- 101 - failure

Example

- DELETE `http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank`
- Deletes the user `Frank`

XML Output

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

users / getgroups

Retrieves a list of groups the specified user is a member of. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}/groups

- HTTP method: GET

Status codes:

- 100 - successful

Example

- GET `http://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/groups`
- Retrieves a list of groups of which `Frank` is a member

XML Output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>100</statuscode>
    <status>ok</status>
  </meta>
  <data>
    <groups>
      <element>admin</element>
      <element>group1</element>
    </groups>
  </data>
</ocs>
```

users / addtogroup

Adds the specified user to the specified group. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}/groups

- HTTP method: POST
- POST argument: groupid, string - the group to add the user to

Status codes:

- 100 - successful
- 101 - no group specified
- 102 - group does not exist
- 103 - user does not exist
- 104 - insufficient privileges
- 105 - failed to add user to group

Example

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

XML Output

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

users / removefromgroup

Removes the specified user from the specified group. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}/groups

- HTTP method: DELETE
- POST argument: groupid, string - the group to remove the user from

Status codes:

- 100 - successful
- 101 - no group specified
- 102 - group does not exist
- 103 - user does not exist
- 104 - insufficient privileges
- 105 - failed to remove user from group

Example

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

XML Output

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

users / createsubadmin

Makes a user the subadmin of a group. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}/subadmins

- HTTP method: POST
- POST argument: groupid, string - the group of which to make the user a subadmin

Status codes:

- 100 - successful
- 101 - user does not exist
- 102 - group does not exist
- 103 - unknown failure

Example

- POST `https://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/subadmins -d groupid="group"`
- Makes the user `Frank` a subadmin of the `group` group

XML Output

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

users / removesubadmin

Removes the subadmin rights for the user specified from the group specified. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}/subadmins

- HTTP method: DELETE
- DELETE argument: groupid, string - the group from which to remove the user's subadmin rights

Status codes:

- 100 - successful
- 101 - user does not exist
- 102 - user is not a subadmin of the group / group does not exist
- 103 - unknown failure

Example

- DELETE `https://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/subadmins -d groupid="oldgroup"`
- Removes `Frank's` subadmin rights from the `oldgroup` group

XML Output

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

users / getsubadmingroups

Returns the groups in which the user is a subadmin. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/users/{userid}/subadmins

- HTTP method: GET

Status codes:

- 100 - successful
- 101 - user does not exist
- 102 - unknown failure

Example

- GET `https://admin:secret@example.com/ocs/v1.php/cloud/users/Frank/subadmins`
- Returns the groups of which `Frank` is a subadmin

XML Output

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

5.6.2 Instruction Set For Groups

groups / getgroups

Retrieves a list of groups from the ownCloud server. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/groups

- HTTP method: GET
- url arguments: search - string, optional search string
- url arguments: limit - int, optional limit value
- url arguments: offset - int, optional offset value

Status codes:

- 100 - successful

Example

- GET `http://admin:secret@example.com/ocs/v1.php/cloud/groups?search=adm`
- Returns list of groups matching the search string.

XML Output

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

groups / addgroup

Adds a new group. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/groups

- HTTP method: POST
- POST argument: groupid, string - the new groups name

Status codes:

- 100 - successful
- 101 - invalid input data
- 102 - group already exists

- 103 - failed to add the group

Example

- POST `http://admin:secret@example.com/ocs/v1.php/cloud/groups -d groupid="newgroup"`
- Adds a new group called `newgroup`

XML Output

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

groups / getgroup

Retrieves a list of group members. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/groups/{groupid}

- HTTP method: GET

Status codes:

- 100 - successful

Example

- POST `http://admin:secret@example.com/ocs/v1.php/cloud/groups/admin`
- Returns a list of users in the `admin` group

XML Output

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

groups / getsubadmins

Returns subadmins of the group. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/groups/{groupid}/subadmins

- HTTP method: GET

Status codes:

- 100 - successful
- 101 - group does not exist
- 102 - unknown failure

Example

- GET `https://admin:secret@example.com/ocs/v1.php/cloud/groups/mygroup/subadmins`
- Return the subadmins of the group: `mygroup`

XML Output

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

groups / deletegroup

Removes a group. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/groups/{groupid}

- HTTP method: DELETE

Status codes:

- 100 - successful
- 101 - group does not exist
- 102 - failed to delete group

Example

- DELETE `http://admin:secret@example.com/ocs/v1.php/cloud/groups/mygroup`
- Delete the group `mygroup`

XML Output

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

5.6.3 Instruction Set For Apps

apps / getapps

Returns a list of apps installed on the ownCloud server. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/apps/

- HTTP method: GET
- url argument: filter, string - optional (`enabled` or `disabled`)

Status codes:

- 100 - successful
- 101 - invalid input data

Example

- GET `http://admin:secret@example.com/ocs/v1.php/cloud/apps?filter=enabled`
- Gets enabled apps

XML Output

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

apps / getappinfo

Provides information on a specific application. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/apps/{appid}

- HTTP method: GET

Status codes:

- 100 - successful

Example

- GET `http://admin:secret@example.com/ocs/v1.php/cloud/apps/files`
- Get app info for the `files` app

XML Output

```
<?xml version="1.0"?>
<ocs>
  <meta>
    <statuscode>100</statuscode>
    <status>ok</status>
  </meta>
  <data>
    <info/>
    <remote>
      <files>appinfo/remote.php</files>
      <webdav>appinfo/remote.php</webdav>
      <filesync>appinfo/filesync.php</filesync>
    </remote>
    <public/>
    <id>files</id>
    <name>Files</name>
    <description>File Management</description>
    <licence>AGPL</licence>
    <author>Robin Appelman</author>
    <require>4.9</require>
    <shipped>true</shipped>
    <standalone></standalone>
    <default_enable></default_enable>
    <types>
      <element>filesystem</element>
    </types>
  </data>
</ocs>
```

apps / enable

Enable an app. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/apps/{appid}

- HTTP method: POST

Status codes:

- 100 - successful

Example

- POST `http://admin:secret@example.com/ocs/v1.php/cloud/apps/files_texteditor`
- Enable the `files_texteditor` app

XML Output

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

apps / disable

Disables the specified app. Authentication is done by sending a Basic HTTP Authorization header.

Syntax: ocs/v1.php/cloud/apps/{appid}

- HTTP method: DELETE

Status codes:

- 100 - successful

Example

- DELETE `http://admin:secret@example.com/ocs/v1.php/cloud/apps/files_texteditor`
- Disable the `files_texteditor` app

XML Output

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

CHAPTER

SIX

FILE SHARING AND MANAGEMENT

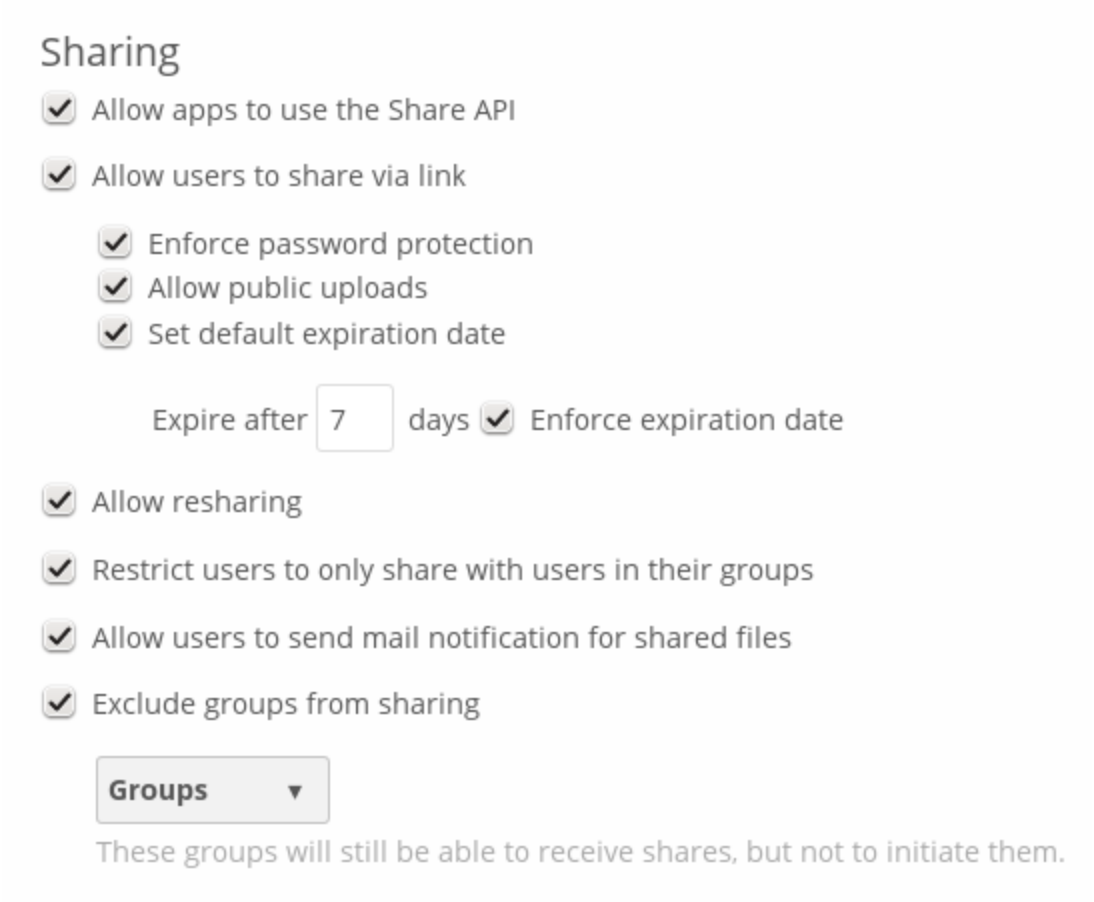
6.1 File Sharing

ownCloud users can share files with their ownCloud groups and other users on the same ownCloud server, and create public shares for people who are not ownCloud users. You have control of a number of user permissions on file shares:

- Allowing users to share files
- Allowing users to create public shares
- Requiring a password on public shares
- Allowing public uploads to public shares
- Requiring an expiration date on public share links
- Allowing resharing
- Restricting sharing to group members only
- Allowing email notifications of new public shares
- Excluding groups from creating shares

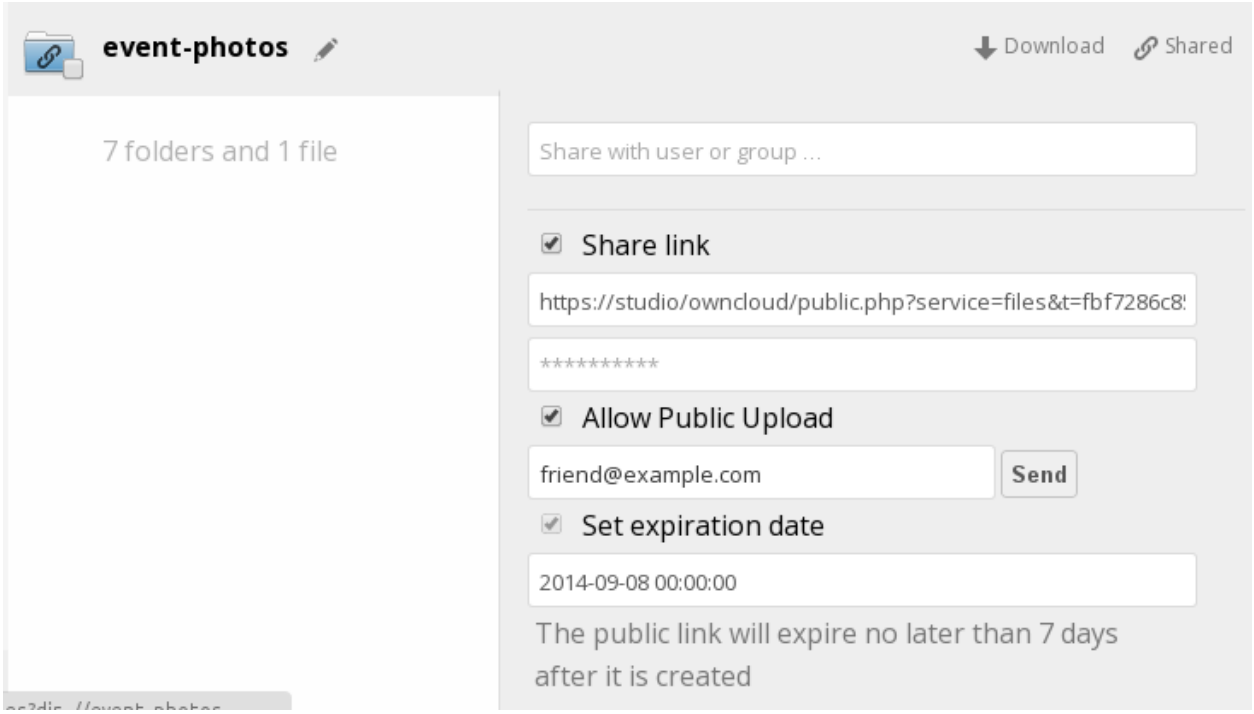
Configure your sharing policy on your Admin page in the Sharing section.

- Check `Allow apps to use the Share API` to enable users to share files. If this is not checked, no users can create file shares
- Check `Allow users to share via link` to enable creating public shares for people who are not ownCloud users. This creates a hyperlink, just like a Web page, so your ownCloud server needs to be accessible to whoever you are sharing with
- Check `Enforce password protection` to force users to set a password on all public share links. This does not affect local user and group shares
- Check `Allow public uploads` to allow outside users to upload files to public shares
- Checking `Set default expiration date` sets a default expiration date on public shares, and checking `Enforce expiration date` makes it a requirement
- Check `Allow resharing` to enable users to re-share files shared with them
- Check `Restrict users to only share with users in their groups` to confine sharing within group memberships
- Check `Allow users to send mail notification for shared files` so that users can check "notify by email" when they create new file shares. This sends an email notification to everyone the file is shared with (everyone who has entered an email address on their Personal page)

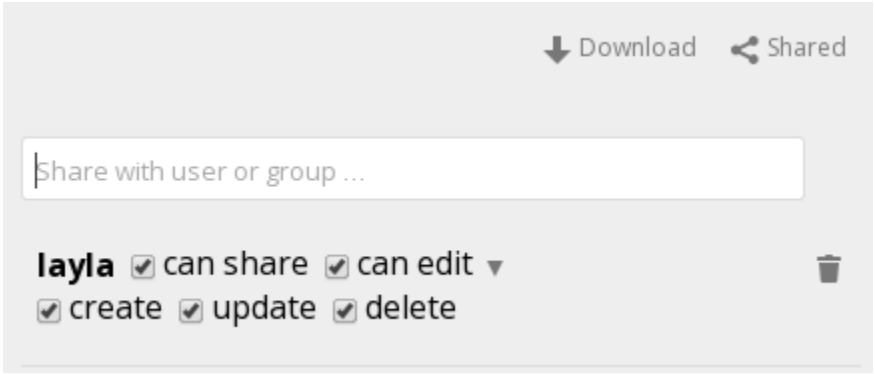

Sharing
Allow apps to use the Share API
Allow users to share via link
Enforce password protection
Allow public uploads
Set default expiration date
Expire after 7 days
Enforce expiration date
Allow resharing
Restrict users to only share with users in their groups
Allow users to send mail notification for shared files
Exclude groups from sharing
Groups
These groups will still be able to receive shares, but not to initiate them.

- Check `Exclude groups from sharing` to prevent members of specific groups from creating any file shares in those groups. When you check this, you'll get a dropdown list of all your groups to choose from. Members of excluded groups can still receive shares, but not create any

This is how it looks when a user creates a public share with passwords and expiration dates required:



This what a local share looks like. The user creating the share controls re-sharing, editing, updating, and deletion privileges:



Note: In older versions of ownCloud, you could set an expiration date on both local and public shares. Now you can set an expiration date only on public shares, and local shares do not expire when public shares expire. The only way to "expire" a local share is to click the trash can icon to un-share your files.

Note: ownCloud does not preserve the mtime (modification time) of directories, though it does update the mtimes on files. See Wrong folder date when syncing for discussion of this.

6.1.1 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– at a minumum `create`– 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 creator of the share regardless of who adds or edits them.

6.2 Uploading big files > 512MB

The default maximum file size for uploads is 512MB. You can increase this limit up to what your filesystem and operating system allows. There are certain hard limits that cannot be exceeded:

- < 2GB on 32Bit OS-architecture
- < 2GB on Windows (32Bit and 64Bit)
- < 2GB with Server Version 4.5 or older
- < 2GB with IE6 - IE8
- < 4GB with IE9 - IE11

64-bit filesystems have much higher limits; consult the documentation for your filesystem.

Note: The ownCloud sync client is not affected by these upload limits as it is uploading files in smaller chunks.

6.2.1 System Configuration

- Make sure that the latest version of PHP (at least 5.4.9) is installed
- Disable user quotas, which makes them unlimited
- Your temp file or partition has to be big enough to hold multiple parallel uploads from multiple users; e.g. if the max upload size is 10GB and the average number of users uploading at the same time is 100: temp space has to hold at least 10x100 GB

6.2.2 Configuring Your Webserver

Note: 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 .ini file:

```
php_value upload_max_filesize = 16G
php_value post_max_size = 16G
```

Adjust these values for your needs. If you see PHP timeouts in your logfiles, increase the timeout values, which are in seconds:

```
php_value max_input_time 3600
php_value max_execution_time 3600
```

The mod_reqtimeout Apache module could also stop large uploads from completing. If you're using this module and getting failed uploads of large files either disable it in your Apache config or raise the configured `RequestReadTimeout` timeouts.

There are also several other configuration options in your webserver config which could prevent the upload of larger files. Please see the manual of your webserver for how to configure those values correctly:

Apache

- LimitRequestBody
- SSLRenegBufferSize

Apache with mod_fcgid

- FcgidMaxRequestInMem
- FcgidMaxRequestLen

Note: If you are using Apache/2.4 with mod_fcgid, as of February/March 2016, `FcgidMaxRequestInMem` still needs to be significantly increased from its default value to avoid the occurence of segmentation faults when uploading big files. This is not a regular setting but serves as a workaround for Apache with mod_fcgid bug #51747.

Setting `FcgidMaxRequestInMem` significantly higher than normal may no longer be necessary, once bug #51747 is fixed.

---

nginx

- client_max_body_size
- fastcgi_read_timeout
- client_body_temp_path

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

Note: Make sure that `client_body_temp_path` points to a partition with adequate space for your upload file size, and on the same partition as the `upload_tmp_dir` or `tempdirectory` (see below). For optimal performance, place these on a separate hard drive that is dedicated to swap and temp storage.

6.2.3 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` pertaining to upload size, if you entered any.

To view your current PHP configuration and to see the location of your `php.ini` file, create a plain text file named `phpinfo.php` with just this single line of code in it: `<?php phpinfo(); ?>`. Place this file in your Web root, for example `/var/www/html`, and open it in your Web browser, for example `http://localhost/phpinfo.php`. This will display your complete current PHP configuration. Look for the **Loaded Configuration File** section to see which `php.ini` file your server is using. This is the one you want to edit.

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:

```
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:

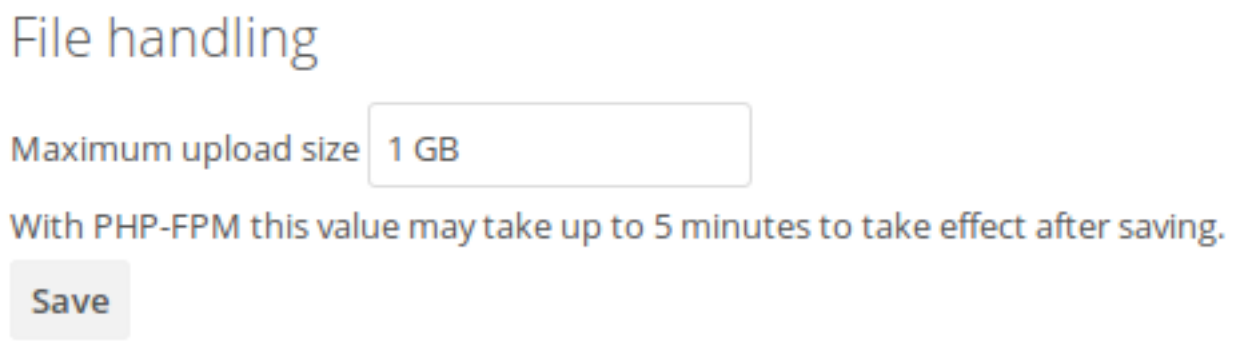
- `output_buffering = 0`

6.2.4 Configuring ownCloud

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

6.2.5 Configuring upload limits within the GUI

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



Depending on your environment this input box won't show up so you need to make sure that:

- you're running Apache with mod_php
- your web server is be able to use the `.htaccess` file shipped by ownCloud
- the user your web server is running as has write permissions to the file `.htaccess`

Setting Strong Directory Permissions might prevent write access to these files. As an admin you need to decide between the ability to use the input box and a more secure ownCloud installation where you need to manually modify the upload limits in the `.htaccess` file described above.

6.3 Configuring the Collaborative Documents App

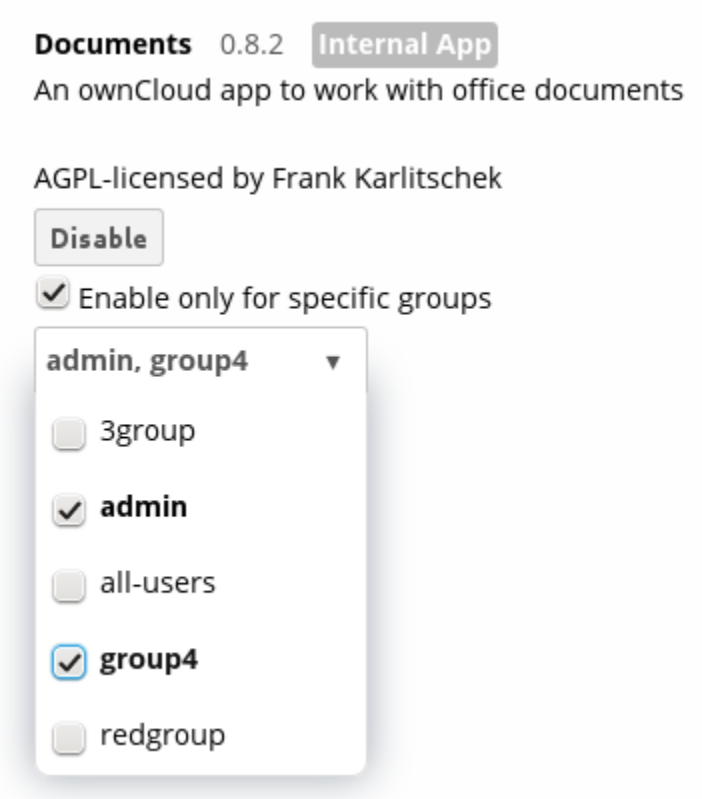
The Documents application supports editing documents within ownCloud, without the need to launch an external application. The Documents app supports these features:

- Cooperative edit, with multiple users editing files simultaneously.
- Document creation within ownCloud.
- Document upload.
- Share and edit files in the browser, and then share them inside ownCloud or through a public link.

Supported file formats are *.odt*, *.doc*, and *.docx*. *.odt* is supported natively in ownCloud, and you must have LibreOffice or OpenOffice installed on the ownCloud server to convert *.doc*, and *.docx* documents.

6.3.1 Enabling the Documents App

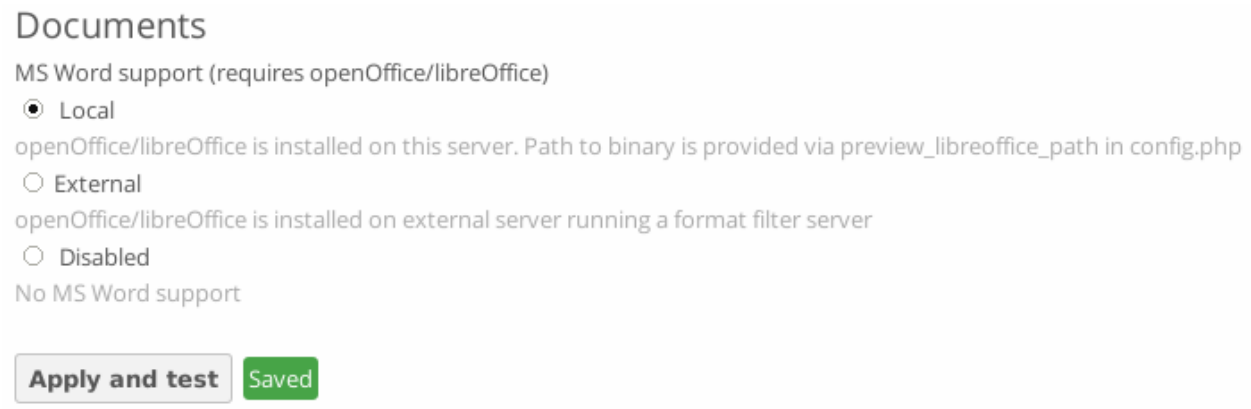
Go to your Apps page and click the `Enable` button. You also have the option to grant access to the Documents apps to selected user groups. By default it is available to all groups.



See "Collaborative Document Editing" in the User manual to learn how to create and share documents in the Documents application.

6.3.2 Enabling and testing MS Word support

Go to your admin settings menu. After choosing `Local` or `External` click on the `Apply and test` button. If you have a working LibreOffice or OpenOffice installation a green `Saved` icon should appear.



Troubleshooting

If the mentioned test fails please make sure that:

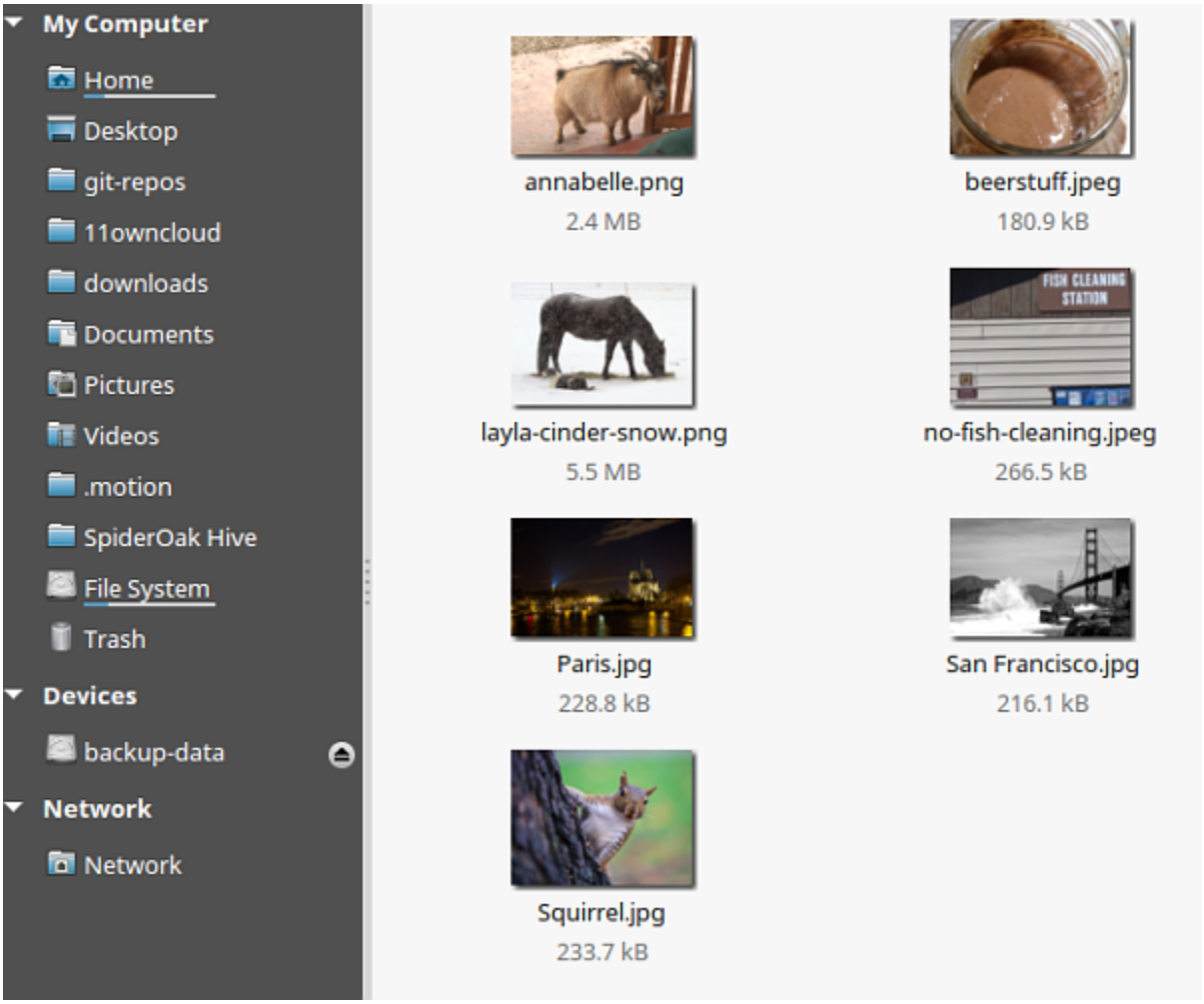
- the PHP functions `escapeshellarg` and `shell_exec` are not disabled in your PHP configuration
- the libreoffice/openoffice binary is within your PATH and is executeable for the HTTP user
- your SELinux configuration is not blocking the execution of the binary
- the PHP `open_basedir` is correctly configured to allow the access to the binary

More hints why the test is failing can be found in your `data/owncloud.log`.

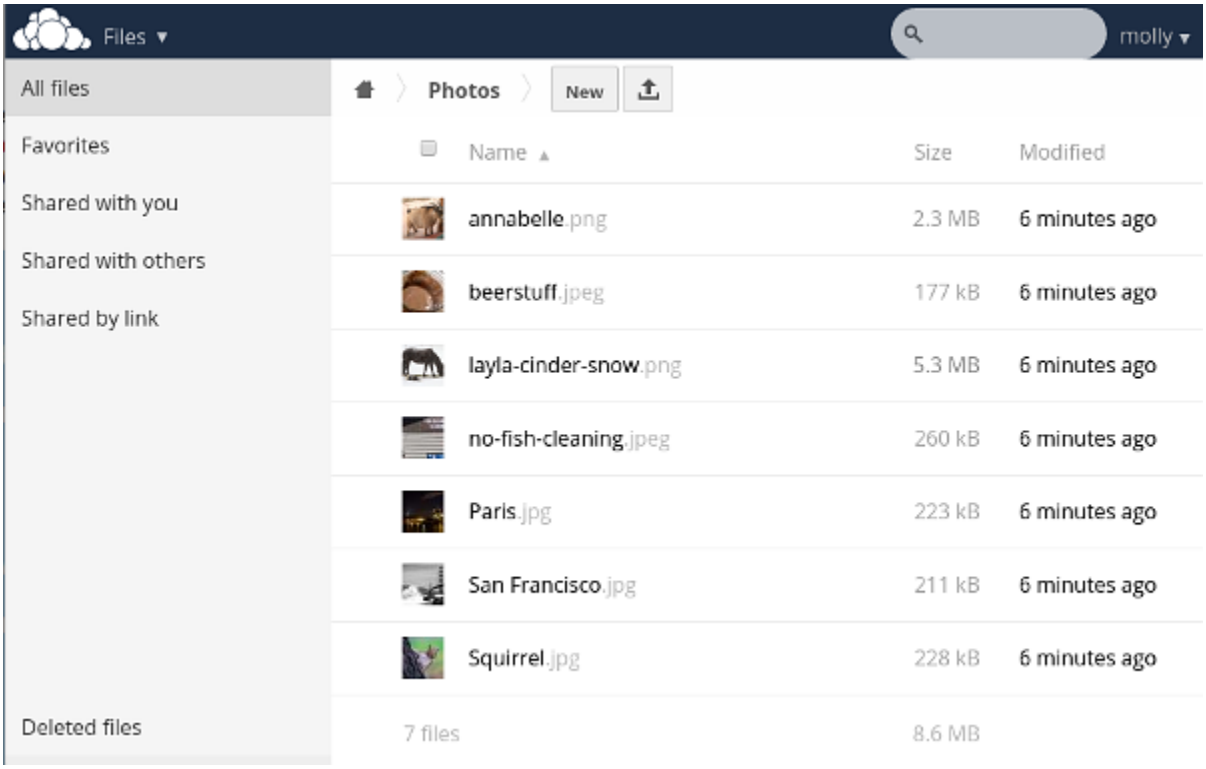
6.4 Providing Default Files

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.



6.4.1 Additional Configuration

The configuration option `skeletondirectory` available in your `config.php` (See *Config.php Parameters*) allows you to configure the directory where the skeleton files are located. These files will be copied to the data directory of new users. Leave empty to not copy any skeleton files.

6.5 Encryption Configuration

ownCloud includes a server-side encryption application. The Encryption app encrypts all files stored on the ownCloud server, and all files on remote storage that is connected to your ownCloud server. Encryption and decryption are performed on the ownCloud server. All files sent to remote storage (for example Dropbox and Google Drive) will be encrypted by the ownCloud server, and upon retrieval, decrypted before serving them to you and anyone you have shared them with.

Encrypting files increases their size by roughly 35%, so you must take this into account when you are provisioning storage and setting storage quotas. User's quotas are based on the unencrypted file size, and not the encrypted file size.

When files on external storage are encrypted in ownCloud, you cannot share them directly from the external storage services, but only through ownCloud sharing because the key to decrypt the data never leaves the ownCloud server.

The main purpose of the Encryption app is to protect users' files on remote storage, and to do it easily and seamlessly from within ownCloud.

The Encryption app generates a strong encryption key, which is unlocked by user's passwords. So your users don't need to track an extra password, but simply log in as they normally do.

Encryption is applied server-wide; it cannot be applied to selected users or files.

The Encryption app encrypts only the contents of files, and not filenames and folder structures.

You should regularly backup all encryption keys to prevent permanent data loss. The encryption keys are stored in the following folders:

data/owncloud_private_key Recovery key, if enabled, and public share key

data/public-keys Public keys for all users

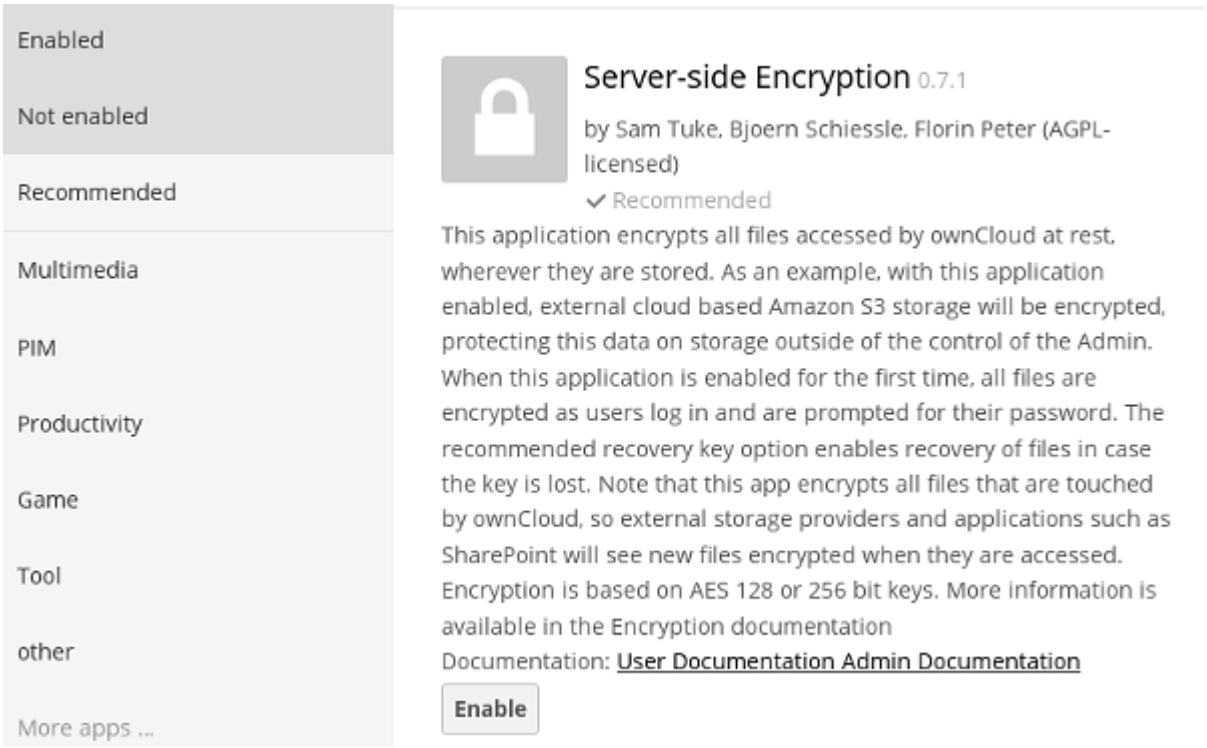
data/<user>/files_encryption Users' private keys and all other keys necessary to decrypt the users' files

data/files_encryption private keys and all other keys necessary to decrypt the files stored on a system wide external storage

Warning: Encryption keys are stored only on the ownCloud server, eliminating exposure of your data to third party storage providers. The encryption app does **not** protect your data if your ownCloud server is compromised, and it does not prevent ownCloud administrators from reading user's files. This would require client-side encryption, which this app does not provide. If your ownCloud server is not connected to any external storage services then it is better to use other encryption tools, such as file-level or whole-disk encryption. Read How ownCloud uses encryption to protect your data for more information.

6.5.1 Enabling the Encryption App

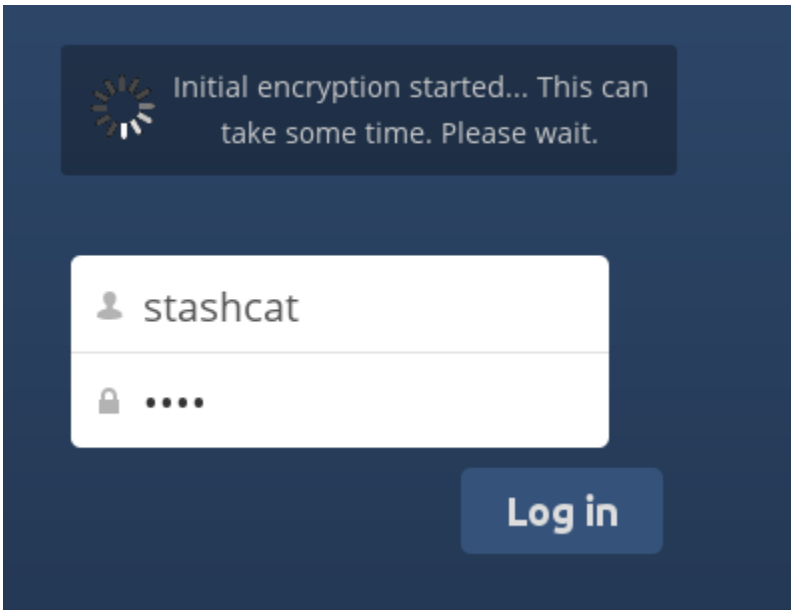
The Encryption app is bundled with ownCloud, so first go to your Apps page to enable it.



After you click the `Enable` button you must log out, and then log back in. If you continue to work without logging out, you'll see a yellow banner at the top of your Files page that warns you "Encryption App is enabled but your keys are not initialized, please log-out and log-in again."

When you log out and then log back in, your encryption keys are initialized and your files are encrypted. This is a one-time process, and it will take a few minutes depending on how many files you have.

Note: The more files you have, the longer the initial encryption will take. It is better to activate the encryption app after a new ownCloud installation, to avoid possible timeouts.

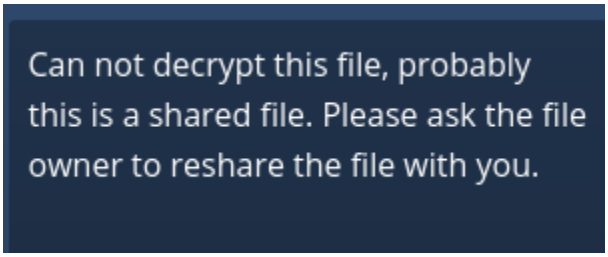


When the encryption process is complete you'll be returned to your default ownCloud page. Every user will go through this process when they log in after you enable encryption, and each user will get unique encryption keys. Users can change their passwords whenever they want on their Personal pages, and ownCloud will update their encryption keys automatically.

6.5.2 Sharing Encrypted Files

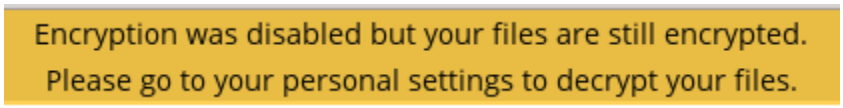
Only users who have private encryption keys have access to shared encrypted files and folders. Users who have not yet created their private encryption keys will not have access to encrypted shared files; they will see folders and filenames, but will not be able to open or download the files. They will see a yellow warning banner that says "Encryption App is enabled but your keys are not initialized, please log-out and log-in again."

Share owners may need to re-share files after encryption is enabled; users trying to access the share will see a message advising them to ask the share owner to re-share the file with them. For individual shares, un-share and re-share the file. For group shares, share with any individuals who can't access the share. This updates the encryption, and then the share owner can remove the individual shares.

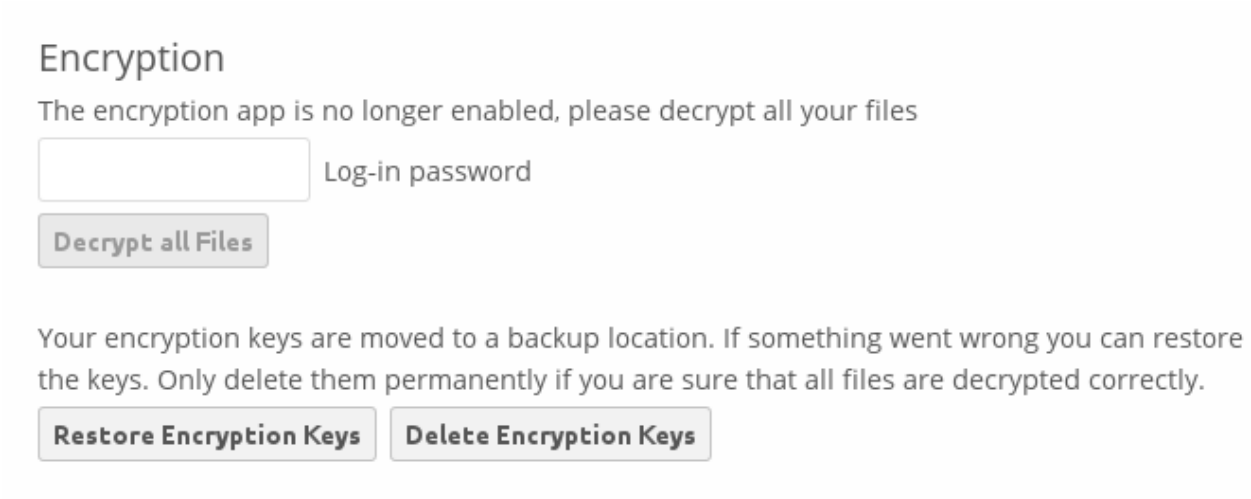


6.5.3 Decrypting Encrypted Files

You have the option of changing your mind and disabling the Encryption app. Just click its Disable button on the Apps page, and when you go to your Files page you'll see the yellow banner warning "Encryption was disabled but your files are still encrypted. Please go to your personal settings to decrypt your files".



Go to your Personal page and enter your password in the Encryption removal form, and your files will all be decrypted.



Your users will also have to follow this step to decrypt their files. If something goes wrong with decryption, click the `Restore Encryption Keys` button to re-encrypt your files, and then review your logfile to see what happened.

6.5.4 Enabling a File Recovery Key

If you lose your ownCloud password, then you lose access to your encrypted files. If one of your users loses their ownCloud password their files are unrecoverable. You cannot reset their password in the normal way; you'll see a yellow banner warning "Please provide an admin recovery password, otherwise all user data will be lost".

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

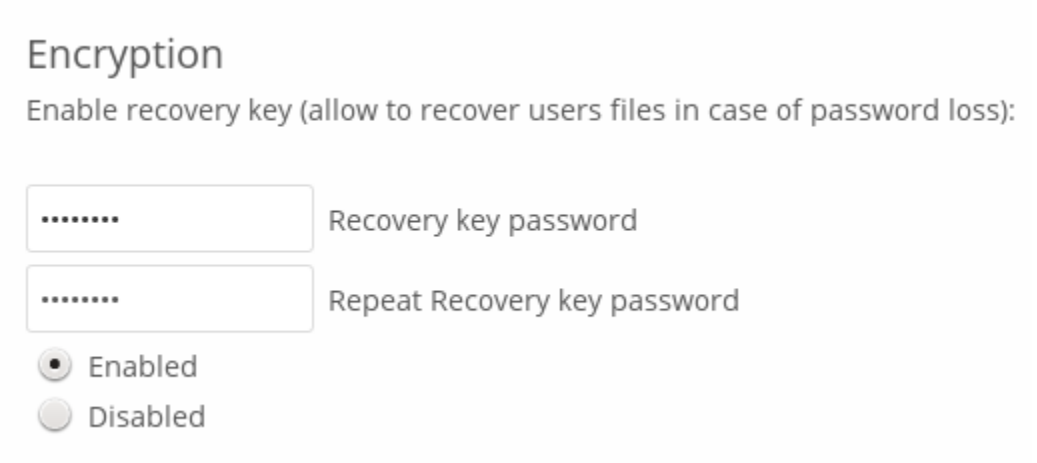
Then your users have the option of enabling password recovery on their Personal pages. If they do not do this, then the Recovery Key won't work for them.

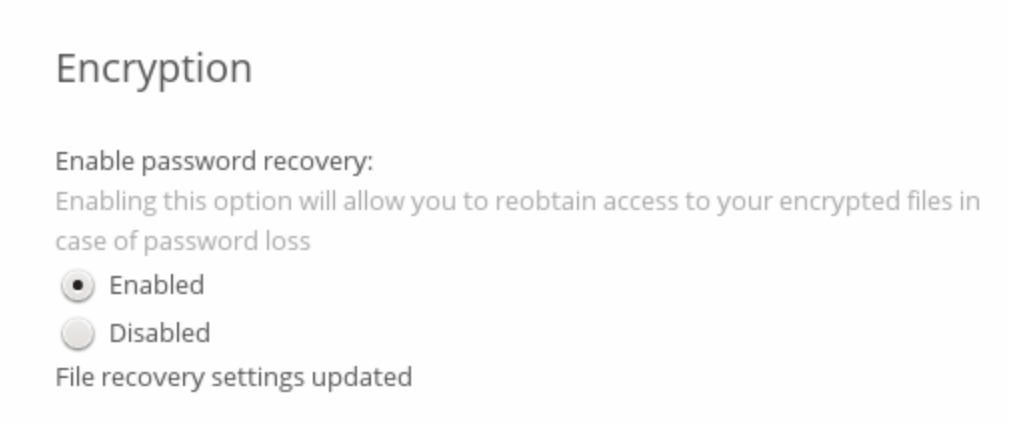
For users who have enabled password recovery, give them a new password and recover access to their encrypted files by supplying the Recovery Key on the Users page.

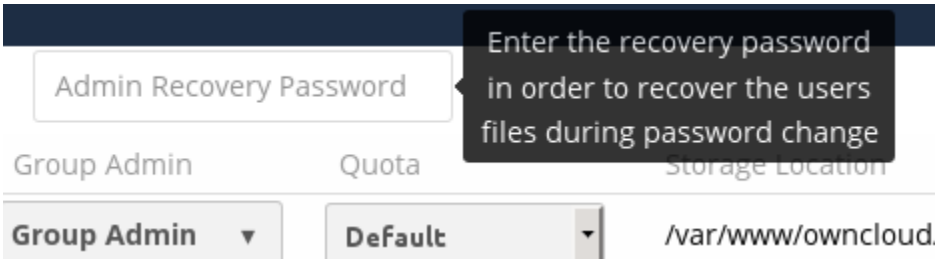
6.5.5 Files Not Encrypted

Only the data in your files is encrypted, and not the filenames or folder structures. These files are never encrypted:

- Old files in the trash bin.


Encryption
Enable recovery key (allow to recover users files in case of password loss):
Recovery key password
Repeat Recovery key password
Enabled
Disabled


Encryption
Enable password recovery:
Enabling this option will allow you to reobtain access to your encrypted files in case of password loss
Enabled
Disabled
File recovery settings updated


Admin Recovery Password
Enter the recovery password in order to recover the users files during password change
Group Admin
Quota
Storage Location
Group Admin
Default
/var/www/owncloud

- Image thumbnails from the Gallery app.
- Previews from the Files app.
- The search index from the full text search app.
- Third-party app data

There may be other files that are not encrypted; only files that are exposed to third-party storage providers are guaranteed to be encrypted.

6.5.6 LDAP and Other External User Back-ends

If you use an external user back-end, such as an LDAP or Samba server, and you change a user's password on the back-end, the user will be prompted to change their 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.

6.5.7 Encryption migration to ownCloud 8.0

When you upgrade from older versions of ownCloud to ownCloud 8.0, you must manually migrate your encryption keys with the *occ* command after the upgrade is complete, like this example for CentOS: *sudo -u apache php occ encryption:migrate-keys* You must run *occ* as your HTTP user. See *Using the occ Command* to learn more about *occ*.

6.6 Configuring External Storage (GUI)

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.

All of these connect to a LAN ownCloud server that is not publicly accessible, with one exception: Google Drive requires an ownCloud server with a registered domain name that is accessible over the Internet.

6.6.1 Supported mounts

ownCloud admins may mount these external storage services and devices:

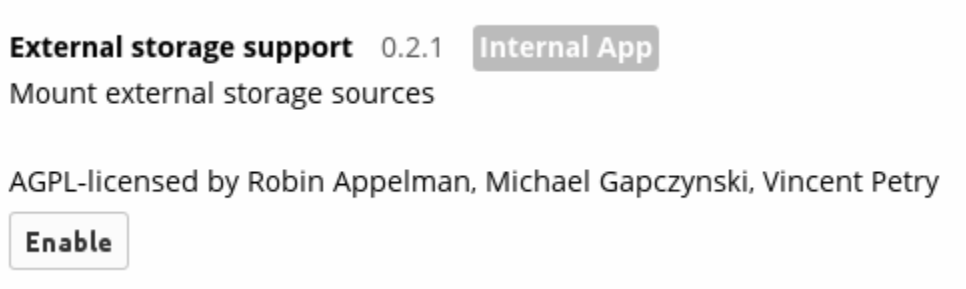
- Local
- Amazon S3 and S3 compliant
- Dropbox
- FTP/SFTP
- Google Drive
- OpenStack Object Storage
- SMB/CIFS
- SMB/CIFS using OC login
- ownCloud
- WebDAV

ownCloud users can be given permission to mount any of these, except local storage.

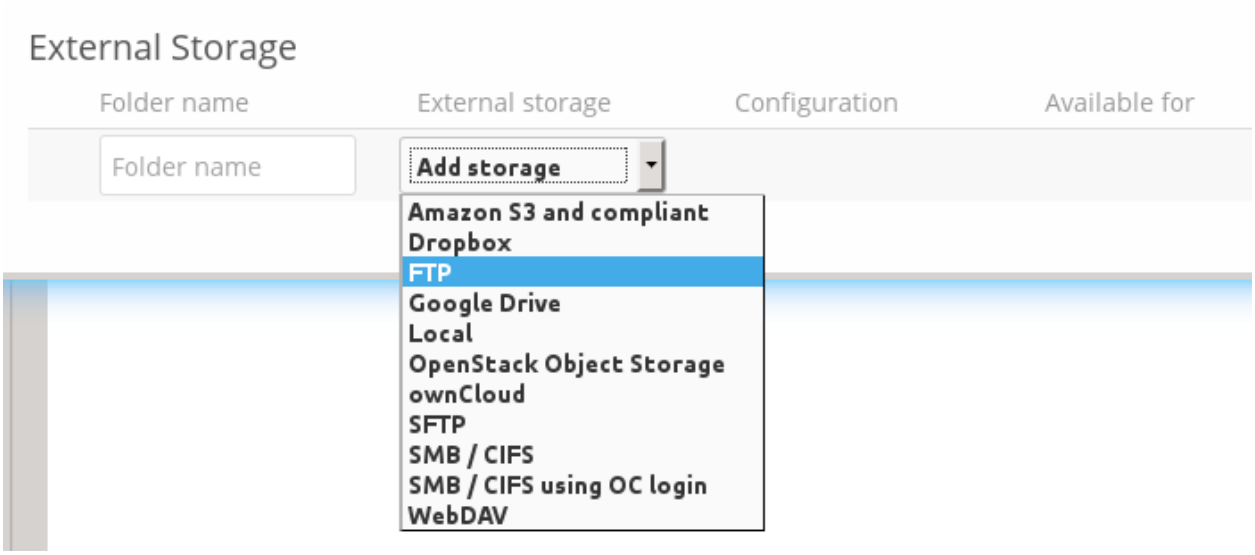
Note: A non-blocking or correctly configured SELinux setup is needed for these backends to work. Please refer to the *SELinux Configuration*.

6.6.2 Enabling External Storage Support

The `External storage support` application is enabled on the `Apps` page.



After enabling it, go to your `Admin` page to set up your external storage mounts.



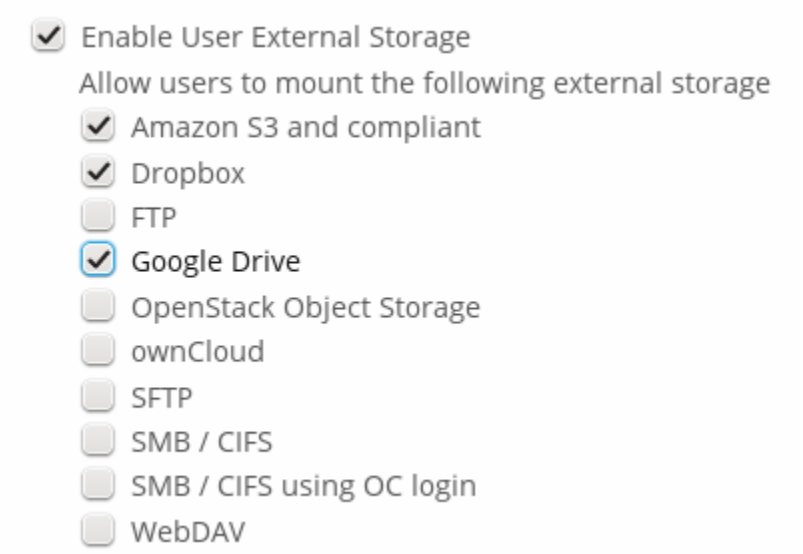
When your configuration is correct you'll see a green light at the left, and if it isn't you'll see a red light.

Check `Enable User External Storage` to allow your users to mount their own external storage services, and check the services you want to allow.

After creating your external storage mounts, you can share them and control permissions just like any other ownCloud share.

6.6.3 Using self-signed certificates

When using self-signed certificates for external storage mounts the certificate needs to be imported in the personal settings of the user. Please refer to this blogpost for more information.



6.6.4 Adding files to external storages

In general it is recommended to configure the background job `Webcron` or `Cron` as described in *Defining Background Jobs* so ownCloud is able to detect files added to your external storages without the need for a user to be browsing your ownCloud installation.

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

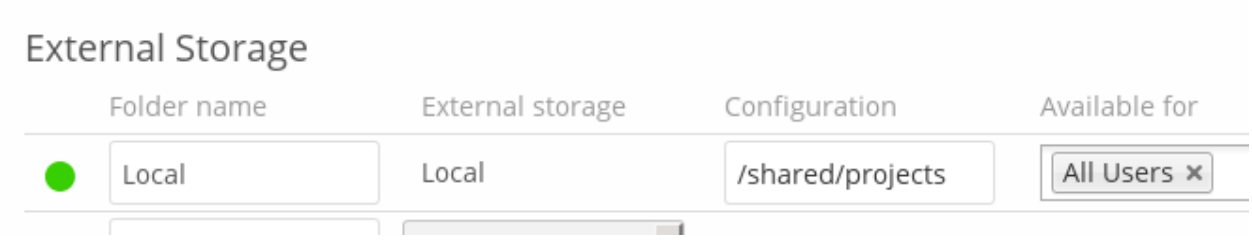
6.6.5 Local Storage

Use this to mount any directory on your ownCloud server that is outside of your ownCloud `data/` directory. This directory must be readable and writable by your HTTP server user.

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 filepath of the directory you want to mount.

In the `Available for` field enter the users or groups who have permission to access the mount.



6.6.6 Amazon S3

All you need to connect your Amazon S3 buckets to ownCloud is your S3 Access Key, Secret Key, and your bucket name.

In the `Folder name` field enter the folder name that you want to appear on your ownCloud `Files` page.

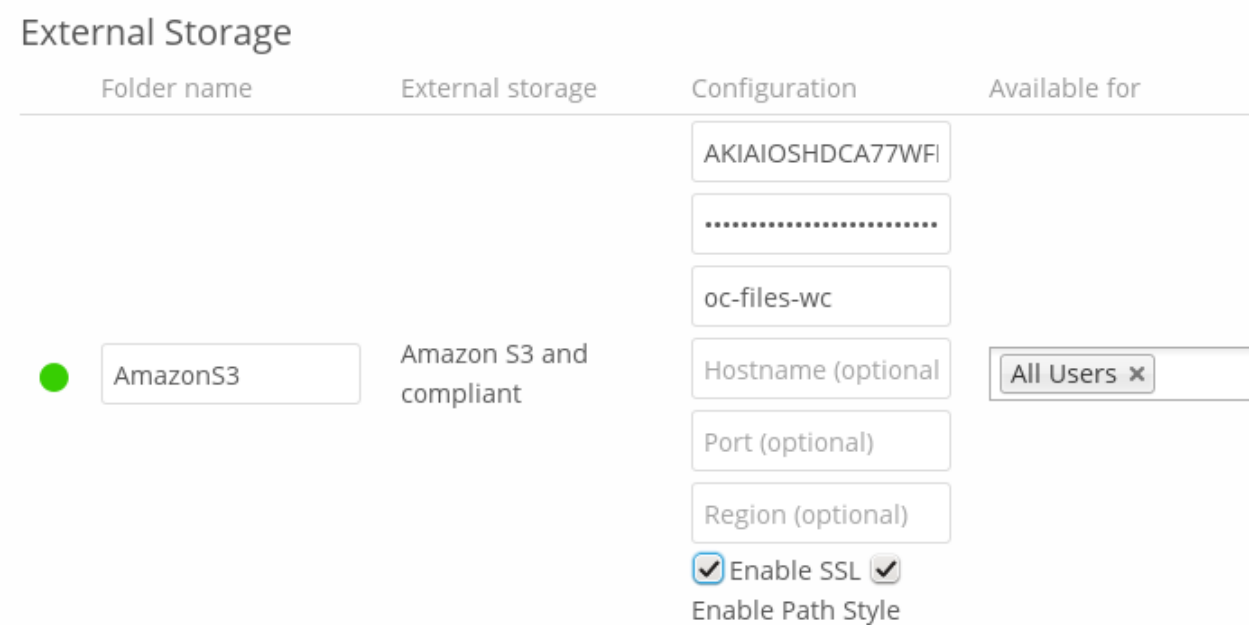
In the `Access Key` field enter your S3 Access Key.

In the `Secret Key` field enter your S3 Secret Key.

In the `Bucket` field enter the name of your S3 bucket you want to share.

In the `Available for` field enter the users or groups who have permission to access your S3 mount.

The hostname, port, and region of your S3 server are optional; you will need to use these for non-Amazon S3-compatible servers.



6.6.7 Dropbox

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

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

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

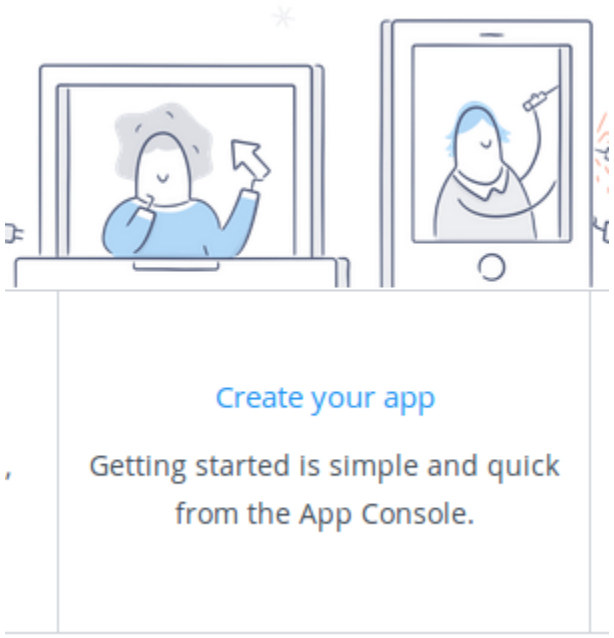
The next option is choosing which folders to share, or to share everything in your Dropbox.

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

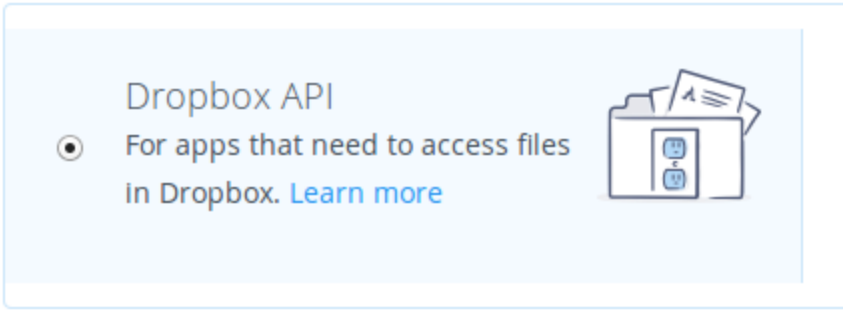
Then click the **Create App** button.

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

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

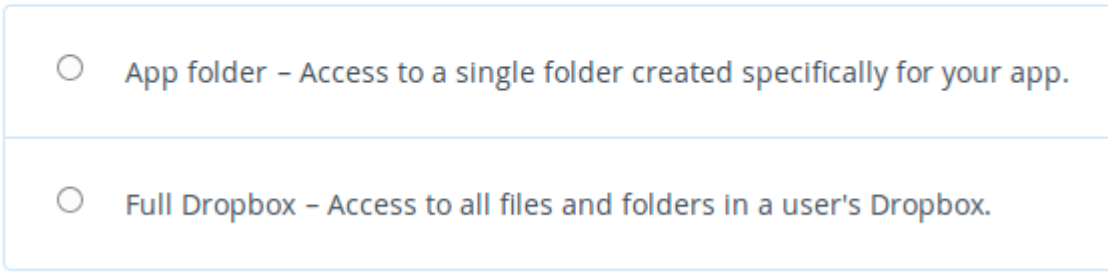


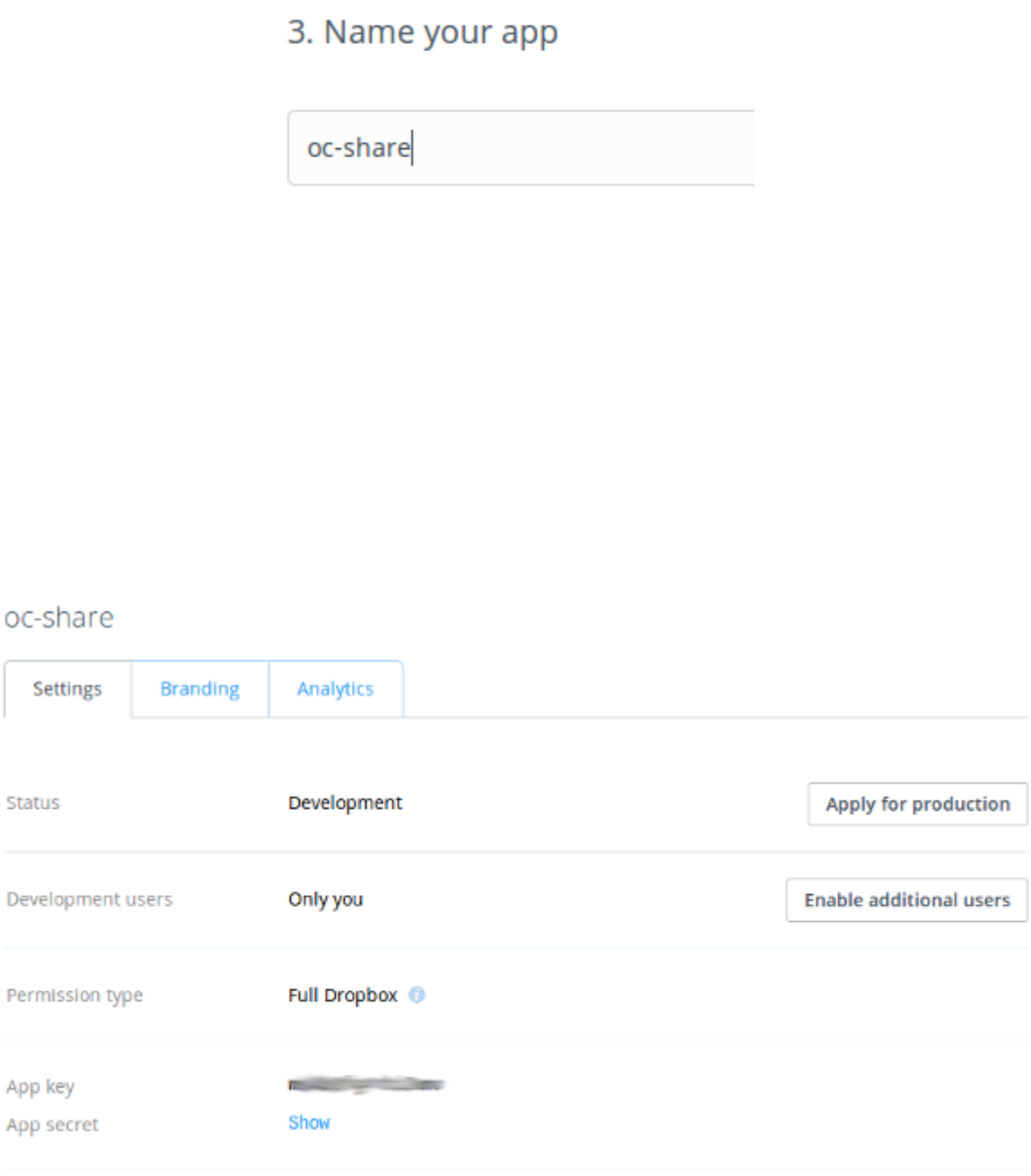
1. Choose an API



2. Choose the type of access you need

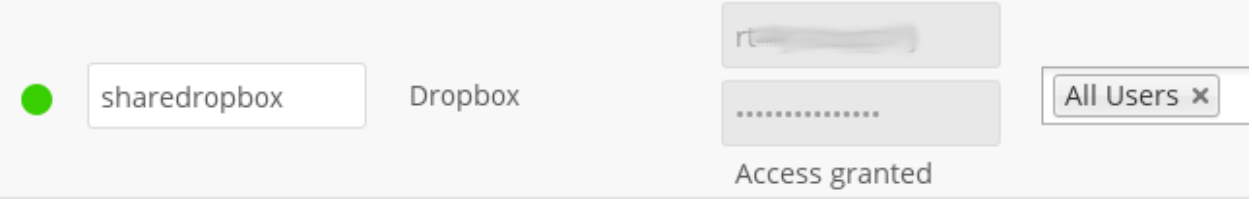
Learn more about access types



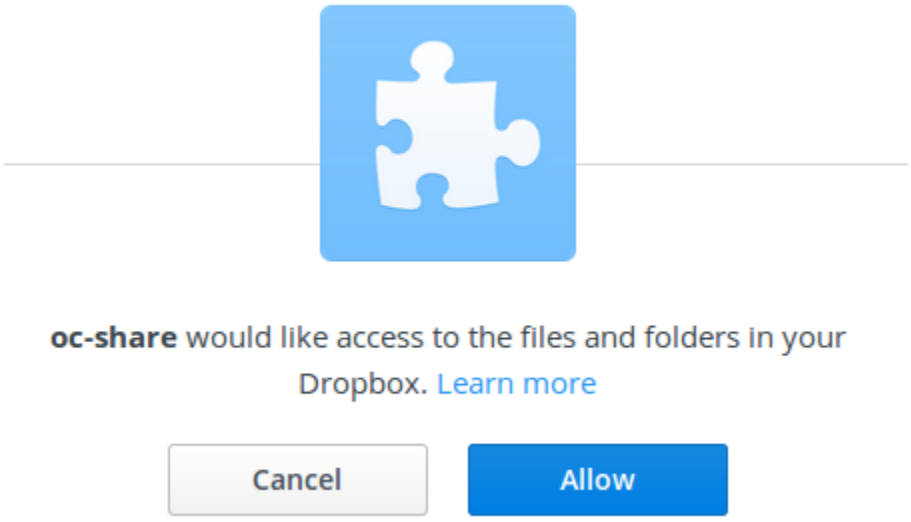

3. Name your app
oc-share
oc-share
Settings
Branding
Analytics
Status
Development
Apply for production
Development users
Only you
Enable additional users
Permission type
Full Dropbox
App key
App secret
Show

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

After entering your local mount name, enter **App Key** and **App Secret**.



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



6.6.8 FTP/FTPS/SFTP

Connecting to an FTP server requires:

- Whatever name you want for your local mountpoint.
- The URL of your FTP server, and optionally the port number.
- FTP server username and password.
- Remote Subfolder, the FTP directory to mount in ownCloud. ownCloud defaults to the root directory. When you specify a different directory you must leave off the leading slash. For example, if you want to connect your `public_html/images` directory, then type it exactly like that.
- Choose whether to connect in the clear with `ftp://`, or to encrypt your FTP session with SSL/TLS over `ftps://` (Your FTP server must be configured to support `ftps://`)
- Enter the ownCloud users or groups who are allowed to access the share.

Note: The external storage `FTP/FTPS/SFTP` 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`.

SFTP uses SSH rather than SSL, as FTPS does, so your SFTP sessions are always safely tucked inside an SSH tunnel. To connect an SFTP server you need:

- Whatever name you want for your local mountpoint.
- The URL of your SFTP server.
- SFTP server username and password.
- Remote Subfolder, the SFTP directory to mount in ownCloud.
- The ownCloud users or groups who are allowed to access the share.

6.7 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 **Create Project** button.

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

You'll be returned to your dashboard.

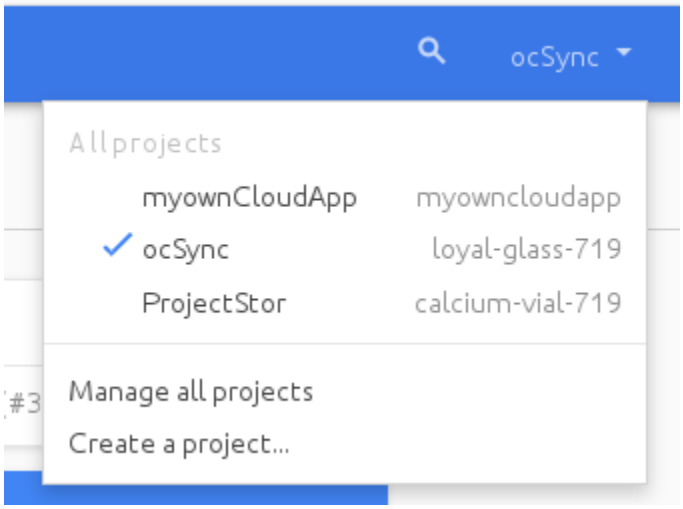
Google helpfully highlights your next step in blue, the **Use Google APIs** box. Make sure that your new project is selected, click on **Use Google APIs** , and it takes you to Google's APIs screen. There are many Google APIs; look for the **Google Apps APIs** and click **Drive API.**

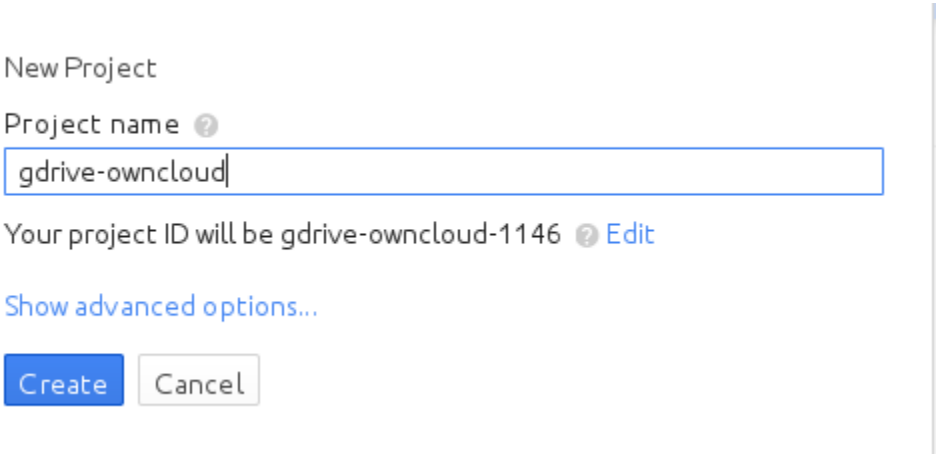
Drive API takes you to the API Manager overview. Click the blue **Enable API** button.

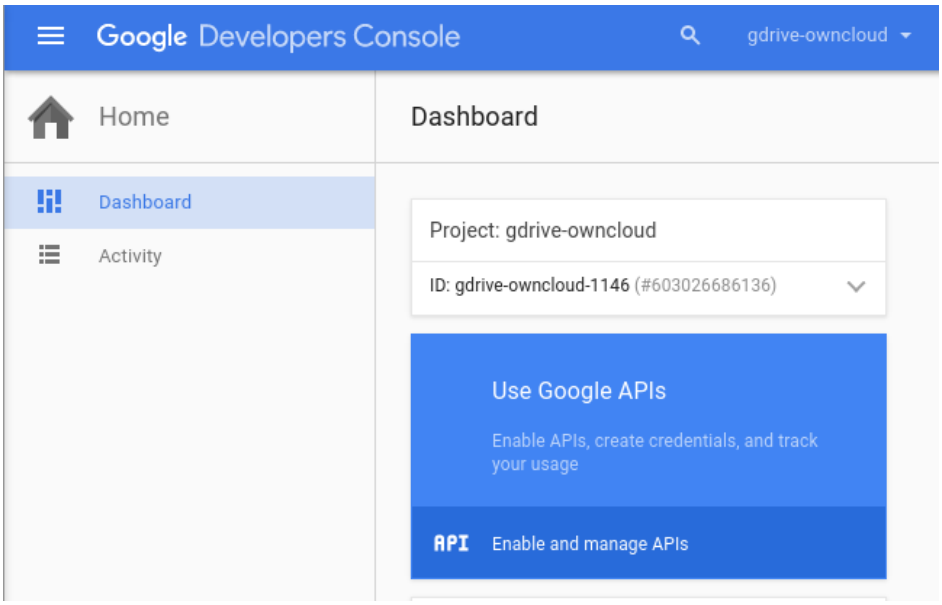
Now you must create your credentials, so click on **Go to credentials**.

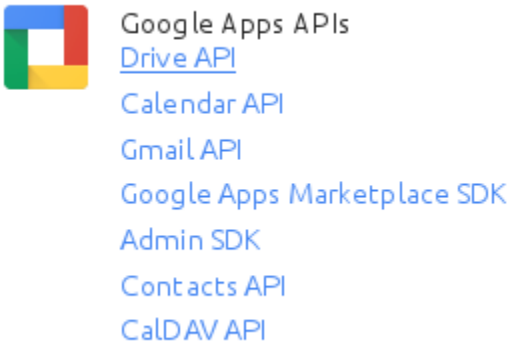
For some reason Google warns us again that we need to create credentials. We will use 0Auth 2.0.

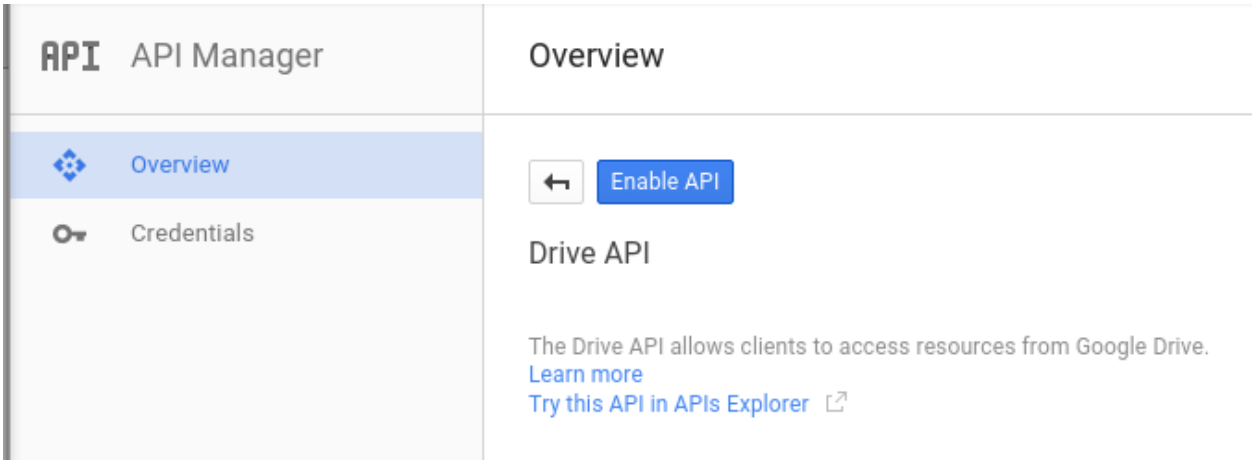
Now we have to create a consent screen. This is the information in the screen Google shows you when you connect your new Google app to ownCloud the first time. Click **Configure consent screen**. Then fill in the required form fields. Your logo must be hosted, as you cannot upload it, so enter its URL.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

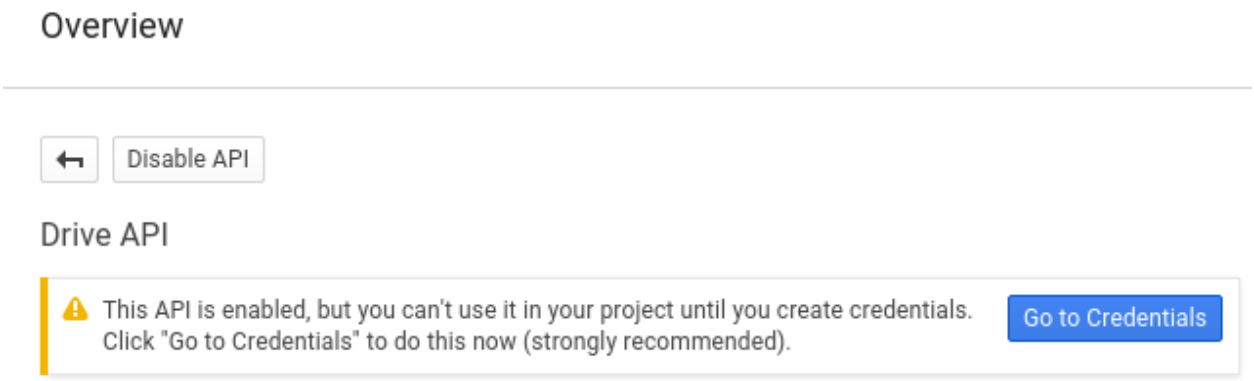

ocSync
All projects
myownCloudApp
myowncloudapp
ocSync
loyal-glass-719
ProjectStor
calcium-vial-719
Manage all projects
Create a project...

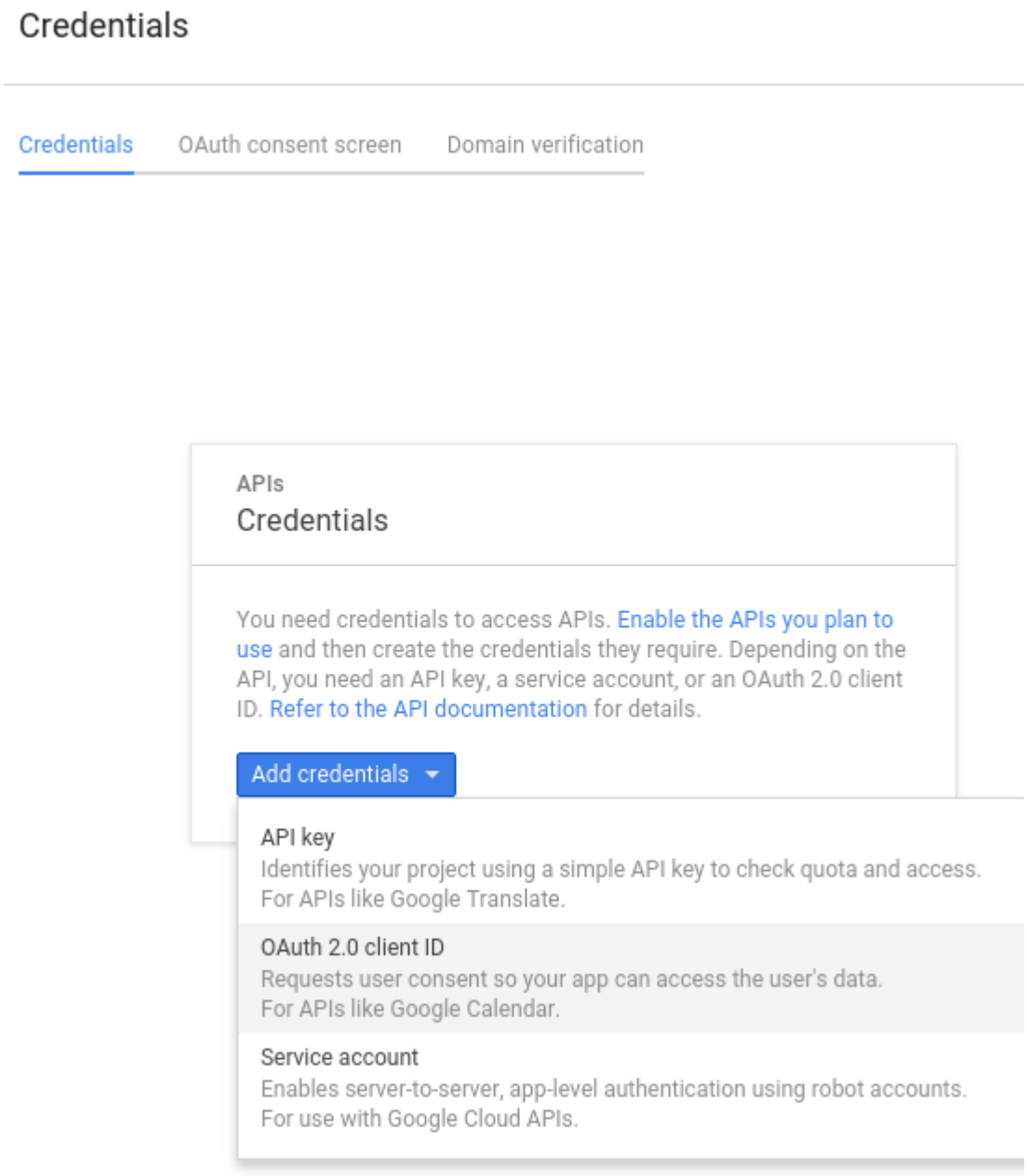

New Project
Project name
gdrive-owncloud
Your project ID will be gdrive-owncloud-1146 Edit
Show advanced options...
Create
Cancel

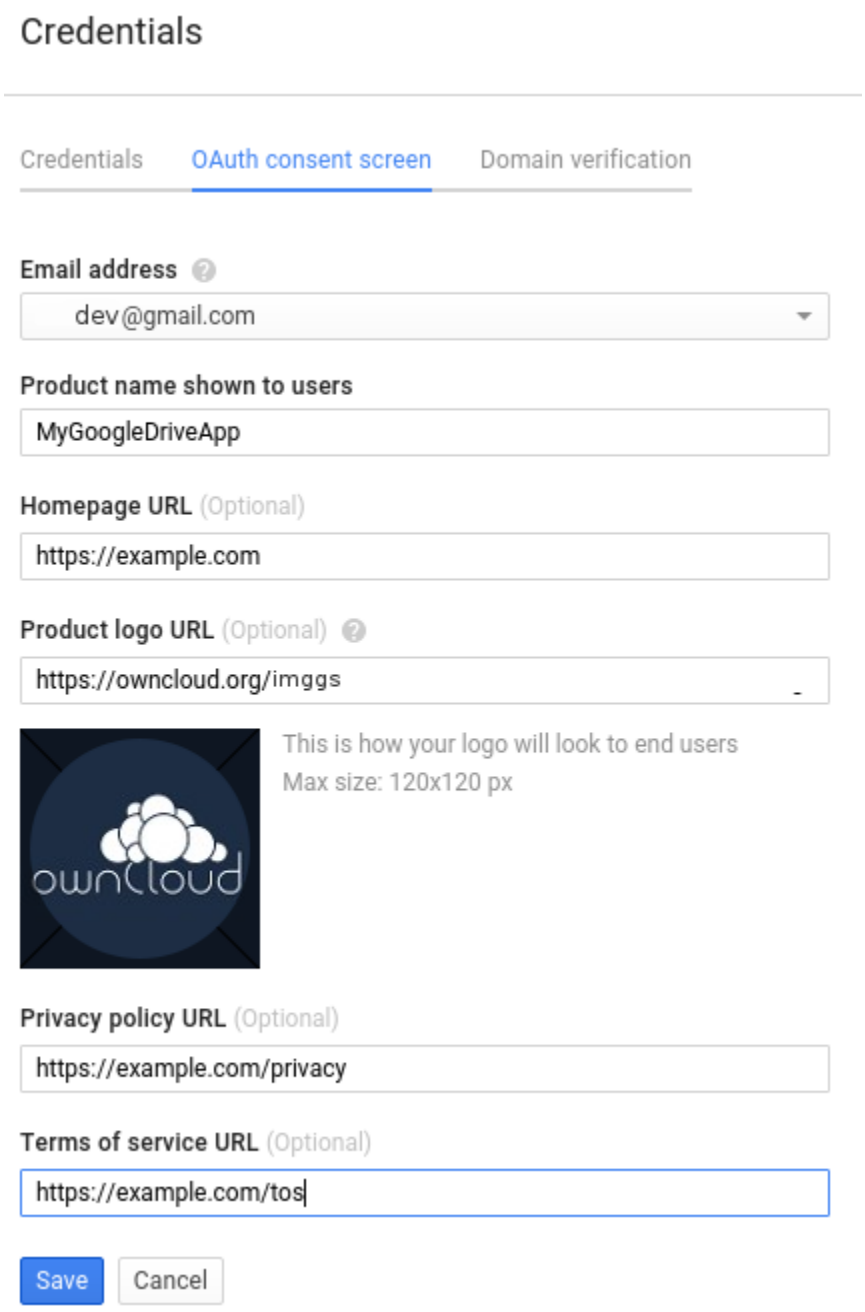

Google Developers Console
gdrive-owncloud
Home
Dashboard
Dashboard
Activity
Project: gdrive-owncloud
ID: gdrive-owncloud-1146 (#603026686136)
Use Google APIs
Enable APIs, create credentials, and track your usage
API
Enable and manage APIs


Google Apps APIs
Drive API
Calendar API
Gmail API
Google Apps Marketplace SDK
Admin SDK
Contacts API
CalDAV API


API Manager
Overview
Credentials
Overview
Enable API
Drive API
The Drive API allows clients to access resources from Google Drive.
Learn more
Try this API in APIs Explorer


Overview
Disable API
Drive API
This API is enabled, but you can't use it in your project until you create credentials.
Click "Go to Credentials" to do this now (strongly recommended).
Go to Credentials


Credentials
Credentials
OAuth consent screen
Domain verification
APIs
Credentials
You need credentials to access APIs. Enable the APIs you plan to use and then create the credentials they require. Depending on the API, you need an API key, a service account, or an OAuth 2.0 client ID. Refer to the API documentation for details.
Add credentials
API key
Identifies your project using a simple API key to check quota and access.
For APIs like Google Translate.
OAuth 2.0 client ID
Requests user consent so your app can access the user's data.
For APIs like Google Calendar.
Service account
Enables server-to-server, app-level authentication using robot accounts.
For use with Google Cloud APIs.


Credentials
Credentials
OAuth consent screen
Domain verification
Email address
dev@gmail.com
Product name shown to users
MyGoogleDriveApp
Homepage URL (Optional)
https://example.com
Product logo URL (Optional)
https://owncloud.org/imggs
This is how your logo will look to end users
Max size: 120x120 px
Privacy policy URL (Optional)
https://example.com/privacy
Terms of service URL (Optional)
https://example.com/tos
Save
Cancel

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

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

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

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

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

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

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

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

6.7.1 SMB/CIFS

You can mount SMB/CIFS file shares on ownCloud servers that run on Linux. This only works on Linux ownCloud servers because you must have `smbclient` installed. SMB/CIFS file servers include any Windows file share, Samba servers on Linux and other Unix-type operating systems, and NAS appliances.

You need the following information:

- Folder name – Whatever name you want for your local mountpoint.
- Host – The URL of the Samba server.
- Username – The username or domain/username used to login to the Samba server.
- Password – The password to login to the Samba server.
- Share – The share on the Samba server to mount.
- Remote Subfolder – The remote subfolder inside the Samba share to mount (optional, defaults to `/`). To assign the ownCloud logon username automatically to the subfolder, use `$user` instead of a particular subfolder name.

And finally, the ownCloud users and groups who get access to the share.

6.7.2 SMB/CIFS using OC login

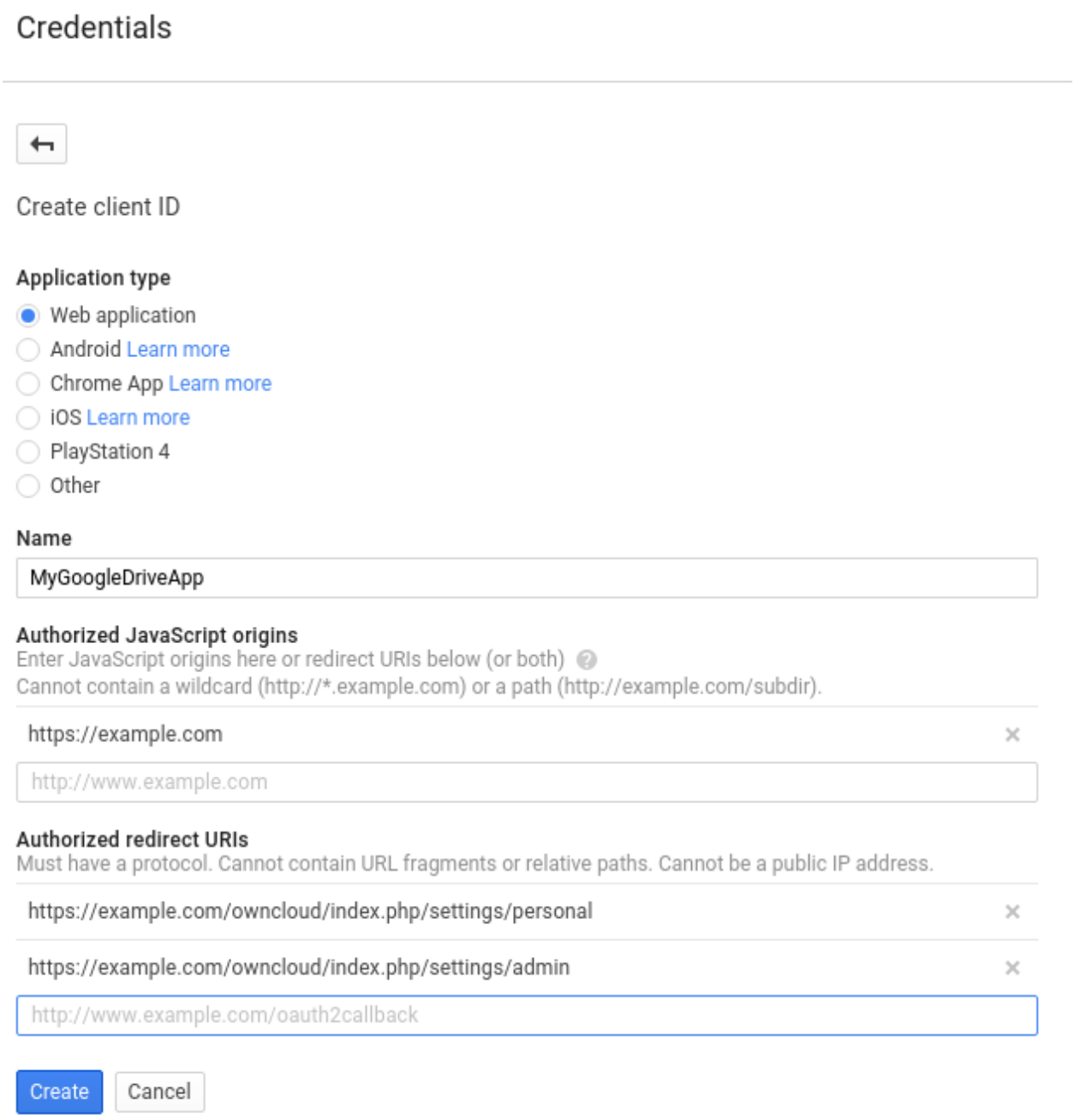
This works the same way as setting up a SMB/CIFS mount, except you can use your ownCloud logins intead of the SMB/CIFS server logins. To make this work, your ownCloud users need the same login and password as on the SMB/CIFS server.

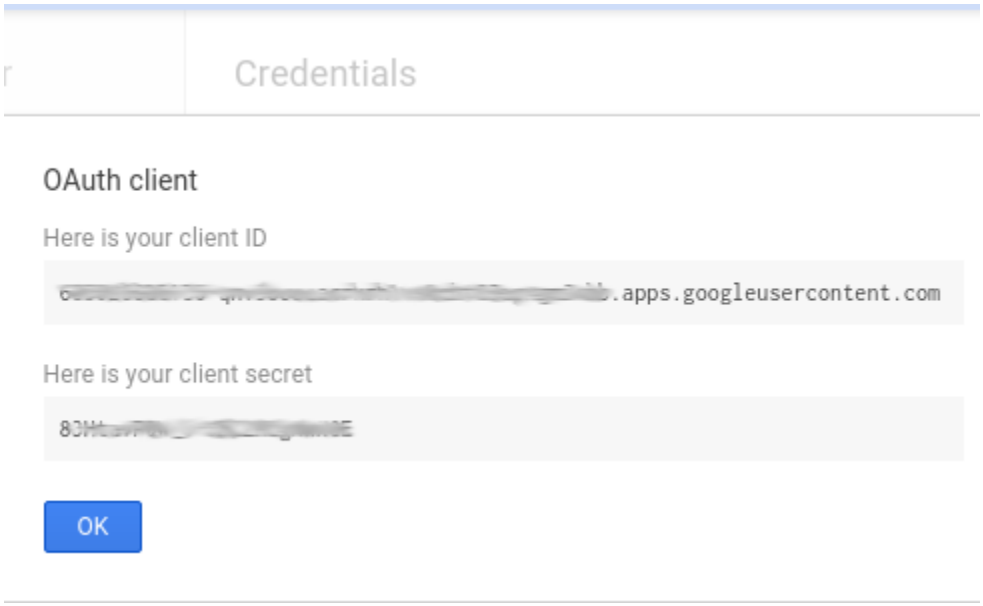
Note: Shares set up with `SMB/CIFS using OC login` cannot be shared in ownCloud. If you need to share your SMB/CIFS mount, then use the SMB/CIFS mount without oC login.

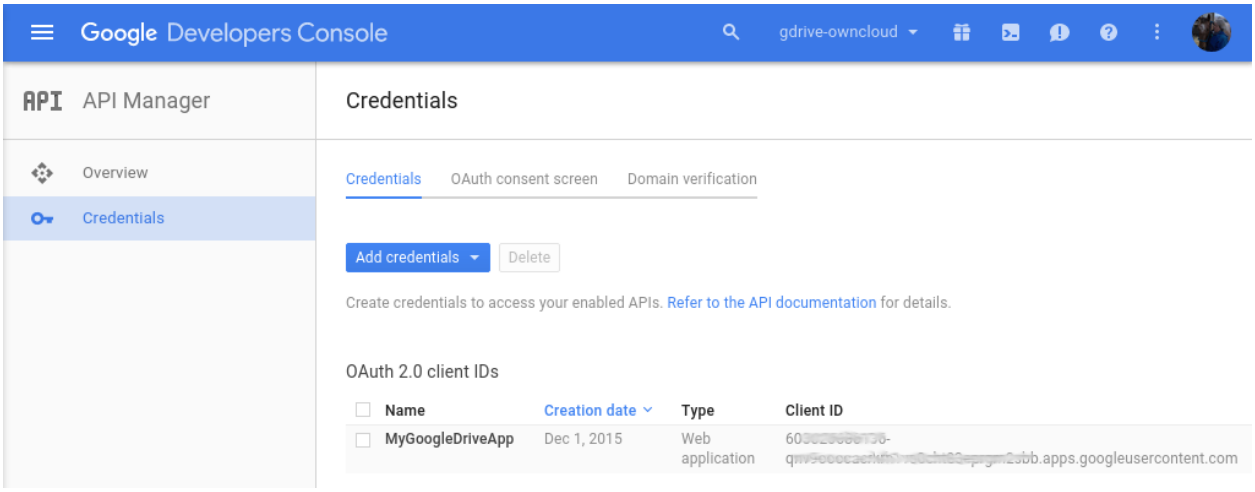
6.7.3 ownCloud and WebDAV

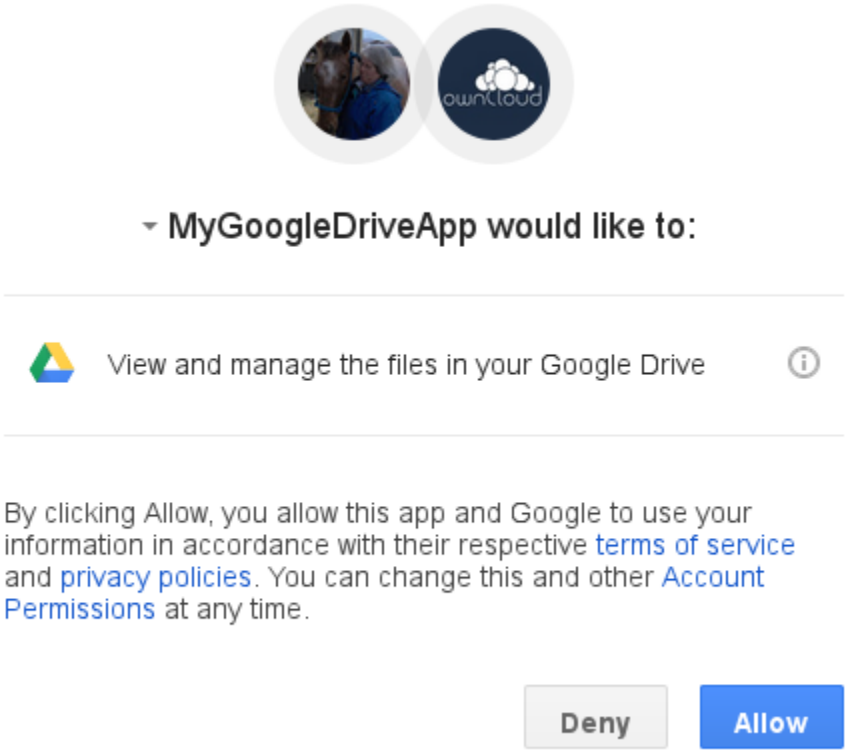
Use these to mount a directory from any WebDAV server, or another ownCloud server.

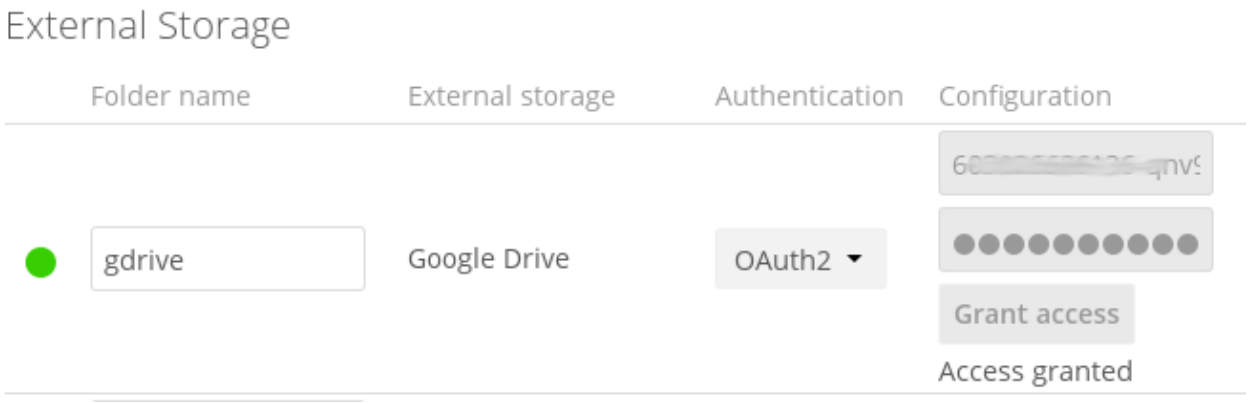
- Folder name – Whatever name you want for your local mountpoint.

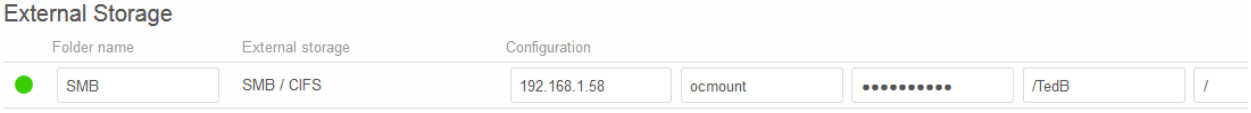

Credentials
Create client ID
Application type
Web application
Android Learn more
Chrome App Learn more
iOS Learn more
PlayStation 4
Other
Name
MyGoogleDriveApp
Authorized JavaScript origins
Enter JavaScript origins here or redirect URIs below (or both)
Cannot contain a wildcard (http://*.example.com) or a path (http://example.com/subdir).
https://example.com
http://www.example.com
Authorized redirect URIs
Must have a protocol. Cannot contain URL fragments or relative paths. Cannot be a public IP address.
https://example.com/owncloud/index.php/settings/personal
https://example.com/owncloud/index.php/settings/admin
http://www.example.com/oauth2callback
Create
Cancel


Credentials
OAuth client
Here is your client ID
.apps.googleusercontent.com
Here is your client secret
OK

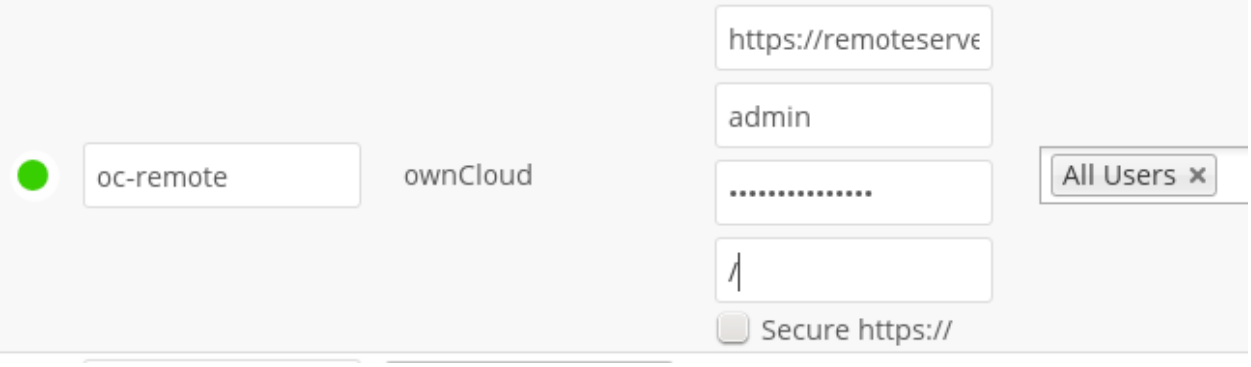

Google Developers Console
gdrive-owncloud
API Manager
Overview
Credentials
Credentials
Credentials
OAuth consent screen
Domain verification
Add credentials
Delete
Create credentials to access your enabled APIs. Refer to the API documentation for details.
OAuth 2.0 client IDs
Name
Creation date
Type
Client ID
MyGoogleDriveApp
Dec 1, 2015
Web application


MyGoogleDriveApp would like to:
View and manage the files in your Google Drive
By clicking Allow, you allow this app and Google to use your information in accordance with their respective terms of service and privacy policies. You can change this and other Account Permissions at any time.
Deny
Allow


External Storage
Folder name
External storage
Authentication
Configuration
gdrive
Google Drive
OAuth2
Grant access
Access granted


External Storage
Folder name
External storage
Configuration
SMB
SMB / CIFS
192.168.1.58
ocmount
/TedB
/

- URL – The URL of the WebDAV or ownCloud server.
- Username and password for the remote server
- Remote Subfolder – The remote subfolder you want to mount (optional, defaults to `/`)
- Secure `https://` - Whether to use `https://` to connect to the WebDav server instead of `http://` (We always recommend `https://` for security)



Note: CPanel users should install Web Disk to enable WebDAV functionality.

6.7.4 OpenStack Object Storage

Use this to mount a container on an OpenStack Object Storage server. You need the following information:

- Username
- Bucket
- Region
- API Key
- Tenantname
- Password
- Service Name
- URL of identity Endpoint
- Timeout of HTTP request

6.7.5 Configuration File

The configuration of mounts created within the External Storage App are stored in the `data/mount.json` file. This file contains all settings in JSON (JavaScript Object Notation) format. Two different types of entries exist:

- Group mounts: Each entry configures a mount for each user in group.
- User mount: Each entry configures a mount for a single user or all users.

For each type, there is a JSON array with the user/group name as key and an array of configuration values as the value. Each entry consist of the class name of the storage backend and an array of backend specific options (described above) and will be replaced by the user login.

Although configuration may be done by making modifications to the `mount.json` file, it is recommended to use the Web-GUI in the administrator panel (as described in the above section) to add, remove, or modify mount options to prevent any problems. See *Configuring External Storage (Configuration File)* for configuration examples.

6.8 Configuring External Storage (Configuration File)

You may configure external storage mounts by creating and editing the `data/mount.json`. This file contains all settings in JSON (JavaScript Object Notation) format. At the moment two different types of entries exist:

- **Group mounts:** each entry configures a mount for each user in group.
- **User mounts:** each entry configures a mount for a single user or for all users.

For each type, there is a JSON array with the user/group name as key, and an array of configuration entries as value. Each entry consists of the class name of the storage backend and an array of backend specific options and will be replaced by the user login. The template **$user** can be used in the mount point or backend options. As of writing the following storage backends are available for use:

- Local file system
- FTP (or FTPS)
- SFTP
- SMB
- WebDAV
- Amazon S3
- Dropbox
- Google Drive
- OpenStack Swift

You need to enable the *External storage support* app first. You can do this on the Apps page of your ownCloud Web GUI, or use the `occ` command. This example shows how to list apps, and how to enable the *External storage support* app on Debian Linux and its derivatives:

```
$ sudo -u www-data php occ app:list
Enabled:
 [snip]
Disabled:
- files_external
 [snip]

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

See *Using the occ Command* to learn more about `occ`.

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

Please keep in mind that some formatting has been applied and carriage returns have been added for better readability. In the `data/mount.json` all values need to be concatenated and written in a row without these modifications!

It is recommended to use the *Web-GUI* in the administrator panel to add, remove or modify mount options to prevent any problems!

6.8.1 Using self-signed certificates

When using self-signed certificates for external storage mounts the certificate needs to be imported in the personal settings of the user. Please refer to this blogpost for more information.

6.8.2 Adding files to external storages

In general it is recommended to configure the background job `Webcron` or `Cron` as described in *Defining Background Jobs* so ownCloud is able to detect files added to your external storages without the need for a user to be browsing your ownCloud installation.

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

6.8.3 Example

```
{"group":{
    "admin":{
        "\/$user\/files\/Admin_Stuff":{
            "class":"\\OC\\Files\\Storage\\Local",
            "options":{ ... },
            "priority":150
            }
        }
    }
 "user":{
    "all":{
        "\/$user\/files\/Pictures":{
            "class":"\\OC\\Files\\Storage\\DAV",
            "options":{ ... },
            "priority":100
            }
        }
    "someuser":{
        "\/someuser\/files\/Music":{
            "class":"\\OC\\Files\\Storage\\FTP",
            "options":{ ... },
            "priority":100
            }
        }
    }
}
```

6.8.4 Priorities

An advanced feature is available, only configurable directly in `data/mount.json`, which allows mount configurations to have an associated priority. When two or more valid mount configurations exist for the same mount point, the one with the highest priority (defined by the largest number) will take precedence and become the active mount for the user.

Each backend has a default priority, assigned when a mount configuration with that backend is created. The default priority will be shown in the example section for each backend below. Should a backend not provide a default priority, a value of 100 will be used.

There is also a concept of priority types, to preserve compatibility with previous mount configuration parsing. Mount configurations are evaluated in the following order, with later mount types always overriding a previous mount type:

- user -> all : global mount configurations
- group : group mount configurations
- user (not all) : per-user mount configurations
- `data/$user/mount.json` : personal mount configurations

6.8.5 Backends

Local Filesystem

The local filesystem backend mounts a folder on the server into the virtual filesystem, the class to be used is **\OC\Files\Storage\Local** and takes the following options:

- **datadir** : the path to the local directory to be mounted

Example

```
{ "class":"\\OC\\Files\\Storage\\Local",
  "options":{ "datadir":"\/mnt\/additional_storage" },
  "priority":150
}
```

Note: You must ensure that the web server has sufficient permissions on the folder.

FTP (or FTPS)

The FTP backend mounts a folder on a remote FTP server into the virtual filesystem and is part of the 'External storage support' app, the class to be used is **\OC\Files\Storage\FTP** and takes the following options:

- **host**: the hostname of the ftp server, and optionally the port number
- **user**: the username used to login to the ftp server
- **password**: the password to login to the ftp server
- **secure**: whether to use ftps:// (FTP over TLS) to connect to the ftp server instead of ftp:// (optional, defaults to false)
- **root**: the remote subfolder inside the ftp server to mount (optional, defaults to '/')

Example

```
{   "class":"\\OC\\Files\\Storage\\FTP",
    "options":{
        "host":"ftp.myhost.com:21",
        "user":"johndoe",
```

```
        "password":"secret",
        "root":"\/Videos",
        "secure":"false"
    },
    "priority":100
}
```

Note: PHP needs to be built with FTP support for this backend to work.

Note: The external storage `FTP/FTPS/SFTP` 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`.

SFTP

The SFTP backend mounts a folder on a remote SSH server into the virtual filesystem and is part of the 'External storage support' app. The class to be used is **\OC\Files\Storage\SFTP** and takes the following options:

- **host**: the hostname of the SSH server
- **user**: the username used to login to the SSH server
- **password**: the password to login to the SSH server
- **root**: the remote subfolder inside the SSH server to mount (optional, defaults to '/')

Example

```
{   "class":"\\OC\\Files\\Storage\\SFTP",
    "options":{
        "host":"ssh.myhost.com",
        "user":"johndoe",
        "password":"secret",
        "root":"\/Books"
    },
    "priority":100
}
```

Note: PHP needs to be built with SFTP support for this backend to work.

Note: The external storage `FTP/FTPS/SFTP` 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`.

SMB

The SMB backend mounts a folder on a remote Samba server, a NAS appliance or a Windows machine into the virtual file system. It is part of the 'External storage support' app, the class to be used is **\OC\Files\Storage\SMB** and takes the following options:

- **host**: the host name of the samba server
- **user**: the username or domain/username to login to the samba server
- **password**: the password to login to the samba server

- **share**: the share on the samba server to mount
- **root**: the remote subfolder inside the samba share to mount (optional, defaults to '/'). To assign the ownCloud logon username automatically to the subfolder, use `$user` instead of a particular subfolder name.

Note: The SMB backend requires **smbclient** to be installed on the server.

Example

With username only:

```
{    "class":"\\OC\\Files\\Storage\\SMB",
     "options":{
          "host":"myhost.com",
          "user":"johndoe",
          "password":"secret",
          "share":"\/test",
          "root":"\/Pictures"
     },
     "priority":100
}
```

With domainname and username:

```
{    "class":"\\OC\\Files\\Storage\\SMB",
     "options":{
          "host":"myhost.com",
          "user":"domain\/johndoe",
          "password":"secret",
          "share":"\/test",
          "root":"\/Pictures"
     },
     "priority":100
}
```

WebDAV

The WebDAV backend mounts a folder on a remote WebDAV server into the virtual filesystem and is part of the 'External storage support' app, the class to be used is **\OC\Files\Storage\DAV** and takes the following options:

- **host**: the hostname of the webdav server.
- **user**: the username used to login to the webdav server
- **password**: the password to login to the webdav server
- **secure**: whether to use https:// to connect to the webdav server instead of http:// (optional, defaults to false)
- **root**: the remote subfolder inside the webdav server to mount (optional, defaults to '/')

Example

```
{    "class":"\\OC\\Files\\Storage\\DAV",
     "options":{
          "host":"myhost.com\/webdav.php",
          "user":"johndoe",
```

```
        "password":"secret",
        "secure":"true"
    },
    "priority":100
}
```

Amazon S3

The Amazon S3 backend mounts a bucket in the Amazon cloud into the virtual filesystem and is part of the 'External storage support' app, the class to be used is **\OC\Files\Storage\AmazonS3** and takes the following options:

- **key**: the key to login to the Amazon cloud
- **secret**: the secret to login to the Amazon cloud
- **bucket**: the bucket in the Amazon cloud to mount

Example

```
{   "class":"\\OC\\Files\\Storage\\AmazonS3",
    "options":{
        "key":"key",
        "secret":"secret",
        "bucket":"bucket"
    },
    "priority":100
}
```

Dropbox

The Dropbox backend mounts a dropbox in the Dropbox cloud into the virtual filesystem and is part of the 'External storage support' app, the class to be used is **\OC\Files\Storage\Dropbox** and takes the following options:

- **configured**: whether the drive has been configured or not (true or false)
- **app_key**: the app key to login to your Dropbox
- **app_secret**: the app secret to login to your Dropbox
- **token**: the OAuth token to login to your Dropbox
- **token_secret**: the OAuth secret to login to your Dropbox

Example

```
{   "class":"\\OC\\Files\\Storage\\Dropbox",
    "options":{
        "configured":"#configured",
        "app_key":"key",
        "app_secret":"secret",
        "token":"#token",
        "token_secret":"#token_secret"
    },
    "priority":100
}
```

Google Drive

The Google Drive backend mounts a share in the Google cloud into the virtual filesystem and is part of the 'External storage support' app, the class to be used is **\OC\Files\Storage\Google** and is done via an OAuth2.0 request. That means that the App must be registered through the Google APIs Console. The result of the registration process is a set of values (incl. client_id, client_secret). It takes the following options:

- **configured**: whether the drive has been configured or not (true or false)
- **client_id**: the client id to login to the Google drive
- **client_secret**: the client secret to login to the Google drive
- **token**: a compound value including access and refresh tokens

Example

```
{    "class":"\\OC\\Files\\Storage\\Google",
     "options":{
         "configured":"#configured",
         "client_id":"#client_id",
         "client_secret":"#client_secret",
         "token":"#token"
     },
     "priority":100
}
```

OpenStack Swift

The Swift backend mounts a container on an OpenStack Object Storage server into the virtual filesystem and is part of the 'External storage support' app, the class to be used is **\OC\Files\Storage\SWIFT** and takes the following options:

- **host**: the hostname of the authentication server for the swift storage.
- **user**: the username used to login to the swift server
- **token**: the authentication token to login to the swift server
- **secure**: whether to use ftps:// to connect to the swift server instead of ftp:// (optional, defaults to false)
- **root**: the container inside the swift server to mount (optional, defaults to '/')

Example

```
{    "class":"\\OC\\Files\\Storage\\SWIFT",
     "options":{
         "host":"swift.myhost.com\/auth",
         "user":"johndoe",
         "token":"secret",
         "root":"\/Videos",
         "secure":"true"
     },
     "priority":100
}
```

6.8.6 External Storage Password Management

ownCloud handles passwords for external mounts differently than regular ownCloud user passwords.

The regular user and file share passwords (when you use the default ownCloud user backend) are stored using a strong cryptographically secure hashing mechanism in the database. On a new user account with a new password, the password is hashed and stored in the ownCloud database. The plain-text password is never stored. When the user logs in, the hash of the password they enter is compared with the hash in the database. When the hashes match the user is allowed access. These are not recoverable, so when a user loses a password the only option is to create a new password.

Passwords which are used to connect against external storage (e.g. SMB or FTP), there we have to differentiate again between different implementations:

1. **Login with ownCloud credentials**

When a mountpoint has this option, for example `SMB / CIFS using OC login`, the password will be intercepted when a user logs in and written to the PHP session (which is a file on the filesystem), and written encrypted into the session with a key from the configuration file. Every time that password is required ownCloud reads it from the PHP session file.

When you use this option, features such as sharing will not work properly from that mountpoint when the user is not logged-in.

Depending on the implementation of the application, this means that the password could get leaked in the `ps` output, as we use `smbclient` for SMB storage access in the community version. There is a bug report on this. Consequently, we're currently evaluating an alternative approach accessing the library directly, and thus not leaking the password anymore. This is already implemented in the Enterprise Edition in our Windows Network Drive application, and it will get into the community version once we have streamlined the code of the `files_external` application a little bit more.

2. **Stored credentials**

When you enter credentials into the `files_external` dialog those are stored on the filesystem and encrypted with a key stored in `config.php`. This is required since ownCloud needs access to those files and shares even when the user is not logged-in to have sharing and other key features properly working.

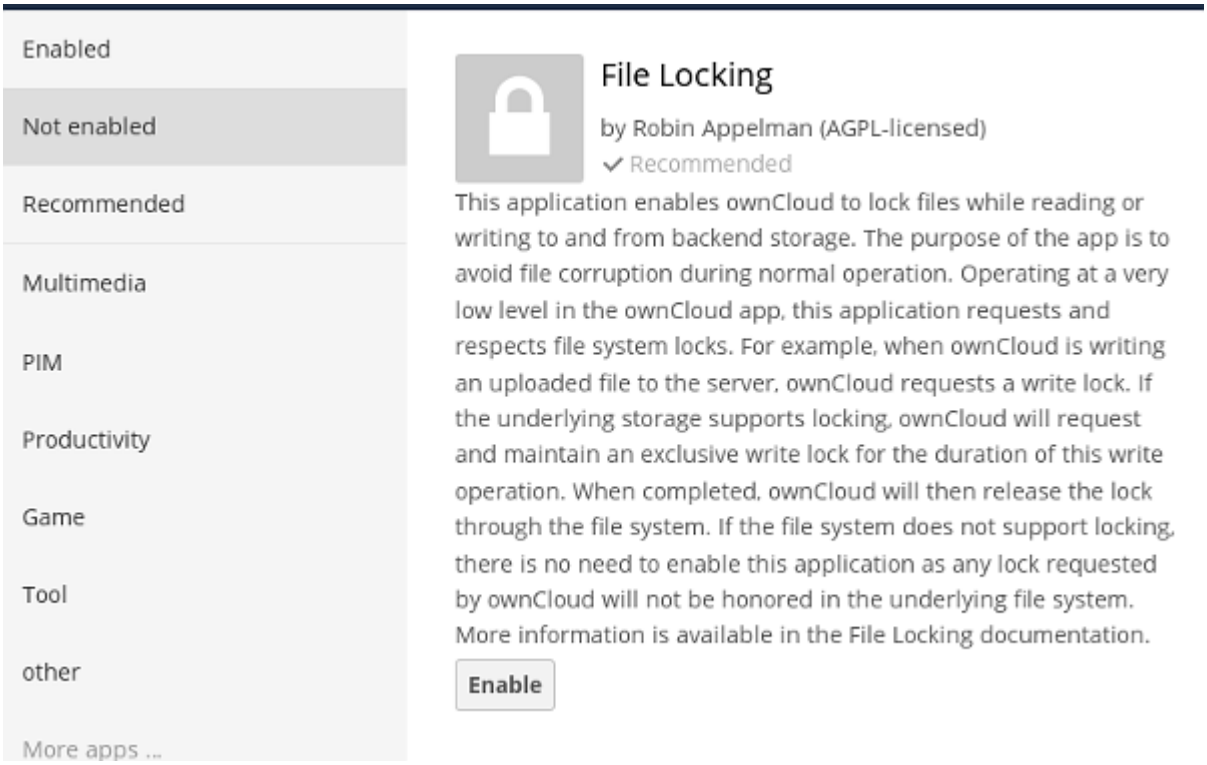
To sum up:

The "login with ownCloud credentials" SMB function in the community edition exposes the password in the server system's process list. If you want to get around this limitation without waiting for it to be addressed in CE you can get the Enterprise Edition. However, even then the password is stored in the PHP session and a malicious admin could access it. You can protect your PHP session files using protections available in your filesystem. Stored credentials are always accessible to the ownCloud instance.

6.9 Using the Files Locking App

The Files Locking application enables ownCloud to lock files while reading or writing to and from backend storage. The purpose of the app is to avoid file corruption during normal operation. Operating at a very low level in ownCloud, this application requests and respects file system locks. For example, when ownCloud is writing an uploaded file to the server, ownCloud requests a write lock. If the underlying storage supports locking, ownCloud will request and maintain an exclusive write lock for the duration of this write operation. When completed, ownCloud will then release the lock through the filesystem. If the file system does not support locking, there is no need to enable this application as any lock requested by ownCloud will not be honored in the underlying filesystem.

The Files Locking app has no configuration options; all you need to do is enable or disable it on your Apps page.



6.10 Configuring Federated Cloud Sharing

In ownCloud 7 this was called server-to-server sharing. Now it is called federated cloud sharing. With just a few clicks you can easily and securely link file shares between ownCloud servers, in effect creating a cloud of ownClouds. You can automatically send an email notification when you create the share, share directly with users on other ownCloud servers, add password protection, allow users to upload files, and set an expiration date.

Note: Currently, federated shares cannot be re-shared, and the only visible option when you create the share is **Can edit**.

6.10.1 Sharing With ownCloud 7

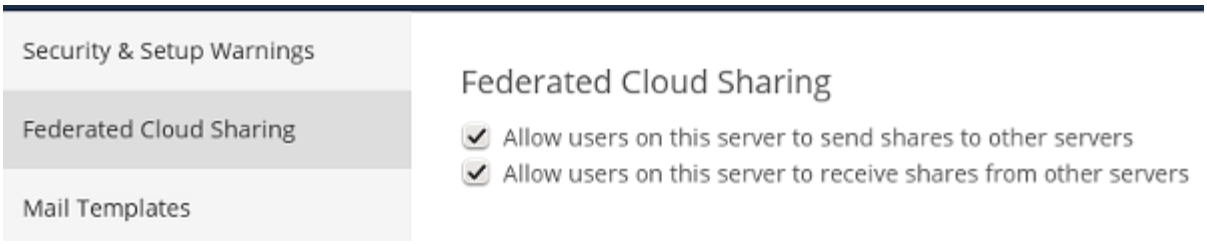
Direct share links (*Creating a Direct Share Link*) are not supported in ownCloud 7, so you must create federated cloud shares with public links (*Creating Federated Cloud Shares via Public Link Share*).

6.10.2 Creating a Direct Share Link

It is not labeled in the user interface, but you can create a direct share link with a user on a remote ownCloud server in the `Share with user or group` form field in the sharing dialog. Follow these steps:

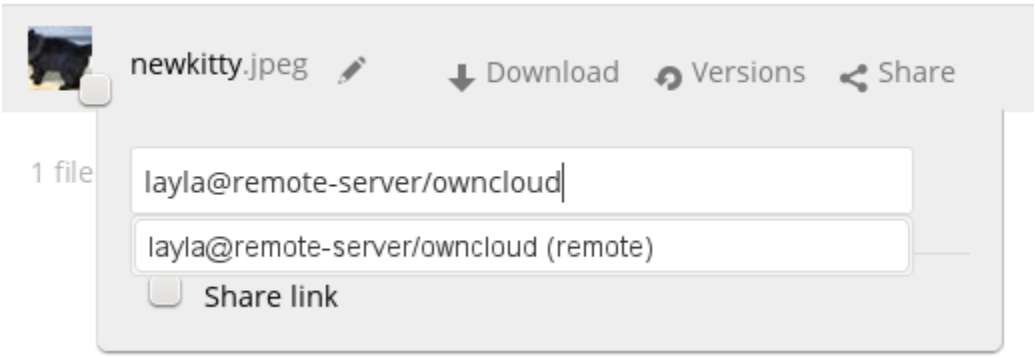
1. Go to your ownCloud Admin page and scroll to the Federated Cloud Sharing section.

2. Check `Allow other users on this server to send shares to other servers` and `Allow users on this server to receive shares from other servers`. Leaving the checkboxes blank disables federated cloud sharing.

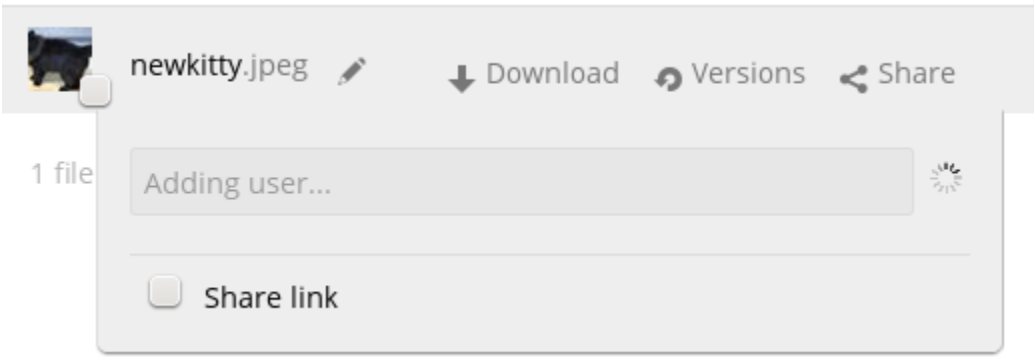


3. In the Sharing section, check `Allow users to share via link` and `Allow users to send mail notification for shared files`.

4. Now you and your users can go to your Files pages to create a new direct share link. Click the Share icon on the file or directory you want to share to expose your first sharing option. This dialog allows you to create local shares with users and groups on your local ownCloud server, and also to create federated cloud shares with users on remote ownCloud servers by typing a link to the remote server in the form of `<user>@<link-to-owncloud>`. In this screenshot the remote ownCloud server is on the local network, so the URL form is `user@hostname/owncloud`, or `layla@remote-server/owncloud` in the example. The URL you type is echoed by the form, and labeled as `(remote)`.



Press the return key, and then wait for the link to be established. You'll see a status message while it is working.

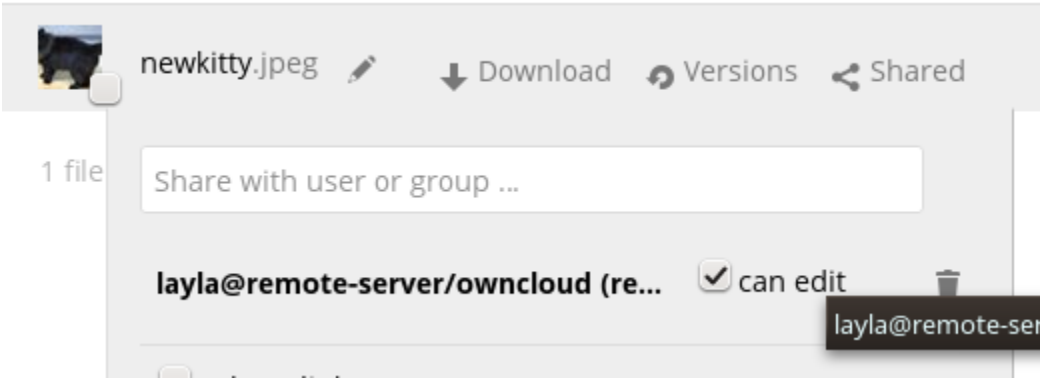


When the remote server has been successfully contacted you'll see a confirmation.

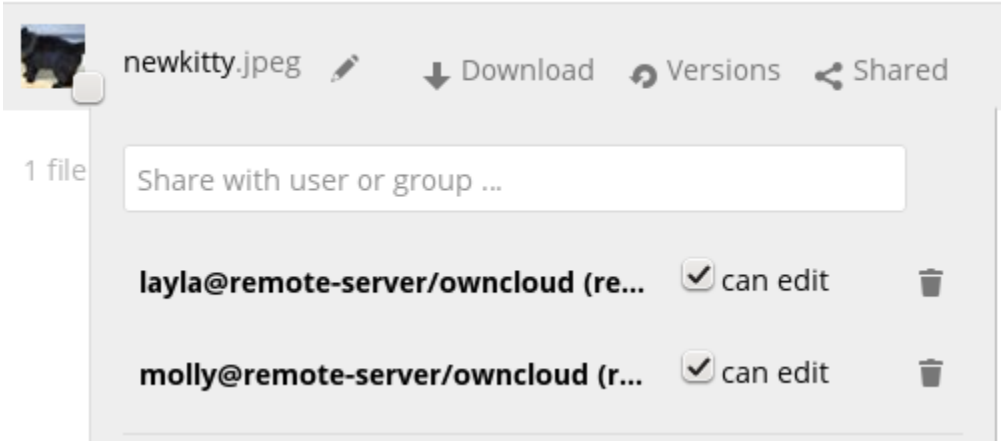
The link is created when your remote user confirms the share by clicking the **Add remote share** button on the confirmation dialog.

You can return to the share dialog any time to see a list of everyone you have shared with, and federated cloud shares are labeled as `(remote)`.

Click the trash can icons to disconnect the share.

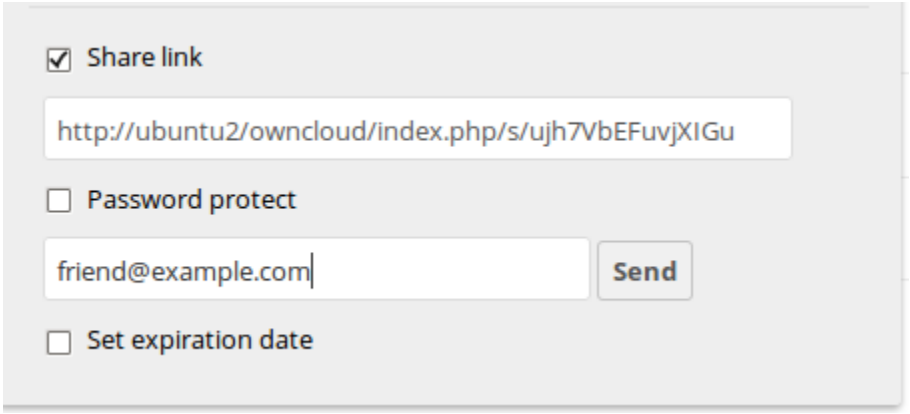

newkitty.jpeg
Download
Versions
Shared
1 file
Share with user or group ...
layla@remote-server/owncloud (re...
can edit
layla@remote-ser


Remote share
Do you want to add the remote share /newkitty.jpeg from stashcat@studio/owncloud/?
Cancel
Add remote share


newkitty.jpeg
Download
Versions
Shared
1 file
Share with user or group ...
layla@remote-server/owncloud (re...
can edit
molly@remote-server/owncloud (r...
can edit

6.10.3 Creating Federated Cloud Shares via Public Link Share

Check the `Share Link` checkbox to expose more sharing options (which are described more fully in *File Sharing*). You may create a federated cloud share by allowing ownCloud to create a public link for you, and then email it to the person you want to create the share with.



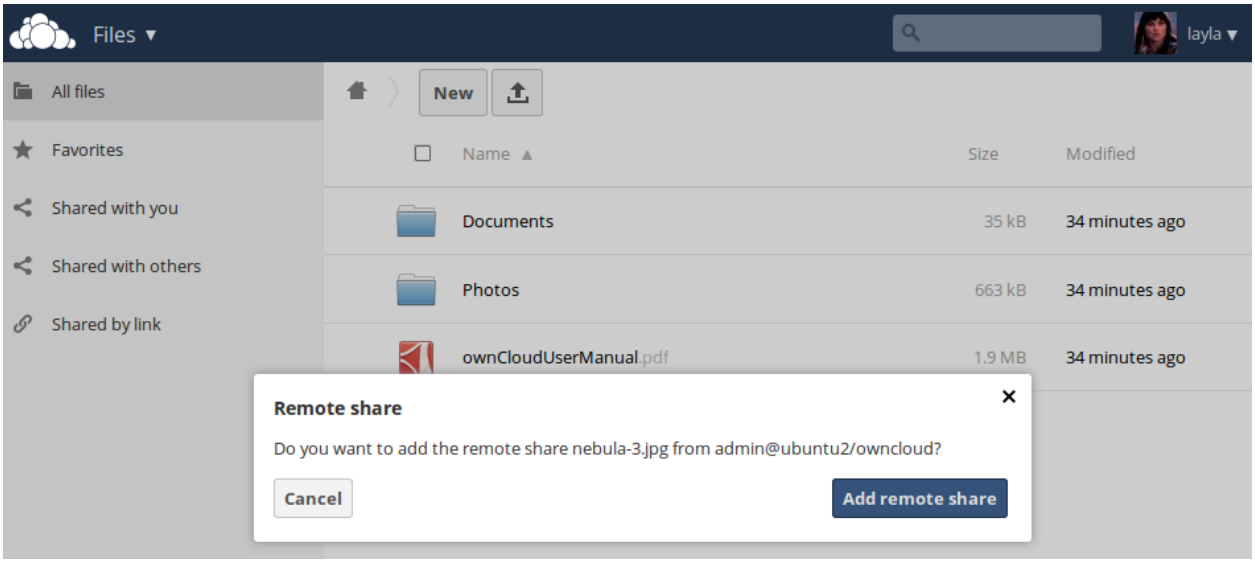
You may optionally set a password and expiration date on it. When your recipient receives your email they must click the link, or copy it to a Web browser. They will see a page displaying a thumbnail of the file, with a button to **Add to your ownCloud**.

Your recipient should click the **Add to your ownCloud** button. On the next screen your recipient needs to enter the URL to their ownCloud server, and then press the return key.

Your recipient has to take one more step, and that is to confirm creating the federated cloud share link by clicking the **Add remote share** button.

Un-check the `Share Link` checkbox to disable any federated cloud share created this way.


ıote-server/owncloud
Download


Files
layla
All files
Favorites
Shared with you
Shared with others
Shared by link
New
Name
Size
Modified
Documents
35 kB
34 minutes ago
Photos
663 kB
34 minutes ago
ownCloudUserManual.pdf
1.9 MB
34 minutes ago
Remote share
Do you want to add the remote share nebula-3.jpg from admin@ubuntu2/owncloud?
Cancel
Add remote share

6.10.4 Configuration Tips

The Sharing section on your Admin page allows you to control how your users manage federated cloud shares:

- Check `Enforce password protection` to require passwords on link shares.
- Check `Set default expiration date` to require an expiration date on link shares.
- Check `Allow public uploads` to allow two-way file sharing.

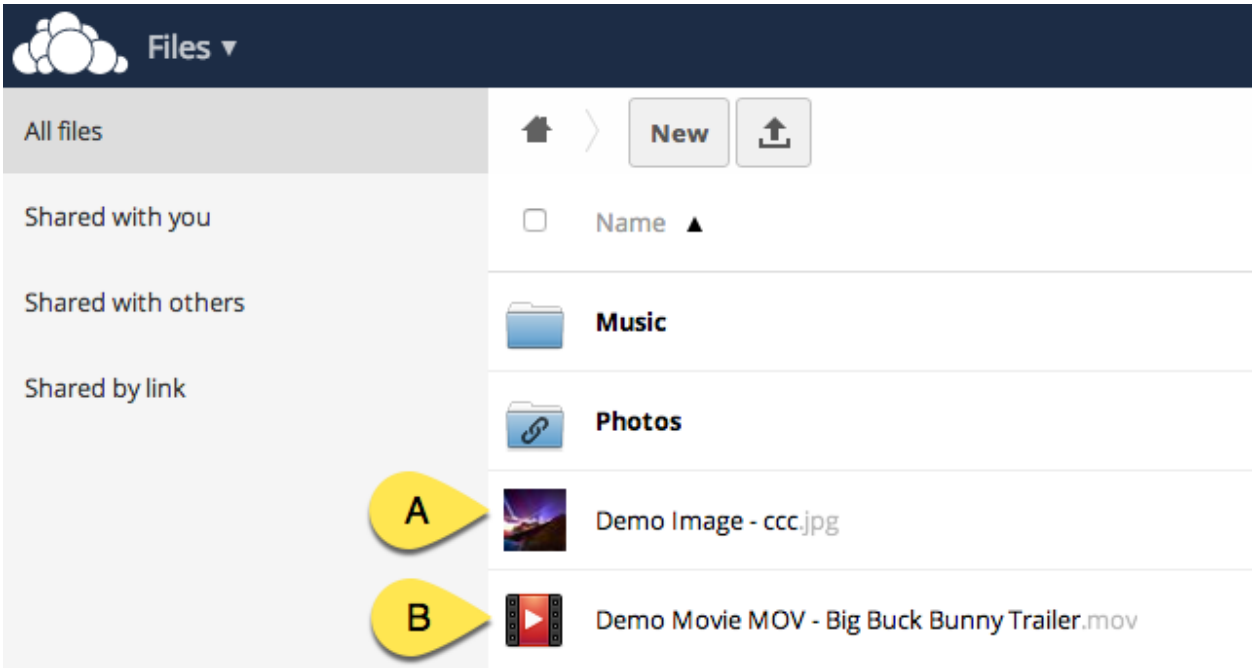
Your Apache Web server must have `mod_rewrite` enabled, and you must have `trusted_domains` correctly configured in `config.php` to allow external connections (see *Installation Wizard*). Consider also enabling SSL to encrypt all traffic between your servers .

Your ownCloud server creates the share link from the URL that you used to log into the server, so make sure that you log into your server using a URL that is accessible to your users. For example, if you log in via its LAN IP address, such as `http://192.168.10.50`, then your share URL will be something like `http://192.168.10.50/owncloud/index.php/s/jWfCfTVztGlWTJe`, which is not accessible outside of your LAN. This also applies to using the server name; for access outside of your LAN you need to use a fully-qualified domain name such as `http://myserver.example.com`, rather than `http://myserver`.

6.11 Previews Configuration

The ownCloud thumbnail system generates previews of files for all ownCloud apps that display files, such as Files and Gallery.

The following image shows a file (A) where the ownCloud server generates a preview image, and another file (B) that it could not generate a preview for. In this case a generic icon is displayed.



By default, ownCloud can generate previews for the following filetypes:

- Images files
- Cover of MP3 files

- Text documents

Note: 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.

6.11.1 Parameters

Please notice that the ownCloud preview system comes already with sensible defaults, and therefore it is usually unnecessary to adjust those configuration values.

Disabling previews:

Under certain circumstances, for example if the server has limited resources, you might want to consider disabling the generation of previews. Note that if you do this all previews in all apps are disabled, including the Gallery app, and will display generic icons instead of thumbnails.

Set the configuration option `enable_previews` in config.php to `false`:

```
<?php
  'enable_previews' => false,
```

Maximum preview size:

There are two configuration options to set the maximum size of a preview.

```
<?php
  'preview_max_x' => null,
  'preview_max_y' => null,
```

By default, both options are set to null. 'Null' is equal to no limit. Numeric values represent the size in pixels. The following code limits previews to a maximum size of 100×100px:

```
<?php
  'preview_max_x' => 100,
  'preview_max_y' => 100,
```

'preview_max_x' represents the x-axis and 'preview_max_y' represents the y-axis.

Maximum scale factor:

If a lot of small pictures are stored on the 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:

```
<?php
  'preview_max_scale_factor' => 10,
```

If you want to disable scaling at all, you can set the config value to '1':

```
<?php
  'preview_max_scale_factor' => 1,
```

If you want to disable the maximum scaling factor, you can set the config value to 'null':

```
<?php
  'preview_max_scale_factor' => null,
```

6.12 Serving Static Files for Better Performance

ownCloud can serve static files, which may improve performance.

Note: This feature can currently only be activated for local files, i.e. files inside the **data/** directory and local mounts. It also does not work with the Encryption App enabled. Controlled file serving **does not work for generated zip files**. This is due to zip files being generated and streamed back directly to the client.

6.12.1 Apache2 (X-Sendfile)

It is possible to let Apache handle static file serving via mod_xsendfile.

Installation

On Debian and Ubuntu systems use:

```
apt-get install libapache2-mod-xsendfile
```

Configuration

Configuration of mod_xsendfile for ownCloud depends on its version. For versions below 0.10 (Debian squeeze ships with 0.9)

```
<Directory /var/www/owncloud>
    ...
    SetEnv MOD_X_SENDFILE_ENABLED 1
    XSendFile On
    XSendFileAllowAbove On
</Directory>
```

For versions >=0.10 (e.g. Ubuntu 12.10)

```
<Directory /var/www/owncloud>
    ...
    SetEnv MOD_X_SENDFILE_ENABLED 1
    XSendFile On
    XSendFilePath /home/valerio
</Directory>
```

- **SetEnv MOD_X_SENDFILE_ENABLED**: tells ownCloud scripts that they should add the X-Sendfile header when serving files
- **XSendFile**: enables web server handling of X-Sendfile headers (and therefore file serving) for the specified Directory
- **XSendFileAllowAbove (<0.10)**: enables file serving through the web server on a path outside the specified Directory. This is needed for configured local mounts which may reside outside the data directory.

- **XSendFilePath (>=0.10)**: a white list of paths that the web server is allowed to serve outside of the specified Directory. Other paths which correspond to local mounts should be configured here as well. For a more in-depth documentation of this directive refer to the mod_xsendfile website linked above.

6.12.2 Nginx (X-Accel-Redirect)

Nginx supports handling of static files differently from Apache. Documentation can be found in the Nginx Wiki section Mod X-Sendfile and section X-Accell. The header used by Nginx is X-Accel-Redirect.

Installation

X-Accel-Redirect is supported by default in Nginx and no additional operation should be needed to install it.

Configuration - Method 1

Method 1 is preferred because it limits what files can be served through X-Accel.

```
location ~ \.php(?:$|/) {
    ...
    fastcgi_param MOD_X_ACCEL_REDIRECT_ENABLED on;
}

location ^~ /data {
    internal;
    # Set 'alias' if not using the default 'datadirectory'
    #alias /path/to/non-default/datadirectory;

#    LOCAL-MOUNT-NAME should match "Folder name" and 'alias' value should match "Configuration"
#    A 'Local' External Storage Mountpoint available to a single user
#    location /data/USER/files/LOCAL-FS-MOUNT-NAME {
#        alias /path/to/local-mountpoint;
#    }

#    A 'Local' External Storage Mountpoint available to multiple users
#    location ~ ^/data/(?:USER1|USER2)/files/LOCAL-FS-MOUNT-NAME/(.+)$ {
#        alias /path/to/local-mountpoint/$1;
#    }

#    A 'Local' External Storage Mountpoint available to all users
#    location ~ ^/data/[^/]+/files/LOCAL-FS-MOUNT-NAME/(.+)$ {
#        alias /path/to/local-mountpoint/$1;
#    }

}
```

- **fastcgi_param MOD_X_ACCEL_REDIRECT_ENABLED** ~ Tells ownCloud scripts that they should add the X-Accel-Redirect header when serving files.
- **/data** ~ The ownCloud data directory. Any 'Local' External Storage Mounts must also have nested locations here.
  - set alias if you are using a non-default data directory
  - **/data/USER/files/LOCAL-MOUNT-NAME** ~ a local external storage mount available to a single user
  - **~ ^/data/(?:USER1|USER2)/files/LOCAL-MOUNT-NAME/(.+)$** ~ a local external storage mount available to multiple users

  - **~ ^/data/[^/]+/files/LOCAL-MOUNT-NAME/(.+)$** ~ a local external storage mount available to all users

Configuration - Method 2

Method 2 is simpler to setup when using local external storage mounts, especially when they are available to many, but not all users. This method may be preferred if you are regularly adding users that should not all have access to the same local external storage mount(s).

```
location ~ \.php(?:$|/) {
    ...
    fastcgi_param MOD_X_ACCEL_REDIRECT_ENABLED on;
    fastcgi_param MOD_X_ACCEL_REDIRECT_PREFIX /xaccel;
}

location ^~ /xaccel {
    internal;
    alias /;
}
```

- **fastcgi_param MOD_X_ACCEL_REDIRECT_ENABLED** ~ Tells ownCloud scripts that they should add the X-Accel-Redirect header when serving files.
- **fastcgi_param MOD_X_ACCEL_REDIRECT_PREFIX** ~ A prefix to internally serve files from, in this example "/xaccel" is used but this is configurable.
- **location ^~ /xaccel** ~ The location to internally serve files from, must match MOD_X_ACCEL_REDIRECT_PREFIX.

6.12.3 How to check if it's working?

You are still able to download stuff via the web interface and single, local file downloads can be paused and resumed.

CHAPTER

SEVEN

OWNCLOUD SERVER CONFIGURATION

7.1 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`:

```
* Manage apps
* Manage users
* Convert the ownCloud database
* Reset passwords, including administrator passwords
* Convert the ownCloud database from SQLite to a more performant DB
* Query and change LDAP settings
```

`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.

The HTTP user is different on the various Linux distributions. See the **Setting Strong Directory Permissions** section of *Installation Wizard* to learn how to find your HTTP user.

- 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`.

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

```
$ sudo -u www-data php occ
ownCloud version 8.0.3
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
 upgrade                     run upgrade routines after installation of a new
                             release. The release has to be installed before.
```

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

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

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

Display your ownCloud version:

```
$ sudo -u www-data php occ -V
  ownCloud version 8.0.3
```

Query your ownCloud server status:

```
$ sudo -u www-data php occ status
  - installed: true
  - version: 8.0.3.4
  - versionstring: 8.0.3
  - edition: Enterprise
```

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

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

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

```
$ sudo -u www-data php occ help maintenance:mode
  Usage:
  maintenance:mode [--on] [--off]

  Options:
  --on                  enable maintenance mode
  --off                 disable maintenance mode
  --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.
```

7.1.1 Apps Commands

The `app` commands list, enable, and disable apps. This lists all of your installed apps, and shows whether they are enabled or disabled:

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

Enable an app:

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

Disable an app:

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

7.1.2 Database Conversion

The SQLite database is good for testing, and for ownCloud servers with small workloads, 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. 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 to SQLite MySQL/MariaDB:

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

For a more detailed explanation see *Converting Database Type*

7.1.3 Encryption

When you are using encryption, you must manually migrate your encryption keys after upgrading your ownCloud server:

```
$ sudo -u www-data php occ encryption:migrate-keys
```

7.1.4 File Operations

The `files:scan` command scans for new files and updates the file cache. You may rescan all files, per-user, a space-delimited list of users, and limit the search path:

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

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

Options:
  --path (-p)           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
  --all                 will rescan all files of all known users
```

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

7.1.5 l10n, Create javascript Translation Files for Apps

Use the `l10n:createjs` to translate apps into various languages, using this syntax:

```
l10n:createjs appname language_name
```

This example converts the Activity app to Bosnian:

```
$ sudo -u www-data php occ l10n:createjs activity bs
```

These are the supported language codes, and Codes for the Representation of Names of Languages may be helpful:

```
ach                 gu      ml      sr
ady         eo      he      ml_IN   sr@latin
af_ZA       es      hi      mn      su
ak          es_AR   hi_IN   ms_MY   sv
am_ET       es_BO   hr      mt_MT   sw_KE
ar          es_CL   hu_HU   my_MM   ta_IN
ast         es_CO   hy      nb_NO   ta_LK
az          es_CR   ia      nds     te
be          es_EC   id      ne      tg_TJ
bg_BG       es_MX   io      nl      th_TH
bn_BD       es_PE   is      nn_NO   tl_PH
bn_IN       es_PY   it      nqo     tr
bs          es_US   ja      oc      tzm
ca          es_UY   jv      or_IN   ug
ca@valencia et_EE   ka_GE   pa      uk
cs_CZ       eu      km      pl      ur
cy_GB       eu_ES   kn      pt_BR   ur_PK
da          fa      ko      pt_PT   uz
de          fi      ku_IQ   ro      vi
de_AT       fi_FI   lb      ru      yo
de_CH       fil     lo      si_LK   zh_CN
de_DE       fr      lt_LT   sk      zh_HK
el          fr_CA   lv      sk_SK   zh_TW
en_GB       fy_NL   mg      sl
en_NZ       gl      mk      sq
```

7.1.6 LDAP Commands

You can run the following LDAP commands with `occ`.

Search for an LDAP user, using this syntax:

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

This example searches for usernames that start with “rob”:

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

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

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

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

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

`ldap:create-empty-config` creates an empty LDAP configuration. The first one you create has no `configID`, like this example:

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

This is a holdover from the early days, when there was no option to create additional configurations. The second, and all subsequent, configurations that you create are automatically assigned IDs:

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

Then you can list and view your configurations:

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

And view the configuration for a single configID:

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

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

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

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

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

`ldap:show-remnants` is for cleaning up the LDAP mappings table, and is documented in *LDAP User Cleanup*.

7.1.7 Maintenance Commands

These maintenance commands put your ownCloud server into maintenance and single-user mode, and run repair steps during updates.

You must put your ownCloud server into maintenance mode whenever you perform an update or upgrade. This locks the sessions of all logged-in users, including administrators, and displays a status screen warning that the server is in maintenance mode. Users who are not already logged in cannot log in until maintenance mode is turned off. When you take the server out of maintenance mode logged-in users must refresh their Web browsers to continue working:

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

Putting your 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:

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

And turn it off when you're finished:

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

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

```
 $ sudo -u www-data php occ maintenance:repair
    - Repair mime types
- Repair legacy storages
- Repair config
- Clear asset cache after upgrade
    - Asset pipeline disabled -> nothing to do
- Generate ETags for file where no ETag is present.
    - ETags have been fixed for 0 files/folders.
- Clean tags and favorites
    - 0 tags for delete files have been removed.
    - 0 tag entries for deleted tags have been removed.
    - 0 tags with no entries have been removed.
- Re-enable file app
```

7.1.8 User Commands

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

You can reset any user's password, including administrators (see *Resetting a Lost Admin Password*):

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

You can delete users:

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

View a user's most recent login:

```
$ sudo -u www-data php occ user:lastseen layla
  layla's last login: 09.01.2015 18:46
```

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

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

7.1.9 Upgrade Command

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 *Upgrading Your ownCloud Server*) use this command to upgrade your databases:

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

Before completing the upgrade, ownCloud first runs a simulation by copying all database tables to new tables, and then performs the upgrade on them, to ensure that the upgrade will complete correctly. The copied tables are deleted after the upgrade. This takes twice as much time, which on large installations can be many hours, so you can omit this step with the `--skip-migration-test` option:

```
$ sudo -u www-data php occ upgrade --skip-migration-test
```

You can perform this simulation manually with the `--dry-run` option:

```
$ sudo -u www-data php occ upgrade --dry-run
```

7.2 Configuring the Activity App

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 sharee.

7.2.1 Enabling the Activity App

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

7.2.2 Configuring your ownCloud for the Activity App

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

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

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

7.3 Configuring the ClamAV Antivirus Scanner

You can configure your ownCloud server to automatically run a virus scan on newly-uploaded files with the Antivirus App for Files. The Antivirus App for Files integrates the open source anti-virus engine ClamAV with ownCloud. ClamAV detects all forms of malware including Trojan horses, viruses, and worms, and it operates on all major file types including Windows, Linux, and Mac files, compressed files, executables, image files, Flash, PDF, and many others. ClamAV's Freshclam daemon automatically updates its malware signature database at scheduled intervals.

ClamAV runs on Linux and any Unix-type operating system, and Microsoft Windows. However, it has only been tested with ownCloud on Linux, so these instructions are for Linux systems. You must first install ClamAV, and then install and configure the Antivirus App for Files on ownCloud.

7.3.1 Installing ClamAV

As always, the various Linux distributions manage installing and configuring ClamAV in different ways.

Debian, Ubuntu, Linux Mint On Debian and Ubuntu systems, and their many variants, install ClamAV with these commands:

```
apt-get install clamav clamav-daemon
```

The installer automatically creates default configuration files and launches the `clamd` and `freshclam` daemons. You don't have to do anything more, though it's a good idea to review the ClamAV documentation and your settings in `/etc/clamav/`. Enable verbose logging in both `clamd.conf` and `freshclam.conf` until you get any kinks worked out.

Red Hat 7, CentOS 7 On Red Hat 7 and related systems you must install the Extra Packages for Enterprise Linux (EPEL) repository, and then install ClamAV:

```
yum install epel-release
yum install clamav clamav-scanner clamav-scanner-systemd clamav-server
clamav-server-systemd clamav-update
```

This installs two configuration files: `/etc/freshclam.conf` and `/etc/clamd.d/scan.conf`. You must edit both of these before you can run ClamAV. Both files are well-commented, and `man clamd.conf` and `man freshclam.conf` explain all the options. Refer to `/etc/passwd` and `/etc/group` when you need to verify the ClamAV user and group.

First edit `/etc/freshclam.conf` and configure your options. `freshclam` updates your malware database, so you want it to run frequently to get updated malware signatures. Run it manually post-installation to download your first set of malware signatures:

```
freshclam
```

The EPEL packages do not include an init file for `freshclam`, so the quick and easy way to set it up for regular checks is with a cron job. This example runs it every hour at 47 minutes past the hour:

```
# m   h  dom mon dow  command
  47  *  *   *    *  /usr/bin/freshclam --quiet
```

Please avoid any multiples of 10, because those are when the ClamAV servers are hit the hardest for updates.

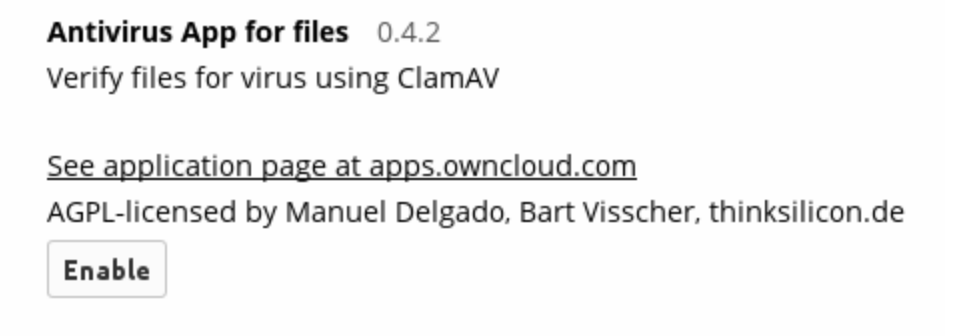
Next, edit `/etc/clamd.d/scan.conf`. When you're finished you must enable the `clamd` service file and start `clamd`:

```
systemctl enable clamd@scan.service
systemctl start clamd@scan.service
```

That should take care of everything. Enable verbose logging in `scan.conf` and `freshclam.conf` until it is running the way you want.

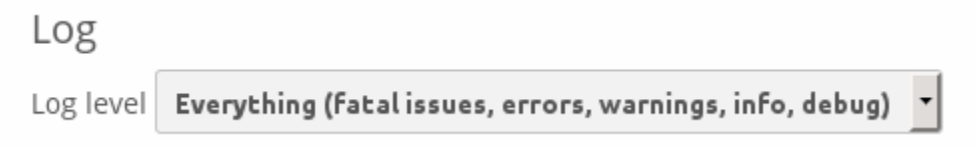
7.3.2 Enabling the Antivirus App for Files

Simply go to your ownCloud Apps page to enable it.

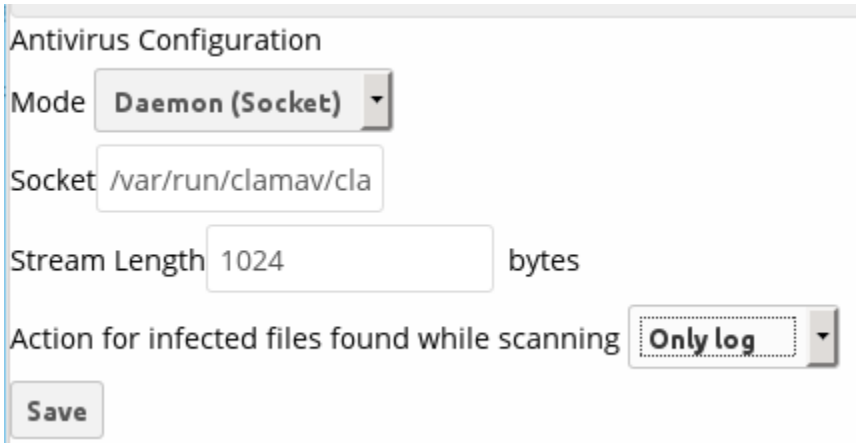


7.3.3 Configuring ClamAV on ownCloud

Next, go to your ownCloud Admin page and set your ownCloud logging level to Everything.



Now find your Antivirus Configuration panel on your Admin page.

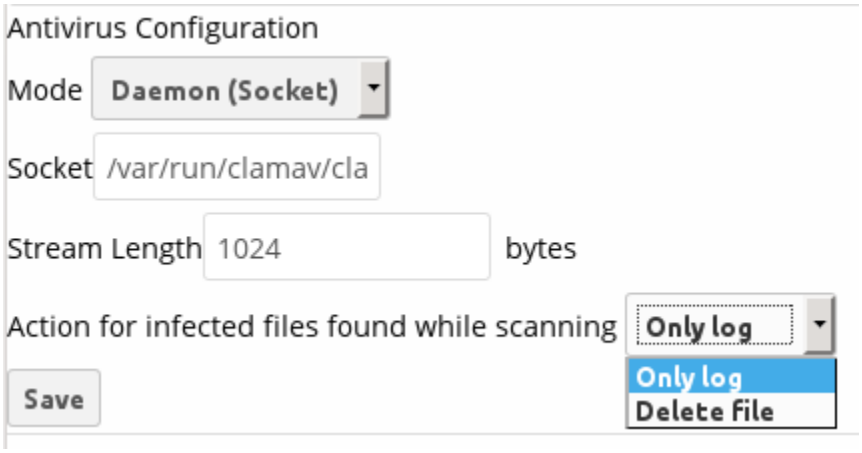


ClamAV runs in one of three modes:

- Daemon (Socket): ClamAV is running on the same server as ownCloud. The ClamAV daemon, `clamd`, runs in the background. When there is no activity `clamd` places a minimal load on your system. If your users upload large volumes of files you will see high CPU usage.
- Daemon: ClamAV is running on a different server. This is a good option for ownCloud servers with high volumes of file uploads.
- Executable: ClamAV is running on the same server as ownCloud, and the `clamscan` command is started and then stopped with each file upload. `clamscan` is slow and not always reliable for on-demand usage; it is better to use one of the daemon modes.

Daemon (Socket) 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:

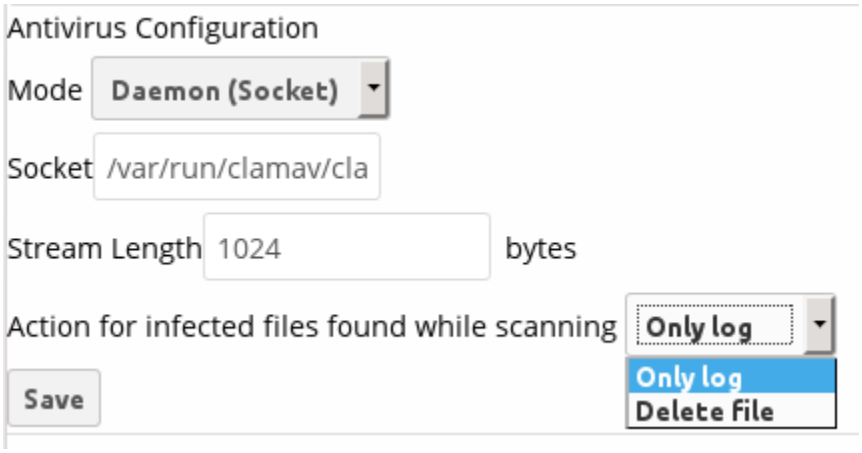
```
netstat -a|grep clam
unix 2 [ ACC ] STREAM LISTENING 15857 /var/run/clamav/clamd.ctl
```



The `Stream Length` value sets the number of bytes read in one pass. 10485760 bytes, or ten megabytes, is the default. This value should be no larger than the PHP `memory_limit` settings, or physical memory if `memory_limit` is set to -1 (no limit).

`Action for infected files found while scanning` gives you the choice of logging any alerts without deleting the files, or immediately deleting infected files.

Daemon For the Daemon option you need the hostname or IP address of the remote server running ClamAV, and the server's port number.

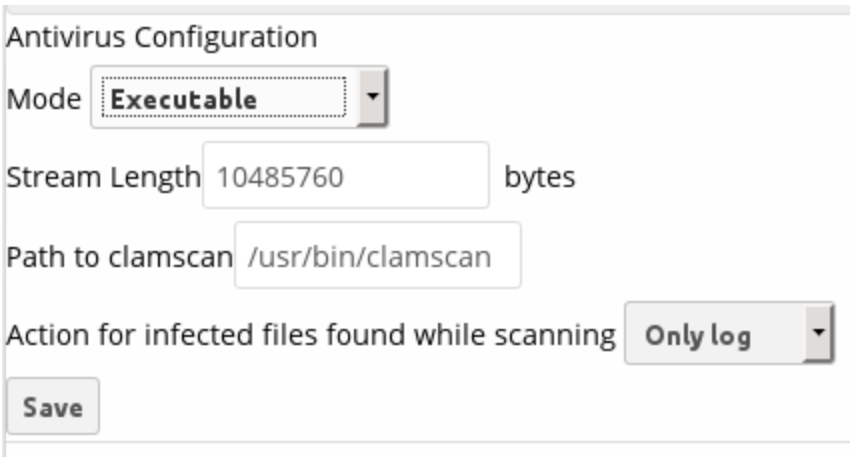


Executable The Executable option requires the path to `clamscan`, which is the interactive ClamAV scanning command. ownCloud should find it automatically.

When you are satisfied with how ClamAV is operating, you might want to go back and change all of your logging to less verbose levels.

7.4 Automatic Configuration Setup

If you need to install ownCloud on multiple servers, you normally do not want to set up each instance separately as described in *Database Configuration*. For this reason, ownCloud provides an automatic configuration feature.



To take advantage of this feature, you must create a configuration file, called `../owncloud/config/autoconfig.php`, and set the file parameters as required. You can specify any number of parameters in this file. Any unspecified parameters appear on the "Finish setup" screen when you first launch ownCloud.

The `../owncloud/config/autoconfig.php` is automatically removed after the initial configuration has been applied.

7.4.1 Parameters

When configuring parameters, you must understand that two parameters are named differently in this configuration file when compared to the standard `config.php` file.

autoconfig.php	config.php
directory	datadirectory
dbpass	dbpassword

7.4.2 Automatic Configurations Examples

The following sections provide sample automatic configuration examples and what information is requested at the end of the configuration.

Data Directory

Using the following parameter settings, the "Finish setup" screen requests database and admin credentials settings.

```
<?php
$AUTOCONFIG = array(
  "directory"     => "/www/htdocs/owncloud/data",
);
```

SQLite Database

Using the following parameter settings, the "Finish setup" screen requests data directory and admin credentials settings.

```
<?php
$AUTOCONFIG = array(
  "dbtype"        => "sqlite",
  "dbname"        => "owncloud",
  "dbtableprefix" => "",
);
```

MySQL Database

Using the following parameter settings, the "Finish setup" screen requests data directory and admin credentials settings.

```
<?php
$AUTOCONFIG = array(
  "dbtype"        => "mysql",
  "dbname"        => "owncloud",
  "dbuser"        => "username",
  "dbpass"        => "password",
  "dbhost"        => "localhost",
  "dbtableprefix" => "",
);
```

Note: Keep in mind that the automatic configuration does not eliminate the need for creating the database user and database in advance, as described in *Database Configuration*.

PostgreSQL Database

Using the following parameter settings, the "Finish setup" screen requests data directory and admin credentials settings.

```
<?php
$AUTOCONFIG = array(
  "dbtype"        => "pgsql",
  "dbname"        => "owncloud",
  "dbuser"        => "username",
  "dbpass"        => "password",
  "dbhost"        => "localhost",
  "dbtableprefix" => "",
);
```

Note: Keep in mind that the automatic configuration does not eliminate the need for creating the database user and database in advance, as described in *Database Configuration*.

All Parameters

Using the following parameter settings, because all parameters are already configured in the file, the ownCloud installation skips the "Finish setup" screen.

```
<?php
$AUTOCONFIG = array(
  "dbtype"        => "mysql",
  "dbname"        => "owncloud",
  "dbuser"        => "username",
```

```
  "dbpass"        => "password",
  "dbhost"        => "localhost",
  "dbtableprefix" => "",
  "adminlogin"    => "root",
  "adminpass"     => "root-password",
  "directory"     => "/www/htdocs/owncloud/data",
);
```

Note: Keep in mind that the automatic configuration does not eliminate the need for creating the database user and database in advance, as described in *Database Configuration*.

7.5 Defining Background Jobs

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

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

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

7.5.1 Parameters

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

- AJAX
- Webcron
- Cron

7.5.2 Cron Jobs

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

AJAX

The AJAX scheduling method is the default option. Unfortunately, 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 is that it does not require access to the system nor registration with a third party service. The disadvantage of this mechanism, when compared to the Webcron service, is that it requires regular visits to the page for it to be triggered.

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

Webcron

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

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

Cron

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

To run a cron job on a *nix system, every 15 minutes, under the default web server user (often, `www-data` or `wwwrun`), you must set up the following cron job to call the **cron.php** script:

```
# crontab -u www-data -e
*/15  *  *  *  * php -f /var/www/owncloud/cron.php > /dev/null 2>&1
```

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

```
# crontab -u www-data -l
*/15  *  *  *  * php -f /var/www/owncloud/cron.php > /dev/null 2>&1
```

Note: You have to replace the path `/var/www/owncloud/cron.php` with the path to your current ownCloud installation.

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

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

7.6 Configuring Memory Caching

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

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

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

If you are using PHP 5.4, which is the oldest supported PHP version for ownCloud, you may install the Alternative PHP Cache (APC). This is both an opcache and data cache. APC has not been updated since 2012 and is essentially dead, and PHP 5.4 is old and lags behind later releases. If it is possible to upgrade to a later PHP release that is the best option.

Data caching is supplied by the Alternative PHP Cache, user (APCu) in PHP 5.5+, Memcached, or Redis.

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

- **APC** A local cache for systems running PHP 5.4.
- **APCu, APCu 4.0.6 and up required.** A local cache for systems running PHP 5.5 and up.
- **Memcached** Distributed cache for multi-server ownCloud installations.
- **Redis, PHP module 2.2.5 and up required.** For distributed caching.

Memcaches must be explicitly configured in ownCloud 8.1 and up by installing and enabling your desired cache, and then adding the appropriate entry to `config.php` (See *Config.php Parameters* for an overview of all possible config parameters).

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

7.6.1 APC

APC is only for systems running PHP 5.4 and older. The oldest supported PHP version in ownCloud is 5.4.

Note: RHEL 6 and CentOS 6 ship with PHP 5.3 and must be upgraded to PHP 5.4 to run ownCloud.

On Red Hat/CentOS/Fedora systems running PHP 5.4, install `php-pecl-apc`. On Debian/Ubuntu/Mint systems install `php-apc`. Then restart your Web server.

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

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

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

7.6.2 APCu

PHP 5.5 and up include the Zend OPcache in core, and on most Linux distributions it is enabled by default. However, it does not bundle a data cache. APCu is a data cache, and it is available in most Linux distributions. On Red Hat/CentOS/Fedora systems running PHP 5.5 and up install `php-pecl-apcu`. On Debian/Ubuntu/Mint systems install `php5-apcu`. On Ubuntu 14.04LTS, the APCu version is 4.0.2, which is too old to use with ownCloud. ownCloud requires 4.0.6+. You may install 4.0.7 from Ubuntu backports with this command:

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

Then restart your Web server.

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

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

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

7.6.3 Memcached

Memcached is a reliable oldtimer for shared caching on distributed servers, and performs well with ownCloud.

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

Setting up Memcached is easy. On Debian/Ubuntu/Mint install `memcached` and `php5-memcached`. The installer will automatically start `memcached` and configure it to launch at startup.

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

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

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

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

```
'memcache.local' => '\OC\Memcache\APCu',
'memcache.distributed' => '\OC\Memcache\Memcached',
'memcached_servers' => array(
     array('localhost', 11211),
     array('server1.example.com', 11211),
     array('server2.example.com', 11211),
     ),
```

7.6.4 Redis

The Redis PHP module must be version 2.2.5+. If you are running a Linux distribution that does not package the supported versions of this module, or does not package Redis at all, see *Additional Redis Installation Help*.

On Debian/Ubuntu/Mint install `redis-server` and `php5-redis`. The installer will automatically launch `redis-server` and configure it to launch at startup.

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

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

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

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

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

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

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

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

```
'memcache.local' => '\OC\Memcache\Redis',
'redis' => array(
```

```
     'host' => '/var/run/redis/redis.sock',
     'port' => 0,
      ),
```

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

7.6.5 Cache Directory Location

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

7.6.6 Recommendations Based on Type of Deployment

Small/Private Home Server

Only use APCu:

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

Small Organization, Single-server Setup

Use APCu for local caching, Redis for file locking:

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

Large Organization, Clustered Setup

Use Redis for everything except local memcache:

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

7.6.7 Additional Redis Installation Help

If your version of Mint or Ubuntu does not package the required version of `php5-redis`, then try this Redis guide on Tech and Me for a complete Redis installation on Ubuntu 14.04 using PECL. These instructions are adaptable for any distro that does not package the supported version, or that does not package Redis at all, such as SUSE Linux Enterprise Server and Red Hat Enterprise Linux.

The Redis PHP module must be at least version 2.2.5. Please note that the Redis PHP module versions 2.2.5 - 2.2.7 will only work for:

```
PHP version 6.0.0 or older
PHP version 5.2.0 or newer
```

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

On Debian/Mint/Ubuntu, use `apt-cache` to see the available `php5-redis` version, or the version of your installed package:

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

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

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

7.7 Config.php Parameters

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`.

Note: The installer creates a configuration containing the essential parameters. Only manually add configuration parameters to `config/config.php` if you need to use a special value for a parameter. **Do not copy everything from ‘‘config/config.sample.php‘‘. Only enter the parameters you wish to modify!**

---

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`.

7.7.1 Default Parameters

These parameters are configured by the ownCloud installer, and are required for your ownCloud server to operate.

```
'instanceid' => '',
```

This is a unique identifier for your ownCloud installation, created automatically by the installer. This example is for documentation only, and you should never use it because it will not work. A valid `instanceid` is created when you install ownCloud.

‘instanceid’ => ‘d3c944a9a’,

```
'passwordsalt' => '',
```

The salt used to hash all passwords, 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.

```
'hashingCost' => 10,
```

The hashing cost used by hashes generated by ownCloud Using a higher value requires more time and CPU power to calculate the hashes

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

Your list of trusted domains that users can log into. Specifying trusted domains prevents host header poisoning. Do not remove this, as it performs necessary security checks.

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

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)

```
'version' => '',
```

The current version number of your ownCloud installation. This is set up during installation and update, so you shouldn't need to change it.

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

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

Available:

- sqlite (SQLite3 - Not in Enterprise Edition)
- mysql (MySQL/MariaDB)
- pgsql (PostgreSQL)
- oci (Oracle - Enterprise Edition Only)

```
'dbhost' => '',
```

Your host server name, for example `localhost`, `hostname`, `hostname.example.com`, or the IP address. To specify a port use `hostname:####`; to specify a Unix socket use `localhost:/path/to/socket`.

```
'dbname' => 'owncloud',
```

The name of the ownCloud database, which is set during installation. You should not need to change this.

```
'dbuser' => '',
```

The user that ownCloud uses to write to the database. This must be unique across ownCloud instances using the same SQL database. This is set up during installation, so you shouldn't need to change it.

```
'dbpassword' => '',
```

The password for the database user. This is set up during installation, so you shouldn't need to change it.

```
'dbtableprefix' => '',
```

Prefix for the ownCloud tables in the database.

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

Additional driver options for the database connection, eg. to enable SSL encryption in MySQL.

```
'sqlite.journal_mode' => 'DELETE',
```

sqlite3 journal mode can be specified using this config parameter - can be 'WAL' or 'DELETE' see for more details https://www.sqlite.org/wal.html

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

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

7.7.2 Default config.php Examples

When you use SQLite as your ownCloud database, your `config.php` looks like this after installation. The SQLite database is stored in your ownCloud `data/` directory. SQLite is a simple, lightweight embedded database that is good for testing and for simple installations, but for production ownCloud systems you should use MySQL, MariaDB, or PosgreSQL.

```
<?php
$CONFIG = array (
  'instanceid' => 'occ6f7365735',
  'passwordsalt' => '2c57784763467863O3',
  'trusted_domains' =>
  array (
    0 => 'localhost',
    1 => 'studio',
  ),
  'datadirectory' => '/var/www/owncloud/data',
  'dbtype' => 'sqlite3',
  'version' => '7.0.2.1',
  'installed' => true,
);
```

This example is from a new ownCloud installation using MariaDB:

```
<?php
$CONFIG = array (
  'instanceid' => 'oc8c0fd71e03',
  'passwordsalt' => '515a13302a6b3950a9d0fdb970191a',
  'trusted_domains' =>
  array (
    0 => 'localhost',
    1 => 'studio',
    2 => '192.168.10.155'
  ),
  'datadirectory' => '/var/www/owncloud/data',
  'dbtype' => 'mysql',
   'version' => '7.0.2.1',
  'dbname' => 'owncloud',
  'dbhost' => 'localhost',
  'dbtableprefix' => 'oc_',
  'dbuser' => 'oc_carla',
  'dbpassword' => '67336bcdf7630dd80b2b81a413d07',
  'installed' => true,
);
```

7.7.3 User Experience

These optional parameters control some aspects of the user interface. Default values, where present, are shown.

```
'default_language' => 'en',
```

This sets the default language on your ownCloud server, using ISO_639-1 language codes such as `en` for English, `de` for German, and `fr` for French. It overrides automatic language detection on public pages like login or shared items. User's language preferences configured under "personal -> language" override this setting after they have logged in.

```
'defaultapp' => 'files',
```

Set the default app to open on login. Use the app names as they appear in the URL after clicking them in the Apps menu, such as documents, calendar, and gallery. You can use a comma-separated list of app names, so if the first app is not enabled for a user then ownCloud will try the second one, and so on. If no enabled apps are found it defaults to the Files app.

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

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

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

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

```
'allow_user_to_change_display_name' => true,
```

`true` allows users to change their display names (on their Personal pages), and `false` prevents them from changing their display names.

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

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

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

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

```
'session_keepalive' => true,
```

Enable or disable session keep-alive when a user is logged in to the Web UI.

Enabling this sends a "heartbeat" to the server to keep it from timing out.

```
'skeletondirectory' => '/path/to/owncloud/core/skeleton',
```

The directory where the skeleton files are located. These files will be copied to the data directory of new users. Leave empty to not copy any skeleton files.

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

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

7.7.4 Mail Parameters

These configure the email settings for ownCloud notifications and password resets.

```
'mail_domain' => 'example.com',
```

The return address that you want to appear on emails sent by the ownCloud server, for example `oc-admin@example.com`, substituting your own domain, of course.

```
'mail_from_address' => 'owncloud',
```

FROM address that overrides the built-in `sharing-noreply` and `lostpassword-noreply` FROM addresses.

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

Enable SMTP class debugging.

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

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

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

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

For the `sendmail` option you need an installed and working email system on the server, with `/usr/sbin/sendmail` installed on your Unix system.

For `qmail` the binary is /var/qmail/bin/sendmail, and it must be installed on your Unix system.

```
'mail_smtphost' => '127.0.0.1',
```

This depends on `mail_smtpmode`. Specify the IP address of your mail server host. This may contain multiple hosts separated by a semi-colon. If you need to specify the port number append it to the IP address separated by a colon, like this: `127.0.0.1:24`.

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

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

```
'mail_smtptimeout' => 10,
```

This depends on `mail_smtpmode`. This sets the SMTP server timeout, in seconds. You may need to increase this if you are running an anti-malware or spam scanner.

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

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

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

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

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

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

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

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

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

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

7.7.5 Proxy Configurations

```
'overwritehost' => '',
```

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`.

```
'overwriteprotocol' => '',
```

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`.

```
'overwritewebroot' => '',
```

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.

```
'overwritecondaddr' => '',
```

This option allows you to define a manual override condition as a regular expression for the remote IP address. For example, defining a range of IP addresses starting with `10.0.0.` and ending with 1 to 3: `^10\.0\.0\.[1-3]$`

```
'overwrite.cli.url' => '',
```

Use this configuration parameter to specify the base URL for any URLs which are generated within ownCloud using any kind of command line tools (cron or occ). The value should contain the full base URL: `https://www.example.com/owncloud`

```
'proxy' => '',
```

The URL of your proxy server, for example `proxy.example.com:8081`.

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

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

The format is: `username:password`.

7.7.6 Deleted Items (trash bin)

These parameters control the Deleted files app.

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

When the trash bin app is enabled (default), this is the number of days a file will be kept in the trash bin. Default is 30 days.

```
'trashbin_auto_expire' => true,
```

Disable or enable auto-expiration for the trash bin. By default auto-expiration is enabled.

7.7.7 ownCloud Verifications

ownCloud performs several verification checks. There are two options, `true` and `false`.

```
'appcodechecker' => true,
```

Checks an app before install whether it uses private APIs instead of the proper public APIs. If this is set to true it will only allow to install or enable apps that pass this check.

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

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

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

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

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

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

```
'check_for_working_htaccess' => true,
```

This is a crucial security check on Apache servers that should always be set to `true`. This verifies that the `.htaccess` file is writable and works.

If it is not, then any options controlled by `.htaccess`, such as large file uploads, will not work. It also runs checks on the `data/` directory, which verifies that it can't be accessed directly through the web server.

```
'config_is_read_only' => false,
```

In certain environments it is desired to have a read-only config 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 config file writable again for the update process.

7.7.8 Logging

```
'log_type' => 'owncloud',
```

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.

```
'logfile' => 'owncloud.log',
```

Change the ownCloud logfile name from `owncloud.log` to something else.

```
'loglevel' => 2,
```

Loglevel to start logging at. Valid values are: 0 = Debug, 1 = Info, 2 = Warning, 3 = Error. The default value is Warning.

```
'logdateformat' => 'F d, Y H:i:s',
```

This uses PHP.date formatting; see http://php.net/manual/en/function.date.php

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

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

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

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

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

Log successful cron runs.

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

Enables log rotation and limits the total size of logfiles. The default is 0, or no rotation. Specify a size in bytes, for example 104857600 (100 megabytes = 100 * 1024 * 1024 bytes). A new logfile is created with a new name when the old logfile reaches your limit. The total size of all logfiles is double the `log_rotate_sizerotation` value.

7.7.9 Alternate Code Locations

Some of the ownCloud code may be stored in alternate locations.

```
'3rdpartyroot' => '',
```

ownCloud uses some 3rd party PHP components to provide certain functionality.

These components are shipped as part of the software package and reside in `owncloud/3rdparty`. Use this option to configure a different location.

```
'3rdpartyurl' => '',
```

If you have an alternate `3rdpartyroot`, you must also configure the URL as seen by a Web browser.

```
'customclient_desktop' =>
        'http://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',
```

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

7.7.10 Apps

Options for the Apps folder, Apps store, and App code checker.

```
'appstoreenabled' => true,
```

When enabled, admins may install apps from the ownCloud app store.

```
'appstoreurl' => 'https://api.owncloud.com/v1',
```

The URL of the appstore to use.

```
'apps_paths' => array(
        array(
                'path'=> '/var/www/owncloud/apps',
                'url' => '/apps',
                'writable' => true,
        ),
),
```

Use the `apps_paths` parameter to set the location of the Apps directory, which should be scanned for available apps, and where user-specific apps should be installed from the Apps store. The `path` defines the absolute file system path to the app folder. The key `url` defines the HTTP web path to that folder, starting from the ownCloud web root. The key `writable` indicates if a web server can write files to that folder.

```
'appcodechecker' => true,
```

Checks an app before install whether it uses private APIs instead of the proper public APIs. If this is set to true it will only allow to install or enable apps that pass this check.

7.7.11 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_previews' => true,
```

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

```
'preview_max_x' => null,
```

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

```
'preview_max_y' => null,
```

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

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

If a lot of small pictures are stored on the 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.

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

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

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

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

custom path for LibreOffice/OpenOffice binary

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

Use this if LibreOffice/OpenOffice requires additional arguments.

```
'enabledPreviewProviders' => array(
        'OC\Preview\Image',
        'OC\Preview\MP3',
        'OC\Preview\TXT',
        'OC\Preview\MarkDown'
),
```

Only register providers that have been explicitly enabled

The following providers are enabled by default:

- OC\Preview\Image
- 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

Note: Troubleshooting steps for the MS Word previews are available at the *Configuring the Collaborative Documents App* section of the Administrators Manual.

The following providers are not available in Microsoft Windows:

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

- OC\Preview\MSOffice2007
- OC\Preview\OpenDocument
- OC\Preview\StarOffice

7.7.12 LDAP

Global settings used by LDAP User and Group Backend

```
'ldapUserCleanupInterval' => 51,
```

defines the interval in minutes for the background job that checks user existence and marks them as ready to be cleaned up. The number is always minutes. Setting it to 0 disables the feature.

See command line (occ) methods ldap:show-remnants and user:delete

7.7.13 Maintenance

These options are for halting user activity when you are performing server maintenance.

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

Enable maintenance mode to disable ownCloud

If you want to prevent users to login 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.

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

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

7.7.14 SSL

```
'forcessl' => false,
```

Change this to `true` to require HTTPS for all connections, and to reject HTTP requests.

```
'forceSSLforSubdomains' => false,
```

Change this to `true` to require HTTPS connections also for all subdomains.

Works only together when *forcessl* is set to true.

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

Extra SSL options to be used for configuration.

7.7.15 Memory caching backend configuration

```
'redis' => array(
        'host' => 'localhost', // can also be a unix domain socket: '/tmp/redis.sock'
        'port' => 6379,
        'timeout' => 0.0
),
```

Connection details for redis to use for memory caching.

Redis is only used if other memory cache options (xcache, apc, apcu) are not available.

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

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

Memcache is only used if other memory cache options (xcache, apc, apcu, redis) are not available.

```
'cache_path' => '',
```

Location of the cache folder, defaults to `data/$user/cache` where `$user` is the current user. When specified, the format will change to `$cache_path/$user` where `$cache_path` is the configured cache directory and `$user` is the user.

7.7.16 Using Object Store with ownCloud

```
'objectstore' => array(
        'class' => 'OC\\Files\\ObjectStore\\Swift',
        'arguments' => array(
                // trystack will user your facebook id as the user name
                'username' => 'facebook100000123456789',
                // in the trystack dashboard go to user -> settings -> API Password to
                // generate a password
                'password' => 'Secr3tPaSSWoRdt7',
                // must already exist in the objectstore, name can be different
                'container' => 'owncloud',
                // create the container if it does not exist. default is false
                'autocreate' => true,
                // required, dev-/trystack defaults to 'RegionOne'
                'region' => 'RegionOne',
                // The Identity / Keystone endpoint
                'url' => 'http://8.21.28.222:5000/v2.0',
                // required on dev-/trystack
                'tenantName' => 'facebook100000123456789',
                // dev-/trystack uses swift by default, the lib defaults to 'cloudFiles'
                // if omitted
                'serviceName' => 'swift',
        ),
),
```

This example shows how to configure ownCloud to store all files in a swift object storage.

It is important to note that ownCloud in object store mode will expect exclusive access to the object store container because it only stores the binary data for each file. The metadata is currently kept in the local database for performance reasons.

WARNING: The current implementation is incompatible with any app that uses direct file IO and circumvents our virtual filesystem. That includes Encryption and Gallery. Gallery will store thumbnails directly in the filesystem and encryption will cause severe overhead because key files need to be fetched in addition to any requested file.

One way to test is applying for a trystack account at http://trystack.org/

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

Database types that are supported for installation.

Available:

- sqlite (SQLite3 - Not in Enterprise Edition)
- mysql (MySQL)
- pgsql (PostgreSQL)
- oci (Oracle - Enterprise Edition Only)
- mssql (Microsoft SQL Server - Enterprise Edition Only)

```
'custom_csp_policy' =>
        "default-src 'self'; script-src 'self' 'unsafe-eval'; ".
        "style-src 'self' 'unsafe-inline'; frame-src *; img-src *; ".
        "font-src 'self' data:; media-src *; connect-src *",
```

Custom CSP policy, changing this will overwrite the standard policy

7.7.17 All other config options

```
'blacklisted_files' => array('.htaccess'),
```

Blacklist a specific file or files and disallow the upload of files with this name. `.htaccess` is blocked by default.

WARNING: USE THIS ONLY IF YOU KNOW WHAT YOU ARE DOING.

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

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

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

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

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

```
'xframe_restriction' => true,
```

X-Frame-Restriction is a header which prevents browsers from showing the site inside an iframe. This is be used to prevent clickjacking. It is risky to disable this, so leave it set at `true`.

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

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

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

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

```
'filesystem_check_changes' => 1,
```

Specifies how often the filesystem is checked for changes made outside ownCloud.

0 -> Never check the filesystem for outside changes, provides a performance increase when it's certain that no changes are made directly to the filesystem

1 -> Check each file or folder at most once per request, recommended for general use if outside changes might happen.

2 -> Check every time the filesystem is used, causes a performance hit when using external storages, not recommended for regular use.

```
'asset-pipeline.enabled' => false,
```

All css and js files will be served by the web server statically in one js file and one css file if this is set to `true`. This improves performance.

```
'assetdirectory' => '/var/www/owncloud',
```

The parent of the directory where css and js assets will be stored if pipelining is enabled; this defaults to the ownCloud directory. The assets will be stored in a subdirectory of this directory named 'assets'. The server *must* be configured to serve that directory as $WEBROOT/assets.

You will only likely need to change this if the main ownCloud directory is not writeable by the web server in your configuration.

```
'mount_file' => 'data/mount.json',
```

Where `mount.json` file should be stored, defaults to `data/mount.json`

```
'filesystem_cache_readonly' => false,
```

When `true`, prevent ownCloud from changing the cache due to changes in the filesystem for all storage.

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

Secret used by ownCloud for various purposes, e.g. to encrypt data. If you lose this string there will be data corruption.

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

List of trusted proxy servers

```
'forwarded_for_headers' => array('HTTP_X_FORWARDED', 'HTTP_FORWARDED_FOR'),
```

Headers that should be trusted as client IP address in combination with *trusted_proxies*

```
'max_filesize_animated_gifs_public_sharing' => 10,
```

max file size for animating gifs on public-sharing-site.

If the gif is bigger, it'll show a static preview

Value represents the maximum filesize in megabytes. Default is 10. Set to -1 for no limit.

```
'copied_sample_config' => true,
```

This entry is just here to show a warning in case somebody copied the sample configuration. DO NOT ADD THIS SWITCH TO YOUR CONFIGURATION!

If you, brave person, have read until here be aware that you should not modify *ANY* settings in this file without reading the documentation.

7.7.18 App config options

Retention for activities of the activity app:

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

Every day a cron job is ran, which deletes all activities for all users which are older then the number of days that is set for `activity_expire_days`

7.8 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
```

7.9 Email Configuration

ownCloud is capable of sending password reset emails, notifying users of new file shares, changes in files, and activity notifications. Your users configure which notifications they want to receive on their Personal pages.

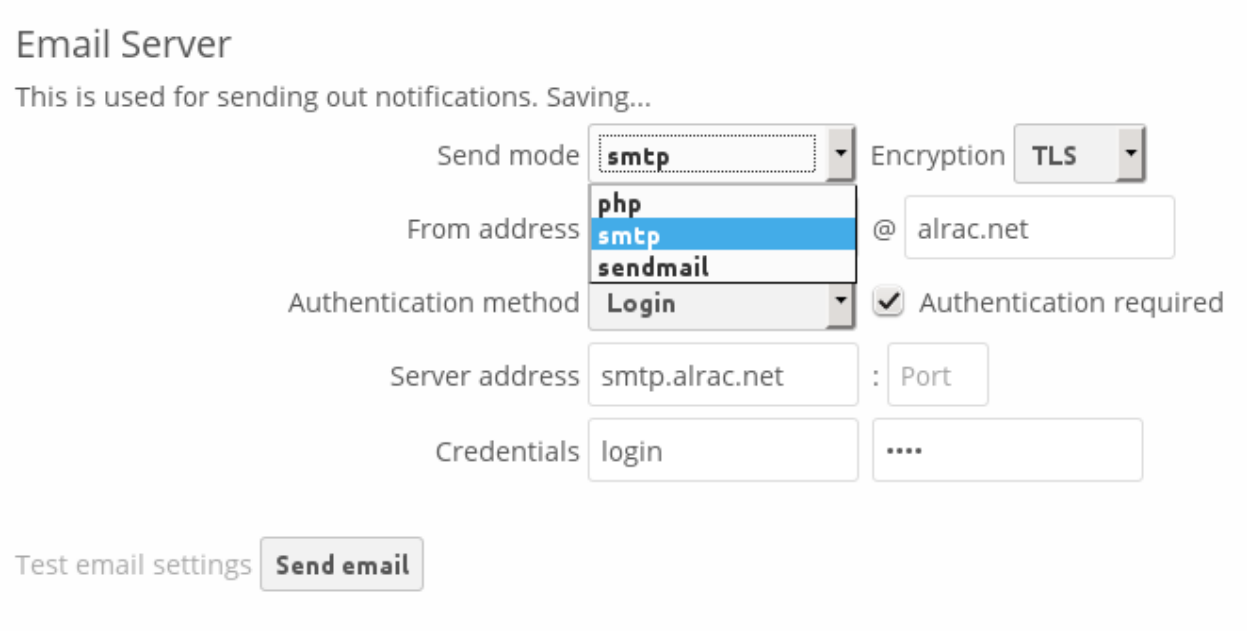
ownCloud does not contain a full email server, but rather connects to your existing mail server. You must have a functioning mail server for ownCloud to be able to send emails. You may have a mail server on the same machine as ownCloud, or it may be a remote server.

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

With the new wizard, connecting ownCloud to your mail server is fast and easy. The wizard fills in the values in `config/config.php`, so you may use either or both as you prefer.

The ownCloud Email wizard supports three types of mail server connections: SMTP, PHP, and Sendmail. Use the SMTP configurator for a remote server, and PHP or Sendmail when your mail server is on the same machine as ownCloud.

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



7.9.1 Configuring an SMTP Server

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

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

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

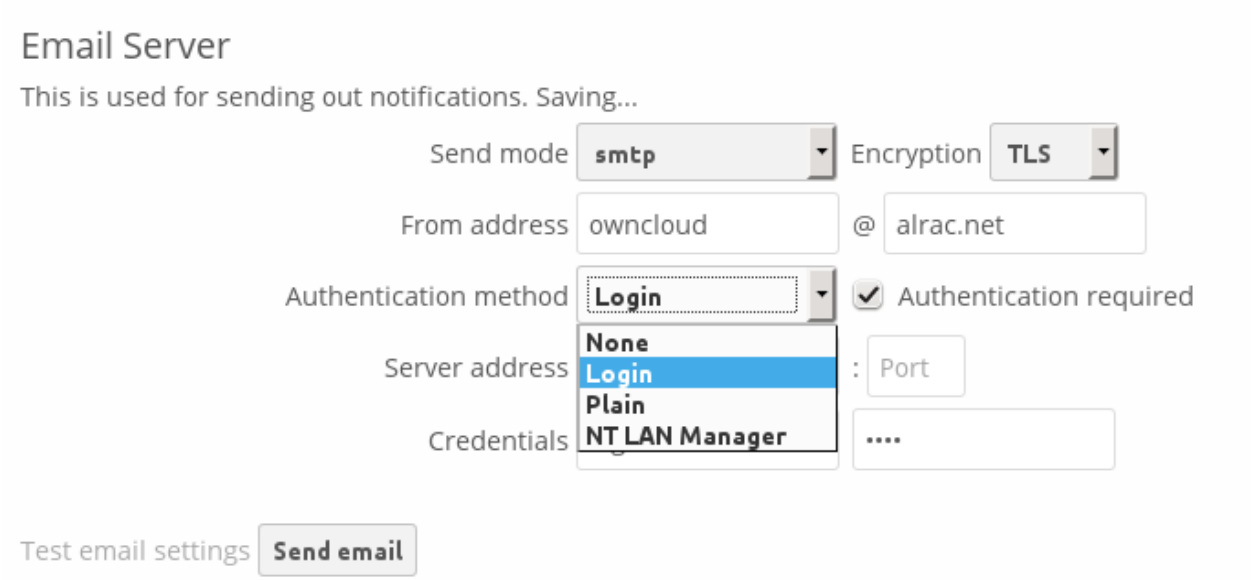
```
If you received this email, the settings seem to be correct.

--
ownCloud
web services under your control
```

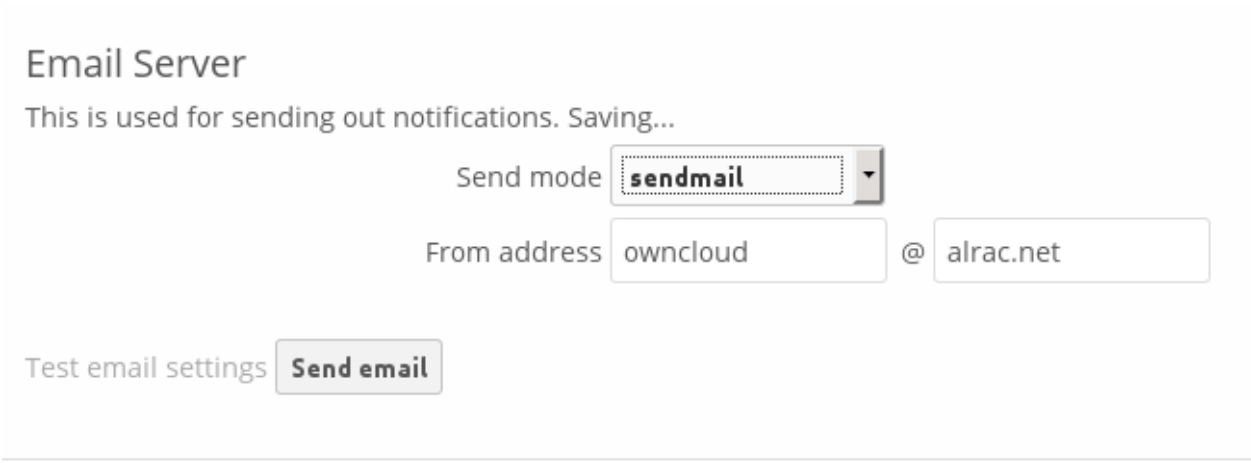
7.9.2 Configuring PHP and Sendmail

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.


Email Server
This is used for sending out notifications. Saving...
Send mode
smtp
Encryption
TLS
From address
owncloud
@
alrac.net
Authentication method
Login
Authentication required
None
Login
Plain
NT LAN Manager
Server address
:
Port
Credentials
••••
Test email settings
Send email


Email Server
This is used for sending out notifications. Saving...
Send mode
sendmail
From address
owncloud
@
alrac.net
Test email settings
Send email

7.9.3 Using Email Templates

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

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

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

To modify an email template to users:

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

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

```
<?php

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

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

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

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

7.9.4 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

A secure connection will be initiated using the outdated SMTPS protocol which uses the port 465/tcp:

```
<?php

  "mail_smtphost"     => "smtp.server.dom:465",
  "mail_smtpsecure"   => 'ssl',
```

TLS

A secure connection will be initiated using the STARTTLS protocol which uses the default port 25/tcp:

```
<?php

  "mail_smtphost"     => "smtp.server.dom",
  "mail_smtpsecure"   => 'tls',
```

And finally it is necessary to configure if the SMTP server requires authentication, if not, the default values can be taken as is.

```
<?php

  "mail_smtpauth"     => false,
  "mail_smtpname"     => "",
  "mail_smtppassword" => "",
```

If SMTP authentication is required you have to set the required username and password and can optionally choose between the authentication types **LOGIN** (default) or **PLAIN**.

```
<?php

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

PHP mail

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

7.9.5 Send a Test Email

To test your email configuration, save your email address in your personal settings and then use the **Send email** button in the *Email Server* section of the Admin settings page.

7.9.6 Troubleshooting

If you are unable to send email, try turning on debugging. Do this by enabling the `mail_smtpdebug parameter` in `config/config.php`.

```
<?php

  "mail_smtpdebug" => true;
```

Note: Immediately after pressing the **Send email** button, as described before, several **SMTP -> get_lines(): ...** messages appear on the screen. This is expected behavior and can be ignored.

Question: Why is my web domain different from my mail domain?

Answer: The default domain name used for the sender address is the hostname where your 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
```

Question: How can I find out if an SMTP server is reachable?

Answer: Use the ping command to check the server availability:

```
ping smtp.server.dom

PING smtp.server.dom (ip-address) 56(84) bytes of data.
64 bytes from your-server.local.lan (192.168.1.10): icmp_req=1 ttl=64
time=3.64ms
```

Question: How can I find out if the SMTP server is listening on a specific TCP port?

Answer: The best way to get mail server information is to ask your mail server admin. If you are the mail server admin, or need information in a hurry, you can use the `netstat` command. This example shows all active servers on your system, and the ports they are listening on. The SMTP server is listening on localhost port 25.

```
# netstat -pant

Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address   Foreign Address  State  ID/Program name
tcp    0      0    0.0.0.0:631     0.0.0.0:*        LISTEN   4418/cupsd
tcp    0      0    127.0.0.1:25    0.0.0.0:*        LISTEN   2245/exim4
tcp    0      0    127.0.0.1:3306  0.0.0.0:*        LISTEN   1524/mysqld
```

- 25/tcp is unencrypted smtp
- 110/tcp/udp is unencrypted pop3
- 143/tcp/udp is unencrypted imap4
- 465/tcp is encrypted ssmtp
- 993/tcp/udp is encrypted imaps
- 995/tcp/udp is encrypted pop3s

Question: How can I determine if the SMTP server supports the outdated SMTPS protocol?

Answer: A good indication that the SMTP server supports the SMTPS protocol is that it is listening on port **465**.

Question: How can I determine what authorization and encryption protocols the mail server supports?

Answer: SMTP servers usually announce the availability of STARTTLS immediately after a connection has been established. You can easily check this using the `telnet` command.

Note: You must enter the marked lines to obtain the information displayed.

```
telnet smtp.domain.dom 25

Trying 192.168.1.10...
Connected to smtp.domain.dom.
Escape character is '^]'.
220 smtp.domain.dom ESMTP Exim 4.80.1 Tue, 22 Jan 2013 22:39:55 +0100
EHLO your-server.local.lan                  # <<< enter this command
250-smtp.domain.dom Hello your-server.local.lan [ip-address]
250-SIZE 52428800
250-8BITMIME
250-PIPELINING
250-AUTH PLAIN LOGIN CRAM-MD5               # <<< Supported auth protocols
250-STARTTLS                                # <<< Encryption is supported
250 HELP
QUIT                                        # <<< enter this command
221 smtp.domain.dom closing connection
Connection closed by foreign host.
```

7.9.7 Enabling Debug Mode

If you are unable to send email, it might be useful to activate further debug messages by enabling the mail_smtpdebug parameter:

```
<?php

  "mail_smtpdebug" => true,
```

Note: Immediately after pressing the **Send email** button, as described before, several **SMTP -> get_lines(): ...**

messages appear on the screen. This is expected behavior and can be ignored.

7.10 Linking External Sites

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

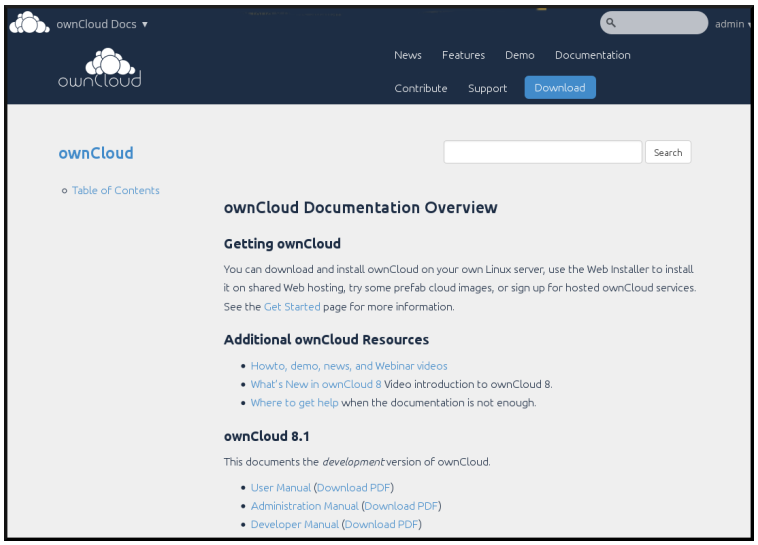


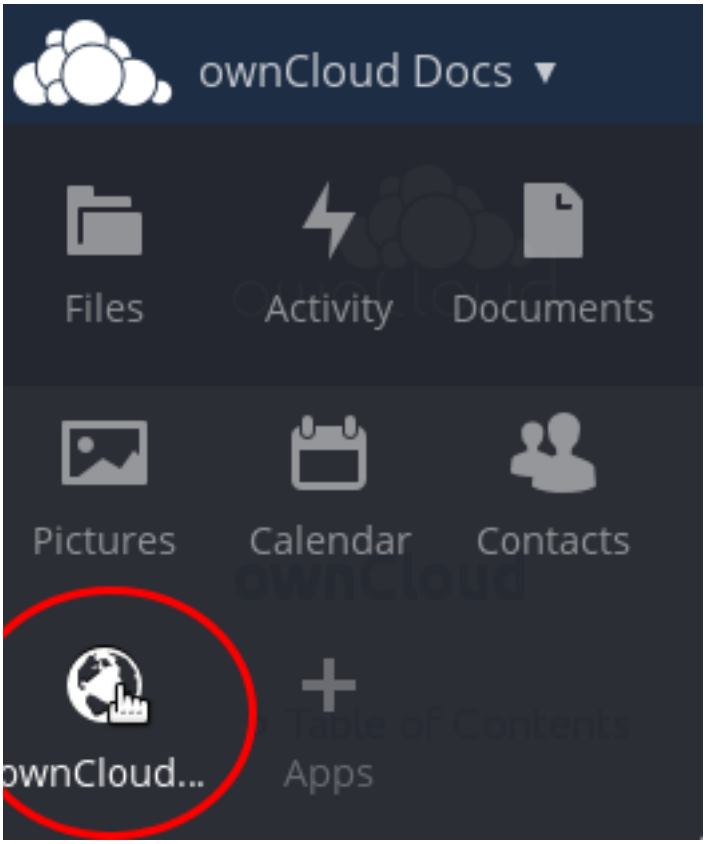
Figure 7.1: *Click to enlarge*

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.

Figure 7.2: *Click to enlarge*

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.

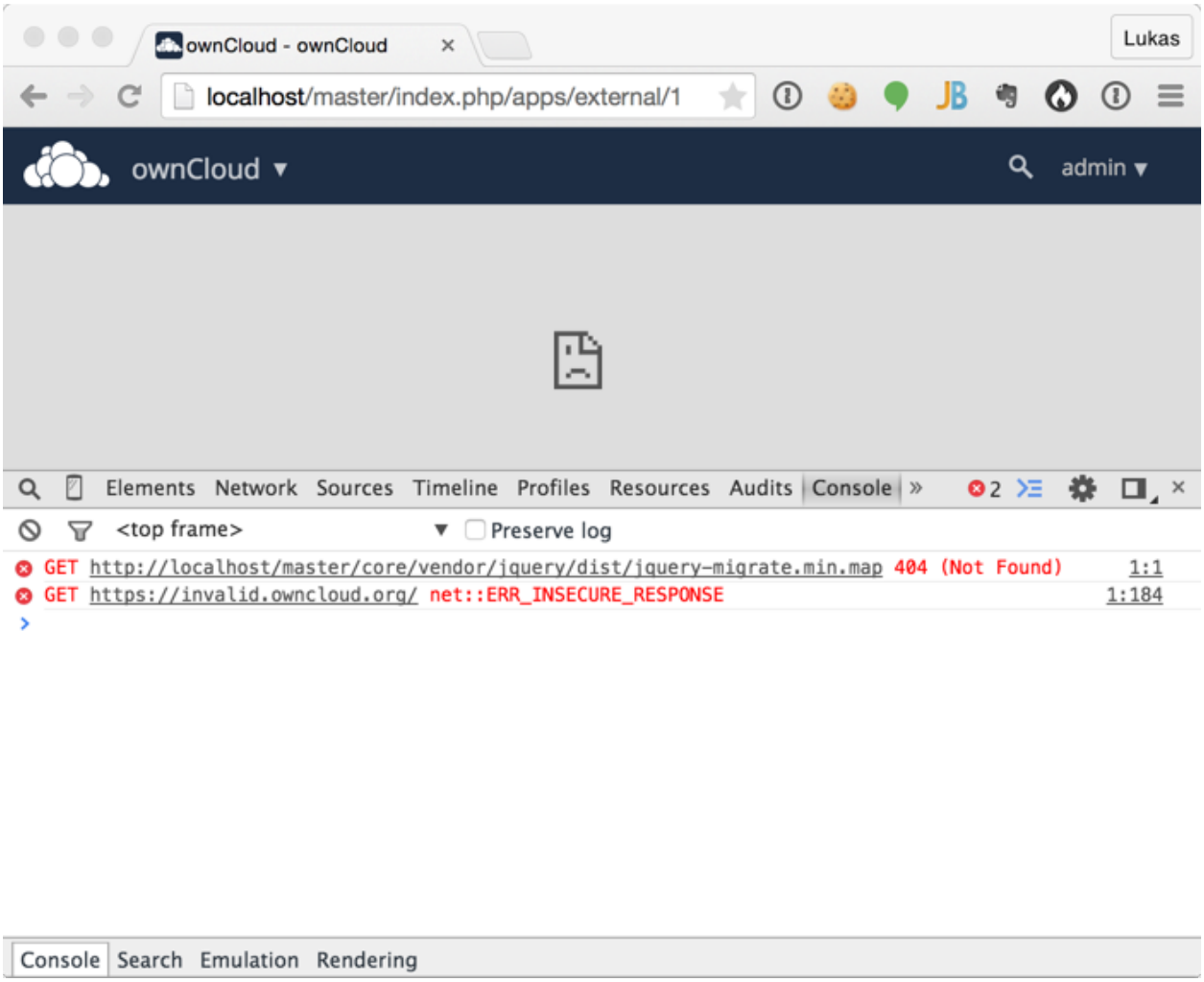
There isn't much you can do about these issues, but if you're curious you can see what is happening.

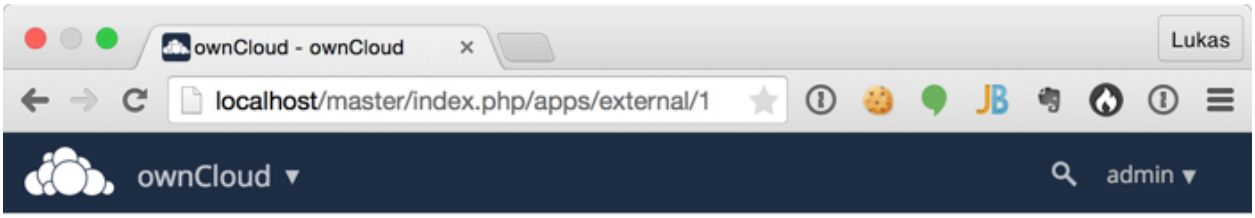
7.11 JavaScript and CSS Asset Management

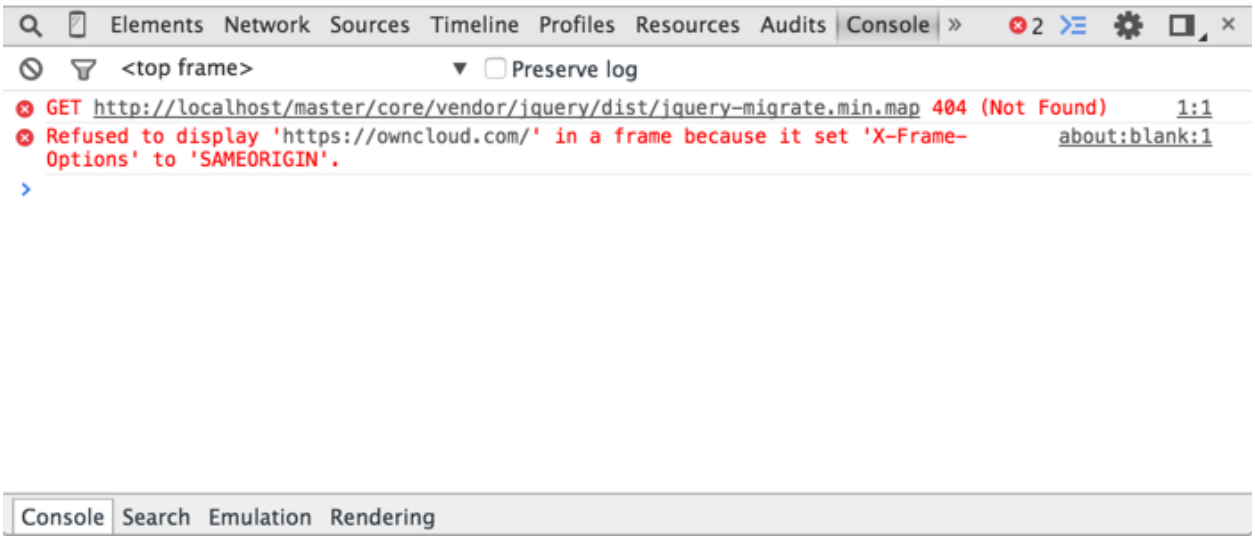
In production environments, JavaScript and CSS files should be delivered in a concatenated and compressed format.

ownCloud can automatically collect all JavaScript and CSS files, aggregate and compress them to then save the result in a folder called 'assets' which can be found in the folder where ownCloud has been installed.

If your web server has write access to your ownCloud installation, then the 'assets' folder will be automatically created for you, otherwise, you need to create it yourself before enabling that option and you must give write access to your web server user.


ownCloud - ownCloud
Lukas
localhost/master/index.php/apps/external/1
ownCloud
admin
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame>
Preserve log
GET http://localhost/master/core/vendor/jquery/dist/jquery-migrate.min.map 404 (Not Found) 1:1
GET https://invalid.owncloud.org/ net::ERR_INSECURE_RESPONSE 1:184
Console Search Emulation Rendering


ownCloud - ownCloud
Lukas
localhost/master/index.php/apps/external/1
ownCloud
admin


Elements Network Sources Timeline Profiles Resources Audits Console
<top frame>
Preserve log
GET http://localhost/master/core/vendor/jquery/dist/jquery-migrate.min.map 404 (Not Found) 1:1
Refused to display 'https://owncloud.com/' in a frame because it set 'X-Frame-Options' to 'SAMEORIGIN'. about:blank:1
Console Search Emulation Rendering

Assets found in that folder will from now on be served as static files by your web server and will be automatically updated whenever a change is detected.

7.11.1 Parameters

```
<?php
  $CONFIG = array (
    ...
    'asset-pipeline.enabled' => true,
    ...
  );
```

You can set this parameter in the `config/config.php`

7.12 Knowledge Base Configuration

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

7.12.1 Parameters

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

```
<?php

  "knowledgebaseenabled" => true,
```

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

7.13 Language Configuration

In normal cases ownCloud will automatically detect the language of the Web-GUI. If this does not work properly or you want to make sure that ownCloud always starts with a given language, you can use the **default_language** parameter.

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

Please check `settings/languageCodes.php` for the list of supported language codes.

7.13.1 Parameters

```
<?php

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

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

7.14 Logging Configuration

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.

7.14.1 Parameters

Logging levels range from **DEBUG**, which logs all activity, to **FATAL**, which logs only fatal errors.

- **0**: DEBUG: All activity; the most detailed logging.
- **1**: INFO: Activity such as user logins and file activities, plus warnings, errors, and fatal errors.
- **2**: WARN: Operations succeed, but with warnings of potential problems, plus errors and fatal errors.
- **3**: ERROR: An operation fails, but other services and operations continue, plus fatal errors.
- **4**: FATAL: The server stops.

By default the log level is set to **2** (WARN). Use **DEBUG** when you have a problem to diagnose, and then reset your log level to a less-verbose level as **DEBUG** outputs a lot of information, and can affect your server performance.

Logging level parameters are set in the `config/config.php` file, or on the Admin page of your 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",
```

7.15 Hardening and Security Guidance

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.

Note: 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.

7.15.1 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.

7.15.2 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.

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

Enable hardening modules such as SELinux

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

7.15.3 Deployment

Place data directory outside of the web root

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

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.

7.15.4 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 a setting such as the following in the Apache VirtualHosts config:

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

Enable HTTP Strict Transport Security

While redirecting all traffic to HTTPS is good, it may not completely prevent man-in-the-middle attacks. Thus administrators are encouraged to set the HTTP Strict Transport Security header, which instructs browsers to not allow any connection to the 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:

```
<VirtualHost *:443>
  ServerName cloud.owncloud.com
    <IfModule mod_headers.c>
      Header always set Strict-Transport-Security "max-age=15768000; includeSubDomains; preload"
    </IfModule>
 </VirtualHost>
```

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.

7.15.5 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.

7.15.6 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 Click-jacking 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.

7.16 Reverse Proxy Configuration

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.

7.16.1 Parameters

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.

7.16.2 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`.

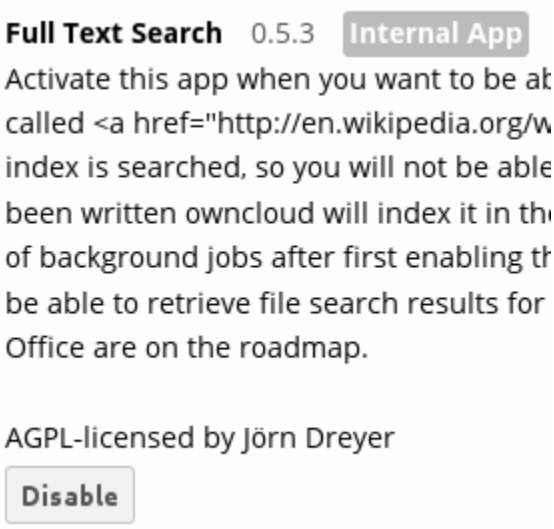
```
<?php
$CONFIG = array (
  "overwritehost"     => "ssl-proxy.tld",
  "overwriteprotocol" => "https",
  "overwritewebroot"  => "/domain.tld/owncloud",
  "overwritecondaddr" => "^10\.0\.0\.1$",
);
```

Note: If you want to use the SSL proxy during installation you have to create the `config/config.php` otherwise you have to extend the existing **$CONFIG** array.

7.17 Enabling Full-Text Search

The Full-Text Search app indexes plain text, `.docx`, `.xlsx`, `.pptx`, `.odt`, `.ods` and `.pdf` files stored in ownCloud. It is based on Zend Search Lucene, which is a good general purpose text search engine written in PHP 5. The Zend Lucene index is stored on the filesystem (in `owncloud/data/$user/lucene_index`) and does not require a database server.

Using the Full-Text Search app is literally set-it-and-forget-it: all you do is enable it on your Apps page, and then it automatically indexes all documents on your ownCloud server. It does not index files on remote storage services or devices.

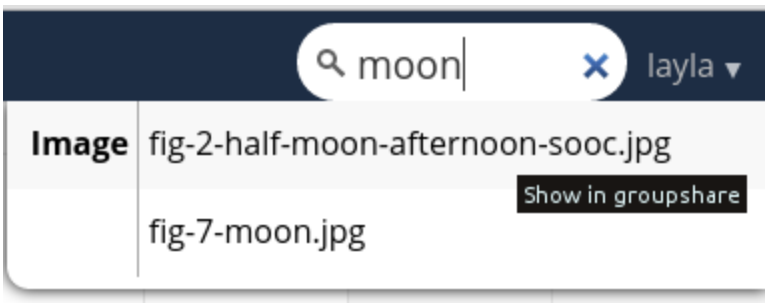


When you want to find a document, enter your search term in the search field at the upper right of your ownCloud Web interface. You can run a search from any ownCloud page. Hover your cursor over any of your search results to see what folder it is in, or click on the filename and it takes you to the folder.

Known limitations

It does not work with the Encryption app, because the background indexing process does not have access to the key needed to decrypt files when the user is not logged in.

Not all PDF versions can be indexed because its text extraction may be incompatible with newer PDF versions.



7.18 Warnings on Admin Page

Your ownCloud server has a built-in security and 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.

7.18.1 Cache Warnings

"No memory cache has been configured. To enhance your performance please configure a memcache if available." ownCloud supports multiple php caching extentions:

- APC (PHP 5.4 only)
- APCu (PHP 5.5+, minimum required PHP extension version 4.0.6)
- Memcached
- Redis (minimum required php extension version: 2.2.5)

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

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

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

7.18.2 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 manual pages describe how to enable HTTPS on the Apache and Nginx Web servers.

Enabling SSL (on Apache)

Use HTTPS

Nginx Configuration

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

Some environments are not passing a valid PATH variable to ownCloud. The *php-fpm Configuration Notes* provides the information about how to configure your environment.

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

"The "Strict-Transport-Security" HTTP header is not configured to least "15768000" seconds. For enhanced security we recommend enabling HSTS as described in our security tips."

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

7.18.5 /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.

7.18.6 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.

7.18.7 Outdated NSS / OpenSSL version

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

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

There are known bugs in older OpenSSL and NSS versions leading to misbehaviour in combination with remote hosts using SNI. A technology used by most of the HTTPS websites. To ensure that 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.

7.19 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.

7.19.1 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",
```

7.20 Server Tuning & Performance Tips

7.20.1 ownCloud Server Tuning

Serving static files via web server

See *Serving Static Files for Better Performance* for a description and the benefits.

Using cron to perform background jobs

See *Defining Background Jobs* for a description and the benefits.

Enable JavaScript and CSS Asset Management

See *JavaScript and CSS Asset Management* for a description and the benefits.

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 *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 *Converting Database Type*.

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.

CHAPTER

EIGHT

DATABASE CONFIGURATION

8.1 Converting Database Type

You can convert a SQLite database to a more performing MySQL, MariaDB or PostgreSQL database with the ownCloud command line tool. SQLite is good for testing and simple single-user ownCloud servers, but it does not scale for multiple-user production users.

Note: ownCloud Enterprise Subscription does not support SQLite.

8.1.1 Run the conversion

First setup the new database, here called "new_db_name". In ownCloud root folder call

```
php occ db:convert-type [options] type username hostname database
```

The Options

- `--port="3306"` the database port (optional)
- `--password="mysql_user_password"` password for the new database. If omitted the tool will ask you (optional)
- `--clear-schema` clear schema (optional)
- `--all-apps` by default, tables for enabled apps are converted, use to convert also tables of deactivated apps (optional)

Note: The converter searches for apps in your configured app folders and uses the schema definitions in the apps to create the new table. So tables of removed apps will not be converted even with option `--all-apps`

For example

```
php occ db:convert-type --all-apps mysql oc_mysql_user 127.0.0.1 new_db_name
```

To successfully proceed with the conversion, you must type `yes` when prompted with the question `Continue with the conversion?`

On success the converter will automatically configure the new database in your ownCloud config `config.php`.

8.1.2 Unconvertible Tables

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

```
The following tables will not be converted:
oc_permissions
...
```

You can ignore these tables. Here is a list of known old tables:

- oc_calendar_calendars
- oc_calendar_objects
- oc_calendar_share_calendar
- oc_calendar_share_event
- oc_fscache
- oc_log
- oc_media_albums
- oc_media_artists
- oc_media_sessions
- oc_media_songs
- oc_media_users
- oc_permissions
- oc_queuedtasks
- oc_sharing

8.2 Database Configuration

ownCloud requires a database in which administrative data is stored. The following databases are currently supported:

- MySQL / MariaDB
- PostgreSQL
- Oracle

The MySQL or MariaDB databases are the recommended database engines.

8.2.1 Requirements

Choosing to use MySQL / MariaDB, PostgreSQL, or Oracle (ownCloud Enterprise Subscription only) as your database requires that you install and set up the server software first. (Oracle users, see *Oracle Database Setup*.)

Note: The steps for configuring a third party database are beyond the scope of this document. Please refer to the documentation for your specific database choice for instructions.

MySQL / MariaDB with Binary Logging Enabled

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. See Overview of the Binary Log and The Binary Log for detailed information.

8.2.2 Parameters

For setting up ownCloud to use any database, use the instructions in *Installation Wizard*. You should not have to edit the respective values in the `config/config.php`. However, in special cases (for example, if you want to connect your ownCloud instance to a database created by a previous installation of ownCloud), some modification might be required.

Configuring a MySQL or MariaDB Database

If you decide to use a MySQL or MariaDB database, ensure the following:

- That you have installed and enabled the MySQL extension in PHP
- That the **mysql.default_socket** points to the correct socket (if the database runs on the same server as ownCloud).

Note: MariaDB is backwards compatible with MySQL. All instructions work for both. You will not need to replace mysql with anything.

The PHP configuration in `/etc/php5/conf.d/mysql.ini` could look like this:

```
# configuration for PHP MySQL module
extension=pdo_mysql.so
extension=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 login for the first time.

To start the MySQL command line mode use:

```
mysql -uroot -p
```

Then a **mysql>** or **MariaDB [root]>** prompt will appear. Now enter the following lines and confirm them with the enter key:

```
CREATE USER 'username'@'localhost' IDENTIFIED BY 'password';
CREATE DATABASE IF NOT EXISTS 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_",
```

PostgreSQL Database

If you decide to use a PostgreSQL database make sure that you have installed and enabled the PostgreSQL extension in PHP. The PHP configuration in `/etc/php5/conf.d/pgsql.ini` could look like this:

```
# configuration for PHP PostgreSQL module
extension=pdo_pgsql.so
extension=pgsql.so

[PostgresSQL]
pgsql.allow_persistent = On
pgsql.auto_reset_persistent = Off
pgsql.max_persistent = -1
pgsql.max_links = -1
pgsql.ignore_notice = 0
pgsql.log_notice = 0
```

The default configuration for PostgreSQL (at least in Ubuntu 14.04) is to use the peer authentication method. Check `/etc/postgresql/9.3/main/pg_hba.conf` to find out which authentication method is used in your setup. To start the postgres command line mode use:

```
sudo -u postgres psql -d template1
```

Then a **template1=#** prompt will appear. Now enter the following lines and confirm them with the enter key:

```
CREATE USER username CREATEDB;
CREATE DATABASE owncloud OWNER username;
```

You can quit the prompt by entering:

```
\q
```

An ownCloud instance configured with PostgreSQL would contain the path to the socket on which the database is running as the hostname, the system username the php process is using, and an empty password to access it, and the name of the database. The `config/config.php` as created by the *Installation Wizard* would therefore contain entries like this:

```
<?php

  "dbtype"        => "pgsql",
  "dbname"        => "owncloud",
  "dbuser"        => "username",
  "dbpassword"    => "",
  "dbhost"        => "/var/run/postgresql",
  "dbtableprefix" => "oc_",
```

Note: The host actually points to the socket that is used to connect to the database. Using localhost here will not work if postgreSQL is configured to use peer authentication. Also note, that no password is specified, because this authentication method doesn't use a password.

If you use another authentication method (not peer), you'll need to use the following steps to get the database setup: Now you need to create a database user and the database itself by using the PostgreSQL command line interface. The database tables will be created by ownCloud when you login for the first time.

To start the postgres command line mode use:

```
psql -hlocalhost -Upostgres
```

Then a **postgres=#** prompt will appear. Now enter the following lines and confirm them with the enter key:

```
CREATE USER username WITH PASSWORD 'password';
CREATE DATABASE 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 therefore contain entries like this:

```
<?php

  "dbtype"        => "pgsql",
  "dbname"        => "owncloud",
  "dbuser"        => "username",
  "dbpassword"    => "password",
  "dbhost"        => "localhost",
  "dbtableprefix" => "oc_",
```

8.2.3 Troubleshooting

How can I find out if my MySQL/PostgreSQL server is reachable?

To check the server's network availability, use the ping command on the server's host name (db.server.com in this example):

```
ping db.server.dom

PING db.server.dom (ip-address) 56(84) bytes of data.
64 bytes from your-server.local.lan (192.168.1.10): icmp_req=1 ttl=64 time=3.64 ms
64 bytes from your-server.local.lan (192.168.1.10): icmp_req=2 ttl=64 time=0.055 ms
64 bytes from your-server.local.lan (192.168.1.10): icmp_req=3 ttl=64 time=0.062 ms
```

For a more detailed check whether the access to the database server software itself works correctly, see the next question.

How can I find out if a created user can access a database?

The easiest way to test if a database can be accessed is by starting the command line interface:

MySQL:

Assuming the database server is installed on the same system you're running the command from, use:

```
mysql -uUSERNAME -p
```

To acess a MySQL installation on a different machine, add the -h option with the respective host name:

```
mysql -uUSERNAME -p -h HOSTNAME

mysql> SHOW VARIABLES LIKE "version";
+---------------+--------+
| Variable_name | Value  |
+---------------+--------+
| version       | 5.1.67 |
+---------------+--------+
1 row in set (0.00 sec)
mysql> quit
```

PostgreSQL:

Assuming the database server is installed on the same system you're running the command from, use:

```
psql -Uusername -downcloud
```

To acess a MySQL installation on a different machine, add the -h option with the respective host name:

```
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     : SELECT User,Host FROM mysql.user;
PostgreSQL: SELECT * FROM pg_user;
```

Show available Databases:

```
MySQL     : SHOW DATABASES;
PostgreSQL: \l
```

Show ownCloud Tables in Database:

```
MySQL     : USE owncloud; SHOW TABLES;
PostgreSQL: \c owncloud; \d
```

Quit Database:

```
MySQL     : quit
PostgreSQL: \q
```

CHAPTER

NINE

MAINTENANCE

9.1 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:

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

You may also put your server into this mode by editing `config/config.php`. Change `"maintenance" => false` to `"maintenance" => true`:

```
<?php

 "maintenance" => true,
```

Then change it back to `false` when you are finished.

9.2 Backing up ownCloud

To backup an ownCloud installation there are three main things you need to retain:

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

9.2.1 Backup Folders

Simply copy your config and data folder (or even your whole ownCloud install and data folder) to a place outside of your ownCloud environment. You could use this command:

```
rsync -Aax owncloud/ owncloud-dirbkp_`date +"%Y%m%d"`/
```

9.2.2 Backup Database

MySQL/MariaDB

MySQL or MariaDB, which is a drop-in MySQL replacement, is the recommended database engine. To backup MySQL/MariaDB:

```
mysqldump --lock-tables -h [server] -u [username] -p[password] [db_name] > owncloud-sqlbkp_`date +"%Y
```

SQLite

```
sqlite3 data/owncloud.db .dump > owncloud-sqlbkp_`date +"%Y%m%d"`.bak
```

PostgreSQL

```
PGPASSWORD="password" pg_dump [db_name] -h [server] -U [username] -f owncloud-sqlbkp_`date +"%Y%m%d"`
```

9.3 Upgrading Your ownCloud Server

Starting with version 8.0.9, ownCloud will be automatically put into maintenance mode after downloading upgraded packages. You must take it out of maintenance mode and then run the upgrade wizard to complete the upgrade.

It is best to keep your ownCloud server upgraded regularly, and to install all point releases and major releases without skipping any of them, as skipping releases increases the risk of errors. Major releases are 8.0, 8.1, 8.2, and 9.0. Point releases are intermediate releases for each major release. For example, 8.0.9 and 8.1.3 are point releases. **Skipping major releases is not supported.**

There are multiple ways to keep your ownCloud server upgraded: with the *Updater App* (Server Edition only), with your Linux package manager, and by manually upgrading. In this chapter you will learn how to keep your ownCloud installation current with your Linux package manager, and by manually upgrading.

Note: Enterprise Subscription customers will use their Enterprise software repositories to install ownCloud packages, rather than the Open Build Service. Then follow the instructions on this page for completing upgrades. Please see *Installing ownCloud Enterprise Subscription on Linux* for more information.

When you are upgrading to a major release, evaluate any third-party apps for compatibility with the upgrade, and then disable them before upgrading. You may re-enable them after the upgrade is completed.

Warning: **Downgrading is not supported** and 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 see if your issue can be resolved without downgrading.

9.3.1 Upgrade Quickstart

The best method for keeping ownCloud on Linux servers current is by configuring your system to use the openSUSE Build Service (see *Preferred Linux Installation Method*); just follow the instructions on oBS for setting up your

package manager. Then stay current by using your Linux package manager to upgrade fresh packages, and then 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*.
- Install new packages from the openSUSE Build Service.
- Take your ownCloud server out of *maintenance mode* (ownCloud 8.1.4+).
- Run the *upgrade wizard* (optionally disabling the *migration test*).
- Log in and *apply strong permissions* to your ownCloud directories.
- Re-enable third-party apps.

Note: If you are using the Encryption app and upgrading from older versions of ownCloud to ownCloud 8.0, you must manually migrate your encryption keys with the *occ* command after the upgrade is complete, like this example for CentOS: *sudo -u apache php occ encryption:migrate-keys* You must run *occ* as your HTTP user. See *Using the occ Command* to learn more about *occ*

9.3.2 Prerequisites

You should always maintain regular backups and make a fresh backup before every upgrade.

Then review any third-party apps you have installed for compatibility with the new ownCloud release. Any apps that are not developed by ownCloud show a 3rd party designation. **Install unsupported apps at your own risk**. Then, before the upgrade, they must all be disabled. After the upgrade is complete and you are sure they are compatible with the new ownCloud release you may re-enable them.

Upgrading is disruptive. Your ownCloud server will be automatically put into maintenance mode, so your users will be locked out until the upgrade is completed. Large installations may take several hours to complete the upgrade.

9.3.3 Encryption migration from oC 7.0 to 8.0

The encryption backend was changed between ownCloud 7.0 and 8.0. If you're upgrading from this older version please refer to *Encryption migration to ownCloud 8.0* for the needed migration steps.

9.3.4 Upgrading With Your Linux Package Manager

When an ownCloud upgrade is available from the openSUSE Build Service repository, apply it just like any normal Linux upgrade. For example, on Debian or Ubuntu Linux this is the standard system upgrade command:

```
$ sudo apt-get update && sudo apt-get upgrade
```

Or you can upgrade just ownCloud with this command:

```
$ sudo apt-get update && sudo apt-get install owncloud
```

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:

```
$ sudo yum update
```

Or update only ownCloud:

```
$ sudo yum update owncloud
```

Your Linux package manager only downloads the current ownCloud packages. Then your ownCloud server is automatically put into maintenance mode.

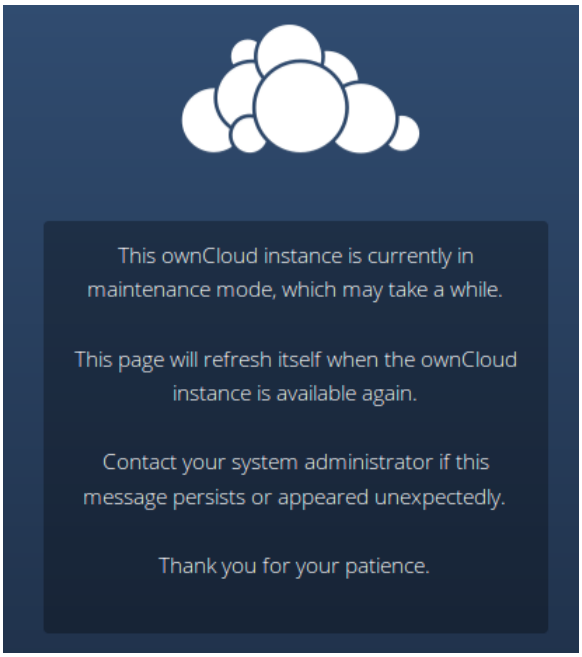


Figure 9.1: *Click to enlarge*

Next, take your server out of maintenance mode. You can do this by changing `'maintenance' => true,` to `'maintenance' => false,` in `config.php`, or use the *occ command*, like this example on Ubuntu:

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

9.3.5 Upgrade Wizard

The final step is to run the upgrade wizard to perform the final steps of updating your apps and database. You will see a screen with a summary of apps that are updated, and a **Start Update** button. If you have shell access it is better to **not** click the Start Update button, but rather to use `occ upgrade`, like this example on CentOS:

```
$ sudo -u apache php occ upgrade
```

`occ upgrade` is more reliable, especially on installations with large datasets and large numbers of users because it avoids the risk of PHP timeouts.

When the upgrade is completed you will be returned to the login screen.

Migration Test

Before completing the upgrade, ownCloud first runs a simulation by copying all database tables to a temporary directory and then performing the upgrade on them, to ensure that the upgrade will complete correctly. This can delay

large installations by several hours, so you can omit this step with the `--skip-migration-test` option, like this example on CentOS:

```
$ sudo -u apache php occ upgrade --skip-migration-test
```

Setting Strong Permissions

After upgrading, verify that your ownCloud directory permissions are set according to *Setting Strong Directory Permissions*.

If the upgrade fails, then you must try a manual upgrade.

9.3.6 Manual Upgrade Procedure

Always start by making a fresh backup.

If you are upgrading to a major release, for example from 8.1.3 to 8.2, you must review all third party applications (not core apps) for compatibility with your new ownCloud version. Then disable all of them before starting the upgrade.

Next put your server in maintenance mode. This prevents new logins, locks the sessions of logged-in users, and displays a status screen so users know what is happening. There are two ways to do this, and the preferred method is to use the *occ command*, which you must run as your HTTP user. This example is for Ubuntu Linux:

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

The other way is by entering your `config.php` file and changing `'maintenance' => false,` to `'maintenance' => true,`.

1. Back up your existing ownCloud Server database, data directory, and `config.php` file. (See *Backing up ownCloud*.)
2. Download and unpack the latest ownCloud Server release (Archive file) from owncloud.org/install/ into an empty directory outside of your current installation. For example, if your current ownCloud is installed in `/var/www/owncloud/` you could create a new directory called `/var/www/owncloud2/`

Note: To unpack your new tarball:: tar xjf owncloud-latest.tar.bz2

---

3. Stop your Web server.
4. Rename or move your current ownCloud directory (named `owncloud/` if installed using defaults) to another location.
5. This creates a new `owncloud/` directory populated with your new server files. Copy this directory and its contents to the original location of your old server, for example `/var/www/`, so that once again you have `/var/www/owncloud`.
6. Copy and paste the `config.php` file from your old version of ownCloud to your new ownCloud version.
7. If you keep your `data/` directory in your `owncloud/` directory, copy it from your old version of ownCloud to the `owncloud/` directory of your new ownCloud version. If you keep it outside of `owncloud/` 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.

Note: We recommend storing your `data/` directory in a location other than your `owncloud/` directory.

---

8. Restart your Web server.

9. **Now you should be able to open a Web browser to your ownCloud server and** log in as usual. You have a couple more steps to go: You should see a **Start Update** screen, just like in the **Upgrading With Your Linux Package Manager** section, above. Review the prerequisites, and if you have followed all the steps click the **Start Update** button.

   If you are running a large installation with a lot of files and users, you should launch the upgrade from the command line using `occ` to avoid PHP timeouts, like this example on Ubuntu Linux:

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

Note: The `occ` command does not download ownCloud updates. You must first download and install the updated code (steps 1-3), and then `occ` performs the final upgrade steps.

---

10. The upgrade operation takes a few minutes to a few hours, depending on the size of your installation. When it is finished you will see a success message, or an error message that will tell where it went wrong.

Assuming your upgrade succeeded, take a look at the bottom of your Admin page to verify the version number. Check your other settings to make sure they're correct. Go to the Apps page and review the core apps to make sure the right ones are enabled. Re-enable your third-party apps. Then apply strong permissions to your ownCloud directories (*Setting Strong Directory Permissions*).

9.3.7 Reverse Upgrade

If you need to reverse your upgrade, see *Restoring ownCloud*.

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

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

Occasionally, *files do not show up after a upgrade*. A rescan of the files can help:

```
$ sudo -u www-data php console.php files:scan --all
```

See the 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:

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

Then start the manual process:

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

If this does not work properly, try the repair function:

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

9.4 Upgrading ownCloud with the Updater App

The Updater app automates many of the steps of updating an ownCloud installation. You should keep your ownCloud server updated and not skip any releases. The Updater app is enabled in your ownCloud Server instance by default, which you can confirm by looking on your Apps page.

Note: The Updater app is **not enabled and not supported** in ownCloud Enterprise Subscription.

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.

The Updater App is not required, and it is recommended to use other methods for keeping your ownCloud server up-to-date, if possible. (See *Upgrading Your ownCloud Server*.) The Updater App is useful for installations that do not have root access, such as shared hosting, for installations with a smaller number of users and data, and it automates updating *manual installations*.

Note: If you are using the Encryption app and upgrading from older versions of ownCloud to ownCloud 8.0, you must manually migrate your encryption keys with the *occ* command after the upgrade is complete, like this example for CentOS: *sudo -u apache php occ encryption:migrate-keys* You must run *occ* as your HTTP user. See *Using the occ Command* to learn more about *occ*

You should maintain regular backups (see *Backing up ownCloud*), and make a backup before every update. The Updater app does not backup your database or data directory.

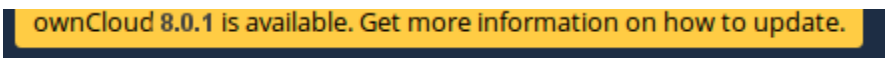
The Updater app performs these operations:

- Creates an `updater_backup` directory under your ownCloud data directory
- Download 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`, `config` and `themes` 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

Follow these steps in order

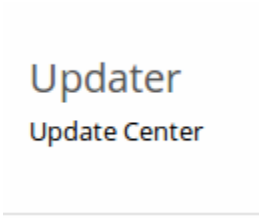
Using the Updater app to update your ownCloud installation is just a few steps:

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 Strong Permissions* section below.
4. Navigate to your Admin page and click the *Update Center* button under Updater:



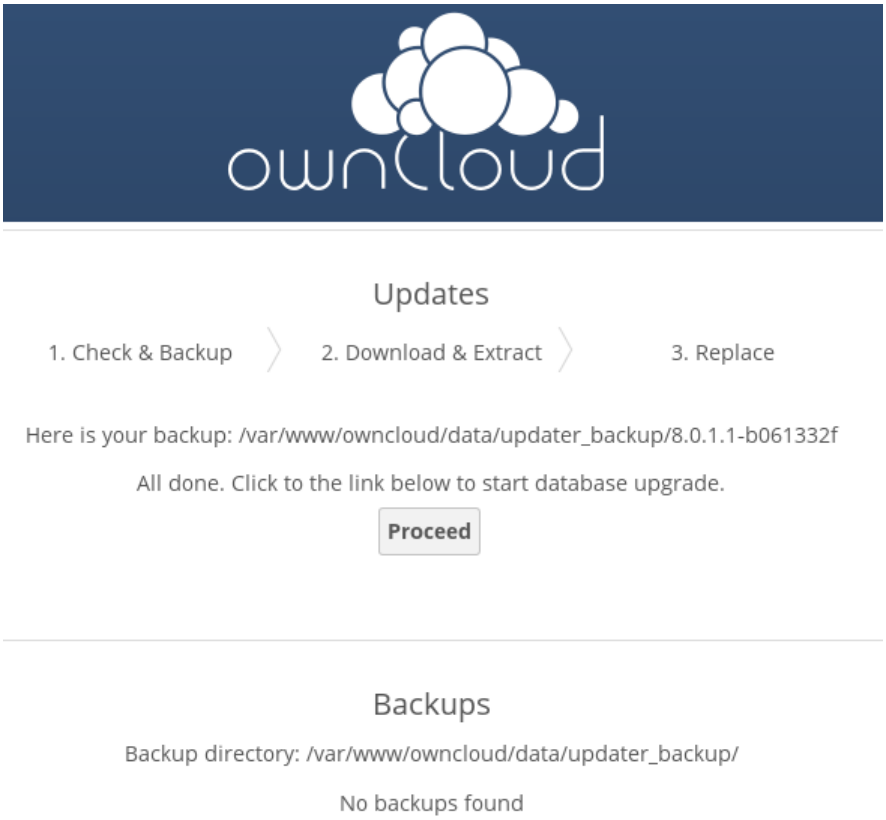
5. This takes you to the Updater control panel.

6. Click 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 *Setting Strong Permissions*.) Otherwise you will see messages about checking your installation and making backups.
7. Click Proceed, and then it performs the remaining steps, which takes a few minutes.
8. 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 Start Update button to complete your update:

Note: If you have a large ownCloud installation, at this point you should use the `occ upgrade` command, running it as your HTTP user, instead of clicking the Start Update button, in order to avoid PHP timeouts. This example is for Ubuntu Linux:

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

Before completing the upgrade, ownCloud first runs a simulation by copying all database tables to new tables, and then performs the upgrade on them, to ensure that the upgrade will complete correctly. The copied tables are deleted



after the upgrade. This takes twice as much time, which on large installations can be many hours, so you can omit this step with the `--skip-migration-test` option, like this example on Ubuntu:

```
$ sudo -u www-data php occ upgrade --skip-migration-test
```

See *Using the occ Command* to learn more.

9. 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.

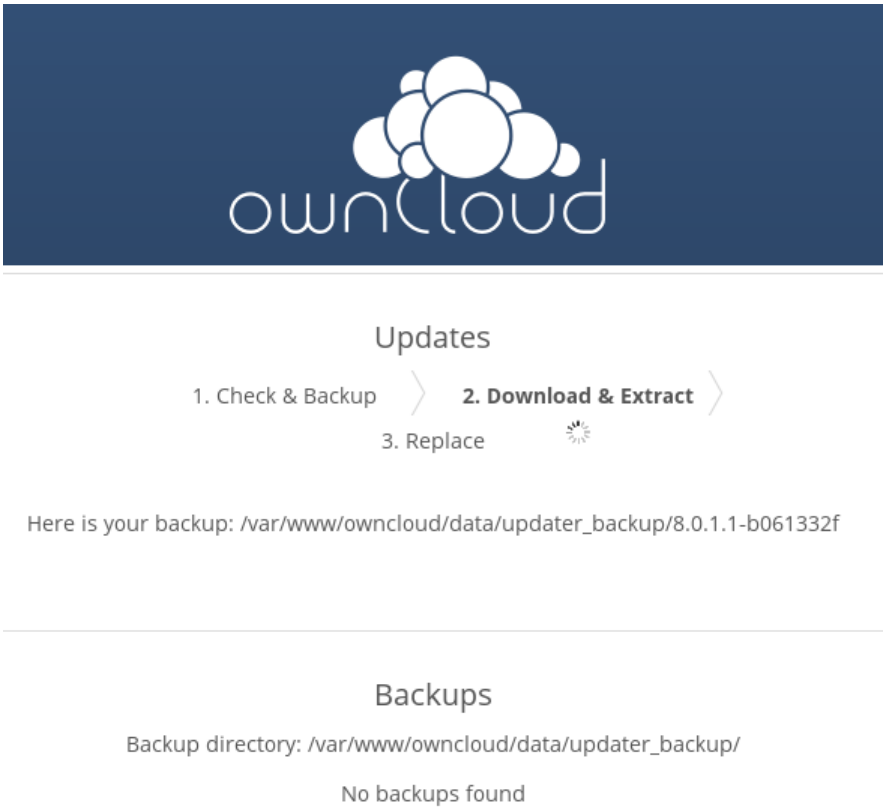
If the update fails, then you must update manually. (See *Upgrading Your ownCloud Server*.)

9.4.1 Can't Login Without Updating

If you can't login to your ownCloud installation without performing an update first, this means that updated ownCloud files have already been downloaded to your server, most likely via your Linux package manager during a routine system update. So you only need to click the Start Update button, or run the `occ` command to complete the update.

9.4.2 Setting Strong Permissions

For hardened security we highly recommend setting the permissions on your ownCloud directory as strictly as possible. These commands should be executed immediately after the initial installation. Please follow the steps in the **Setting**



Strong Directory Permissions section of *Installation Wizard*.

These strict permissions will prevent the Updater app from working, as it needs your whole ownCloud directory to be owned by the HTTP user. The generic command to change ownership of all files and subdirectories in a directory to the HTTP user is:

```
chown -R <http-user>:<http-user> /path/to/owncloud/
```

- This example is for Ubuntu 14.04 LTS server:

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

- Arch Linux:

  ```
  chown -R http:http /path/to/owncloud/
  ```

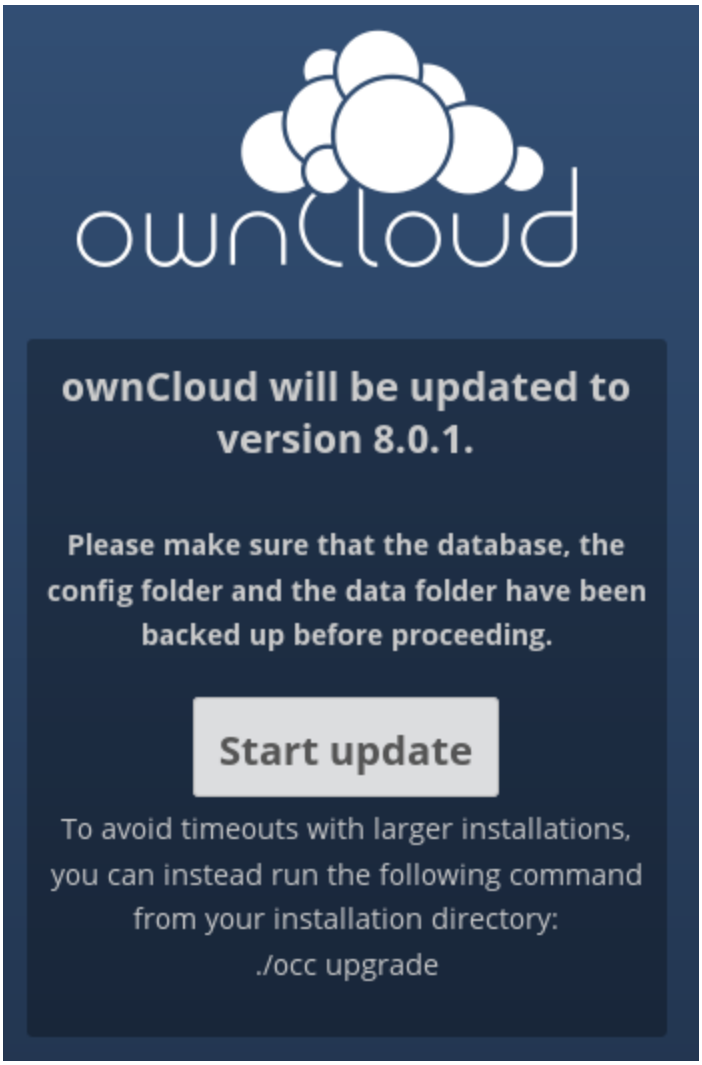
- Fedora:

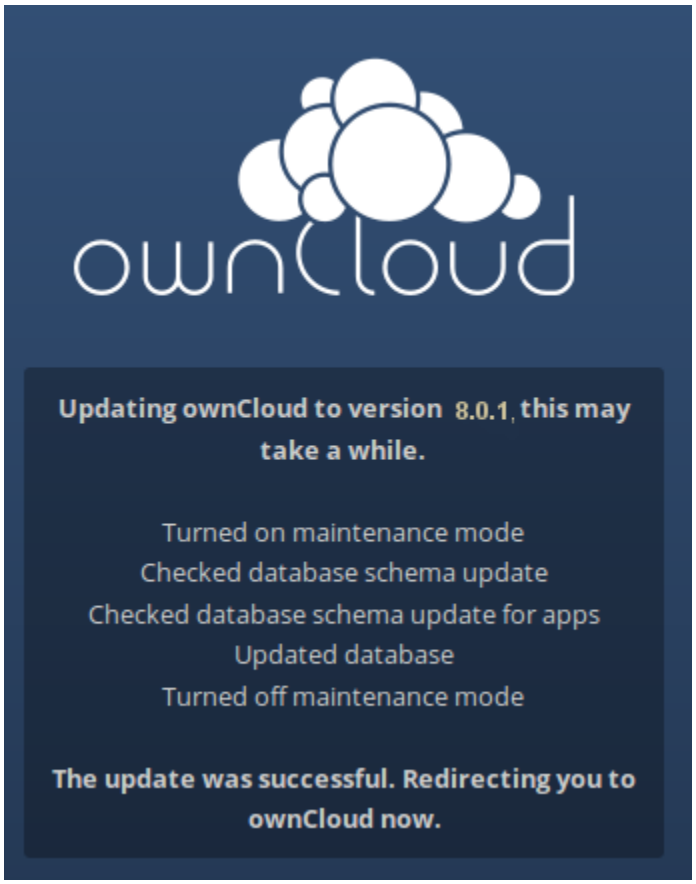
  ```
  chown -R apache:apache /path/to/owncloud/
  ```

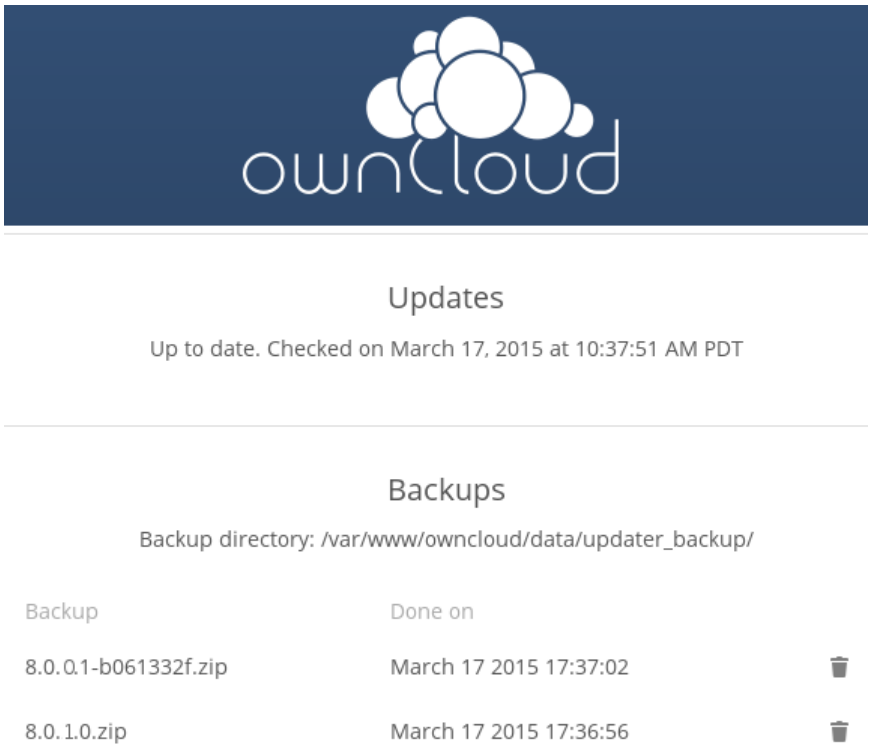
- openSUSE:

  ```
  chown -R wwwrun:www /path/to/owncloud/
  ```

After the Updater app has run, you should re-apply the strict permissions.


ownCloud
ownCloud will be updated to version 8.0.1.
Please make sure that the database, the config folder and the data folder have been backed up before proceeding.
Start update
To avoid timeouts with larger installations, you can instead run the following command from your installation directory:
./occ upgrade


Updating ownCloud to version 8.0.1, this may take a while.
Turned on maintenance mode
Checked database schema update
Checked database schema update for apps
Updated database
Turned off maintenance mode
The update was successful. Redirecting you to ownCloud now.


Updates
Up to date. Checked on March 17, 2015 at 10:37:51 AM PDT
Backups
Backup directory: /var/www/owncloud/data/updater_backup/
Backup
Done on
8.0.0.1-b061332f.zip
March 17 2015 17:37:02
8.0.1.0.zip
March 17 2015 17:36:56

9.5 Restoring ownCloud

To restore an ownCloud installation there are three main things you need to restore:

1. The configuration directory
2. The data directory
3. The database

Note: You must have both the database and data directory. You cannot complete restoration unless you have both of these.

When you have completed your restoration, see the `Setting Strong Directory Permissions` section of *Installation Wizard*.

9.5.1 Restore Folders

Note: This guide assumes that your previous backup is called "owncloud-dirbkp"

Simply copy your configuration and data folder (or even your whole ownCloud install and data folder) to your ownCloud environment. You could use this command:

```
rsync -Aax owncloud-dirbkp/ owncloud/
```

9.5.2 Restore Database

Note: This guide assumes that your previous backup is called "owncloud-sqlbkp.bak"

MySQL

MySQL is the recommended database engine. To restore MySQL:

```
mysql -h [server] -u [username] -p[password] [db_name] < owncloud-sqlbkp.bak
```

SQLite

```
rm data/owncloud.db
sqlite3 data/owncloud.db < owncloud-sqlbkp.bak
```

PostgreSQL

```
PGPASSWORD="password" pg_restore -c -d owncloud -h [server] -U [username]
owncloud-sqlbkp.bak
```

9.6 Migrating to a Different Server

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 virtual Appliance to a physical server. All migrations have to be performed with ownCloud offline and no accesses being made. Online migration is supported by ownCloud only when implementing industry standard clustering and HA solutions before ownCloud is installed for the first time.

To start let us be specific about the use case. A configured ownCloud instance runs reliably on one machine. For some reason (e.g. more powerful machine is available but a move to a clustered environment not yet needed) the instance needs to be moved to a new machine. Depending on the size of the ownCloud instance the migration might take several hours. As a prerequisite it is assumed that the end users reach the ownCloud instance via a virtual hostname (a `CNAME` record in DNS) which can be pointed at the new location. It is also assumed that the authentication method (e.g. LDAP) remains the same after the migration.

Warning: At NO TIME any changes to the **ORIGINAL** system are required **EXCEPT** putting ownCloud into maintenance mode.
This ensures, should anything unforseen happen you can go back to your existing installation and provide your users with a running ownCloud while debugging the problem.

1. Set up the new machine with the desired OS, install and configure the web server as well as PHP for ownCloud (e.g. permissions or file upload size limits) and make sure the PHP version matches ownCloud supported configuration and all relevant PHP extensions are installed. Also set up the database and make sure it is an ownCloud supported configuration. If your original machine was installed recently just copying that base configuration is a safe best practice.
2. On the original machine then stop ownCloud. First activate the maintenance mode. After waiting for 6-7 minutes for all sync clients to register the server as in maintenance mode stop the application and/or web server that serves ownCloud.
3. Create a dump from the database and copy it to the new machine. There import it in the database (See *Backing up ownCloud* and *Restoring ownCloud*).
4. Copy all files from your ownCloud instance, the ownCloud program files, the data files, the log files and the configuration files, to the new machine (See *Backing up ownCloud* and *Restoring ownCloud*). The data files should keep their original timestamp (can be done by using `rsync` with `-t` option) otherwise the clients will re-download all the files after the migration. Depending on the original installation method and the OS the files are located in different locations. On the new system make sure to pick the appropriate locations. If you change any paths, make sure to adopt the paths in the ownCloud config.php file. Note: This step might take several hours, depending on your installation.
5. While still having ownCloud in maintenance mode (confirm!) and **BEFORE** changing the `CNAME` record in the DNS start up the database, web server / application server on the new machine and point your web browser to the migrated ownCloud instance. Confirm that you see the maintenance mode notice, that a logfile entry is written by both the web server and ownCloud and that no error messages occur. Then take ownCloud out of maintenance mode and repeat. Log in as admin and confirm normal function of ownCloud.
6. Change the `CNAME` entry in the DNS to point your users to the new location.

CHAPTER

TEN

OPERATIONS

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

10.1 Considerations on Monitoring

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

10.1.1 ownCloud Deployment Architecture

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

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

10.1.2 The Important Components of ownCloud

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

Status.php

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

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

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

ownCloud.log

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

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

Apache

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

Database server

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

Clustered Filesystem

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

Load Balancer

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

10.2 Scaling Across Multiple Machines

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

- Application layer
- Database Layer
- Storage Layer

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

10.2.1 Application Layer

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

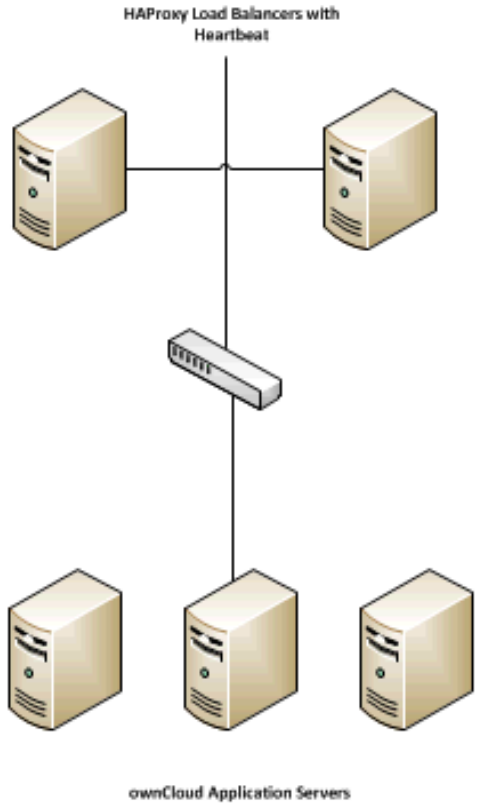
- Apache
- PHP 5.4.x
- PHP-GD
- PHP-XML
- PHP-MYSQL
- PHP-CURL
- SMBCLIENT

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

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

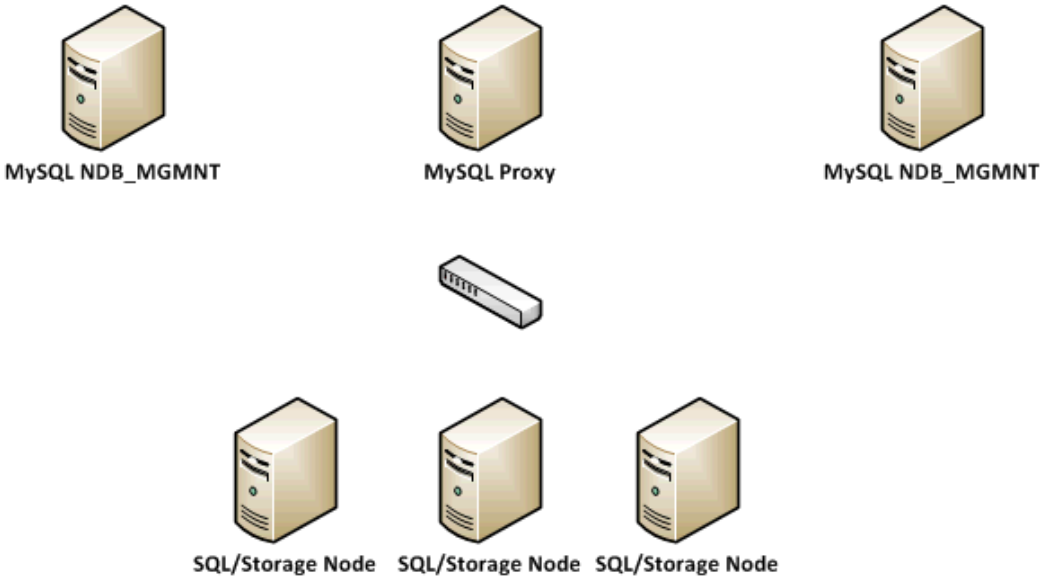
The next step is to create a scalable environment at the application layer, which introduces the load balancer. Because the application servers here are stateless, simply taking the configuration above and replicating (once configured with storage and database connections) and placing behind a load balancer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

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



10.2.2 Database Layer

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



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

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

mysql-cluster

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

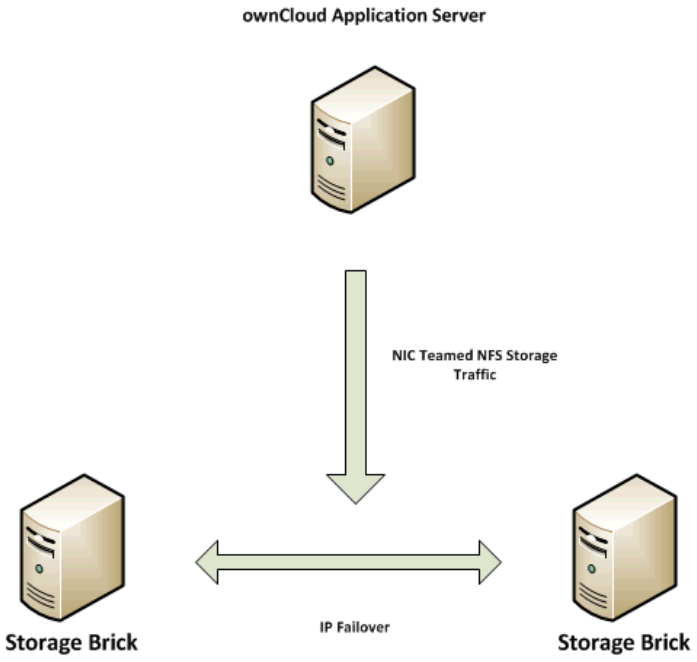
In this example we have also added a second HAProxy server with Heartbeat to prevent any single point of failure. We have also implemented NIC bonding to load balance the traffic across multiple physical NICs.

10.2.3 Storage Layer

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

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

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

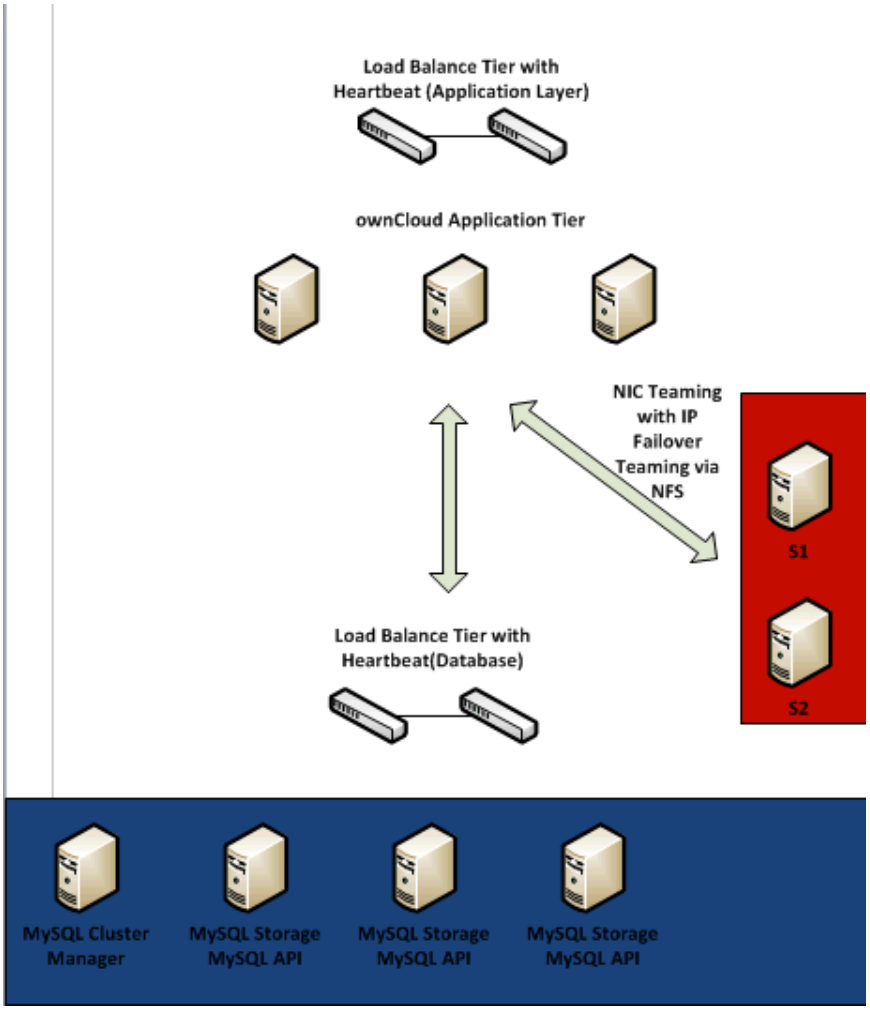


We chose to deploy the storage in this fashion to address both HA and extensibility of the storage, along with managing performance by simply adding additional bricks to the storage volume, back-ended by additional physical disk.

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

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

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



10.3 Theming ownCloud

ownCloud supports themes, so you can customize its look without editing source code. This document covers the basic steps of theming, and then walks through a basic theming example.

10.3.1 Configuration

The themes directory (usually `owncloud/themes`) contains all theming files. A default ownCloud installation comes with a helpful README in this directory. To add a new theme, copy a directory containing your themes files in this directory. The directory name is the theme name, and you configure ownCloud to use this theme by linking it in `config.php` with the `theme` directive:

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

Note: When you upgrade ownCloud custom themes are disabled, as custom themes can sometimes break the upgrade routine. You must re-enable your custom theme after the upgrade is complete by re-entering your theme name in config.php.

This theme overrides any Javascript files, images, templates and CSS files in other locations. Your directory structure should be the same as in `owncloud/core/`:

```
themename/
  core/
   css
   img
   js
```

Never edit the core template files– always use `owncloud/themes` for customizations. Errors in theme files can break ownCloud, so you can always revert to the default theme while you fix your custom theme.

It is possible for a theme to break between major ownCloud releases, as there may be changes to the ownCloud file structure. This can easily be resolved by examining the path to the file being replaced and mirror that path within the themes directory structure.

10.3.2 Customize the logo

Customized logos go in the `themes/core/img` directory. You can adapt the following example code, which goes in `themes/themename/core/css/header.css`, to display your customized logos:

```
#header .logo {
   background-image: url(../img/logo.svg);
   background-repeat: no-repeat;
   width: 252px;
   height: 120px;
   margin: 0 auto;
}

#header .logo-wide {
   background-image: url(../img/logo-wide.svg);
   background-repeat: no-repeat;
   width: 150px;
   height: 34px;
}

#header .logo-icon {
   /* display logo so appname can be shown next to it */
   display: inline-block;
   background-image: url(../img/logo-icon.svg);
   background-repeat: no-repeat;
   width: 62px;
   height: 34px;
}
```

10.3.3 Changing Colors

The color scheme is stored in the `styles.css` file within the `owncloud/core/css` folder.

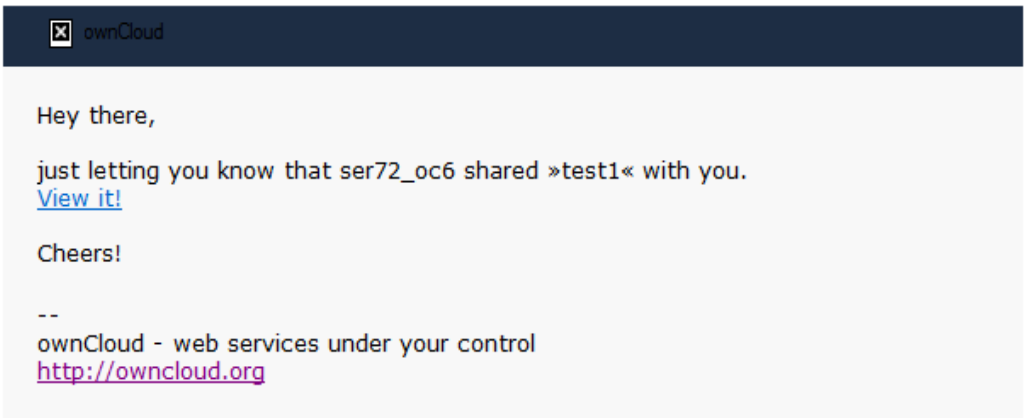
To change the color of the header bars, copy the `core/css/styles.css` to `themes/<themename>/core/css/styles.css` and edit.

This example shows a section in `styles.css` that configures colors. There are several locations within this file that assign colors to various page elements.

```
#body-login {
        text-align: center;
        background: #1d2d44; /* Old browsers */
        background: url('../img/noise.png'), -moz-linear-gradient(top, #35537a 0%, #1d2d44 100%); /*
        background: url('../img/noise.png'), -webkit-gradient(linear, left top, left bottom, color-st
        background: url('../img/noise.png'), -webkit-linear-gradient(top, #35537a 0%,#1d2d44 100%); /
        background: url('../img/noise.png'), -o-linear-gradient(top, #35537a 0%,#1d2d44 100%); /* Ope
        background: url('../img/noise.png'), -ms-linear-gradient(top, #35537a 0%,#1d2d44 100%); /* IE
        background: url('../img/noise.png'), linear-gradient(top, #35537a 0%,#1d2d44 100%); /* W3C */
        filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#35537a', endColorstr='#1d
}
```

10.3.4 Theming emails

Your ownCloud server can send notification and password reset emails, like this notification of a new file share:



Email templates are stored in `owncloud/core/templates/` and may be modified by copying them to `owncloud/themes/core/templates/mail.php`, and then customizing the copies. You may also edit email templates in the graphical template editor on your ownCloud Admin page.

10.3.5 Theming Example

General Hospital wants their own custom ownCloud theme, with custom logos and colors. Here is a step by step guide of how to build this theme.

10.3.6 Establishing a Directory Structure

Avoid spaces in your custom theme names:

```
themes/GeneralHospital
```

Next, create the directories for images and style sheets:

```
themes/GeneralHospital/core/img
themes/GeneralHospital/core/css
```

10.3.7 Configuration File

Next, add your new theme name to `config.php`:

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

10.3.8 Splash Screen Logo

This is the new logo for the login splash screen. It must be a 252x122 pixel SVG file:

The file must be named logo.svg and placed in the image folder:

```
themes/GeneralHospital/core/img/logo.svg
```

10.3.9 Top Left Logo

This is the new custom logo for the top left of the ownCloud navigation frame:

It must be an SVG file at 142x32 pixels, named `logo-wide.svg`. You can alter the width, but the height is fixed. This file also goes in the `themes/GeneralHospital/core/img/` folder.

10.3.10 Modifying Colors

The color definitions are stored in `core/css/styles.css`. The first step is to copy this file to `themes/GeneralHospital/core/css/styles.css`.

As stated in section Changing colors, edit the colors as desired in this section:

```
#body-login {
        text-align: center;
        background: #1d2d44; /* Old browsers */
        background: url('../img/noise.png'), -moz-linear-gradient(top, #35537a 0%, #1d2d44 100%); /*
        background: url('../img/noise.png'), -webkit-gradient(linear, left top, left bottom, color-st
        background: url('../img/noise.png'), -webkit-linear-gradient(top, #35537a 0%,#1d2d44 100%); /
        background: url('../img/noise.png'), -o-linear-gradient(top, #35537a 0%,#1d2d44 100%); /* Ope
        background: url('../img/noise.png'), -ms-linear-gradient(top, #35537a 0%,#1d2d44 100%); /* IE
        background: url('../img/noise.png'), linear-gradient(top, #35537a 0%,#1d2d44 100%); /* W3C */
        filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#35537a', endColorstr='#1d
}
```

This section of code handles the headers for many different browser types. The default colors in the above example are `#33537a` (light blue) and `#1d2d42` (dark blue). Some older browsers have only one color, however, most support gradients.

The login page background is a horizontal gradient. The hex `#33537a` is the top color of the gradient at the login screen. The `#1d2d42` is the bottom color of the gradient at the login screen.

To change the colors, modify these entries with the desired hex color code.

10.3.11 Changing Application icons

App icons can be modified as well. App icons are in the `owncloud/apps/<app>/img` folder. Similarly, the modified icon should reside in the `owncloud/themes/<themename>/apps/<app>/img` folder.

General Hospital would like to modify the activity icon with the following image:

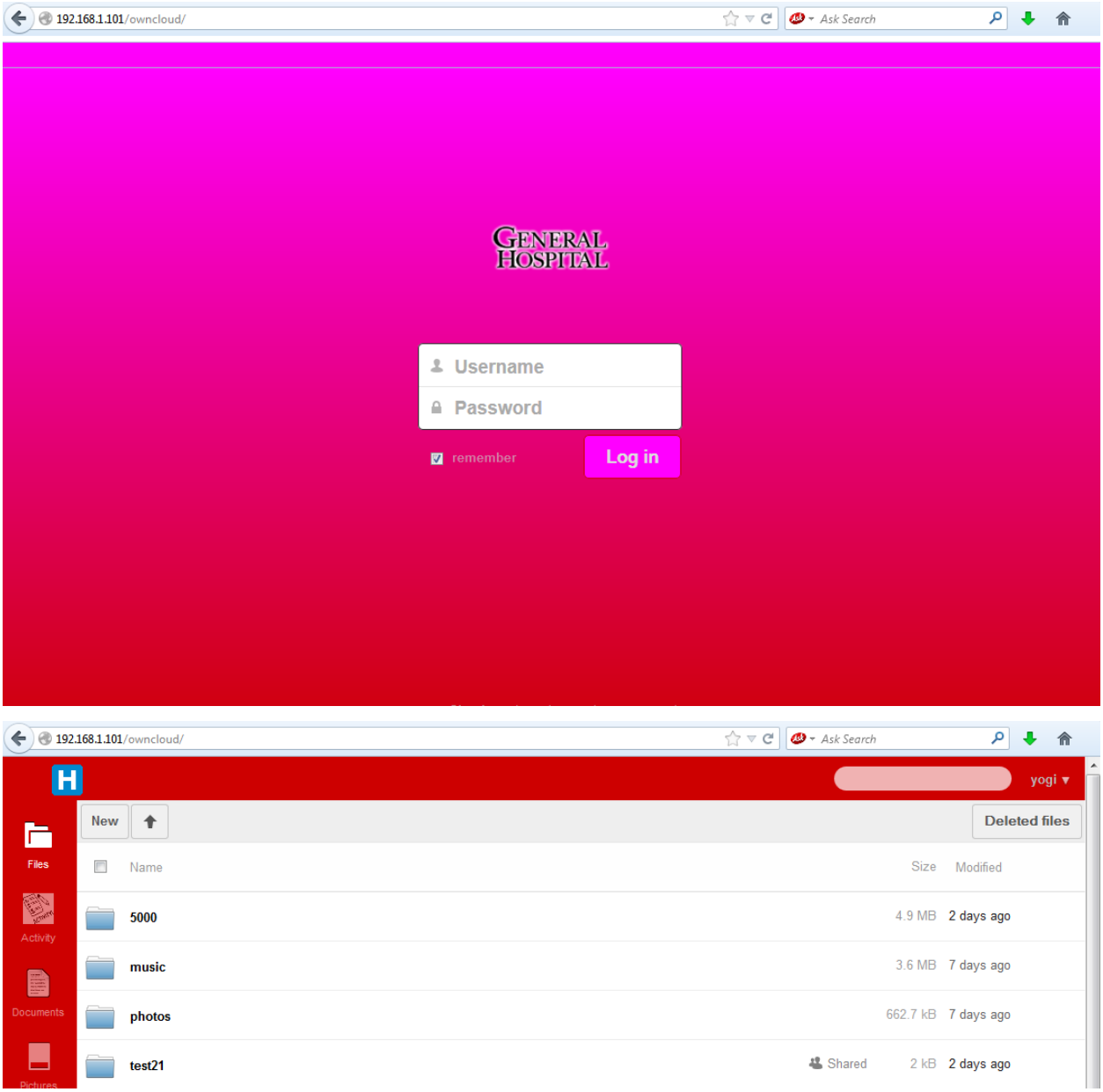
And the Documents icon with the following:

These must be converted to SVG format and placed in these locations:

```
owncloud/themes/GeneralHospital/apps/activity/img/activity.svg
owncloud/themes/GeneralHospital/apps/documents/img/documents.svg
```

10.3.12 Results

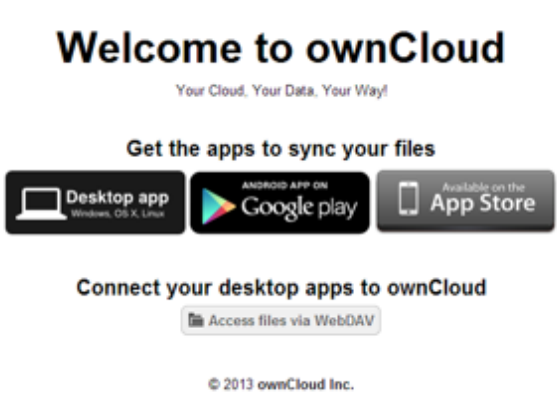
After saving all files and refreshing the Web browser, you can see the new theme:



Theming the First Run Wizard

The ownCloud First Run Wizard is a welcome screen which appears when a user logs into your ownCloud server for the first time, and users can re-run anytime from their Personal pages. (See *:doc: ../installation/installation-wizard*)

You may customize the welcome screen. This is the default:



The base file containing the welcome window configuration is `apps/firstwunwizard/templates/wizard.php`. Copy this file to your custom themes directory. (Make sure the `config.php` file has been updated to point to your themes directory.)

The `wizard.php` file performs checks to verify whether the community or enterprise edition of ownCloud is running. The welcome screen is slightly different in the two instances.

As an enterprise customer, all modifications to the `wizard.php` file should be in the "Else" section of the following condition statements:

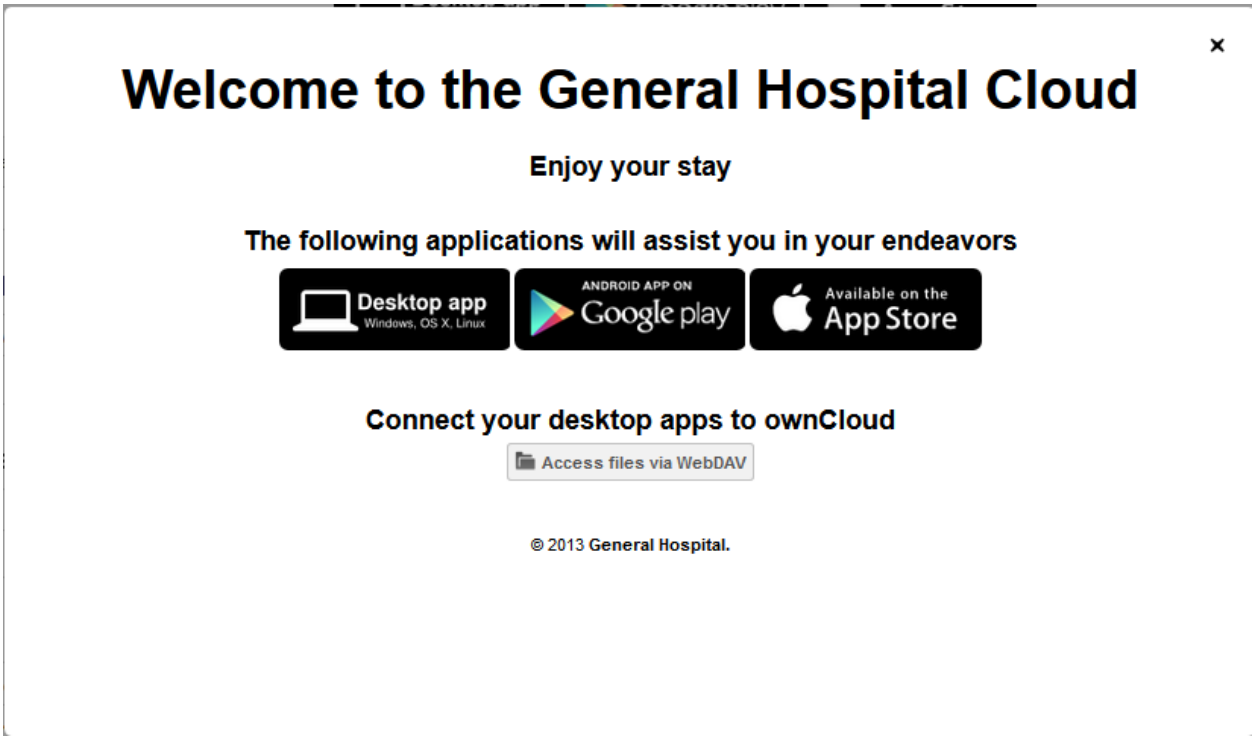
```
<?php if (OC_Util::getEditEditionString() === ''): ?>
  ...
<?php else: ?>
  ...
<?php endif; ?>
```

General Hospital wants to modify the welcome window, so modify `wizard.php` to apply the appropriate customizations. In this example several instances of `ownCloud` are replaced with `General Hospital`.

```
<div id="firstrunwizard">

<a id="closeWizard" class="close">
       <img class="svg" src="<?php print_unescaped(OCP\Util::imagePath('core', 'actions/close.svg'));
</a>
<h1><?php p($l->t('Welcome to the General Hospital Cloud', array($theme->getTitle()))); ?></h1>
<?php if (OC_Util::getEditionString() === ''): ?>
<p><?php p($l->t('Your personal web services. All your files, contacts, calendar and more, in one pla
<?php else: ?>
<h2><?php p($l->t('Enjoy your stay'));?></h2>
<?php endif; ?>


<h2><?php p($l->t('Get the apps to sync your files'));?></h2>
<a target="_blank" href="<?php p($_['clients']['desktop']); ?>">
       <img src="<?php print_unescaped(OCP\Util::imagePath('core', 'desktopapp.png')); ?>" />
```

```
</a>
<a target="_blank" href="<?php p($_['clients']['android']); ?>">
       <img src="<?php print_unescaped(OCP\Util::imagePath('core', 'googleplay.png')); ?>" />
</a>
<a target="_blank" href="<?php p($_['clients']['ios']); ?>">
       <img src="<?php print_unescaped(OCP\Util::imagePath('core', 'appstore.png')); ?>" />
</a>

<?php if (OC_Util::getEditionString() === ''): ?>
<h2><?php p($l->t('Connect your desktop apps to %s', array($theme->getName()))); ?></h2>
<a target="_blank" class="button" href="<?php p(link_to_docs('user-sync-calendars')) ?>">
       <img class="appsmall appsmall-calendar svg" src="<?php print_unescaped(OCP\Util::imagePath('co
</a>
<a target="_blank" class="button" href="<?php p(link_to_docs('user-sync-contacts')) ?>">
       <img class="appsmall appsmall-contacts svg" src="<?php print_unescaped(OCP\Util::imagePath('co
</a>
<a target="_blank" class="button" href="<?php p(link_to_docs('user-webdav')); ?>">
       <img class="appsmall svg" src="<?php print_unescaped(OCP\Util::imagePath('core', 'places/folde
</a>
<?php else: ?>
<br><br><br>
<a target="_blank" class="button" href="<?php p(link_to_docs('user-manual')); ?>">
       <img class="appsmall svg" src="<?php print_unescaped(OCP\Util::imagePath('settings', 'help.svg
</a>
<a target="_blank" class="button" href="<?php p(link_to_docs('user-webdav')); ?>">
       <img class="appsmall svg" src="<?php print_unescaped(OCP\Util::imagePath('core', 'places/folde
</a>
<?php endif; ?>

<p class="footnote">
<?php if (OC_Util::getEditionString() === ''): ?>
<?php print_unescaped($l->t('There's more information in the <a target="_blank" href="%s">documentati
<?php print_unescaped($l->t('If you like ownCloud,
       <a href="mailto:?subject=ownCloud
               &body=ownCloud is a great open software to sync and share your files.
               You can freely get it from http://owncloud.org">
               recommend it to your friends</a>
       and <a href="http://owncloud.org/promote"
               target="_blank">spread the word</a>!')); ?>
<?php else: ?>
© 2014 <a href="https://owncloud.com" target="_blank">General Hospital.</a>
<?php endif; ?>
</p>

</div>
```

The resulting welcome window looks like this:


Welcome to the General Hospital Cloud
Enjoy your stay
The following applications will assist you in your endeavors
Desktop app
Windows, OS X, Linux
ANDROID APP ON
Google play
Available on the
App Store
Connect your desktop apps to ownCloud
Access files via WebDAV
© 2013 General Hospital.

CHAPTER

ELEVEN

ISSUES AND TROUBLESHOOTING

If you have trouble installing, configuring or maintaining ownCloud, please refer to our community support channels:

- The ownCloud Forums

Note: The ownCloud forums have a FAQ page where each topic corresponds to typical mistakes or frequently occurring issues

- The ownCloud User mailing list
- The ownCloud IRC chat channel `irc://#owncloud@freenode.net` on freenode.net, also accessible via webchat

Please understand that all these channels essentially consist of users like you helping each other out. Consider helping others out where you can, to contribute back for the help you get. This is the only way to keep a community like ownCloud healthy and sustainable!

If you are using ownCloud in a business or otherwise large scale deployment, note that ownCloud Inc. offers the Enterprise Subscription with commercial support options.

11.1 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.

11.2 General Troubleshooting

Check the ownCloud *System Requirements*, especially supported browser versions.

11.2.1 Disable 3rdparty / non-shipped apps

It might be possible that 3rd party / non-shipped apps are causing various different issues. Always disable 3rd party apps before upgrades, and for troubleshooting. Please refer to the *Apps Commands* on how to disable an app from command line.

11.2.2 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 manually editing the configuration file. Edit `config/config.php` and add `define('DEBUG', true);`:

```
<?php
define('DEBUG',true);
$CONFIG = array (
    ... configuration goes here ...
);
```

For JavaScript issues you will also need to view the javascript console. All major browsers have developer tools for viewing the console, and you usually access them by pressing F12. For Firefox we recommend to installing the Firebug extension.

Note: The logfile of ownCloud is located in the data directory `owncloud/data/owncloud.log`.

11.2.3 phpinfo

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. e.g. `http://localhost/phpinfo`. 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.

11.2.4 Debugging Sync Issues

Warning: 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.org/owncloud/remote.php/webdav

11.2.5 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` -> The database request takes too long and therefore the MySQL server times out. Its also possible that the server is dropping a packet that is too large. Please refer to the manual of your database for how to raise the config options `wait_timeout` and/or `max_allowed_packet`.
- `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` -> This could be caused by wrong `KeepAlive` settings within your Apache config. Make sure that `KeepAlive` is set to `On` and also try to raise the limits of `KeepAliveTimeout` and `MaxKeepAliveRequests`.
- `No basic authentication headers were found` -> This error is shown in your `data/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 at the forums.

11.3 Troubleshooting Webserver and PHP problems

11.3.1 Logfiles

When having issues the first step is to check the logfiles provided by PHP, the Webserver and ownCloud itself.

Note: In the following the paths to the logfiles of a default Debian installation running Apache2 with mod_php is assumed. On other webservers, 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 Webserver.
- The logfile of ownCloud is located in the data directory `/var/www/owncloud/data/owncloud.log`.

11.3.2 Webserver and PHP modules

There are some Webserver or PHP modules which are known to cause various problems like broken up-/downloads. The following shows a draft overview of these modules:

1. Apache

- mod_pagespeed
- mod_evasive
- mod_security
- mod_reqtimeout

- mod_deflate
- libapache2-mod-php5filter (use libapache2-mod-php5 instead)
- mod_spdy together with libapache2-mod-php5 / mod_php (use fcgi or php-fpm instead)
- mod_dav
- mod_xsendfile / X-Sendfile (causing broken downloads if not configured correctly)

2. NginX
- ngx_pagespeed
- HttpDavModule
- X-Sendfile (causing broken downloads if not configured correctly)

3. Mac OS X server
- mod_auth_apple
- com.apple.webapp.webdavsharing

4. PHP
- eAccelerator

11.4 Troubleshooting WebDAV

ownCloud uses SabreDAV, and the SabreDAV documentation is comprehensive and helpful.

See:

- SabreDAV FAQ
- Webservers (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 webservers)

There is also a well maintained FAQ thread available at the ownCloud Forums which contains various additional information about WebDAV problems.

11.5 Troubleshooting Contacts & Calendar

11.5.1 Service discovery

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

There are several techniques to remedy this, which are described extensively at the Sabre DAV website.

If your ownCloud instance is installed in a subfolder under the web server's document root and the client has difficulties finding the Cal- or CardDAV end-points, configure your web server to redirect from a "well-known" URL to the one used by ownCloud. When using the Apache web server this is easily achieved using a `.htaccess` file in the document root of your site.

Say your instance is located in the `owncloud` folder, so the URL to it is `ADDRESS/owncloud`, create or edit the `.htaccess` file and add the following lines:

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

Now change the URL in the client settings to just use `ADDRESS` instead of e.g. `ADDRESS/remote.php/carddav/principals/username`.

This problem is being discussed in the forum.

11.5.2 Unable to update Contacts or Events

If you get an error like `PATCH https://ADDRESS/some_url 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.

11.6 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.

CHAPTER

TWELVE

OWNCLOUD VIDEOS

Please visit our YouTube channel for howtos, demos, news, and Webinars for both the Server and Enterprise versions of ownCloud.

12.1 Server to Server Sharing on ownCloud 7

Build a Cloud of ownCloud Servers with Server to Server Sharing

Create your own cloud of ownCloud servers with server-to-server sharing. Link specific shares to other ownCloud servers and have two-way synchronization.

12.2 Introducing ownCloud 7 Enterprise Edition

ownCloud 7 Enterprise Edition Enterprise File Sync and Share Software

ownCloud 7 Enterprise Edition introduces Universal File Access, which provides a single interface to all of your disparate systems and data silos. Integrate Sharepoint libraries, Windows network drives, link ownCloud servers with server-to-server sharing, and lots more.

12.3 ownCloud for Enterprise File Sync and Share

ownCloud for Enterprise File Sync and Share

ownCloud is an enterprise-grade file sync and share solution that is hosted in your data center, on your servers, using your storage. ownCloud integrates seamlessly into your IT infrastructure; you can leave data where it lives and still deliver file sharing services that meet your data security and compliance policies.


ownCloud 7 Enterprise Edition
New and Advanced Features
ownCloud


ownCloud

CHAPTER

THIRTEEN

ENTERPRISE SUBSCRIPTION ONLY

13.1 Enterprise Subscription Installation (ES Only)

13.1.1 Installing ownCloud Enterprise Subscription on Linux

The recommended method for installing and maintaining your ownCloud Enterprise Subscription is with your Linux package manager. Configure your package manager to use the ownCloud Enterprise Subscription repository, import the signing key, and then install and update ownCloud like any other software package. Please refer to the `README - ownCloud Package Installation.txt` document in your account at Customer.owncloud.com account for instructions on setting up your Linux package manager.

After you have completed your initial installation of ownCloud as detailed in the README, follow the instructions in *Installation Wizard* to finish setting up ownCloud.

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.

13.1.2 Supported ownCloud Enterprise Subscription Apps

See *Installing and Managing Apps* for instructions on managing ownCloud apps.

These applications are supported in ownCloud Enterprise Edition. The first name on each line is the name of the application as it appears on the Apps page in your ownCloud Web GUI, and the name in parentheses is the filename as it appears in your `owncloud/apps` directory.

Activity (activity)

Antivirus app for files (files_antivirus)

Deleted Files (files_trashbin)

Encryption (files_encryption)

Enterprise License Key (enterprise_key)

External Storage Support (files_external)

External user support (user_external)

Files (files)

File Firewall (Firewall)

File Locking (files_locking)

File Shared access logging app (files_sharing_log)

First Run Wizard (firstrunwizard)

LDAP Home Connector (files_ldap_home)

LDAP user and group backend (user_ldap)

Log audit info (admin_audit)

Mail Template Editor (templateeditor)

Provisioning API (provisioning_api)

Share Files (files_sharing)

SharePoint (sharepoint)

Shibboleth user backend (user_shibboleth)

Text Editor (files_texteditor)

Versions (files_versions)

WebDAV user backend (user_webdavauth)

Windows Network Drive (windows_network_drive)

13.1.3 License Keys

Introduction

You'll need to install a license key to use ownCloud 8 Enterprise Subscription. There are two types of license keys: one is a free 30-day trial key. The other is a full license key for Enterprise Subscription customers.

You can download and try ownCloud 8 Enterprise Subscription 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 Subscription 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 Subscription license key, it needs to be copied to your ownCloud configuration file, `config/config.php` file like this example:

```
'license-key' => 'test-20150101-XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX-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 Subscription to a new version.

13.1.4 Oracle Database Setup

This document will cover the setup and preparation of the ownCloud server to support the use of Oracle as a backend database. For the purposes of testing, we are using Oracle Enterprise Linux as both the Web server that will host ownCloud, and as a host for the Oracle 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, c
```

Substitute an actual password for `password`. Items like TableSpace, Quota etc. will be determined by your DBA.

Downloading and Installing the Oracle Instant Client

As our example system is Oracle Enterprise Linux, it is necessary to go to the Oracle site and download the Oracle Instant Client for your OS Distribution.

Note: Download the instant client and the instant client SDK and place them in a directory on the server, in this example they are RPM packages.

- Install the basic client from the RPM. Use the `rpm -ivh` command
- Install the SDK RPM package. Use the `rpm -ivh` command

At this point, the Oracle Instant client is installed on the ownCloud Host (in the home directory).

Install the OCI8 PHP Extension:

The next step is to compile and install the OCI8 PHP extension for connectivity to the Oracle Database.

- Create a folder for these bits on your server.

- Download the latest version of the extension from http://pecl.php.net/package/oci8.
- Unpack the OCI8 PHP extension and copy it over to the server.
- **There should be two things in the folder:**
  - `package.xml` file
  - `oci8-*.*.*` folder (folder will change based on version of the extension you downloaded).
- **Build the OCI8 module.**
  - Change (`cd`) to the folder where you have copied the downloaded and uncompressed OCI8 bits.
  - Run the following command (there will be a significant amount of output):

    ```
    pecl build
    ```

  - Eventually the output will stop and ask for the *Oracle Home Directory*, just press enter.
- Change directory:

  ```
  cd oci8-<version number>
  ```

- Type the following command:

  ```
  ./configure -with-oci8=instantcleint,/usr/lib/oracle/<version number>/client64/lib
  ```

- Again, there will be significant output
- Enter the following command to compile: `make`
- At this time there should be a folder called modules in the `oci8-<version_>` folder. Within this folder exists the `oci8.so` file.
- Copy this to the directory where the modules are stored in the PHP install. It depends on your distribution. This is the path for RHEL 6 and OEL 6:

  ```
  cp oci8.so /usr/lib64/php/modules
  ```

- **Create an `.ini` file**
  - Navigate to the `php.d` directory: `cd /etc/php.d`
  - Edit a file called oci8.ini: `vi oci8.ini`
  - Make the file look as follows:

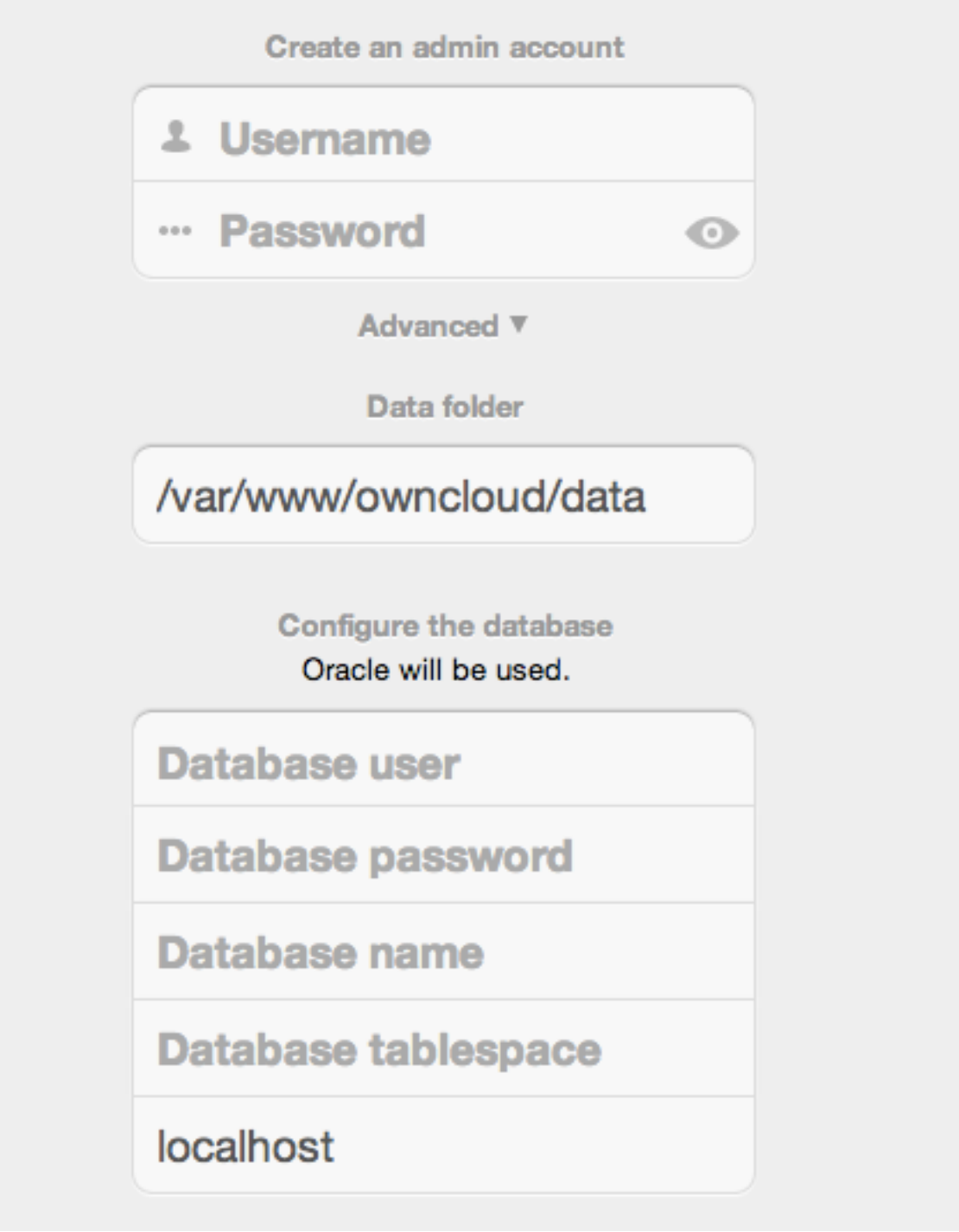
    ```
    ; Oracle Instant Client Shared Object
    extension=oci8.so
    ```

  - Save the document

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:

```
<?php
$CONFIG = array (
'instanceid' => 'abcdefgh',
'passwordsalt' => '01234567890123456789',
'datadirectory' => '/var/data',
'dbtype' => 'oci',
'version' => '8.0.x.y',
'dbname' => 'orcl',
'dbhost' => '192.168.1.57',
'dbtableprefix' => 'oc_',
'dbuser' => 'owncloud1',
'dbpassword' => '********',
'installed' => true,
);
```

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:

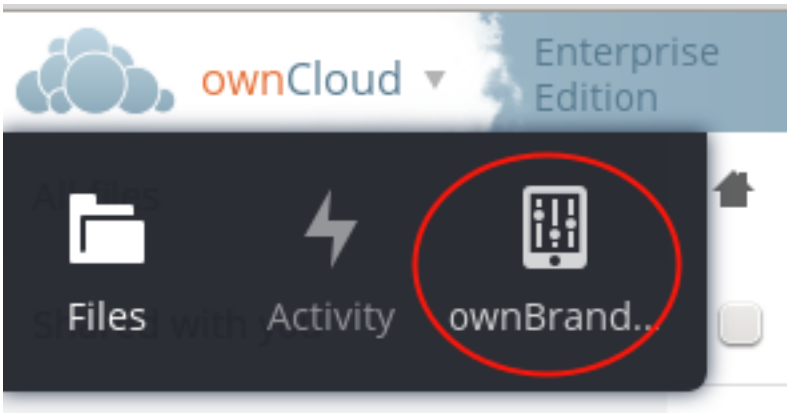
```
Oracle    : quit
```

13.2 Creating Branded ownCloud Clients (ES only)

13.2.1 Creating Branded Client Apps (Enterprise Only)

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 (Enterprise Only) 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 (Enterprise Only) 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 an Android App

Building and distributing your branded Android ownCloud app is fairly simple, and the process is detailed in Building Branded ownCloud Clients (Enterprise Only).

13.2.2 Custom Client Download Repositories

See *Custom Client Download Repositories* to learn how test and configure custom download repository URLs for your branded clients.

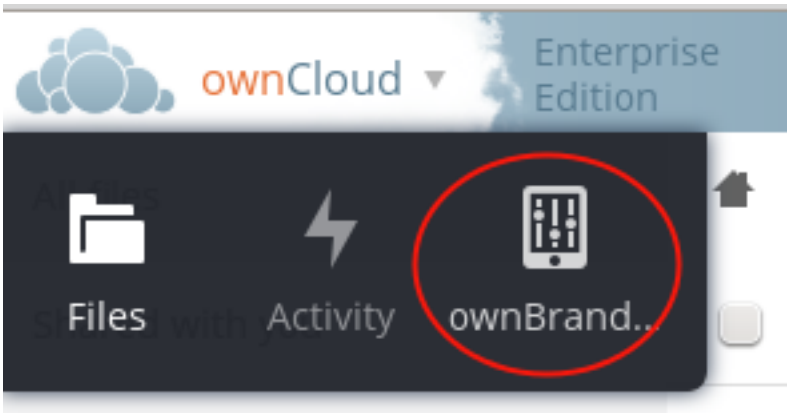
13.3 Enterprise Server Branding (ES only)

13.3.1 Custom Theming ownCloud (ES only)

Overview

ownBrander is an ownCloud build service that is exclusive to Enterprise subscription 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.

Note: 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 and then 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.

Login logo images

Login page logo

This is the image shown on the login page just above the Username and Password fields (svg format) (width: 252px height: 122px) *i*

Upload

Login page logo

This is the image shown on the login page just above the Username and Password fields. This image is used when the browser does not support svg, we recommend this to be the same as the previous one (Login page logo) (png format) (width: 252px height: 122px) *i*

Delete image Upload

Logo icon

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.

Note: Always edit only your custom theme files. Never edit the default theme files.

13.4 External Storage (ES only)

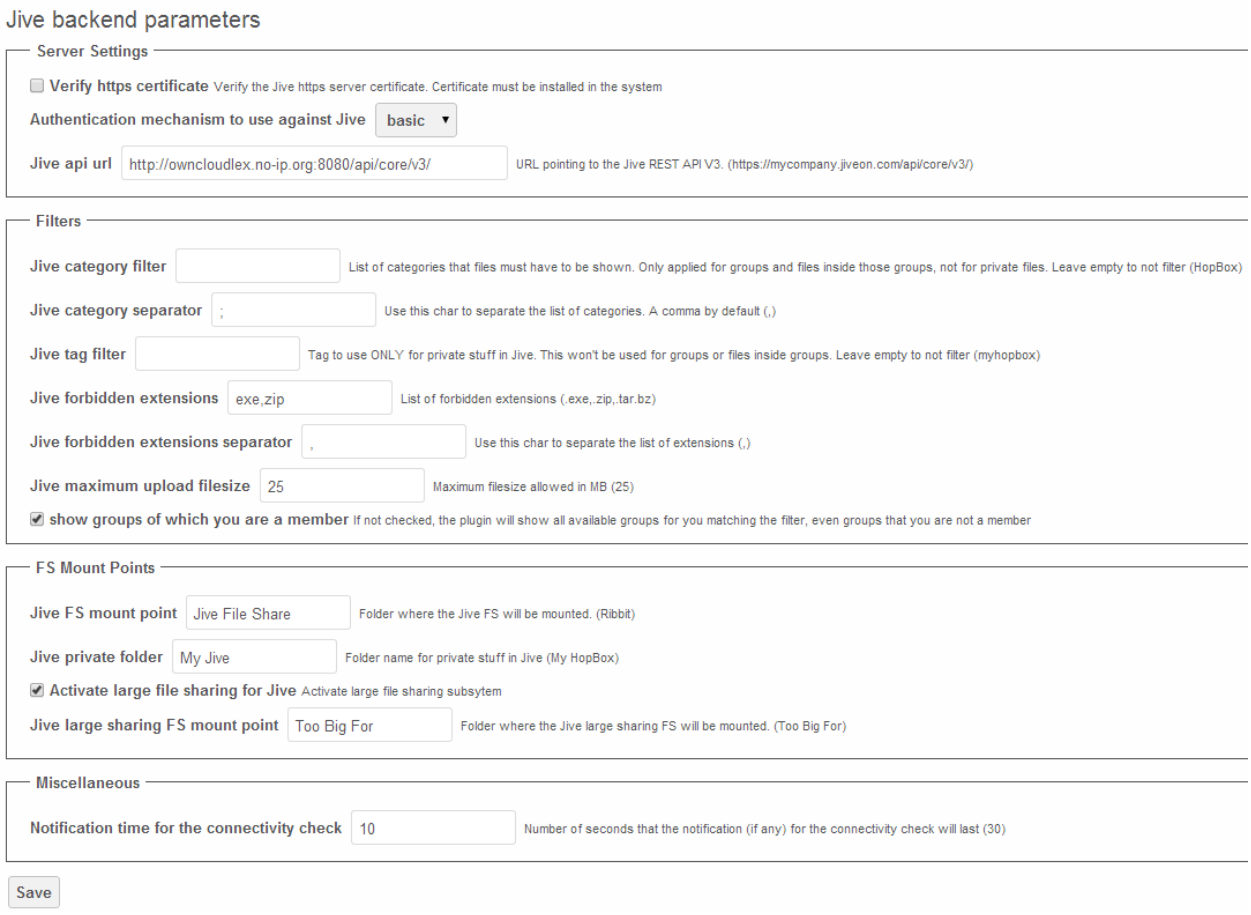
13.4.1 Jive Integration

The Jive application allows Jive users to access files stored in Jive from a mobile device, tablet, or desktop client. Users have complete access through ownCloud Enterprise Subscription to upload, edit or download their files.

Jive can be configured as a data storage location for ownCloud, which means files saved in Jive appear in folders within ownCloud. Jive remains the system of record while ownCloud acts as a proxy, providing end-to-end file access for users at their desks and on the go.

Configuration

The Jive application is installed under the owncloud/apps directory on the server and enabled via the ownCloud admin screen. This app is only available for ownCloud EE v6 or higher. Go to the ownCloud admin screen section "Jive backend parameters" to configure the app to match your Jive server system parameters.

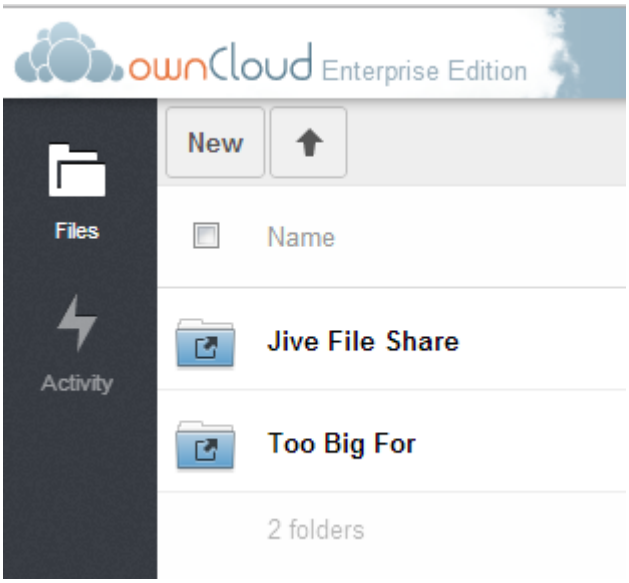

Jive backend parameters
Server Settings
Verify https certificate Verify the Jive https server certificate. Certificate must be installed in the system
Authentication mechanism to use against Jive basic
Jive api url http://owncloudlex.no-ip.org:8080/api/core/v3/ URL pointing to the Jive REST API V3. (https://mycompany.jiveon.com/api/core/v3/)
Filters
Jive category filter List of categories that files must have to be shown. Only applied for groups and files inside those groups, not for private files. Leave empty to not filter (HopBox)
Jive category separator ; Use this char to separate the list of categories. A comma by default (,)
Jive tag filter Tag to use ONLY for private stuff in Jive. This won't be used for groups or files inside groups. Leave empty to not filter (myhopbox)
Jive forbidden extensions exe,zip List of forbidden extensions (.exe,.zip,.tar.bz)
Jive forbidden extensions separator , Use this char to separate the list of extensions (,)
Jive maximum upload filesize 25 Maximum filesize allowed in MB (25)
show groups of which you are a member If not checked, the plugin will show all available groups for you matching the filter, even groups that you are not a member
FS Mount Points
Jive FS mount point Jive File Share Folder where the Jive FS will be mounted. (Ribbit)
Jive private folder My Jive Folder name for private stuff in Jive (My HopBox)
Activate large file sharing for Jive Activate large file sharing subsytem
Jive large sharing FS mount point Too Big For Folder where the Jive large sharing FS will be mounted. (Too Big For)
Miscellaneous
Notification time for the connectivity check 10 Number of seconds that the notification (if any) for the connectivity check will last (30)
Save

Parameter	Description	Values
Https	Verify the https server certificate. Certificate must be installed on the system.	Checkbox – enabled/disable
Authentication	Chose the Authentication mechanism to use against Jive	basic OR oAuth
Jive api url	URL string pointing to the Jive API	Example: https://mycompany.jiveon.com/api/core/v
Jive FS mount point	Folder where the Jive File share will be mounted	String value up to 10 characters max
Jive category filter	List of categories that files have to be shown	Jive categories list, or blank
Jive category separator	Separator for Jive catagories list	Comma by default or any single character
Jive tag filter	Tag to use for private stuff in jive	Jive tag or blank
Jive forbidden extensions	List of forbidden extensions These will not be allowed for upload or download with Jive.	Examples include: .exe,.zip
Jive forbidden extensions separator	Use this character to separate the list of extensions	Comma by default or any single character
Jive maximum upload filesize	Maximum file size allowed in MB. This includes upload and downloads.	Numeric value
Jive private folder	Folder name for private stuff in Jive	String value up to 250 characters max
Activate large file sharing for Jive	Enable the large file sharing subsystem. This allows storage of files that are too large for Jive to be stored on the ownCloud server and available via the ownCloud web, mobile and desktop interfaces.	Checkbox – enable/disable
Jive large file sharing FS mount point	Folder where the Jive large sharing File Share will be mounted	String value up to 10 characters max
Show groups of which you are a member	If this is not checked, the plugin will show all available groups for you matching the filter, even groups that you are not a member	Enable/disable

Use Cases

The ownCloud Jive plugin can be used in various ways to extend the access to the Jive content across multiple devices.

Web Client Use Cases



- Create a folder in the "Jive File Share" Web Client folder to create a new Jive Group.
  - Verify the Group is created in Jive.
- Create a new Group in Jive and upload a file to that Group.
  - Check the Web Client and download the file.
  - Verify that file is the same as the uploaded file.
- Upload a file in the "Too Big For" Jive folder, and create the link in a Jive document.
  - Verify that file link is in Jive.
  - Download the file via the link, and verify it is the same as the uploaded file.
- Upload a file to the private "My Jive" Web Client folder.
  - Check your Jive content and make sure the file has been uploaded.
  - Download the file and verify it is the same as the uploaded file.

Mobile Client Use Cases (iOs and Android)

Create a new folder in the Mobile Client to create a new Jive Group.

Upload a file in the Web Client folder, and see that file in the corresponding Jive Group.

Desktop Client Use Cases

Create a folder in the Desktop Client to create a new Jive Group.

Upload a file in the Desktop Client folder, and see that file in the corresponding Jive Group.

The ownCloud folder structure hierarchy matches the Jive Groups the user can access. Sub folders under the Jive Group folders that are created on the desktop will not sync to ownCloud or Jive because they will not match the Jive "Group" view. If a sub folder is created under the Jive Group desktop folder, the desktop client will display an error

that this operation is not allowed. For example; if the folder structure is "JiveFileShare/GroupA", any sub folder under GroupA will not be synced to ownCloud or Jive.

Configuring the Jive app

This section explains how each configuration parameter changes the behavior of the app.

Verify https certificate

If your Jive server is under https, it must provide a https certificate when a client connects to it. Curl (the client that ownCloud is using to connect to Jive) usually verify that certificate, but to do that you must somehow supply a CA cert so curl can verify against.

This feature is usually turn off to make the Jive app easier to use, because in this case curl won't verify the certificate, so you don't need to have installed the CA cert. However, turning this off could be a security issue: you could be connecting to a fake Jive server without notice.

If you want to turn on this feature, you must get the CA cert of the server (check " http://curl.haxx.se/docs/sslcerts.html " for more information about how you can get the file you need) and install it in your ownCloud server.

In order to know where you should install the CA cert, you can run

curl -v https://yourserver.com/

You should look the output for a line with the CA path:

- successfully set certificate verify locations:
- CAfile: none
- CApath: /etc/ssl/certs

That's the place where you should install the CA cert.

Once you have installed the CA cert, you should run again the same curl:

curl -v https://yourserver.com/

And look for:

- Server certificate:
- subject: *****
- start date: *******
- expire date: *****
- subjectAltName: *******
- issuer: ******
- SSL certificate verify ok.

If the SSL is verified correctly ("SSL certificate verify ok."), you just need to activate the checkbox.

Curl usually comes installed with some CA certs by default, so all the previous steps might not be needed. Just check that curl can connect to your Jive server, and if so, activate this feature.

Authentication mechanism to use against Jive

To be able to access to Jive, the ownCloud plugin needs to use some kind of authentication. At this time, the plugin supports basic and oAuth authentication.

Basic authentication In order to use basic authentication, you should take into account the following things:

- The credentials used to access to ownCloud must match the ones used to connect to Jive. This means that if you access to ownCloud with a user "PeterP" and password "PeterPassword", the same user must exist in Jive with the same password. Otherwise, the user won't be able to access to Jive.
- If the credentials (typically the password) changes in one side, it must change in the other. You'll need to this manually.

The usage of basic authentication isn't recommended due to the following reasons:

- We need to store the password and be able to recover it. Although the password is stored encrypted, this is not strictly secure.
- Passwords are sent to the server in almost plain text. In fact it's a base64 encoded string of user and password, but that's all the security the authentication provides.

If you plan to use basic authentication, at least make sure you connect through HTTPS protocol and inside a local LAN if possible.

oAuth authentication First of all, make sure Jive is prepared to support this authentication.

The usage of this authentication method solves the issue of having the same credentials in both ownCloud and Jive server. This means that the ownCloud user "PeterP" with password "PeterPassword" can access to the contents of the Jive user "John" with password "John007". It's also possible that another ownCloud user "AliceK" access to the contents of the Jive user "John" too at the same time.

Keep in mind that this isn't insecure: any ownCloud user that wants to access to John's Jive content (following this little example) MUST know his Jive password.

If this authentication method is set, we don't store passwords BUT we still need to store some other things. These things are stored in plain text.

These are the steps to make it work (if your Jive server support this authentication):

1. Activate the oAuth authentication in the ownCloud admin settings (just the admin can do this)
2. Go to the ownCloud web interface, in the files app. A popup will appear.
3. Click on the link that appear in the popup
4. You'll get redirected to a Jive page asking for your Jive credentials. If this is not the case, it's recommended to clean the browser cache and start again (to point 2) because you might be accessing to Jive with another user.
5. After entering your Jive credentials, you get redirected a page with an activation code. If you entered the wrong credentials, you might not get redirected to that page. If this is the case click in the link again in the ownCloud popup (point 3) which will redirect you to the activation code page.
6. Copy the activation code into the ownCloud popup, and click in the "send code" button. If there is no error, you're done.

WARNING:

Not all the oAuth flows are covered by the plugin. The expiration of the access token is handled automatically by the plugin, so it will request a new access token if needed. HOWEVER, the expiration of the refresh token isn't covered, so the plugin will likely stop working for that user (we won't be able to get more access tokens)

[Ask for info to know how to solve this issue?]

It's very important that the user access to ownCloud through the web interface first, so the user goes through the oAuth flow for the first time (as described with the steps above) to get an access token. Otherwise, the plugin won't get an access token and the user won't be able to get the files from Jive.

Jive API URL

You'll need to enter the full URL of the Jive API. This includes the protocol (HTTP or HTTPS) and the port (if any).

An example of API URL could be: " https://myjiveserver.com/api/core/v3/ "

Notice the following things:

- You must specify a protocol that is understandable by curl. Under normal circumstances, the protocol is limited to HTTP or HTTPS.
- If your server is under a port different than the 80, you'll need to specify it. Take " https://jserver.prv:9999/api/core/v3/ " as an example
- If your server isn't under the root URL, you can also specify the correct path: " https://myserver.prv:8888/path/to/jive/api/core/v3/ "
- The API URL should end with "/api/core/v3/" (be careful with the slash at the end)

Filters

The Jive plugin comes with a set of filters that the admin can set to filter the content the users can access through ownCloud. The drawback of using filters is that there isn't any performance gain because the filtering is mainly done in the ownCloud side, and even can degrade performance in some cases. We'll explain the filters one by one, and tell you what consequences have each one.

Category filter and separator You can filter files using one or several categories. This filter applies only to groups and files inside those groups. Your private files won't be affected by this filter.

In order to set this filter, you can provide a list of categories, all in one line. In order to separate the different categories, you must use the separator set in the "category separator" text box.

> Jive category filter : syncWithMe,sync,syncMe
>
> Jive category separator : ,

You can also achieve the same behavior with:

> Jive category filter : syncWithMe#sync#syncMe
>
> Jive category separator : #

The plugin will show all groups which have ALL those categories set. If there is a group with any of the categories missing, that group won't be shown. Anyway, you should only need to set one category.

It's important to notice that, although you can set only one category or leave the text box empty, the category separator MUST always be set. In fact, you shouldn't need to change the default value of the category separator.

Files shown inside those groups will be also affected by this filter. This means that all the files shown inside those groups must have all the categories too.

Files uploaded through ownCloud to those groups will have all the categories set in Jive automatically. If you want to add more categories to those files, you'll need to do it manually through Jive.

The usage of the category filter can degrade the performance a lot. We need to make extra calls to Jive to get the categories for each group, one extra call per group returned by Jive in the first place. There is also the limitation of not having more than 25 categories set per group. Use this filter with extreme caution.

You can "disable" this filter just by setting the category filter empty. This will prevent the extra call from being made, and will show all available groups.

Tag filter This filter works only for private files. Files inside groups won't be affected by this filter.

You can only set one tag for the files that will be shown in ownCloud. Make sure you set one of the tags from Jive as they're shown there. It's highly recommended to use only lowercase letters to set the tag to prevent possible issues when the tag is set in Jive.

The usage of this filter won't alter significantly the performance

It's important to notice that the filter will be applied to all users. Users won't be able to set their own tag to sync their own files.

This filter can also be "disabled" by setting the filter empty.

Forbidden extensions filter and separator This filter is set the same way as the category filter: you provide a list of extensions that are separated by the char set in the separator text box.

> Jive forbidden extensions: .exe,.zip,.tar.gz
>
> Jive forbidden extensions separator : ,

You can also achieve the same behavior with:

> Jive forbidden extensions: .exe#.zip#.tar.gz
>
> Jive forbidden extensions separator: #

Keep in mind that the filter is performed against the end of the filename, that's why the "." is important. If you set "exe" as a forbidden extension, a file named "texe" or "f1.lexe" will be affected by this filter.

You must also take into account that, by using only the filename, we avoid to download the file, so the performance isn't significantly degraded. On the other hand, we cannot verify that a ".png" file is what it claims to be.

This filter works for any file, and for uploads and downloads through ownCloud. This means that you won't be able to upload a file with any of those extensions from onwCloud and the Jive files which have those extensions won't be shown (and consequently they won't be downloaded). Of course, you can still upload the files from Jive (if Jive allows it) and have them there.

Maximum upload file size This filter allows you to limit the size of the files that will go through ownCloud. All files uploads and downloads will be affected by this filter. You won't be able to upload files bigger than the file size limit and the Jive files bigger than the limit won't be shown in ownCloud (and consequently they won't be downloaded)

Under normal circumstances, you want to match the limit with the one Jive has. This way you can notify errors regarding the file size faster because the files won't reach the Jive server, and at the same time you allow the users to upload up to the maximum limit that Jive allows. (Note: we can't know this limit from ownCloud, so we can't provide a sensitive default value, plus the value can change among Jive instances. You might need to adjust the value manually).

You can also set the limit to a lower value than what it's in Jive, so only small files will be delivered from ownCloud.

Show groups of which you are member Under normal circumstances, you can see all available groups in Jive, including open, member-only and private groups, only secrets groups are outside. Even if you're not a member of those groups, you can still see their contents.

For small Jive installations (less than 100 available groups per user) this is usually enough, and it has an acceptable performance. However, for larger installations, with more than 500 groups available per user, the performance is degraded a lot.

For these larger installations, this checkbox comes in handy.

Again, under normal circumstances, it's common that a user is member of just a few groups (let's say less than 25) even if there are thousand of groups available that the user can see. It usually makes sense to show the contents of only those 25 groups, not every group available.

By activating this checkbox, the user will see only those 25 groups instead of all available groups. This will increase the performance a lot, specially for larger installations, as long as each user isn't member of too many groups. Anyway, if there are user who are member of too many groups, the performance will still be degraded.

FS mount points

This Jive plugin mounts one (or two) virtual filesystems on the normal one in a transparent way.

From a user point of view, these virtual filesystems appear as new folders inside the root one.

From the settings page, you can change the mount points names. The folders will change accordingly.

Jive FS mount point The name of the folder that will hold the Jive virtual FS. The name shouldn't collide with any existing name in the root folder to prevent possible issues. The virtual FS will be mounted inside the root folder of the ownCloud FS.

As said, the contents of the folder will be the groups that the user can access through ownCloud (recheck the "filters" section).

Jive private folder The folder where your private Jive files will be stored. The name of the folder will be the same for all users, although the contents will likely differ.

This private folder will be inside the Jive mount point, as if it were another group.

Files inside this folder will be only visible to you, but they will be stored in Jive. They won't be visible neither for ownCloud users nor Jive users.

In order to prevent collisions with other groups, the folder name might be changed automatically by adding "(private)" to the end of the folder name if it's needed .

Large file sharing subsystem The large file sharing allow you to share files over the Jive limits (typically size limits). You can enable or disable this subsystem by checking or un-checking the checkbox, and provide the corresponding mount point. Use a non-existent folder name to prevent issues.

Files inside that folder will be stored inside the ownCloud server. However those files can be shared by link to Jive.

The process is like the following:

1. Upload a file (or folder) inside the large file sharing folder (by default named as "Too Big For")
2. Once the file is uploaded, click in the "share" action, and then click in the "Share link" checkbox
3. By default the share link will expire after 1 week. You can change the value and / or protect the link by password
4. Click the "Submit to Jive" button (the name can be changed depending on the actual Jive folder name)

5. A new browser tab should appear with the Jive draft ready to be edited (you might need to enter your Jive credentials first). The draft will have some predefined text, but you can edit it to your needs. Once you publish the document, it's done.

Notifications

This Jive plugin runs a connectivity check between ownCloud and Jive whenever the web page is loaded. This check allows you to know some potential issues between the ownCloud – Jive connection.

When a potential issue is detected, a notification will be shown, so you'll know what's happening.

You can control the time the notification is shown in the "notification time for the connectivity check" configuration. The time is in seconds.

13.4.2 LDAP Home Connector

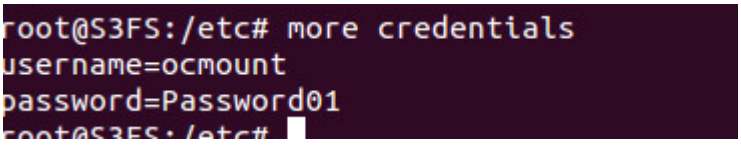
The LDAP Home Connector App provides the ability to mount a user's windows home directory to their ownCloud instance.

This document assumes the ownCloud Enterprise Subscription has been installed and the app has been downloaded from ownCloud.

Mount home directory in Ubuntu

There are two options to mount the home directory.

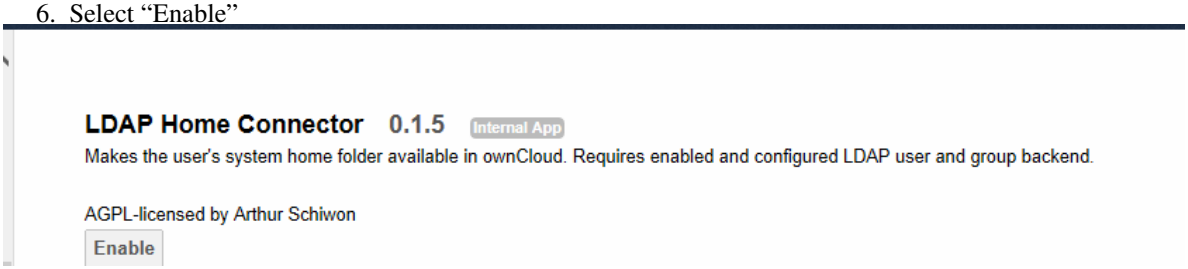
1. This is not persistent across reboots and will need to be entered after each reboot of the Ubuntu server
   - #mount –t cifs –o username=<user>,password=<password>,uid=www-data,gid=www-data //<ip>/<directory> <directory>
     - #mount –t cifs –o username=ocmount,password=Password01,uid=www-data,gid=www-data //192.168.1.58/share /mnt/share
2. Modifying the /etc/fstab file will keep the mount across reboots
   - Add the following line to the fstab file
     - //<ip>/<directory> <directory> cifs credentials=<credential file>,uid=33,gid=33
       - //192.168.1.58/share /mnt/share cifs credentials=/etc/credentials,uid=33,gid=33
   - Then create a <credential file> with the following
     - Username=<user>
     - Password=<password>
       - /etc/credentials is as follows:



Configure ownCloud

Install the app

1. ftp the app package (eg enterprise_files_ldap_home-x.y.z.tar.bz2) to the apps directory of your ownCloud instance
2. Decompress the app package
   - tar jxvf enterprise_files_ldap_home-x.y.z.tar.bz2
3. The decompression creates an enterprise directory – Under the enterprise directory exists the files_ldap_home directory. Move this to up one level
   - #cd enterprise
   - #mv –R files_ldap_home ..
4. Login to your ownCloud instance as admin and proceed to the apps page
5. Find the LDAP Home Connector app on the left and select it
6. Select “Enable”



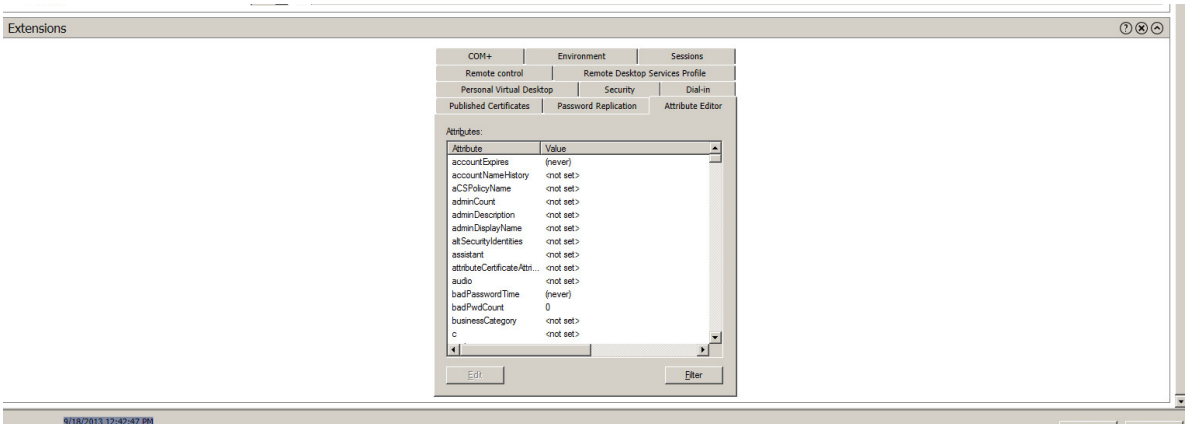
Configure the App

1. Navigate to the Admin page and scroll to the “LDAP User Home” section

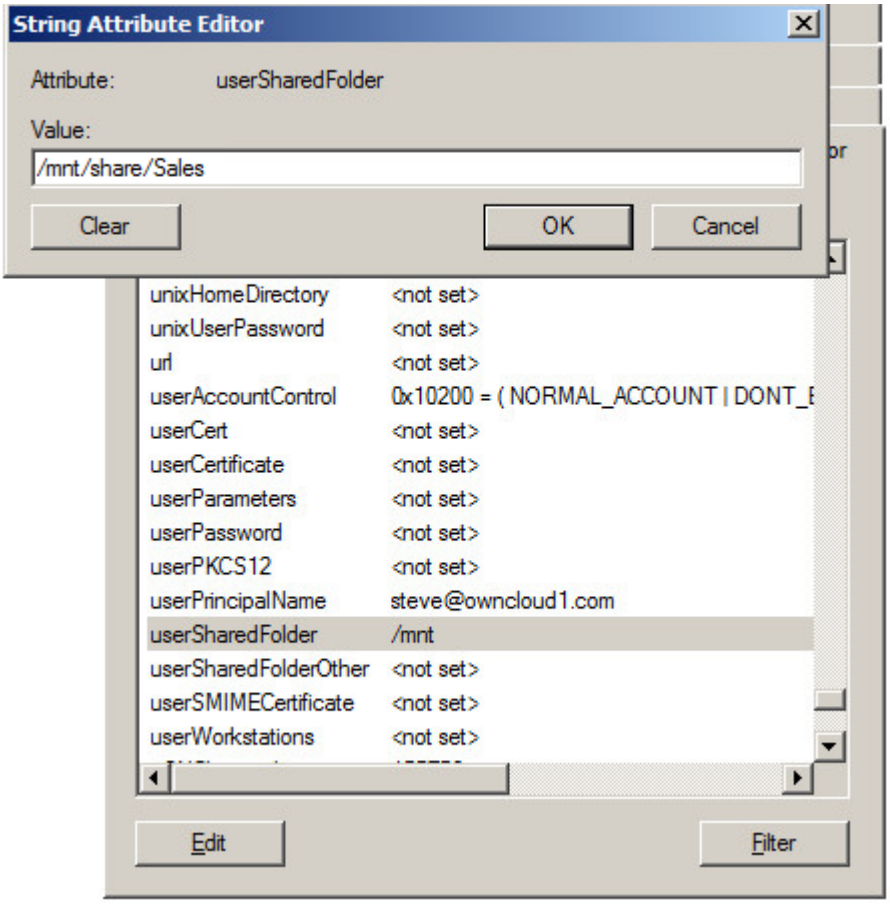
2. Fill in the name that you would like to display the folder to the user as in “Display Folder As”
3. Fill in the attribute name that will contain the homeDirectory. Use any LDAP attribute that is not already in use, In this document we will use the UserSharedFolder attribute.
4. Select “Save”

Configure the LDAP server

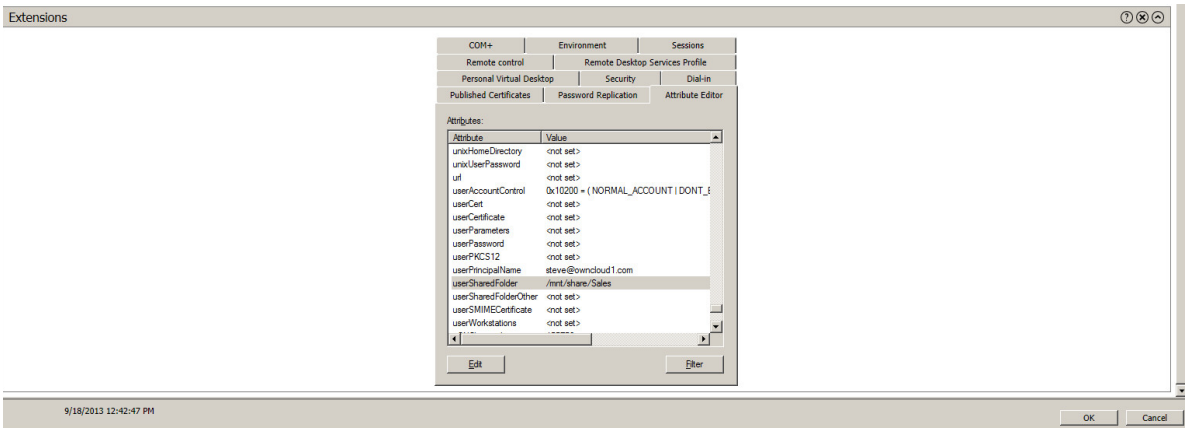
1. In Active directory, open the user profile
2. Scroll to the “Extensions” section and select the “Attribute Editor” tab



3. Scroll to the attribute being used (UserSharedFolder in this instance)
4. Select Edit
5. Enter the user's home directory (from the mount)



6. Select OK
7. Select OK at the bottom of the user page



13.4.3 Configuring S3 and OpenStack Swift Objects as Primary Storage

In ownCloud Enterprise Subscription, you can configure S3 objects as primary storage. This replaces the default data directory, which is `/var/www/owncloud/data` on default Linux installations, and `C:\inetpub\wwwroot\owncloud\data` on Windows servers. However, you may need to keep the `data/` directory for these reasons:

- The ownCloud log file is saved in the data directory
- Legacy apps may not support using anything but the `data/` directory

You can move your logfile by changing its location in `config.php`. You may still need `data/` for backwards compatibility with some apps.

Implications

It is important to note that ownCloud in object store mode will expect exclusive access to the object store container, because it only stores the binary data for each file. The metadata is currently kept in the local database for performance reasons.

The current implementation is incompatible with any app that uses direct file I/O and circumvents the ownCloud virtual filesystem. That includes Encryption and Gallery. Gallery stores thumbnails directly in the filesystem, and Encryption causes severe overhead because key files need to be fetched in addition to any requested file.

Configuration

Look in `config.sample.php` for a example configurations. Copy the relevant part to your `config.php` file. Any objectstore needs to implement `\\OCP\\Files\\ObjectStore\\IObjectStore` and can be passed parameters in the constructor with the `arguments` key:

```
'objectstore' => array(
    'class' => 'Implementation\\Of\\OCP\\Files\\ObjectStore\\IObjectStore',
    'arguments' => array(
        ...
    ),
),
```

Amazon S3

The S3 backend mounts a bucket of the Amazon S3 object store into the virtual filesystem. The class to be used is `OCA\ObjectStore\S3`:

```
'objectstore' => array(
  'class' => 'OCA\ObjectStore\S3',
      'arguments' => array(
        'key' => 'yourkey',
        'secret' => 'yoursecret',
        'bucket' => 'your-oc-bucket',
      ),
),
```

Ceph S3

The S3 backend can also be used to mount the bucket of a ceph object store via the s3 API into the virtual filesystem. The class to be used is `OCA\ObjectStore\S3`:

```
'objectstore' => array(
  'class' => 'OCA\ObjectStore\S3',
      'arguments' => array(
        'key' => 'GEZ550B06Z2ZDB52CT21',
        'secret' => '6Vdo7ObSMBlI4TMRw0jpRE75K6qS9QNTk6nBboxP',
        'bucket' => 'devobjectstore',
        'base_url' => 'http://ceph/',
        'hostname' => 'ceph',
        // you must use this region or the amazon lib will overwrite
        // the path style when resetting the region
        'region' => 's3-eu-west-1.amazonaws.com'
      ),
),
```

OpenStack Swift

The Swift backend mounts a container on an OpenStack Object Storage server into the virtual filesystem. The class to be used is `\\OC\\Files\\ObjectStore\\Swift`:

```
'objectstore' => array(
    'class' => 'OC\\Files\\ObjectStore\\Swift',
    'arguments' => array(
        'username' => 'demo',
        'password' => 'password',
        'container' => 'owncloud',
        'autocreate' => true,
        'region' => 'RegionOne',
        'url' => 'http://devstack:5000/v2.0',
        'tenantName' => 'demo',
        'serviceName' => 'swift',
    ),
),
```

13.4.4 Configuring SharePoint Integration

Native SharePoint support has been added to the ownCloud Enterprise Subscription 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.

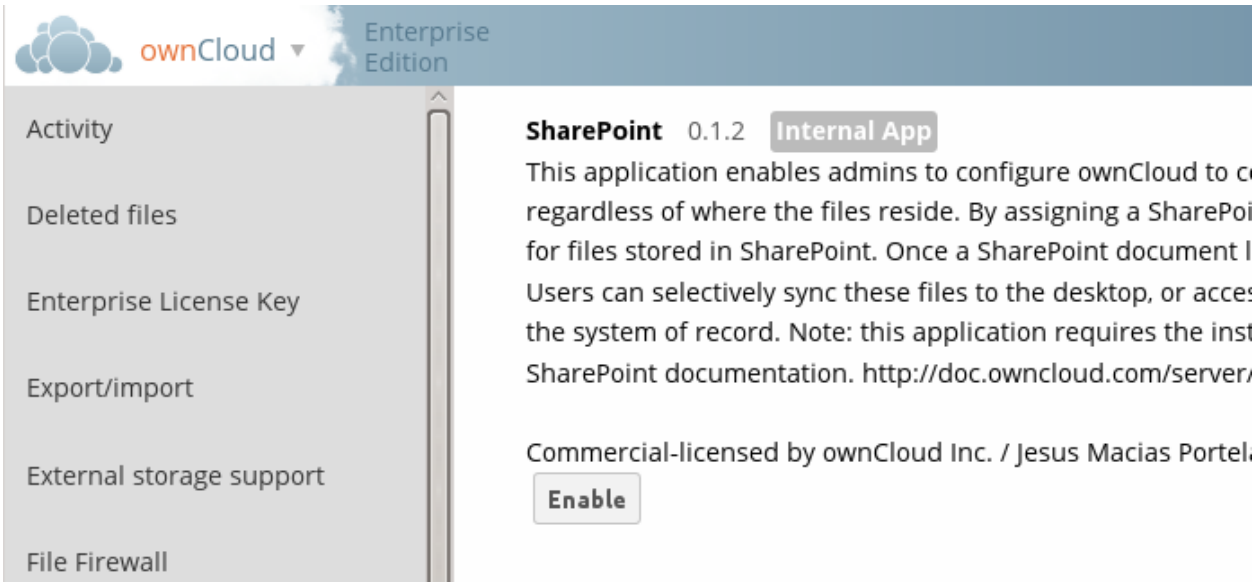
The plugin uses the Simple Object Access Protocol (SOAP) and WebDAV for the uploads and downloads to talk to SharePoint servers. Your ownCloud server must have `php-soap` or `php5-soap` installed. Most Linux distributions and Windows call the package `php-soap`, though there may be some Linux variants that call it `php5-soap`. Starting with ownCloud 7.0.2 EE, Linux packages and ownCloud appliances will install `php5-soap` as a required dependency.

The supported authentication methods are:

- Basic Auth
- NTLM (Recomended)

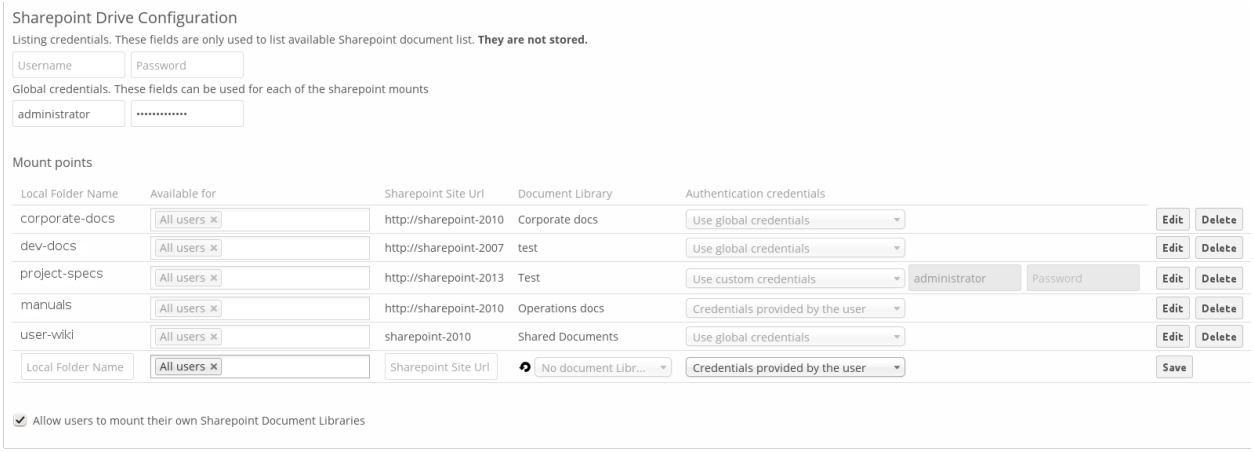
Enabling the SharePoint Plugin

The SharePoint plugin is a native plugin, so the first step is to enter the `Apps` administration page and enable it.



Next, enter the `Admin` panel to set up SharePoint connections in the `SharePoint Drive Configuration` section.

- First, 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 an optional field. If you fill in this field, these credentials will be used on on all SharePoint mounts where you select: "Use global credentials" as the Authentication credentials



- Enter your ownCloud mount point 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 `use 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

Please see `Connecting to SharePoint` in the User Manual to learn how to use your new SharePoint connections.

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

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

Turn on Sharepoint app logging by modifying the following line in `apps/sharepoint/lib/sharepoint.php` to `TRUE`:

```
private static $enableLogs = TRUE;
```

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.

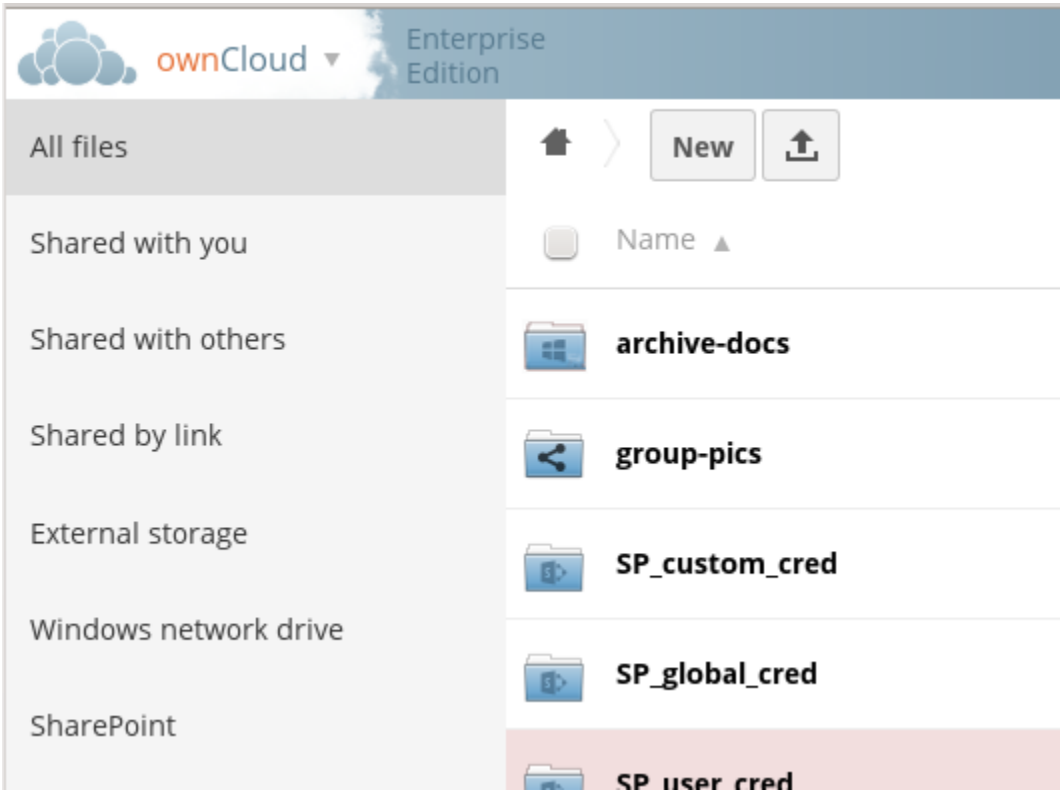
Personal mount points can't be accessed: You have to fill your SharePoint credentials as User on the personal settings page in case your personal mount point doesn't have its own credentials.

A user can't update the credentials: Verify that the correct credentials are configured, and the correct type, either global or custom.

13.4.5 Installing and Configuring the Windows Network Drive App

The Windows Network Drive app creates a control panel on your Admin page for seamless mounting of SMB/CIFS file shares on ownCloud servers that run on Linux. It does not work on Windows IIS or Windows Apache setups, but only Linux servers, because it requires the Samba client. (Samba is the free software implementation of the SMB/CIFS networking protocol.)

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.



Files are synchronized bi-directionally, and you can create, upload, and delete files and folders. You have the option of allowing users to create personal mounts of their own SMB/CIFS shares, and controlling whether they can share them.

ownCloud server admins can create Windows Network Drive mounts, and optionally allow users to create their own personal Windows Network Drive mounts. The password for each mount is encrypted and stored in the ownCloud database, using a long random secret key stored in `config.php`. This allows ownCloud to access the shares when the users who own the mounts are not logged in.

Installation

Enable the Windows Network Drive app on your ownCloud Apps page. Then there are a few dependencies to install.

You must install the ownCloud `php5-libsmbclient` binary; please refer to the README in your customer.owncloud.com account for instructions on obtaining it.

You also need the Samba client installed on your Linux system. This 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`.

Additional Installation Steps

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 in order 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.

Admin-created SMB Mounts

When you create a new SMB share you need the login credentials for the share, the server address, the share name, and the folder you want to connect to.

1. First enter the ownCloud mountpoint for your new SMB share. This must not be an existing folder.
2. Then enter which ownCloud users or groups get access to the share
3. Next, enter the address of the server that contains the SMB share
4. Then the Windows share name
5. Then the root folder of the share

You have four options for login credentials:

- **Global credentials**. Uses the credentials set in the Global credentials fields.
- **User credentials**. For admin-created global mountpoints; users must click on the share and then enter their personal credentials to access the share.
- **Login credentials** is for users to connect to the mountpoint using their Windows domain login credentials.
- **Custom Credentials**. When you select this, a form appears for entering your custom login and password for the share. You must supply this login to users who are granted access to the share.

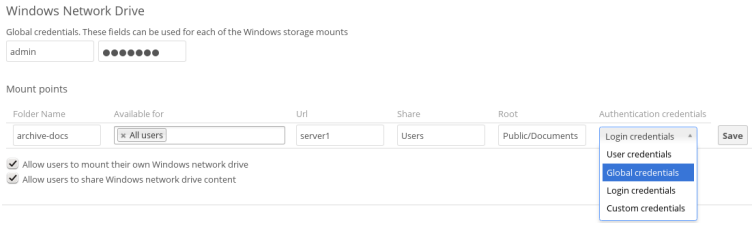


Figure 13.1: *Click to enlarge*

Note: 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.

Personal SMB Mounts

Users create their own personal SMB mounts on their Personal pages. These are created the same way as Admin-created shares. Users have only two options for login credentials:

- **Personal credentials**. Your own login to the share.
- **Custom credentials**. When you select this, a form appears for entering your custom login and password for the share.

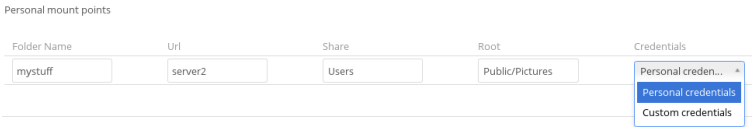


Figure 13.2: *Click to enlarge*

To share your personal SMB mounts, go to your Files page and share your SMB files or folders just like any other file or folder. You should use custom credentials on shared mounts so that you do not give away your own Windows network drive login.

13.5 User Management (ES only)

13.5.1 Shibboleth Integration (Enterprise Subscription only)

Introduction

The ownCloud Shibboleth user backend application integrates ownCloud with a Shibboleth Service Provider (SP) and allows operations in federated and single-sign-on infrastructures. Setting up Shibboleth has three steps:

1. Create the appropriate Apache configuration
2. Enable the Shibboleth app
3. Enable Shibboleth on your ownCloud admin page

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.

The ownCloud Desktop Client and mobile clients can interact with an ownCloud instance running inside a Shibboleth Service Provider by using built-in browser components for authentication against the IdP.

The ownCloud desktop sync client and mobile apps store users' logins, so your users only need to enter their logins the first time they set up their accounts.

Note: The ownCloud clients may use only a single Shibboleth login per ownCloud server; multi-account is not supported with Shibboleth.

These screenshots show what the user sees at account setup. Figure 1 shows a test Shibboleth login screen from Testshib.org on the ownCloud desktop sync client.

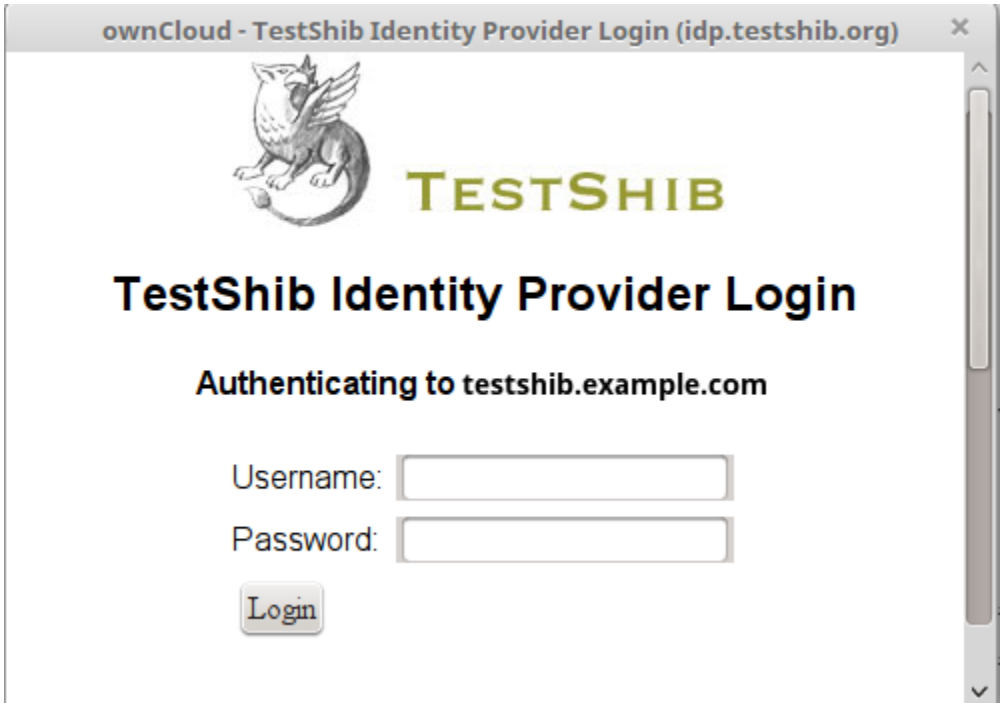


Figure 13.3: *figure 1: First login screen*

Then after going through the setup wizard, the desktop sync client displays the server and login information just like it does for any other ownCloud server connections.

To your users, it doesn't look or behave differently on the desktop sync client, Android app, or iOS app from an ordinary ownCloud account setup. The only difference is the initial setup screen where they enter their account login.

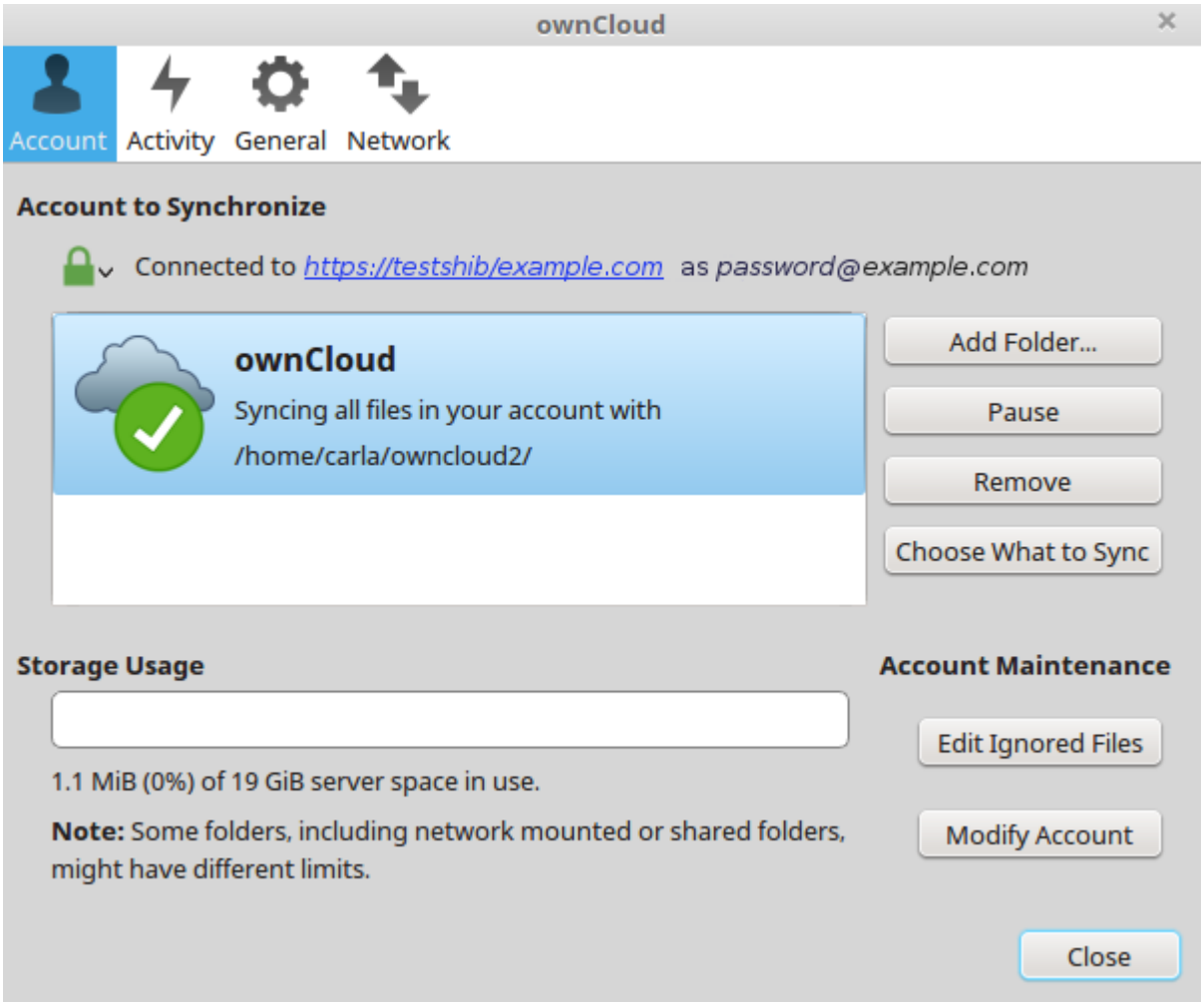


Figure 13.4: *figure 2: ownCloud client displays server information*

Apache Configuration

This is an example configuration as installed and operated on a Linux server running the Apache Web server. These configurations are highly operating system specific and require a high degree of customization.

See the documentation Wiki for more configuration examples.

The ownCloud instance itself is installed in `/var/www/owncloud/`. The following aliases are defined in an Apache virtual host directive:

```
# non-Shibboleth access
Alias /owncloud /var/www/owncloud/
# for Shibboleth access
Alias /oc-shib /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_22.so

#
# Ensures handler will be accessible.
#
<Location /Shibboleth.sso>
  Satisfy Any
  Allow from all
</Location>

#
# Configure the module for content.
#
# Shibboleth is disabled for the following location to allow non
  shibboleth webdav access
<Location ~ "/oc-shib/remote.php/nonshib-webdav">
  Satisfy Any
  Allow from all
  AuthType None
  Require all granted
</Location>

# Shibboleth is disabled for the following location to allow public link
  sharing
<Location ~
  "/oc-shib/(status.php$
  |public.php$
  |cron.php$
  |core/img/
  |index.php/apps/files_sharing/publicpreview.png$
  |index.php/apps/files/ajax/upload.php$
  |index.php/core/ajax/translations.php$
  |apps/files/templates/fileexists.html$
  |index.php/apps/files/ajax/mimeicon.php$)">
  Satisfy Any
  Allow from all
  AuthType None
  Require all granted
</Location>
```

```
# Shibboleth is disabled for the following location to allow public gallery
  sharing
<Location ~
 "/oc-shib/(apps/gallery/templates/slideshow.html$
 |index.php/apps/gallery/ajax/getimages.php
 |index.php/apps/gallery/ajax/thumbnail.php
 |index.php/apps/gallery/ajax/image.php)">
  Satisfy Any
  Allow from all
  AuthType None
  Require all granted
</Location>

# Shibboleth is disabled for the following location to allow public link
  sharing
<Location ~ "/oc-shib/.*\.css">
  Satisfy Any
  Allow from all
  AuthType None
  Require all granted
</Location>

# Shibboleth is disabled for the following location to allow public link
  sharing
<Location ~ "/oc-shib/.*\.js">
  Satisfy Any
  Allow from all
  AuthType None
  Require all granted
</Location>

# Besides the exceptions above this location is now under control of
  Shibboleth
<Location /oc-shib>
  AuthType shibboleth
  ShibRequireSession On
  ShibUseHeaders Off
  ShibExportAssertion On
  require valid-user
</Location>
```

Application Configuration

After installing and enabling the Shibboleth application there are three configuration variables to set up, depending on the data sent back by the IdP. The configuration is stored in `apps/user_shibboleth`.

```
namespace OCA\user_shibboleth {
        const SHIB_SESSION_ID = 'Shib-Session-ID';
        const SHIB_EPPN = 'eppn';
        const SHIB_EMAIL = 'eppn';
        const SHIB_DISPLAY_NAME = 'eppn';
}
```

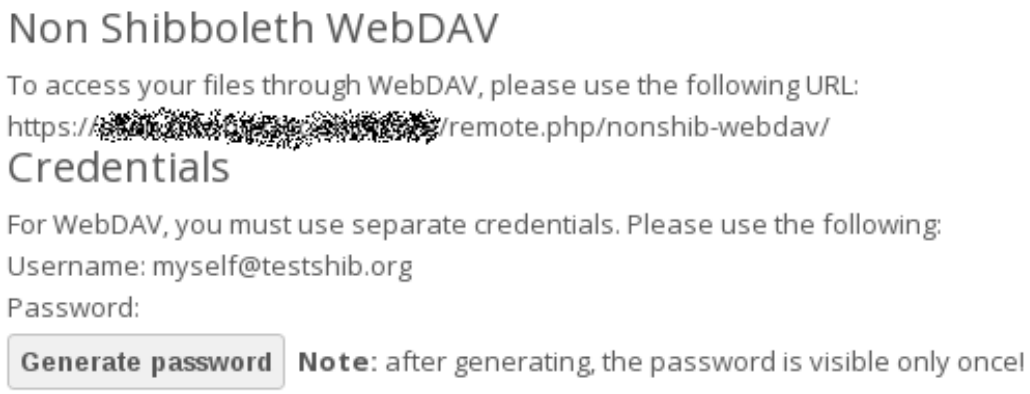
Parameter	Description
SHIB_SESSION_ID	This constant defines the name of the environment variable holding the Shibboleth session id.
SHIB_EPPN	This constant defines the name of the environment variable which holds the EPPN (eduPersonPrincipalName). This is the unique user identifier.
SHIB_EMAIL	The environment variable with this given name holds the email address of the logged-in user.
SHIB_DISPLAY_NAME	This constant defines the name of the environment variable holding the user's display name.

Enabling the Shibboleth App

You must enable the Shibboleth app on your Apps page, and then check **Activate Shibboleth** and click the **Save** button on your ownCloud Admin page. The system information displayed on your Admin page may be useful for troubleshooting; for example you can copy and include it on a support ticket.

WebDAV Support

Users of standard WebDAV clients can use an alternative WebDAV Url, for example `https://cloud.example.com/remote.php/nonshib-webdav/` to log in with their username and password. The password is generated on the Personal settings page.



Note: In pure SSO mode the WebDAV password feature will not work, as we have no way to store the WebDAV password. It does work in auto-provision mode.

For provisioning purpose an OCS API has been added to revoke a generated password for a user:

Syntax: `/v1/cloud/users/{userid}/non_shib_password`

- HTTP method: DELETE

Status codes:

- 100 - successful
- 998 - user unknown

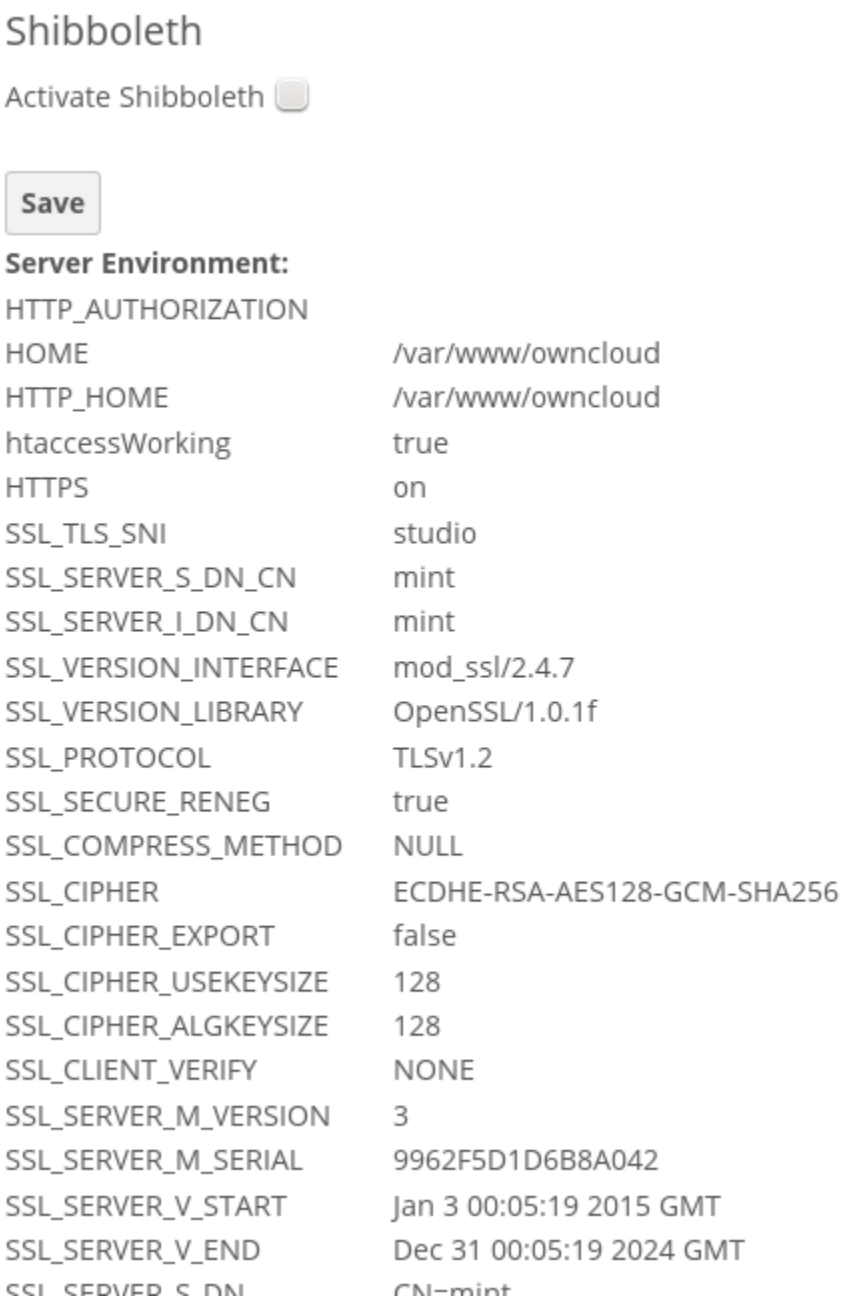


Figure 13.5: *figure 3: Enabling Shibboleth on the Admin page*

Known Limitations

Encryption

File encryption can not be used together with Shibboleth because the 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. Currently, we have no solution to this limitation.

Other Login Mechanisms

Shibboleth is not compatible with any other ownCloud user backend because the login process is handled outside of ownCloud.

You can allow other login mechanisms (e.g. LDAP or ownCloud native) by creating a second Apache virtual host configuration. This second location is not protected by Shibboleth, and you can use your other ownCloud login mechanisms.

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.

The session timeout can be overridden in the service provider, but this requires a source code change of the Apache Shibboleth module. A patch can be provided by the ownCloud support team.

13.6 Enabling Anonymous Uploads with Files Drop (ES Only)

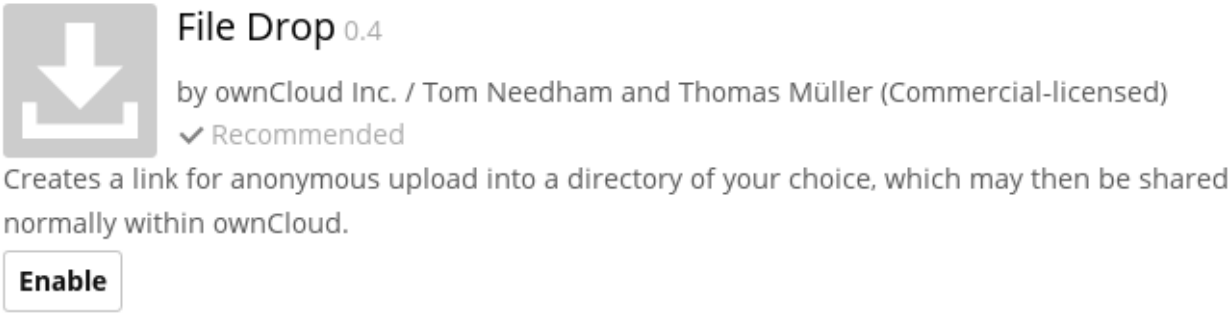
13.6.1 Enabling Anonymous Uploads with Files Drop (ES Only)

The Files Drop application, introduced in ownCloud 8.0.3 Enterprise Subscription, allows anyone to upload files with the click of a button to the directory of your choosing, without needing a login, and they cannot see or change the contents of the directory. It is the perfect replacement for attaching large files to email, maintaining an FTP server, and commercial file-sharing services.

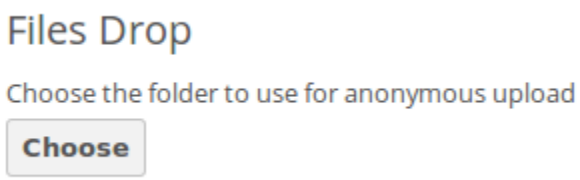
When files are uploaded to your Files Drop directory, you can manage them just like any other ownCloud share: you may share them, restrict access, edit, and delete them.

Setting Up the Files Drop App

Setting up Files Drop is a matter of a few clicks. First go to your Apps page and enable it.



Now your users will see a configuration section on their Personal pages.



Click the **Choose** button to open a dialog to select your upload directory. You may wish to first create a special upload directory (on your Files page), which in the following example is name **upload**.

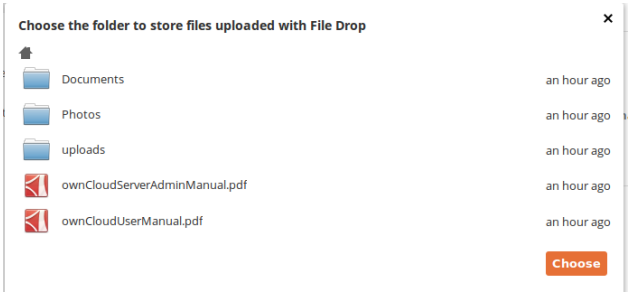
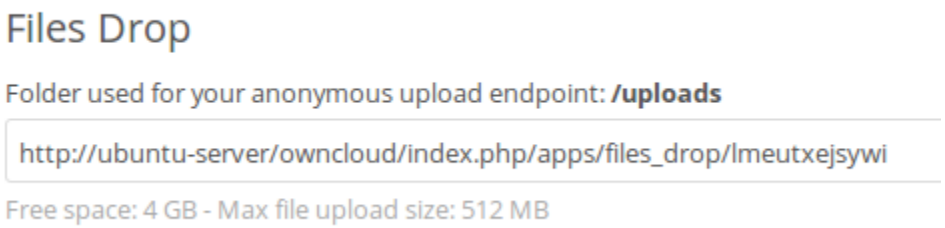


Figure 13.6: *Click to enlarge*

On your Personal page you should now see a URL for your upload directory. Share this URL with anyone you want to allow uploads to your File Drop folder. Note that the maximum upload size in this example is 512MB. (The default ownCloud upload file size limit is 512MB. See *Uploading big files > 512MB* to learn how to customize this.)



Using the Files Drop App

Uploading files via the Files Drop app is simple. Open your Web browser to the share URL created by ownCloud:

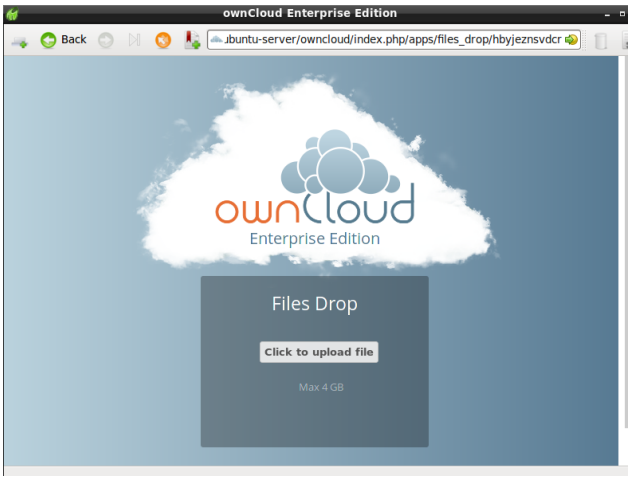


Figure 13.7: *Click to enlarge*

Click the **Click to upload file** button. This opens a file picker, and you select the file or directory you want to upload.

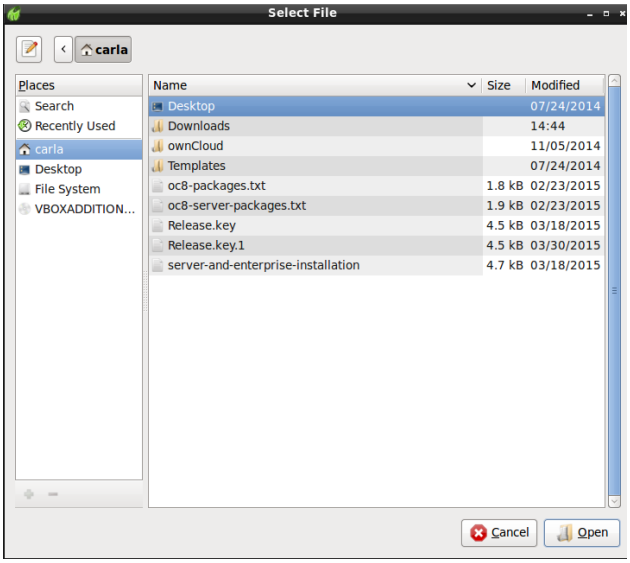


Figure 13.8: *Click to enlarge*

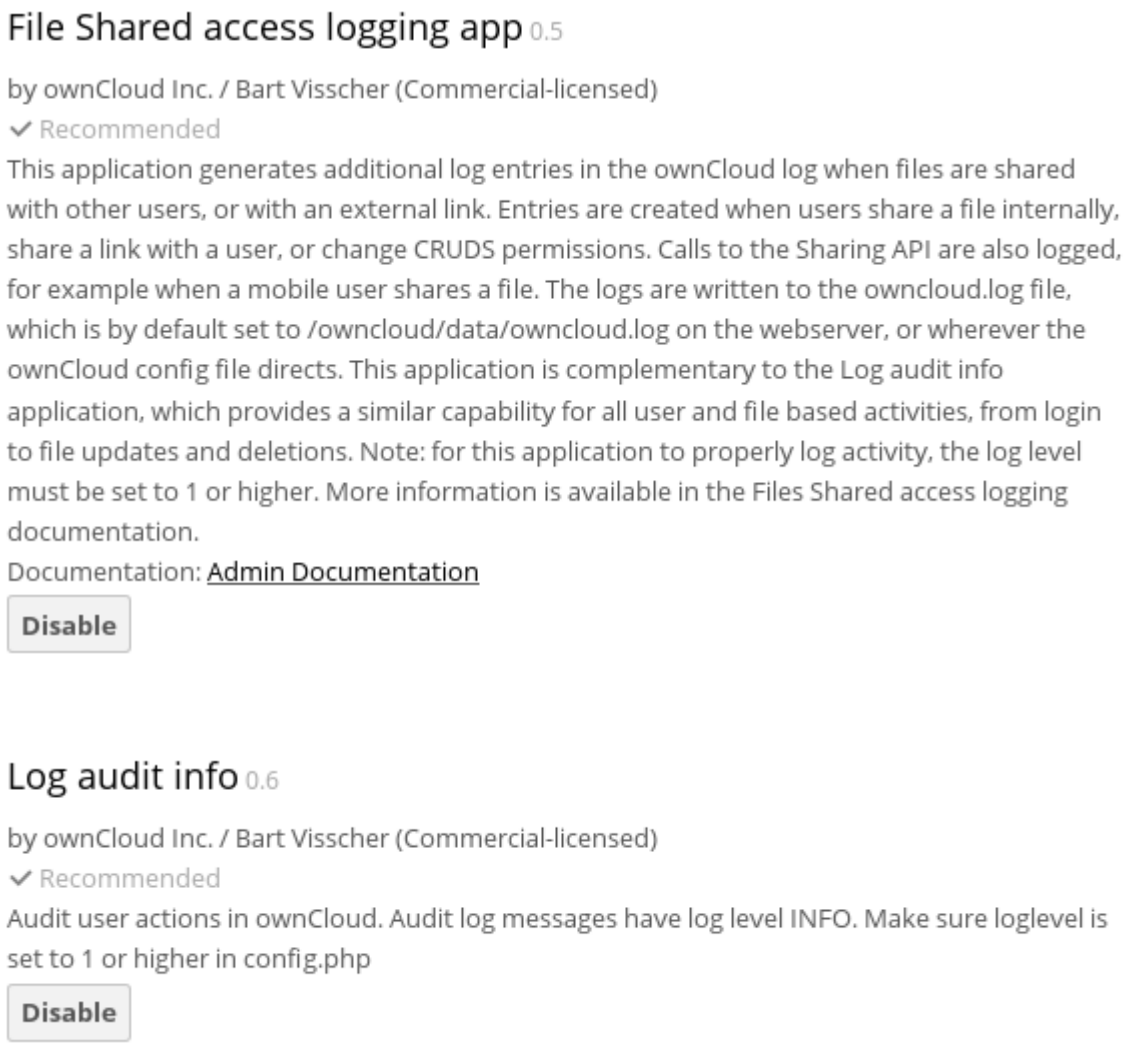
When your upload is completed, you'll see a confirmation message with the filenames.


ownCloud
Enterprise Edition
Files Drop
Click to upload file
Max 512 MB
Following files have been uploaded:
Release.key
server-and-enterprise-installation

13.7 Enterprise Logging Apps (ES only)

13.7.1 Enterprise Logging Apps

There are two enterprise logging apps available to ownCloud Enterprise Subscription customers: **File Shared access logging app** and **Log audit info**. The **File Shared access logging app** records the file sharing activity of your users, and **Log audit info** records user logins and logouts.



These two apps work together, and should be enabled together. 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. This shows which logging app recorded the entries, timestamps, usernames, and their activities:

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`. This is what the raw log looks like:

```
{"reqId":"uaG6sHiutvgzVUCUXM3W","remoteAddr":"::1","app":"admin_audit",
"message":"Rename \"\/molly\/files\/server-and-enterprise-installation\" to
```

Info	sharing_log	User molly shared "/shared" with the user layla, permissions: READ UPDAT E CREATE DELETE SHARE	2015-10-21T22:07:39+00:00
Info	admin_audit	Rename "/molly/files/shared/San%20Francisco.jpg" to "/molly/files/shared /SanFrancisco.jpg" by user molly, owner: molly	2015-10-21T22:07:25+00:00
Info	admin_audit	Rename "/molly/files/server-and-enterprise-installation" to "/molly/files/sh ared/server-and-enterprise-installation" by user molly, owner: molly	2015-10-21T22:07:14+00:00


```
\"\/molly\/files\/shared\/server-and-enterprise-installation\" by user molly,
owner: molly","level":1,"time":"2015-10-21T22:07:14+00:00"}
{"reqId":"Krsnp8BgtLCtuT4zLTWs","remoteAddr":"::1","app":"admin_audit",
"message":"Rename \"\/molly\/files\/shared\/San%20Francisco.jpg\" to
\"\/molly\/files\/shared\/SanFrancisco.jpg\" by user molly, owner:
molly","level":1,"time":"2015-10-21T22:07:25+00:00"}
```


See *Logging Configuration* for more information on logging.