ownCloud Administrators Manual

Release 7.0

The ownCloud developers

August 11, 2015

CHAPTER

ONE

INTRODUCTION

Welcome to the ownCloud Administrator Guide. This guide describes administrator tasks for ownCloud; a flexible, open source, file synchronization and sharing solution. ownCloud is comprised of a server running on either a Linux or Microsoft Windows platform as well as client applications for Microsoft Windows, Mac OS X and Linux (Desktop Client) and mobile clients for both the Android and Apple iOS operating system.

1.1 Target Audience

This guide is targeted towards people who want to install, administer, and optimize the ownCloud server. If you want to learn more about the ownCloud Web user interface or how to install clients on the server, refer to the following:

- User Manual
- Desktop Client Manual

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 Document Structure

This document is broken out into three major sections – Installation, Configuration, and Maintenance. The Issues sections has instructions for reporting bugs. The following sections provide detailed information about various tasks associated with each of these sections.

1.3.1 Installation

This section provides detailed instructions on how to install ownCloud in different scenarios. It contains the following topics:

- *ownCloud Appliances*
- *Installing and Managing Apps*
- *Hiawatha Configuration*
- *Installation Wizard*

- *Lighttpd Configuration*
- *Preferred Linux Installation Method* (recommended)
- *Mac OS X* (not supported)
- *Nginx Configuration*
- *Other Installation Methods*
- *Manual Installation on Linux*
- *Univention Corporate Server*
- *Windows 7 and Windows Server 2008*
- *Yaws Configuration*
- *SELinux Configuration*

Note: If you just want to try out ownCloud in a virtual machine, without any configuration, refer to *ownCloud Appliances*. For your convenience, this topic contains ready-to-use images.

1.3.2 Configuration

This section describes how to configure ownCloud and your Web server. It contains the following topics:

- *Configuring the ClamAV Antivirus Scanner*
- *Automatic Configuration Setup*
- *Defining Background Jobs*
- *Uploading big files > 512MB (as set by default)*
- *Configuring the Collaborative Documents App*
- *Config.php Parameters*
- *Custom Client Configuration*
- *Database Configuration*
- *Email Configuration*
- *Configuring External Storage (GUI)*
- *Configuring External Storage (Configuration File)*
- *File Sharing*
- *Files Locking App Configuration*
- *JavaScript and CSS Asset Management*
- *Knowledge Base Configuration*
- *Language Configuration*
- *Logging Configuration*
- *Previews Configuration*
- *Reverse Proxy Configuration*
- *Enabling Full-Text Search*

- *Encryption Configuration*
- *Configuring Server-to-Server Sharing*
- *Serving Static Files for Better Performance*
- *Using Third Party PHP Components*
- *User Authentication with IMAP, SMB, and FTP*
- *User Authentication with LDAP*
- *LDAP User Cleanup*
- *User Management*
- *Resetting a Lost Admin Password*

1.3.3 Maintenance

This sections describes the maintenance tasks associated with the ownCloud server (for example, updating or migrating to a new version of ownCloud). It contains the following topics:

- *Backing up ownCloud*
- *Converting From SQLite to MySQL, MariaDB, or PostgreSQL*
- *Maintenance Mode Configuration*
- *Migrating ownCloud Installations*
- *Restoring ownCloud*
- *Updating ownCloud with the Updater App*
- *Upgrading Your ownCloud Server*

1.3.4 Issues

What to do when you have problems, and where to report bugs.

- *Issues and Troubleshooting*

CHAPTER

TWO

OWNCLOUD VIDEOS

Please visit our YouTube channel for howtos, demos, news, and Webinars for both the Community and Enterprise versions of ownCloud.

2.1 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.

2.2 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.

2.3 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.

CHAPTER

THREE

WHAT'S NEW FOR ADMINS IN OWNCLOUD 7

3.1 New User Management

Admins can now view all ownCloud users in a single scrolling window, filter user lists by group, and search by user display name using the new text filter. User attributes have also been added, included the file storage location for each user and the last time they logged in. New groups can be added with a click of a button.

3.2 External Storage

Major improvements to the external storage app include support for FTP, Dropbox, Google Drive, SFTP, Swift, S3, WebDAV, SMB/CIFS and more storage locations to the ownCloud instance. You can control which storage types your users can set up in their Personal tabs. Further performance improvements have made externally mounted storage faster and more responsive.

3.3 Object Stores as Primary Storage

Primary storage in ownCloud is where all files and folders are stored by default. In contrast to secondary storage, primary storage is completely managed by the ownCloud application. With ownCloud 7, ownCloud can now leverage SWIFT and S3 (S3 is enterprise only) object stores as primary storage for ownCloud files. Now admins can choose the best option for their specific need, including local storage, network file system mounts, and object stores.

3.4 Server to Server Sharing

ownCloud 7 servers can now connect shares with each other. With just a few clicks you can easily and securely create public shares that are available to other ownCloud 7 users on remote servers, and optionally allow your users to also create their own public shares.

3.5 SharePoint Integration (Enterprise Edition only)

Native SharePoint support has been added to ownCloud 7 Enterprise Edition as a secondary storage location for SharePoint 2007, 2010 and 2013. When this is enabled, users can access and sync all of their SharePoint content via ownCloud, whether in the desktop sync, mobile or Web interfaces. Updated files are bi-directionally synced automatically. SharePoint shares are created by the ownCloud admin, and optionally by any users who have SharePoint credentials. ownCloud preserves SharePoint ACLs to ensure content is restricted per SharePoint rules.

3.6 Windows Network Drive Integration (Enterprise Edition only)

ownCloud has always supported mounting Windows network drives, and in OC7 EE it is easier than ever for the administrator to mount Windows Network Drives for a user, a group or the entire ownCloud instance, and allow each user to access the network drives and preserve their ACLs. The network drives appear as normal folders and files, and changes are bi-directionally synced between user devices and the Windows network drives.

3.7 Sharing

Sharing has been dramatically enhanced and streamlined, making it more flexible, faster and accessible. Improvements include:

- **Force Password:** Admins can now force users to set a password when they create shared links. This ensures that files shared outside of ownCloud via a link are properly secured by users.
- **Share Link Default and Max Expiration:** When sharing a file with a link, admins can now require users to set a specific expiration duration for the link.
- **Antivirus Action Updates:** The Antivirus app has been enhanced to allow – with some minor customization – the use of external virus scanners (rather than the default ClamAV) in scanning files as they arrive on the server.
- **The Shared folder has been removed from new installations of ownCloud 7:** Shared files now appear in the top level of your file tree on your Files page, and you can change the default shared folder to any folder with the `'share_folder'` directive in `config.php`. If you are upgrading from older ownCloud versions you will still have your old Shared folder.
- **Local shares do not expire with public 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.

3.8 Email Configuration Wizard

The new graphical Email configuration wizard connects to your mail server in just a few clicks, so that ownCloud can send automated messages to users. ownCloud connects via PHP, Sendmail, or standard SMTP.

3.9 Editable Email Templates

ownCloud admins can now edit the email templates that ownCloud uses for automatic notifications on the Admin page.

3.10 Active Directory and LDAP Enhancements

Several improvements have been made to the LDAP and Active Directory plug-in application, improving both the performance of the application as well as the compatibility with OpenLDAP and Active Directory.

CHAPTER

FOUR

OWNCLOUD 7.0 RELEASE NOTES

4.1 Recommended Setup for Running ownCloud

ownCloud runs on multiple operating systems and HTTP servers. For best performance, stability, support, and full functionality we recommend:

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

4.2 Supported Platforms

- Hypervisors: Hyper-V, VMware ESX, Xen, KVM
- Server: Windows 2008R2; 2012, Linux (Debian 6, RHEL / Centos 6 and 7, Ubuntu 12 and 14, SLES 11, Univention Corporate Server 3.x)
- Databases: MySQL 5.x; Microsoft SQL Server 2008R2; SQL Server 2012 R2; Oracle 11g
- Desktop: Windows XP, Windows 7, Mac OS X 10.7+ (64-bit only), Linux (CentOS 6, Ubuntu 10.04 and 11.04+, Fedora 16+, openSUSE 11.4+)
- Mobile apps: iOS 7+, Android 4+
- Webserver: IIS (Windows) and Apache2 (Windows and Linux)
- Web browser: IE8+ (but not Compatibility Mode), Firefox v14+, Chrome v18+, Safari v5+

4.2.1 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.

4.2.2 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.

4.2.3 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.

4.2.4 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.

4.2.5 “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.

4.2.6 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.

4.2.7 4GB Limit on SFTP Transfers

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

4.2.8 “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`.

4.2.9 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.

4.2.10 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.

4.3 Enterprise 7 Only

4.3.1 No Federated Cloud Sharing with Shibboleth

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

4.3.2 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.

4.3.3 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.

4.3.4 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

4.3.5 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.

4.3.6 No App Store

The App Store is disabled for the Enterprise Edition.

4.3.7 LDAP Home Connector Linux Only

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

CHAPTER

FIVE

INSTALLATION

5.1 Preferred Linux Installation Method

5.1.1 Supported Distribution Packages

Installing ownCloud 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. Ready-to-use packages are available at the ownCloud repository for a variety of Linux distributions. 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*).

You can still get 7.0.5, which is the latest 7.0 version. It is recommended to use the latest stable 8.x version, but if you wish to roll back an upgrade to 8.x, or delay upgrading, use this direct link to 7.0.5.

Note: Please don't move the folders provided by this packages after the installation. This will break further updates.

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*).

5.1.2 Additional Installation Guides and Notes

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

Archlinux: The are two packages for ownCloud: stable version in the official community repository and development version in AUR.

PCLinuxOS: Follow the Tutorial ownCloud, installation and setup on the PCLinuxOS web site.

Debian/Ubuntu: The package is installing an additional Apache config file to */etc/apache2/conf.d/owncloud.conf* which contains an *Alias* to the owncloud installation directory as well as some more needed configuration options.

5.2 ownCloud Appliances

If you are looking for virtual machine images, check the Software Appliances section. The Hardware Appliances section is of interest for people seeking to run ownCloud on appliance hardware (i.e. NAS filers, routers, etc.).

5.2.1 Software Appliances

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

- SUSE Studio, ownCloud on openSuSE, runnable directly from an USB stick.
- Ubuntu charm, ownCloud
- Amahi home server

5.2.2 ownCloud on Hardware Appliances

These are tutorials provided by the user communities of the respective appliances:

- ownCloud 7 on Raspberry Pi (Arch Linux) using Lighttpd for the popular credit-card sized computer
- QNAP Guide for QNAP NAS appliances
- OpenWrt Guide for the popular embedded distribution for routers and NAS devices.
- Synology Package for Synology NAS products

Todo

Tutorials for running ownCloud on Dreamplug.

5.3 Installing and Managing Apps

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

5.3.1 Viewing Enabled Apps

During the ownCloud installation, some apps are enabled by default. To see which apps are enabled:

1. Click `Apps` in the Apps Selection Menu.

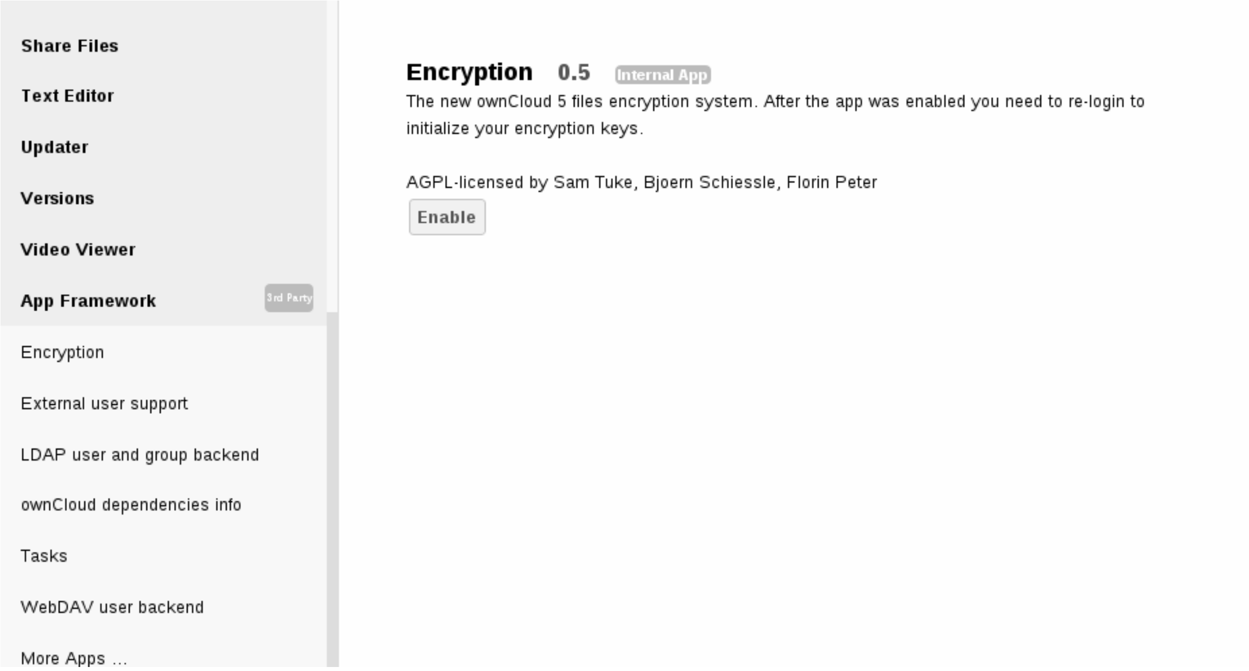
   The apps available for use with ownCloud appear in the Apps Information Field.

2. Scroll down the Apps Information Field to view the enabled apps.

   Apps that are enabled appear at he top of the list of apps.

5.3.2 Managing Apps

In the Apps page, you can enable or disable applications. If an app is already enabled, it appears highlighted in the list. In addition, enabled apps appear at the top of the app list in the Apps Information Field. In contrast, disabled apps appear below any enabled apps in the list and are not highlighted. Some apps have some configurable options on the Apps page, but mainly they are enabled or disabled here, and they are configured on your ownCloud Admin page.



Encryption 0.5

The new ownCloud 5 files encryption system. After the app was enabled you need to re-login to initialize your encryption keys.

AGPL-licensed by Sam Tuke, Bjoern Schiessle, Florin Peter

Figure 5.1: **Administrator application page**

5.3.3 Adding Third Party Apps

Some apps are developed and supported by ownCloud directly, while other apps are created by third parties and either included in or available for your ownCloud server installation. Any apps that are not developed by ownCloud show a *3rd party* designation. Install unsupported apps at your own risk.

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

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 a third party app, it will be downloaded from the app store, installed and enabled.

Though ownCloud provides many apps in the server installation, you can view more 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 app 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`, Ubuntu is `www-data`, and on openSUSE is it `wwwrun:www`.

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 our Developer Center where you can find information about creating and adding your own apps.

5.3.4 Setting App Parameters

Most app parameters are configured on your Admin page, and some are set in `config/config.php`. Always try your Admin page first.

5.3.5 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,
      ),
  ),
```

5.3.6 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",
```

5.4 Hiawatha Configuration

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

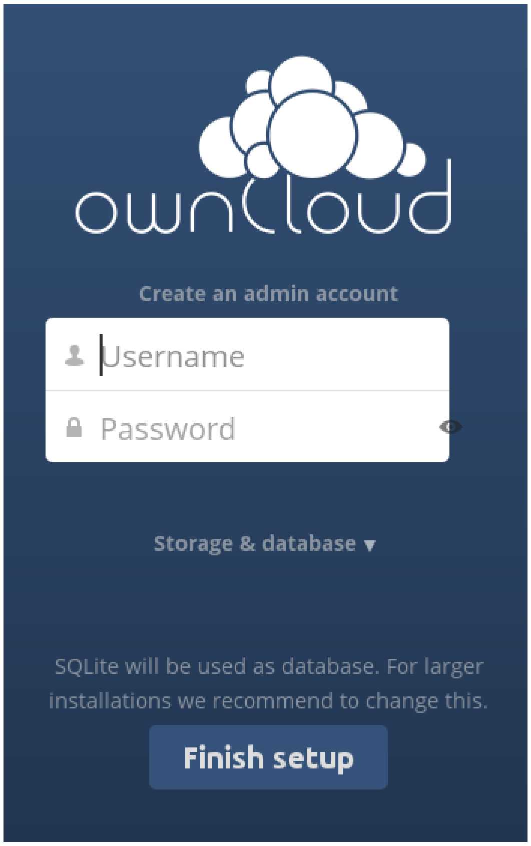
Disable access to data folder:

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

5.5 Installation Wizard

When ownCloud prerequisites are fulfilled and all ownCloud files are installed on the server, the last step to complete the installation is running the Installation Wizard. Open your Web browser to your new ownCloud installation.

- If you are installing ownCloud on the same machine as you are accessing the install wizard from, the URL will be `http://localhost/owncloud`, or `https://localhost/owncloud` if you have enabled SSL.
- If you are installing ownCloud on a different machine, you'll have to access it by its hostname or IP address, e.g. `http://example.com/owncloud`.
- If you are using a self-signed certificate, you will be presented with a security warning about the issuer of the certificate not being trusted which you can ignore.
- You will be presented with the setup screen:



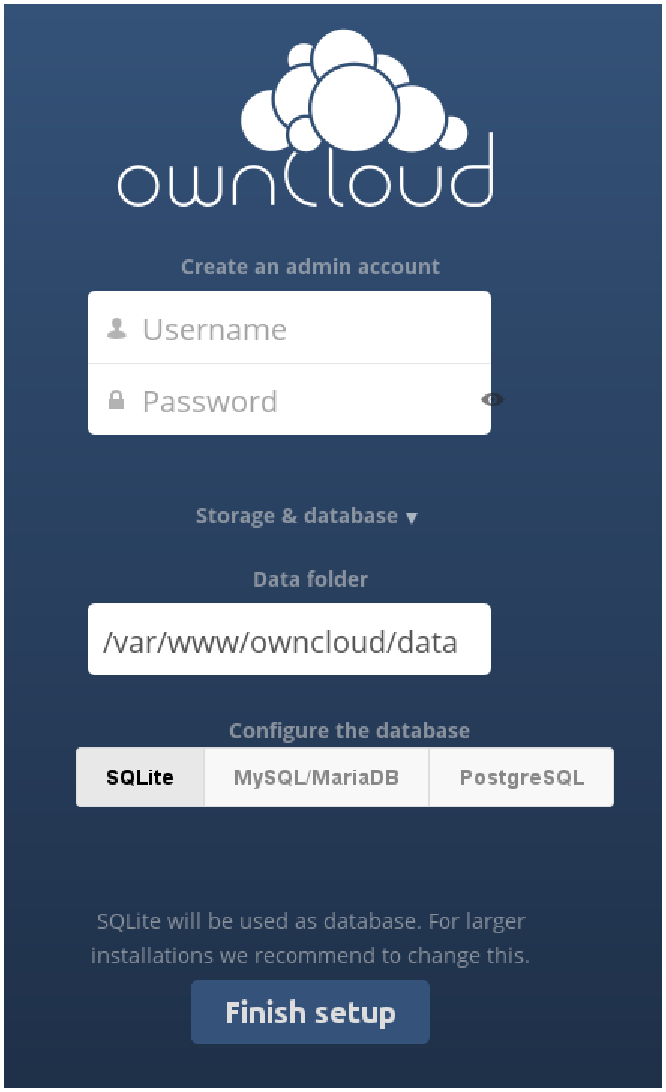
SQLite will be used as database. For larger installations we recommend to change this.

5.5.1 Required Settings

Under `create an admin account` you must enter a username and password for the administrative user account. You may choose any username and password that you want.

5.5.2 Storage & Database

- Click `Storage & Database` to see all of your database options, and to optionally change the default data storage directory.
- The database you want to use must already be installed, and you must have a database admin user and password.
- Enter any arbitrary name for the Database name. This must be a database that does not already exist.
- If you are not using Apache as the web server, it is highly recommended to configure the data directory to a location outside of the document root. Otherwise all user data is potentially publicly visible!



Data folder: /var/www/owncloud/data

SQLite will be used as database. For larger installations we recommend to change this.

5.5.3 Database Choice

- For a guideline on which database system to choose, and on pointers how to set them up for being available for PHP/ownCloud, see *Database Configuration*
- Note that you will only be able to choose among the PHP database connectors which are actually installed on the system.
- It is not easily possible to migrate to another database system once you have set up your ownCloud to use a specific one. So make sure to carefully consider which database system to use.
- When using MySQL/MariaDB or PostgreSQL you have two options regarding the database name and user account you specify:
  - You can specify either an admin or the root user, and the name of a database which does not yet exist. This lets ownCloud create its own database; it will also create a database user account with restricted rights (with the same username as you specified for the administrative user, plus an `oc_` prefix) and will use that for all subsequent database access.
  - There are restrictions as to what characters a database name may or may not contain; see the MySQL Schema Object Names documentation for details);

5.5.4 Finish Installation

- Once you've entered all settings, click "Finish Setup"
- ownCloud will set up your cloud according to the given settings
- When it's finished, it will log you in as administrative user and present the "Welcome to ownCloud" screen.

5.5.5 Setting Strong Directory Permissions

For hardened security we recommend setting the permissions on your ownCloud directory as strictly as possible. This should be done immediately after the initial installation. Your HTTP user must own the `config/`, `data/` and `apps/` directories in your ownCloud directory 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 create a PHP page to find it for you. To do this, create a plain text file with the following lines in it:

```
<?php
echo "User: " . exec('whoami');
echo "Group: " . exec('groups');
?>
```

If the exec php function is disabled (getting a white page with the script above) you can also try to use a script like:

```
<?php
$processUser = posix_getpwuid(posix_geteuid());
echo "User: " . $processUser['name'];
$processGroup = posix_getgrgid($processUser['gid']);
echo " Group: " . $processGroup['name'];
?>
```

Name it `whoami.php` and place it in your `/var/www/html` directory, and then open it in a Web browser, for example `http://localhost/whoami.php`. You should see a single line in your browser page with the HTTP user name.

- 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'

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

chown -R root:${htuser} ${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 root:${htuser} ${ocpath}/.htaccess
chown root:${htuser} ${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 / directory should be owned by `root:[HTTP group]`
- 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

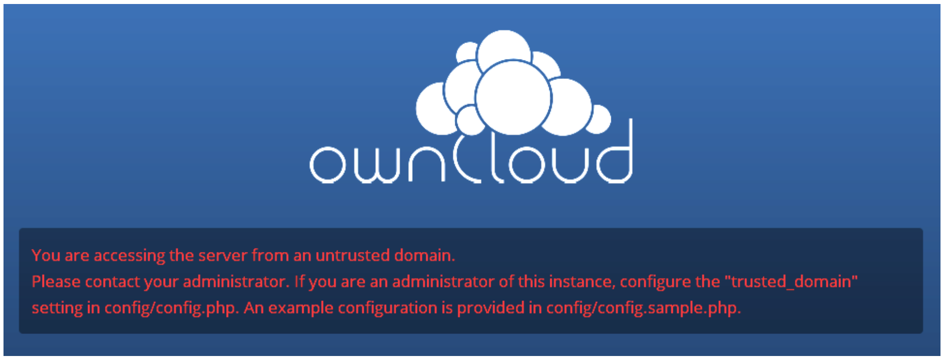
5.5.6 Trusted Domains

ownCloud will take the URL used to access the Installation Wizard and insert that into the `config.php` file for the `trusted_domains` setting. All needed domain names of the ownCloud server go into the `trusted_domains` setting. Users will only be able to log into ownCloud when they point their browsers to a domain name listed in the `trusted_domains` setting. An IPv4 address can be specified instead of a domain name. A typical configuration looks like this:

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

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.

The loopback address, `127.0.0.1`, is whitelisted and therefore users on the ownCloud server who access ownCloud with the loopback interface will be able to successfully login. In the event that an improper URL is used, the following error will appear:



You are accessing the server from an untrusted domain.
Please contact your administrator. If you are an administrator of this instance, configure the "trusted_domain" setting in config/config.php. An example configuration is provided in config/config.sample.php.

For configuration examples, refer to the `config/config.sample.php` document.

5.6 Lighttpd Configuration

This assumes that you are familiar with installing PHP applications on Lighttpd.

It is important to note that the `.htaccess` used by ownCloud to protect the `data` folder is ignored by lighttpd, so you have to secure it by yourself, otherwise your `owncloud.db` database and user data are publicly readable even if directory listing is off. You need to add these two snippets to your Lighttpd configuration file:

Disable access to data folder:

```
$HTTP["url"] =~ "^/owncloud/data/" {
        url.access-deny = ("")
}
```

Disable directory listing:

```
$HTTP["url"] =~ "^/owncloud($|/)" {
        dir-listing.activate = "disable"
}
```

Note for Lighttpd users on Debian stable (wheezy):

Recent versions of ownCloud make use of the **HTTP PATCH** feature, which was added to Lighttpd at version 1.4.32 while Debian stable only ships 1.4.31. The patch is simple, however, and easy to integrate if you're willing to build your own package.

Download the patch from http://redmine.lighttpd.net/attachments/download/1370/patch.patch

Make sure you have the build tools you need:

```
apt-get build-dep lighttpd
apt-get install quilt patch devscripts
```

Patch the package source:

```
apt-get source lighttpd
cd lighttpd-1.4.31
export QUILT_PATCHES=debian/patches # This tells quilt to put the patch in
the right spot
quilt new http-patch.patch
quilt add src/connections.c src/keyvalue.c src/keyvalue.h # Make quilt
watch the files we'll be changing
patch -p1 -i /patch/to/downloaded/patch.patch
quilt refresh
```

Increment the package version with `dch -i`. This will open the changelog with a new entry. You can save as-is or add info to it. The important bit is that the version is bumped so apt will not try to "upgrade" back to Debian's version.

Then build with `debuild` and install the .debs for any Lighttpd packages you already have installed.

5.7 Mac OS X

Note: Due to an issue with Mac OS Unicode support, installing ownCloud Server 7.0 on Mac OS is currently not supported.

5.8 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).

  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/ /index.php;
   }

   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;
   }

   # Optional: set long EXPIRES header on static assets
   location ~* \.(?:jpg|jpeg|gif|bmp|ico|png|css|js|swf)$ {
       expires 30d;
       # Optional: Don't log access to assets
         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.

5.9 Other Installation Methods

5.9.1 PageKite Configuration

You can use this PageKite how to to make your local ownCloud accessible from the internet using PageKite.

5.10 Manual Installation on Linux

Installing ownCloud on Linux from the openSUSE 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.

If there are no packages for your Linux distribution, or you prefer installing from sources, 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 MySQL.

5.10.1 Prerequisites

Note: This tutorial assumes you have terminal access to the machine you want to install ownCloud on. Although this is not an absolute requirement, installation without it is likely to require contacting your hoster (e.g. for installing required modules). Consult the PHP manual for information on modules. Your Linux distribution should have packages for all required modules.

To run ownCloud, your web server must have the following installed:

- php5 (>= 5.3.8, minimum recommended 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 SimpleXML
- PHP module XMLWriter
- PHP module zip
- PHP module zlib

Database connectors (pick at least one):

- PHP module sqlite (>= 3, usually not recommended for performance reasons)
- PHP module mysql
- 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)
- smbclient (for SMB storage / external user authentication)
- 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 only one of the following):

- PHP module apc
- PHP module apcu
- PHP module xcache

For preview generation (*optional*):

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

Remarks:

- Please check your distribution, operating system or hosting partner documentation on how to install and enable these modules.
- Make sure your distribution's PHP version fulfills the version requirements specified above. If it doesn't, there might be custom repositories you can use. If you are e.g. running Ubuntu 10.04 LTS, you can update your PHP using a custom PHP PPA:

  ```
  sudo add-apt-repository ppa:ondrej/php5
  sudo apt-get update
  sudo apt-get install php5
  ```

- You don't need the WebDAV module for your web server (i.e. Apache's `mod_webdav`) to access your ownCloud data via WebDAV. ownCloud has a built-in WebDAV server of its own, SabreDAV.

5.10.2 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 the Prerequisites section (above) for details.
- At the execution of each of the above commands you might be prompted whether you want to continue; press "Y" for Yes (that is if your system language is English. You might have to press a different key if you have a different system language).
- At the installation of the MySQL server, you will be prompted to create a root password. Be sure to remember the password you enter there for later use as you will need it during ownCloud database setup.

Now download the archive of the latest ownCloud version:

- Go to the ownCloud Installation Page.
- Click the **Archive file for server owners** button.
- Click **Download Unix**.
- This downloads a file named owncloud-x.y.z.tar.bz2 (where x.y.z is the version number of the current latest version).
- Save this file on the machine you want to install ownCloud on.
- Verify the MD5 or SHA256 sum:

```
md5sum  owncloud-x.y.z.tar.bz2
sha256sum owncloud-x.y.z.tar.bz2
```

- You may also verify the PGP signature:

  ```
  wget https://download.owncloud.org/community/owncloud-x.y.z.tar.bz2.asc
  wget https://www.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. Open a terminal, navigate to your download directory, and run:

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

- Copy the ownCloud files to their final destination in the document root of your web server:

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

  where `/path/to/webserver/document-root` is replaced by the document root of your Web server. On Ubuntu systems this `/var/www/owncloud`, so your copying command is:

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

5.10.3 Installation Wizard

Finish setting up your ownCloud server by following the *Installation Wizard*.

After running the Installation Wizard your ownCloud installation is complete. However, you should perform the following steps to improve your server's security.

5.10.4 Setting Strong Directory Permissions

We recommend setting the directory permissions in your ownCloud installation as strictly as possible for stronger security. Please refer to the `Setting Strong Directory Permissions` section of *Installation Wizard*.

5.10.5 SELinux

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

Apache is the recommended Web server.

5.10.6 Configuration notes to php.ini files

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

php.ini - used by the webserver:

```
  /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
```

5.10.7 Apache Web Server Configuration

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.

5.10.8 Enabling SSL

An 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 according site. Open a terminal and run:

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

If you are using a different distribution, check your documentation on how to enable SSL.

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 commercial signing authority. Check with your domain name registrar or hosting service, if you're using one, for good deals on commercial certificates.

5.10.9 Configuring ownCloud

Since there was a change in the way versions 2.2 and 2.4 are configured, you'll have to find out which Apache version you are using.

Usually you can do this by running one of the following commands:

```
apachectl -v
apache2 -v
```

Example output:

```
Server version: Apache/2.4.7 (Ubuntu)
Server built:   Jul 22 2014 14:36:38
```

Example config for Apache 2.2:

```
<Directory /path/to/owncloud>
    Options Indexes FollowSymLinks
    AllowOverride All
    Order allow,deny
    allow from all
</Directory>
```

Example config for Apache 2.4:

```
<Directory /path/to/owncloud>
    Options Indexes FollowSymLinks
    AllowOverride All
    Require all granted
</Directory>
```

- This configuration entry needs to go into the configuration file of the “site” you want to use.
- On a Ubuntu system, this typically is the “default-ssl” site (to be found in the `/etc/apache2/sites-available/default-ssl.conf`).
- Add the entry shown above immediately before the line containing:

  ```
  </VirtualHost>
  ```

  (this should be one of the last lines in the file).
- A minimal site configuration file on Ubuntu 14.04 might look like this:

```
<IfModule mod_ssl.c>
<VirtualHost _default_:443>
        ServerName YourServerName
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www
        <Directory />
                Options FollowSymLinks
                AllowOverride None
        </Directory>
        <Directory /var/www/>
                Options Indexes FollowSymLinks
                AllowOverride None
                Order allow,deny
                allow from all
        </Directory>
        ErrorLog ${APACHE_LOG_DIR}/error.log
        LogLevel warn
        CustomLog ${APACHE_LOG_DIR}/ssl_access.log combined
        SSLEngine on
        SSLCertificateFile    /etc/ssl/certs/ssl-cert-snakeoil.pem
        SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
        <FilesMatch "\.(cgi|shtml|phtml|php)$">
                SSLOptions +StdEnvVars
        </FilesMatch>
        <Directory /usr/lib/cgi-bin>
                SSLOptions +StdEnvVars
        </Directory>
        BrowserMatch "MSIE [2-6]" \
                nokeepalive ssl-unclean-shutdown \
                downgrade-1.0 force-response-1.0
        BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown
        <Directory /var/www/owncloud>
                Options Indexes FollowSymLinks
                AllowOverride All
                Allow from all
                Require all granted
                Dav Off
                Satisfy Any
        </Directory>
</VirtualHost>
</IfModule>
```

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

  ```
  a2enmod rewrite
  ```

- In distributions that do not come with `a2enmod`, the module needs to be enabled manually by editing the Apache config files, usually `/etc/httpd/httpd.conf`. Consult the Apache documentation or your Linux distribution's documentation.

- In order for the maximum upload size to be configurable, the `.htaccess` in the ownCloud folder needs to be made writable by the server (this should already be done, see section `Set the Directory Permissions`).
- You should make sure that any built-in WebDAV module of your web server is disabled (at least for the ownCloud directory), as it will interfere with ownCloud's built-in WebDAV support.

  If you need the WebDAV support in the rest of your configuration, you can turn it off specifically for the ownCloud entry by adding the following line in the `<Directory` section for your ownCloud server. Add the following line directly after the `allow from all` / `Require all granted` line:

  ```
  Dav Off
  ```

- 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 directly after the `allow from all` / `Require all granted` line in the `<Directory` section:

  ```
  Satisfy Any
  ```

- When using ssl, take special note on 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.

Note: By default, the certificates' CommonName will be set to the host name at the time the ssl-cert package was installed.

- Finally, restart Apache.
  - On Ubuntu systems run:

    ```
    service apache2 restart
    ```

  - On systemd systems (Fedora, Arch Linux, OpenSUSE), run:

    ```
    systemctl restart httpd.service
    ```

5.10.10 Other Web Servers

Microsoft Internet Information Server (IIS)

See *Windows 7 and Windows Server 2008* for further instructions.

Nginx Configuration

See *Nginx Configuration*

Lighttpd Configuration

See *Lighttpd Configuration*

Yaws Configuration

See *Yaws Configuration*

Hiawatha Configuration

See *Hiawatha Configuration*

5.11 Univention Corporate Server

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

5.11.1 Pre configuration

ownCloud makes use of the UCR, the Univention Configuration Registry. The values are being 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. In case you 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,$ldap_base	The users-subtree in the LDAP directory. If left blank it will fall back to the LDAP base.	2012.4.0.4
owncloud/ldap/base/groups	cn=groups,$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

Continued on next page

Table 5.1 – continued from previous page

Key	Default	Description	Introduced
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,mailPrimaryAddress	Attributes taken into consideration when searching for users (comma separated)	5.0.9
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	Wether ownCloud LDAP schema should be applied to existing users	2012.0.1
owncloud/group/enableDomainUsers	1	Wether the group "Domain Users" shall be enabled for ownCloud on install	2012.4.0.4

Continued on next page

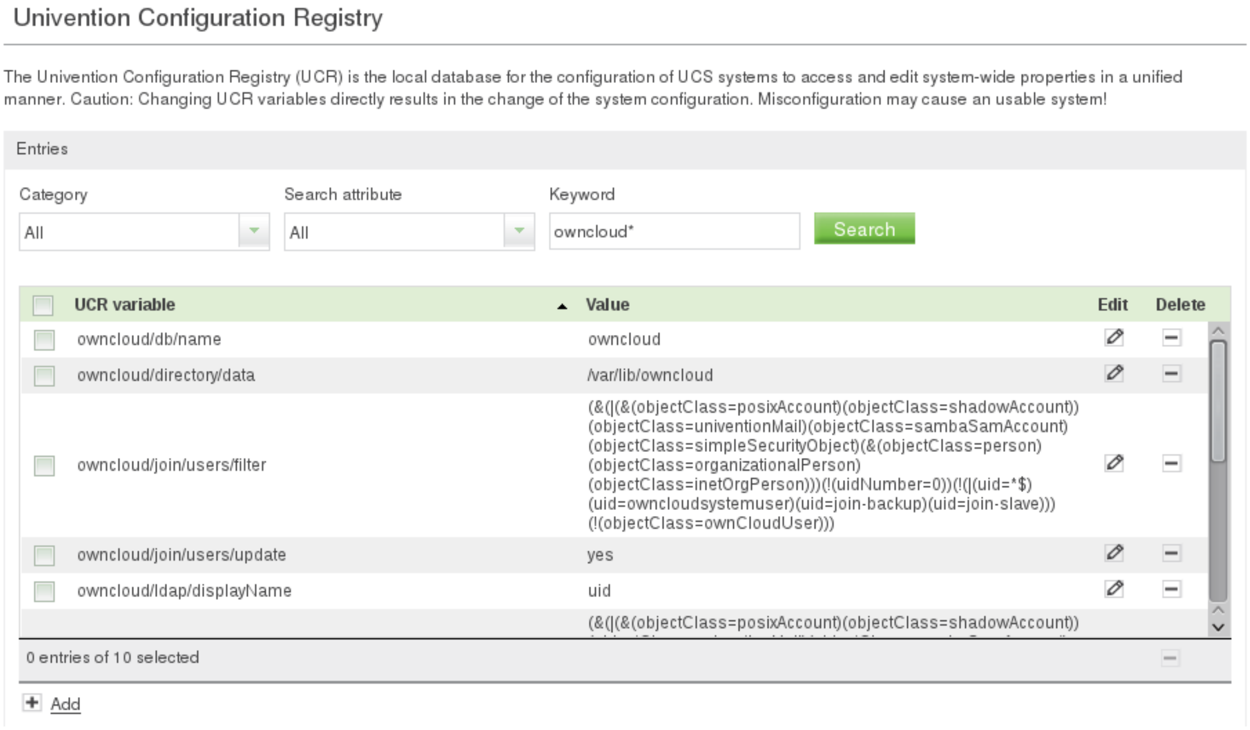
Table 5.1 – continued from previous page

Key	Default	Description	Introduced
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 already 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:



Univention Configuration Registry

The Univention Configuration Registry (UCR) is the local database for the configuration of UCS systems to access and edit system-wide properties in a unified manner. Caution: Changing UCR variables directly results in the change of the system configuration. Misconfiguration may cause an usable system!

Entries

Category: All

Search attribute: All

Keyword: owncloud*

Search

□	UCR variable	Value	Edit	Delete
□	owncloud/db/name	owncloud		
□	owncloud/directory/data	/var/lib/owncloud		
□	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)))		
□	owncloud/join/users/update	yes		
□	owncloud/ldap/displayName	uid		
		(&(|(&(objectClass=posixAccount)(objectClass=shadowAccount))		

0 entries of 10 selected

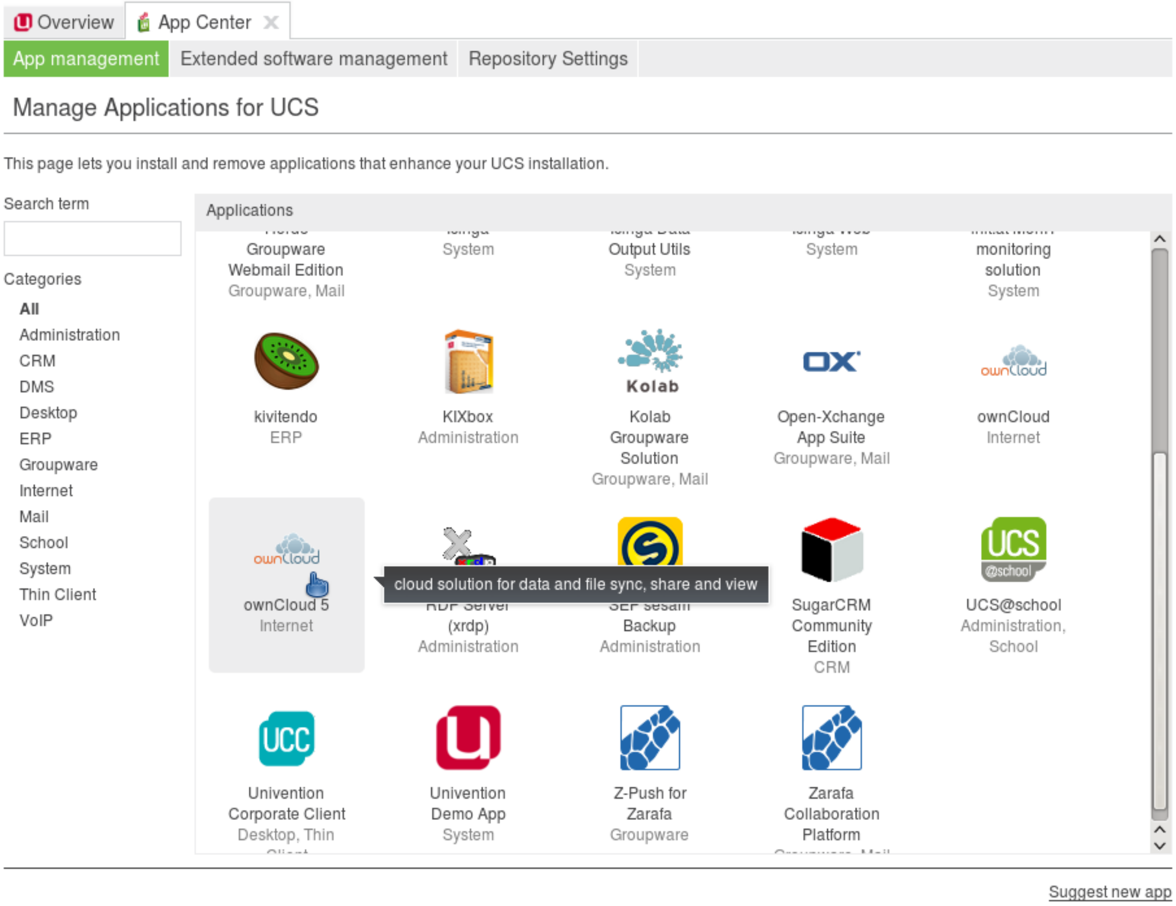
Add

5.11.2 Installation

Now, we are ready to install ownCloud. This can be either done through the UCS App Center (recommended) or by downloading the packages.

UCS App Center

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

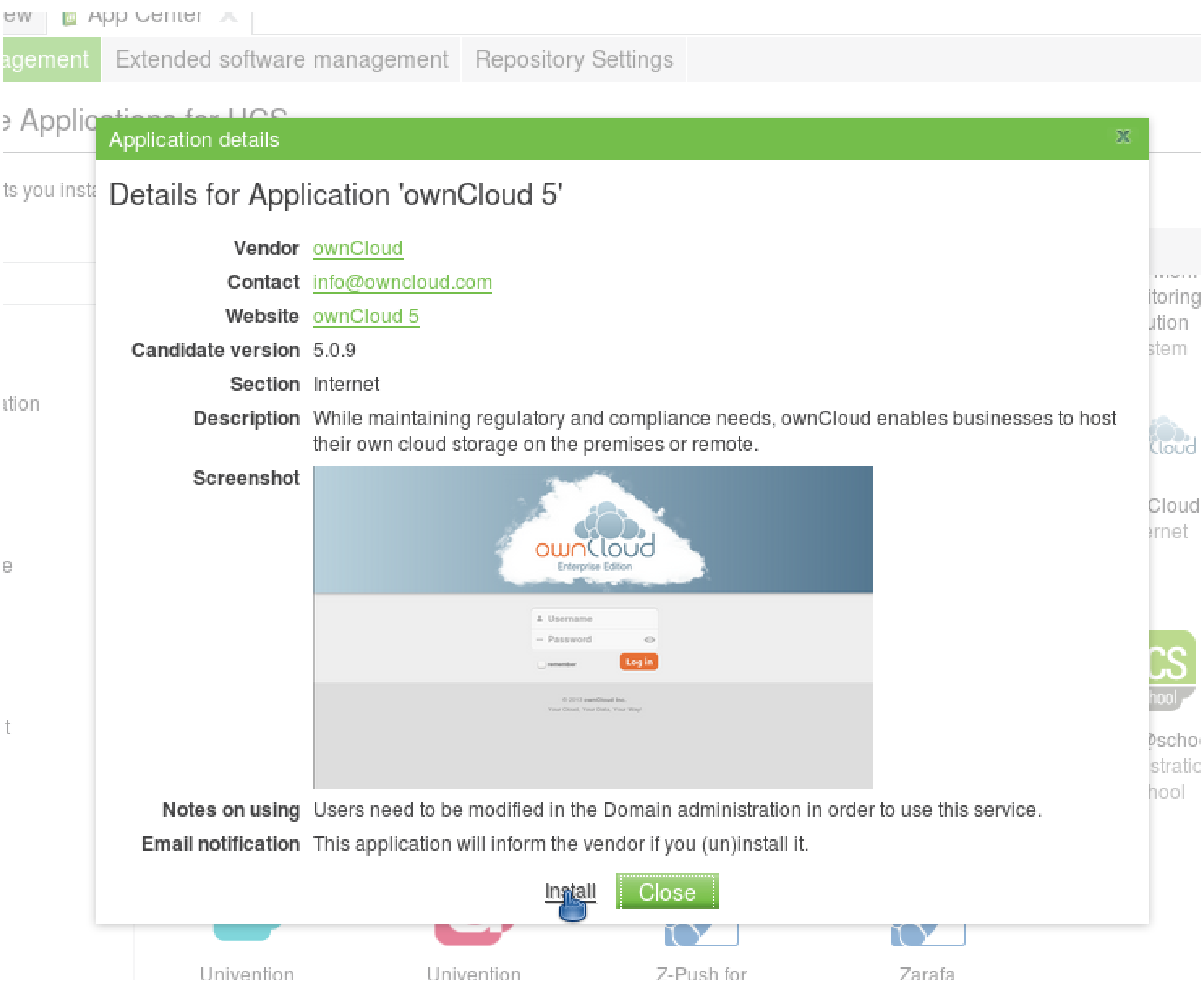


Manage Applications for UCS

This page lets you install and remove applications that enhance your UCS installation.

cloud solution for data and file sync, share and view

Click on ownCloud 5 and follow the instructions.



Details for Application 'ownCloud 5'

Vendor	ownCloud
Contact	info@owncloud.com
Website	ownCloud 5
Candidate version	5.0.9
Section	Internet
Description	While maintaining regulatory and compliance needs, ownCloud enables businesses to host their own cloud storage on the premises or remote.
Screenshot	
Notes on using	Users need to be modified in the Domain administration in order to use this service.
Email notification	This application will inform the vendor if you (un)install it.

In the UCS App Center, you can also upgrade from ownCloud 4.5 by installing ownCloud 5.0. They are provided as separate apps. It is only possible to have one version of ownCloud installed.

Manually by download

Download the integration packages from our website and install them from within your download folder (note: the package owncloud-unsupported is optional) via command line:

```
dpkg -i owncloud*.deb
```

ownCloud will be configured to fully work with LDAP.

Reinstallation

When ownCloud was installed before and uninstalled via AppCenter or via command line using apt-get remove, ownCloud can be simply installed again. The old configuration will be used again.

When an older ownCloud was installed and has been purged (only possible via command line using apt-get purge) the old configuration is gone, but data is left. This blocks an installation. You can either install the old version and upgrade to ownCloud 5 or (re)move the old data. This is done by removing the MySQL database “ownCloud” using the command line:

```
mysql -u root -e "DROP DATABASE owncloud" -p`tail /etc/mysql.secret
```

In this case you probably also want to remove the data directory **/var/lib/owncloud** although this is not mandatory.

5.11.3 Postconfiguration (optional)

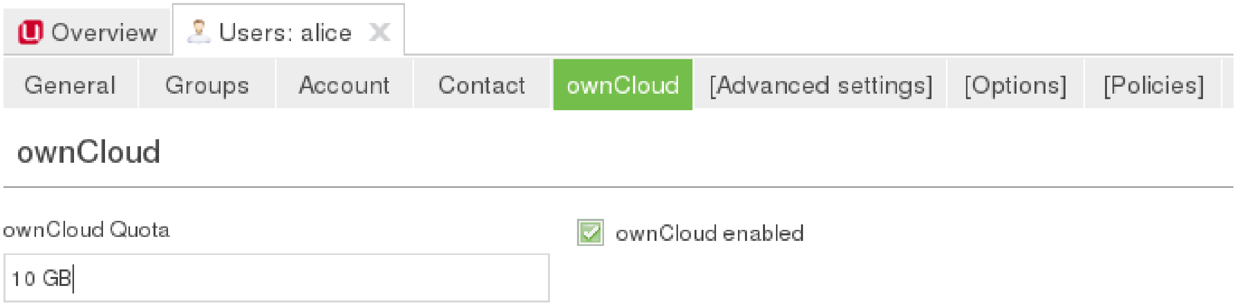
There is only one local admin user “owncloudadmin”, you can find his password in */etc/owncloudadmin.secret*. Use this account, if you want to change basic ownCloud settings.

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 there are also the PHP limits for file transfer specified.

5.11.4 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.



ownCloud Quota

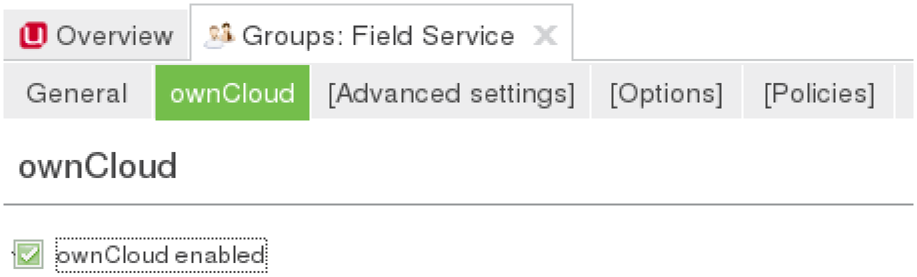
10 GB

☑ ownCloud enabled

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 2012.4.0.4

Since ownCloud Enterprise 2012.4.0.4 group support is enabled. Groups, that are activated for ownCloud usage, can be used to share files to instead of single users, for example. It is also important to note, that users can only share within groups where they belong to. Groups can be enabled and disabled via UCM as shown in the screen shot below.



☑ ownCloud enabled

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 being taken from the UCR variable **owncloud/join/groups/filter**. In case it is empty, a message will be displayed.

5.12 Windows 7 and Windows Server 2008

Note: While ownCloud will run in any standard PHP environment, including IIS or Apache on Windows, there are known issues. For the basic sync and share capabilities of ownCloud, Windows web servers (Apache and IIS) will function properly. However, as apps like external storage are added, particularly SMB mounts, and non-English characters are used in filenames, some of the known Windows and IIS/Apache challenges start to appear as bugs.

ownCloud is not supported on the Internet Server Application Programming Interface (ISAPI).

Microsoft SQL Server is not supported.

For these reasons, while ownCloud server will run on Windows, it is not recommended at this time.

Note: You must move the data directory outside of your public root (See advanced installation settings)

This section describes how to install ownCloud on Windows with IIS (Internet Information Services).

These instructions assume that you have a standard, non-IIS enabled Windows machine using Windows 7 or Server 2008. After enabling IIS, the procedures are essentially identical for both Windows 7 and Windows Server 2008.

For installation, ownCloud physical access or a remote desktop connection is required. We recommend that you leverage MySQL as the backend database for ownCloud. If you do not want to use MySQL, you can use Postgres or SQLite instead. However, Microsoft SQL Server is not yet supported.

Enabling SSL is not yet covered by this section.

Note: If you make your desktop machine or server available outside of your LAN, you must maintain it. Make sure to monitor the logs, manage the access, and apply patches to avoid compromising the system as a whole.

There are four primary steps to the installation, and then an added fifth step required for configuring everything to allow files larger than the default 2 MB size.

1. Install IIS with CGI support – enable IIS on your Windows machine.
2. Install PHP – Grab, download and install PHP.
3. Install MySQL – Setup the MySQL server manager and enable ownCloud to create an instance.
4. Install ownCloud – The whole reason we are here!
5. Configure upload sizes and timeouts to enable large file uploads – So that you can upload larger files.

5.12.1 Activate IIS with CGI Support

Windows 7

To activate IIS on Microsoft Windows 7:

1. Navigate to *Start –> Control Panel –> Programs*.
2. Under Programs and Features, click on the link entitled *Turn Windows Features on and Off*.
3. Expand the box labeled *Internet Information Services*.

4. Expand `World Wide Web Services` and all of the folders beneath it.
5. Select the folders as shown in the image below to launch the IIS server.
6. Because a running FTP server is not required, turn off that feature for your server.
7. Ensure that you have the IIS Management Console. An IIS management console is the easiest way to start, stop, and restart your server. This console also enables you to change certificate options and manage items like file upload size.
8. Check the CGI checkbox under `Application Development Features` in order to enable PHP on IIS.
9. Turn off WebDAV publishing to avoid conflicts between the Windows WebDAV and the ownCloud WebDAV interface.

   > **Note:** This feature might already be turned off for you. However, we recommend that you ensure that it remains off. The common HTTP features are the features you would expect from a web server.

   After implementing the selections on this page, IIS serves up a web page.
10. Restart IIS by going to the IIS manager (*Start –> IIS Manager*).
11. Select your website.

    On the far right side of the opening page you will see a section titled *Manage Server*.
12. Make sure that the service is started, or click *Start* to start the services selected.
13. Go to a web browser and navigate to http://localhost.

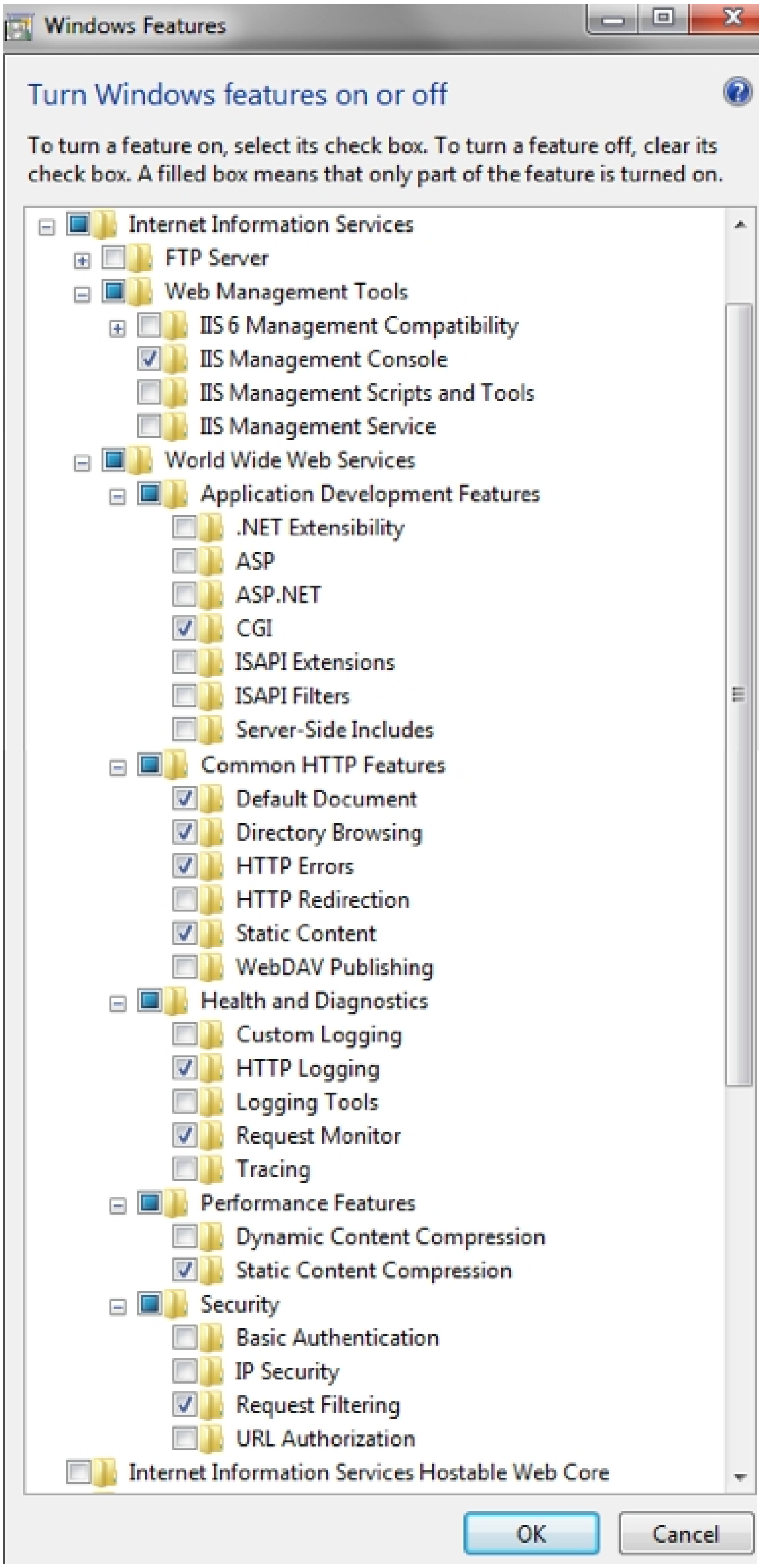
    The standard IIS 7 splash page opens. This page displays a static image that indicates that your web server is running. Assuming you were able to reach splash page, your web server is now up and running.

Continue by installing PHP.

Windows Server 2008

1. Navigate to *Start –> Control Panel –> Programs*.
2. Under Programs and Features, click the link titled *Turn Windows Features on and Off*. The Server Manager starts.
3. In the Server Manager, click `Roles`
4. Click `Add Roles`.
5. Use the *Add Roles Wizard* to add the web server role.
6. Make sure that, at a minimum, the same boxes are checked in this wizard that are checked in the Windows 7 Section. For example, make sure that the CGI box is checked under Application Development Features, and that WebDAV Publishing is turned off. With Remote Desktop Sharing turned on, the detailed role service list looks like the figure “Role Services”.
7. Go to the IIS manager (*Start –> IIS Manager*) and restart IIS.
8. Select your website
9. Once this is complete, you should be able to go to a web browser and type *localhost*. This should open the standard IIS 7 splash page, which is just a static image that says your web server is running. Assuming you were able to get the splash page, it is safe to say your web server is now up and running.

Continue by installing PHP.

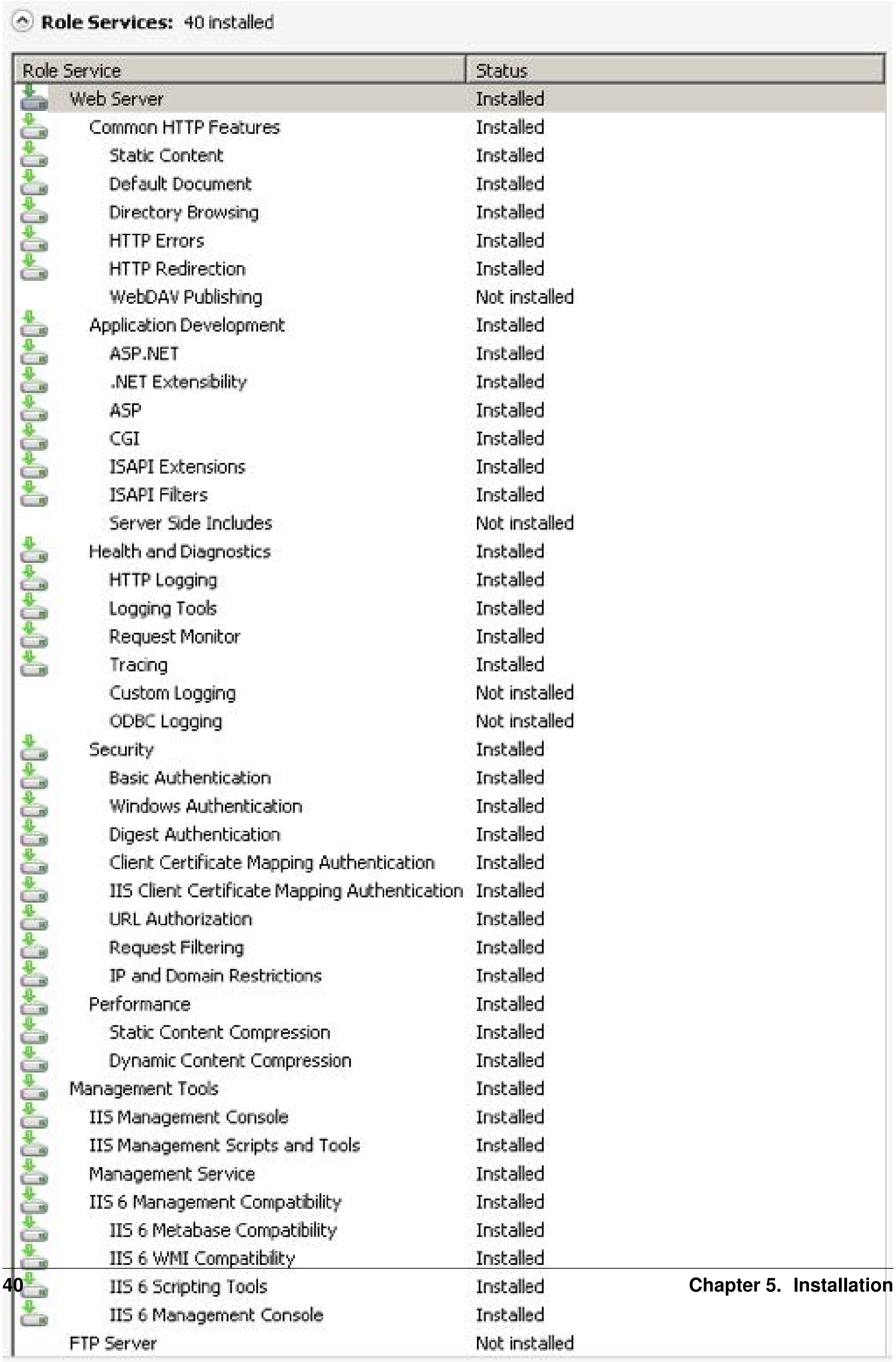


Turn Windows features on or off

To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

- ▣ Internet Information Services
  - □ FTP Server
  - ▣ Web Management Tools
    - □ IIS 6 Management Compatibility
    - ☑ IIS Management Console
    - □ IIS Management Scripts and Tools
    - □ IIS Management Service
  - ▣ World Wide Web Services
    - ▣ Application Development Features
      - □ .NET Extensibility
      - □ ASP
      - □ ASP.NET
      - ☑ CGI
      - □ ISAPI Extensions
      - □ ISAPI Filters
      - □ Server-Side Includes
    - ▣ Common HTTP Features
      - ☑ Default Document
      - ☑ Directory Browsing
      - ☑ HTTP Errors
      - □ HTTP Redirection
      - ☑ Static Content
      - □ WebDAV Publishing
    - ▣ Health and Diagnostics
      - □ Custom Logging
      - ☑ HTTP Logging
      - □ Logging Tools
      - ☑ Request Monitor
      - □ Tracing
    - ▣ Performance Features
      - □ Dynamic Content Compression
      - ☑ Static Content Compression
    - ▣ Security
      - □ Basic Authentication
      - □ IP Security
      - ☑ Request Filtering
      - □ URL Authorization
- □ Internet Information Services Hostable Web Core

Figure 5.2: **Windows Features required for ownCloud on Windows 7**



Role Services: 40 installed

Role Service	Status
Web Server	Installed
Common HTTP Features	Installed
Static Content	Installed
Default Document	Installed
Directory Browsing	Installed
HTTP Errors	Installed
HTTP Redirection	Installed
WebDAV Publishing	Not installed
Application Development	Installed
ASP.NET	Installed
.NET Extensibility	Installed
ASP	Installed
CGI	Installed
ISAPI Extensions	Installed
ISAPI Filters	Installed
Server Side Includes	Not installed
Health and Diagnostics	Installed
HTTP Logging	Installed
Logging Tools	Installed
Request Monitor	Installed
Tracing	Installed
Custom Logging	Not installed
ODBC Logging	Not installed
Security	Installed
Basic Authentication	Installed
Windows Authentication	Installed
Digest Authentication	Installed
Client Certificate Mapping Authentication	Installed
IIS Client Certificate Mapping Authentication	Installed
URL Authorization	Installed
Request Filtering	Installed
IP and Domain Restrictions	Installed
Performance	Installed
Static Content Compression	Installed
Dynamic Content Compression	Installed
Management Tools	Installed
IIS Management Console	Installed
IIS Management Scripts and Tools	Installed
Management Service	Installed
IIS 6 Management Compatibility	Installed
IIS 6 Metabase Compatibility	Installed
IIS 6 WMI Compatibility	Installed
IIS 6 Scripting Tools	Installed
IIS 6 Management Console	Installed
FTP Server	Not installed

5.12.2 Installing PHP

1. Go to the PHP for Windows download page.

Note: The instructions below are for IIS only. If using a different server software, make sure to follow the hints on "Which version do I choose" on the left hand side of the page linked above.

2. Download the Installer for PHP 5.3, the "VC9 Non Thread Safe" version, either 32 or 64 bit, depending on your system.
3. Run the downloaded installation executable.
4. Read the license agreement, agree, select an install directory.
5. Then select IIS FastCGI as the install server.
6. Take the default selections for the items to install, and click next. Then click *install*.
7. Once the installer is finished, PHP is installed.

Continue by installing MySQL.

5.12.3 Installing MySQL

To install MySQL on your Windows machine:

1. Use your browser to migrate to http://dev.mysql.com/downloads/.
2. Download the latest community edition for your operating system, choosing either the 32 or 64 bit version as applicable.
3. Download the **MSI Installer** to assist with the install.
4. Once the download completes, install MySQL (5.5 at the time of writing), selecting the typical installation.
5. Once the installation completes, check the checkbox to launch the MySQL Instance Configuration Wizard and click `Finish`.
6. Select a standard configuration, as this will be the only version of MySQL on this machine.
7. Select the option to install as a windows service, and Check the `Launch the MySQL Server Automatically` button.
8. Select the modify security settings checkbox on the next page, and enter a password.

   Note: Make sure to note your chosen password. You will need this password when you configure ownCloud.

9. Uncheck `enable root access from remote machines` for security reasons.
10. Click execute. The instance is created and launched.
11. Once the instance launches, click Finish.

Take particular note of your MySQL password, as the user name **root** and the password you select will be necessary later on in the ownCloud installation. As an aside, the following link is an excellent resource for questions on how to configure your MySQL instance, and also to configure PHP to work with MySQL. This, however, is not strictly necessary as much of this is handled when you download ownCloud.

More information in this topic can be found in a tutorial on the IIS web site. http://learn.iis.net/page.aspx/353/install-and-configure-mysql-for-php- applications-on-iis-7-and-above/

5.12.4 Installing ownCloud

1. Download the latest version of ownCloud from http://owncloud.org/download. The file is downloaded in tar.bz2 format.
2. Unzip the file and save it locally.

   Note: You can use jZip for a free utility (like Peazip) to unzip the file.

3. Copy the file to your `wwwroot` directory (for example, **C:\inetpub\wwwroot**).

   Note: Only the administrator can install directly into the directory **wwwroot** from an unzipping application. However, you can save the file in a different folder and then move the files into **wwwroot** in windows explorer. This process installs ownCloud locally in your root web directory. You can use a subdirectory called owncloud (or whatever name you choose).

4. To enable write access to the ownCloud directory to the ownCloud server, navigate your windows explorer to **inetpub/wwwroot/owncloud** (or the installation directory you selected).
5. Right click and select properties.
6. Click the security tab, and select the button "to change permissions, click edit".
7. Select the "users" user from the list, and check the box "write".
8. Apply these settings and close the window.

   Continue by following the *Installation Wizard*. Select MySQL as the database, and enter your MySQL database user name, password and desired instance name – use the user name and password you setup during MySQL installation, and pick any name for the database instance.

5.12.5 Ensure Proper HTTP-Verb Handling

IIS must pass all HTTP and WebDAV verbs to the PHP/CGI handler, and must not attempt to handle them by itself or syncrhonizing with the Desktop and Mobile Clients will fail.

To ensure your configuration is correct:

1. Open IIS Manager7.
2. In the *Connections* bar, select your site below *Sites*, or choose the top level entry if you want to modify the machine-wide settings.
3. Choose the *Handler Mappings* feature.
4. Click *PHP_via_fastCGI*.
5. Choose *Request Restrictions* and locate the *Verbs* tab.
6. Ensure *All Verbs* is checked.
7. Click *OK*.

7. Choose the *Request Filtering* feature from the IIS Manager.
8. Ensure that all verbs are permitted (or none are forbidden) in the *Verbs* tab. You need to allow the verbs `GET`, `HEAD`, `POST`, `OPTIONS`, `PROPFIND`, `PUT`, `MKCOL`, `MKCALENDAR`, `DELETE`, `COPY`, and `MOVE`.

   Note: Because ownCloud must be able to use WebDAV on the application level, you must also ensure that you do not enable the WebDAV authoring module.

5.12.6 Configuring ownCloud, PHP and IIS for Large File Uploads

Before you begin to use ownCloud heavily, it is important to make a few configuration changes to enhance the service and make it more useful. For example, you might want to increase the **max upload size**. The default upload is set to **2MB**, which is too small for many files (for example, most MP3 files).

To adjust the maximum upload size, you must access your `PHP.ini` file. You can locate this file in your **C:\Program Files (x86)\PHP** folder.

To adjust the maximum upload size, open the `PHP.ini` file in a text editor, find the following key attributes, and change them to what you want to use:

- **upload_max_filesize** – Changing this value to something like 1G will enable you to upload much larger files.
- **post_max_size** – Change this value to be larger than your max upload size you chose.

You can make other changes in the `PHP.ini` file (for example, the timeout duration for uploads). However, most default settings in the **PHP.ini** file should function appropriately.

To enable file uploads on the web server larger than 30 MB, you must also change some settings in the IIS manager.

To modify the IIS Manager:

1. Go to the start menu, and type **iis manager**. IIS manager launches.
2. Select the website that you want to accept large file uploads.
3. In the main (middle) window, double click the icon **Request filtering**. A window opens displaying a number of tabs across the top.
4. Select *Edit Feature Settings*
5. Modify the *Maximum allowed content length (bytes)* value to 4.1 GB.

> **Note:** This entry is in bytes, not kilobytes.

You should now have ownCloud configured and ready for use.

5.13 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.

5.14 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
```

5.14.1 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
```

5.14.2 Allow access to LDAP server

Use this setting to allow LDAP connections:

```
setsebool -P httpd_can_connect_ldap on
```

5.14.3 Allow access to remote network

ownCloud requires access to remote networks for functionalities 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
```

5.14.4 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
```

CHAPTER

SIX

CONFIGURATION

6.1 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.

6.1.1 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.

6.1.2 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 config option `activity_expire_days` available in your `config.php` (See *Config.php Parameters*) which allows you to clean-up older activies from the database.

6.2 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.

6.2.1 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.

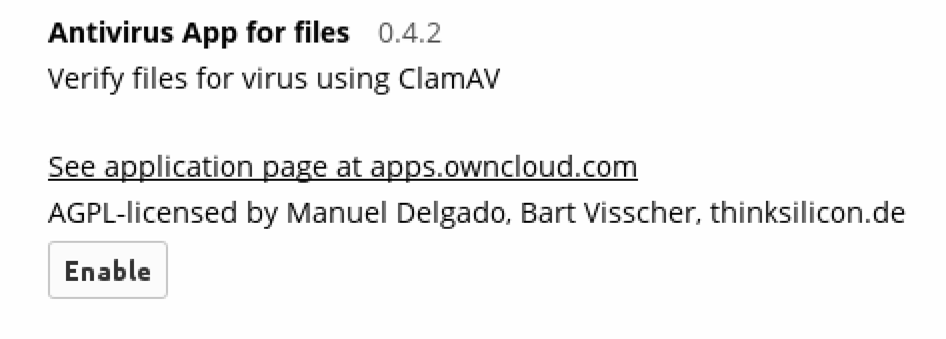
Next, edit `/etc/clamd.d/scan.conf`. When you're finished you must enable the `clamd` service file and start `clamd`:

```
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.

6.2.2 Enabling the Antivirus App for Files

Simply go to your ownCloud Apps page to enable it.



Antivirus App for files 0.4.2

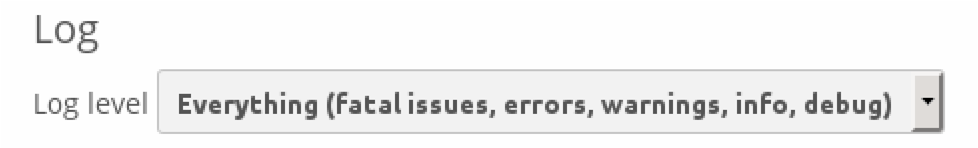
Verify files for virus using ClamAV

See application page at apps.owncloud.com

AGPL-licensed by Manuel Delgado, Bart Visscher, thinksilicon.de

6.2.3 Configuring ClamAV on ownCloud

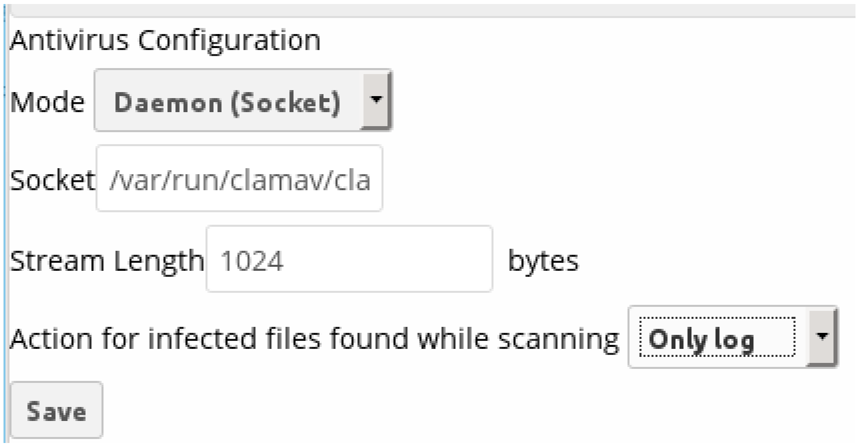
Next, go to your ownCloud Admin page and set your ownCloud logging level to Everything.



Log

Log level: Everything (fatal issues, errors, warnings, info, debug)

Now find your Antivirus Configuration panel on your Admin page.



Antivirus Configuration

Mode: Daemon (Socket)

Socket: /var/run/clamav/cla

Stream Length: 1024 bytes

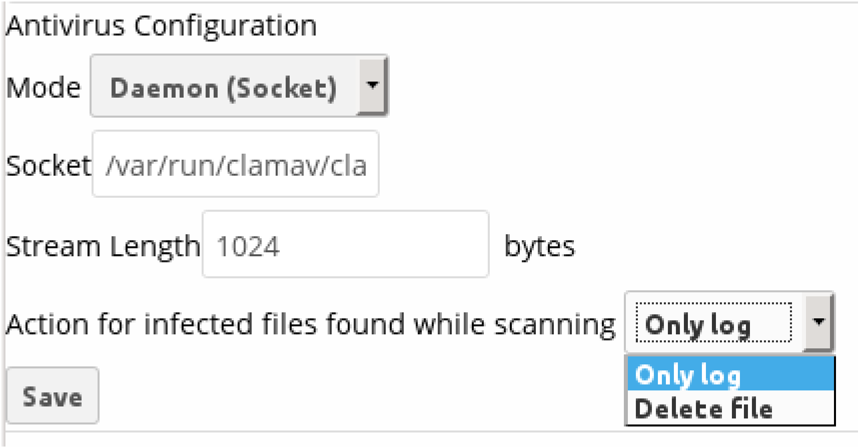
Action for infected files found while scanning: Only log

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
```



Antivirus Configuration

Mode Daemon (Socket)

Socket /var/run/clamav/cla

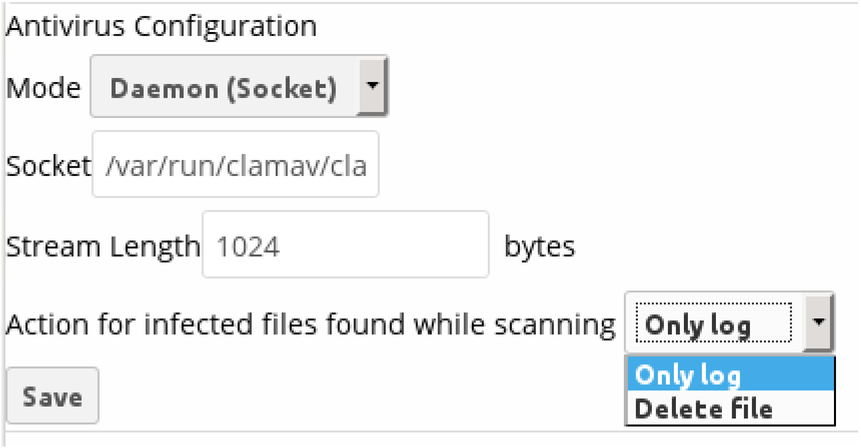
Stream Length 1024 bytes

Action for infected files found while scanning Only log

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.



Antivirus Configuration

Mode Daemon (Socket)

Socket /var/run/clamav/cla

Stream Length 1024 bytes

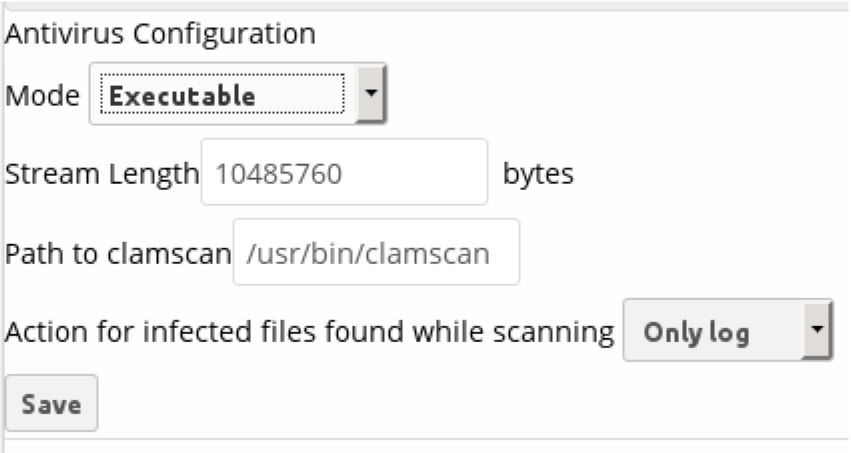
Action for infected files found while scanning Only log

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.

6.3 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 the *Database Configuration*. For this reason, ownCloud provides an automatic configuration feature.



Antivirus Configuration

Mode: Executable

Stream Length: 10485760 bytes

Path to clamscan: /usr/bin/clamscan

Action for infected files found while scanning: Only log

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.

6.3.1 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

6.3.2 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*.

6.4 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.

6.4.1 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

6.4.2 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 is 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: 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.

6.5 Uploading big files > 512MB (as set by default)

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

- < 2GB on 32Bit OS-architecture
- < 2GB with Server Version 4.5 or older
- < 2GB on Windows (32Bit and 64Bit)
- < 2GB with IE6 - IE8
- < 4GB with IE9 - IE10

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

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

6.5.1 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 users uploading the same time is 100: temp space has to hold at least 10x100 GB

6.5.2 Configuring Your Webserver

ownCloud comes with its own `owncloud/.htaccess` file. Set the following two parameters inside this `.htaccess` 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 option in your webserver config which could prevent the upload of larger files. Please see the manual of your webserver how to configure those values correctly:

Apache

- LimitRequestBody
- SSLRenegBufferSize

Apache with mod_fcgid

- FcgidMaxRequestLen

NginX

- client_max_body_size

IIS

- maxAllowedContentLength

6.5.3 Configuring PHP

If you don't want to use the ownCloud `.htaccess` 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 `php.ini`, or PHP will return memory-related errors:

- `output_buffering = 0`

6.6 Configuring the Collaborative Documents App

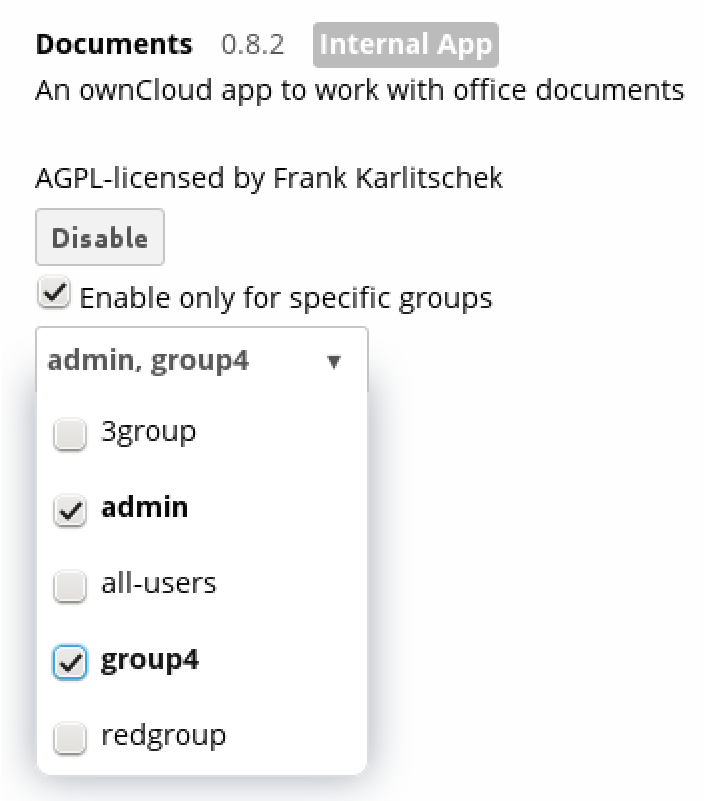
The Documents application supports editing documents within ownCloud, without the need to launch an external application. The Documents app supports these features:

- 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.

6.6.1 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.



Documents 0.8.2 Internal App

An ownCloud app to work with office documents

AGPL-licensed by Frank Karlitschek

☑ Enable only for specific groups

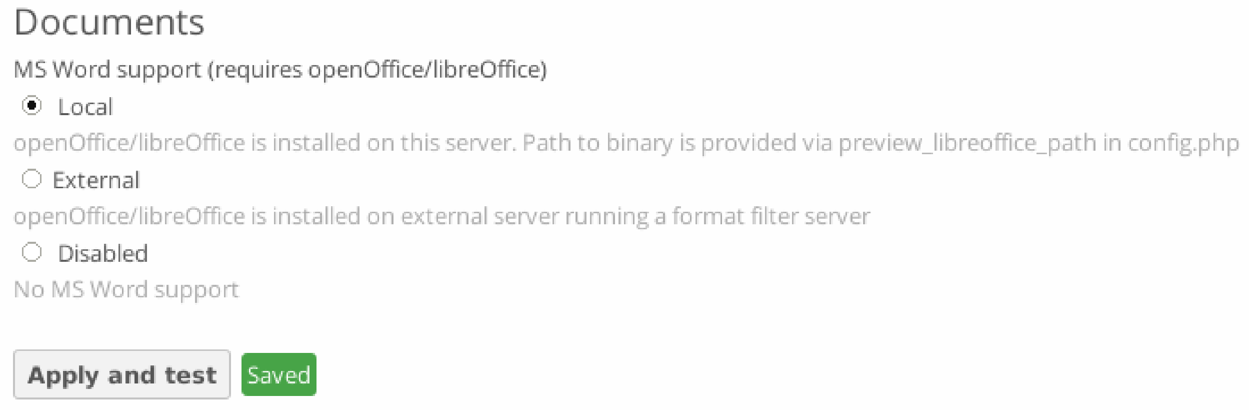
admin, group4 ▾

- □ 3group
- ☑ admin
- □ all-users
- ☑ group4
- □ redgroup

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

6.6.2 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.



Documents

MS Word support (requires openOffice/libreOffice)

● Local

openOffice/libreOffice is installed on this server. Path to binary is provided via preview_libreoffice_path in config.php

○ External

openOffice/libreOffice is installed on external server running a format filter server

○ Disabled

No MS Word support

Troubleshooting

If the mentioned test fails please make sure that:

- you're not running Windows (it is currently not supported)

- 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`.

6.7 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. This document provides a more detailed reference. Many options are configurable on your Admin page, so it is usually not necessary to edit `config/config.php`.

6.7.1 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.

‘passwordsalt’ => ‘d3c944a9af095aa08f’,

```
'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 available only in ownCloud Community 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 - Community Edition Only)
- mysql (MySQL/MariaDB)
- pgsql (PostgreSQL)
- oci (Oracle - Enterprise Edition Only)
- mssql (Microsoft SQL Server - 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.

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

Indicates whether the ownCloud instance was installed successfully; `true` indicates a successful installation, and `false` indicates an unsuccessful installation. .. DEFAULT_SECTION_END .. Generated content above. Don't change this.

6.7.2 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' => '2c5778476346786306303',
  '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,
);
```

6.7.3 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).

6.7.4 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`. Specified 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 set an 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.

6.7.5 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`.

6.7.6 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.

6.7.7 ownCloud Verifications

ownCloud performs several verification checks. There are two options, `true` and `false`.

```
'appcodechecker' => true,
```

Check 3rd party apps to make sure they are using the private API and not the public API. If the app uses the private API it cannot be installed.

```
'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.

6.7.8 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`.

```
'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.

6.7.9 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.

6.7.10 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.

The app store is disabled by default for ownCloud Enterprise Edition

```
'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,
```

Check 3rd party apps to make sure they are using the private API and not the public API. If the app uses the private API it cannot be installed.

6.7.11 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:

- Images 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\Movie

- OC\Preview\MSOffice2003
- OC\Preview\MSOffice2007
- OC\Preview\MSOfficeDoc
- OC\Preview\OpenDocument
- OC\Preview\PDF
- OC\Preview\StarOffice
- OC\Preview\SVG

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

6.7.12 LDAP

Global settings used by LDAP User and Group Backend

```
'ldapUserCleanupInterval' => 51,
```

defines the interval in minutes for the background job that checks user existance 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

6.7.13 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.

6.7.14 SSL

```
'forcessl' => false,
```

Change this to `true` to require HTTPS for all connections, and to reject HTTP requests.

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

Extra SSL options to be used for configuration.

6.7.15 Miscellaneous

```
'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.

```
'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) 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.

```
'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`.

Note: Test this thoroughly on production systems as it should work reliably with core apps, but you may encounter problems with community/third-party apps.

```
'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.

```
'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',
        ),
),
```

The example below 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 - Community Edition Only)
- 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 *",
```

Custom CSP policy, changing this will overwrite the standard policy

```
'secret' => 'ICertainlyShouldHaveChangedTheDefaultSecret',
```

Secret used by ownCloud for various purposes, e.g. to encrypt data. If you lose this string there will be data corruption.

6.7.16 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`

6.8 Custom Client Configuration

If you want to access your ownCloud, you can choose between the standard Web-GUI and various client synchronization applications.

Note: Download links that point to these applications are shown at the top of the Personal Settings Menu.

The following sync applications are currently available by default:

- Desktop sync clients for Windows, MAC and Linux OS
- Mobile sync client for Android devices
- Mobile sync client for iOS devices

6.8.1 Parameters

You can customize the download links to meet your specific requirements for any of the synchronization clients in the `config/config.php` file:

```
<?php

  "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",
```

6.9 Database Configuration

ownCloud requires a database in which administrative data is stored. The following databases are currently supported:

- MySQL / MariaDB
- SQLite
- PostgreSQL
- Oracle

The MySQL or MariaDB databases are the recommended database engines. However, because it is a file based database with the least administrative overhead, SQLite is chosen by default.

Note: Because SQLite does not handle large datasets or large numbers of users well, we recommend that it be used only for single user ownCloud installations, or for simple testing setups.

6.9.1 Requirements

Choosing to use MySQL / MariaDB, PostgreSQL, or Oracle as your database requires that you install and set up the server software first.

Note: The steps for configuring a third party database are beyond the scope of this document. Please refer to the documentation for your specific database choice for instructions.

6.9.2 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 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_",
```

SQLite Database

If you decide to use a SQLite database make sure that you have installed and enabled the SQLite extension in PHP. The PHP configuration in `/etc/php5/conf.d/sqlite3.ini` could look like this:

```
# configuration for PHP SQLite3 module
extension=pdo_sqlite.so
extension=sqlite3.so
```

It is not necessary to create a database and a database user in advance because this will automatically be done by ownCloud when you login for the first time.

An ownCloud instance configured to use sqlite only needs to contain the reference to a writable data directory (which is required for successful operation of ownCloud in general anyway). The `config/config.php` as created by the *Installation Wizard* could therefore contain entries like this:

```
<?php

  "dbtype"        => "sqlite",
  "dbname"        => "owncloud",
  "dbuser"        => "",
  "dbpassword"    => "",
  "dbhost"        => "",
  "dbtableprefix" => "",
  "datadirectory" => "/var/www/html/owncloud/data",
```

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_",
```

Oracle Database

If you are deploying to an Oracle database make sure that you have installed and enabled the Oracle extension in PHP. The PHP configuration in `/etc/php5/conf.d/oci8.ini` could look like this:

```
# configuration for PHP Oracle extension
extension=oci8.so
```

Make sure that the Oracle environment has been set up for the process trying to use the Oracle extension. For a local Oracle XE installation this can be done by exporting the following environment variables (eg. in `/etc/apache2/envvars` for Apache)

```
export ORACLE_HOME=/u01/app/oracle/product/11.2.0/xe
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME/lib
```

Installing and configuring Oracle support for PHP is way out of scope for this document. The official Oracle documentation called The Underground PHP and Oracle Manual should help you through the process.

Creating a database user for ownCloud can be done by using the sqlplus command line interface or the Oracle Application Express web interface. The database tables will be created by ownCloud when you login for the first time.

To start the Oracle command line mode with a DBA account use:

```
sqlplus system AS SYSDBA
```

After entering the password a **SQL>** prompt will appear. Now enter the following lines and confirm them with the enter key:

```
CREATE USER owncloud IDENTIFIED BY password;
ALTER USER owncloud DEFAULT TABLESPACE users
                    TEMPORARY TABLESPACE temp
                    QUOTA unlimited ON users;
GRANT create session
    , create table
    , create procedure
    , create sequence
    , create trigger
    , create view
    , create synonym
    , alter session
   TO owncloud;
```

Note: In Oracle creating a user is the same as creating a database in other RDBMs, so no `CREATE DATABASE` statement is necessary.

You can quit the prompt by entering:

```
exit
```

An ownCloud instance configured with Oracle 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"        => "oci",
  "dbname"        => "XE",
  "dbuser"        => "owncloud",
  "dbpassword"    => "password",
  "dbhost"        => "localhost",
```

Note: This example assumes you are running an Oracle Express Edition on `localhost`. The `dbname` is the name of the Oracle instance. For Oracle Express Edition it is always `XE`.

6.9.3 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:

SQLite:

```
sqlite3 /www/htdocs/owncloud/data/owncloud.db

sqlite> .version
SQLite 3.7.15.1 2012-12-19 20:39:10 6b85b767d0ff7975146156a99ad673f2c1a23318
sqlite> .quit
```

MySQL:

Assuming the database server is installed on the same sytem 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 sytem 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
```

Oracle:

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
```

Useful SQL commands

Show Database Users:

```
SQLite    : No database user is required.
MySQL     : SELECT User,Host FROM mysql.user;
PostgreSQL: SELECT * FROM pg_user;
Oracle    : SELECT * FROM all_users;
```

Show available Databases:

```
SQLite    : .databases (normally one database per file!)
MySQL     : SHOW DATABASES;
PostgreSQL: \l
Oracle    : SELECT name FROM v$database; (requires DBA privileges)
```

Show ownCloud Tables in Database:

```
SQLite    : .tables
MySQL     : USE owncloud; SHOW TABLES;
PostgreSQL: \c owncloud; \d
Oracle    : SELECT table_name FROM user_tables;
```

Quit Database:

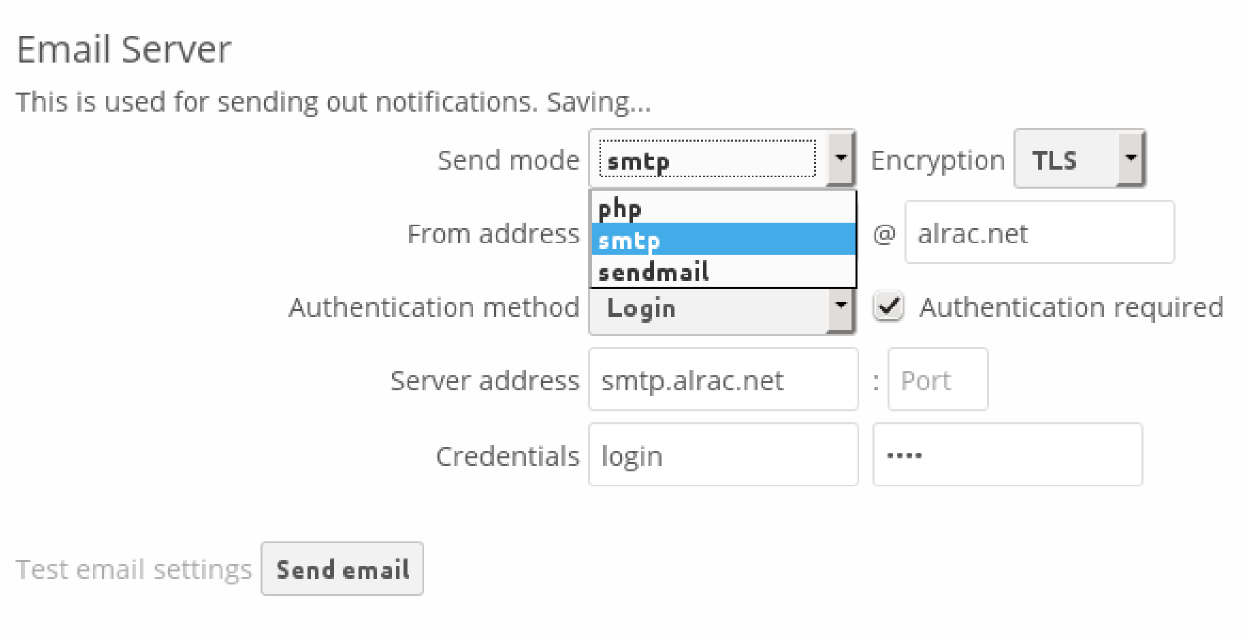
```
SQLite    : .quit
MySQL     : quit
PostgreSQL: \q
Oracle    : quit
```

6.10 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.



Email Server

This is used for sending out notifications. Saving...

Send mode: smtp

Encryption: TLS

From address: @ alrac.net

Authentication method: Login

☑ Authentication required

Server address: smtp.alrac.net : Port

Credentials: login ••••

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.

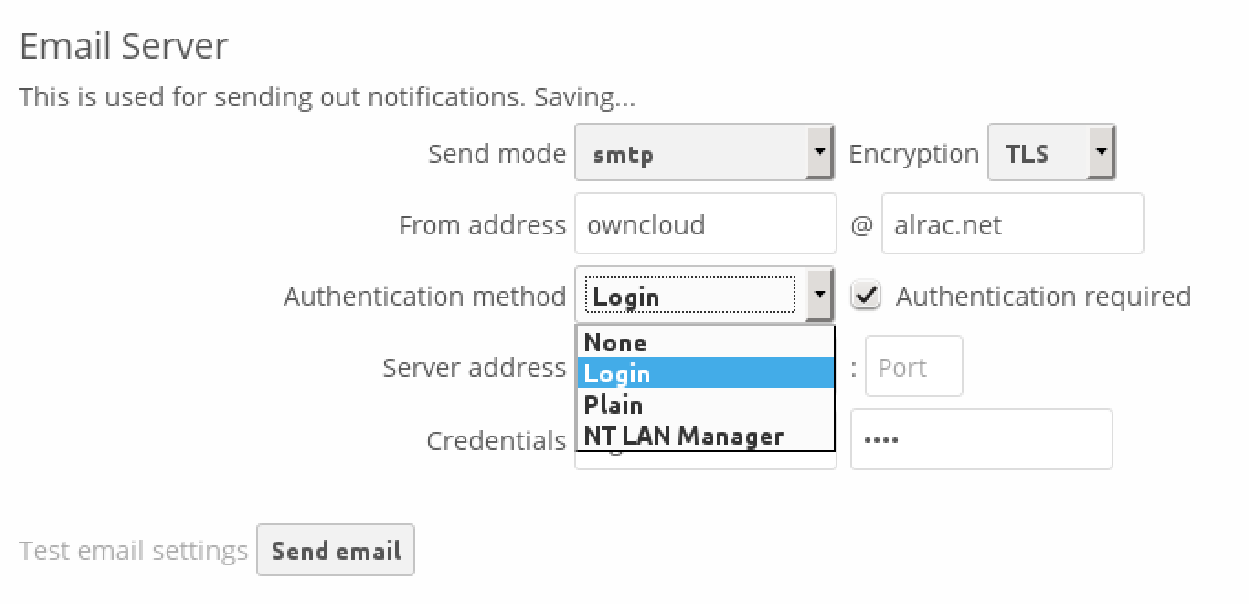
Note: The Sendmail option refers to the Sendmail SMTP server, and any drop-in Sendmail replacement such as Postfix, Exim, or Courier. All of these include a `sendmail` binary, and are freely-interchangeable.

6.10.1 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:



Email Server

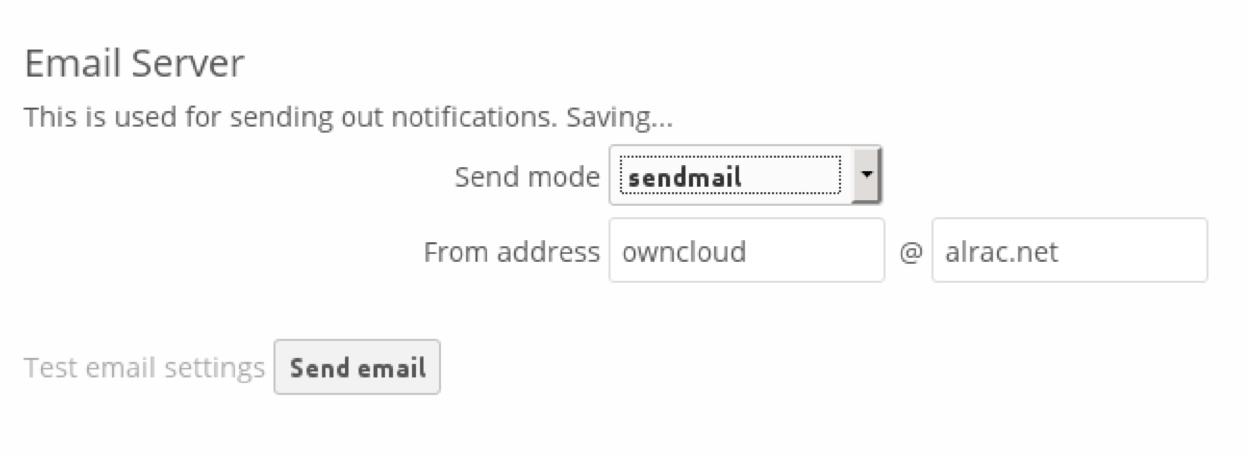
This is used for sending out notifications. Saving...

Send mode: smtp
Encryption: TLS
From address: owncloud @ alrac.net
Authentication method: Login
☑ Authentication required
Server address: : Port
Credentials: ••••

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

--
ownCloud
web services under your control
```

6.10.2 Configuring PHP and Sendmail

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



Email Server

This is used for sending out notifications. Saving...

Send mode: sendmail
From address: owncloud @ alrac.net

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.

6.10.3 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.

6.10.4 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 will change that by modifying the **mail_smtpport** parameter. Multiple servers can be entered, separated by semicolons:

```php
<?php

  "mail_smtpmode"      => "smtp",
  "mail_smtphost"      => "smtp-1.server.dom;smtp-2.server.dom:425",
  "mail_smtpport"      => 25,
```

or

```php
<?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
<?php

  "mail_smtptimeout"  => 30,
```

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

```php
<?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
<?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
<?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
<?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" => "",
```

6.10.5 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 *Email Server* section of the Admin settings page.

6.10.6 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 a 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.
```

6.10.7 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.

6.11 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.

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

When files on external storage are encrypted in 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 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

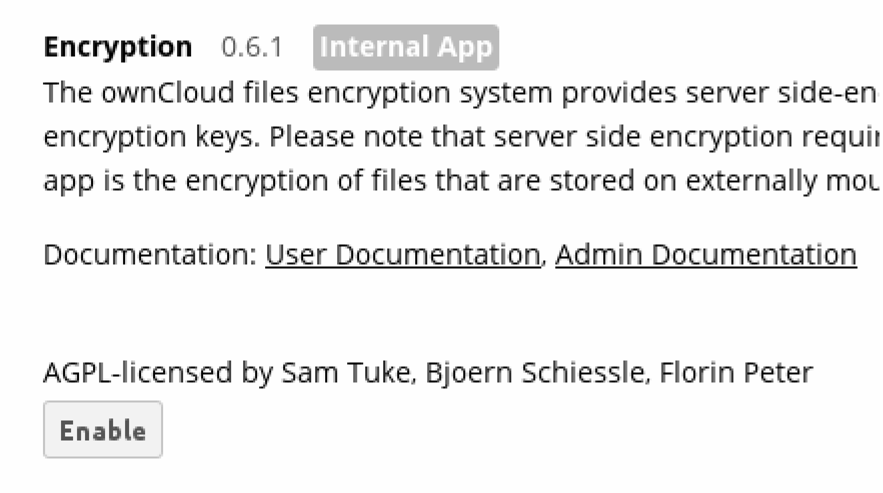
`data/files_encryption` private keys and all other keys necessary to decrypt the files stored on a system wide external storage

Note: 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.

6.11.1 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."

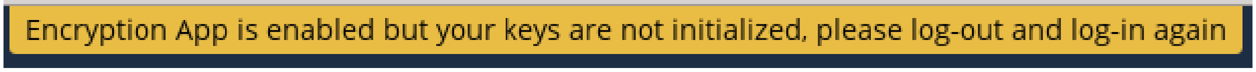


Encryption 0.6.1 Internal App

The ownCloud files encryption system provides server side-en
encryption keys. Please note that server side encryption requi
app is the encryption of files that are stored on externally mou

Documentation: User Documentation, Admin Documentation

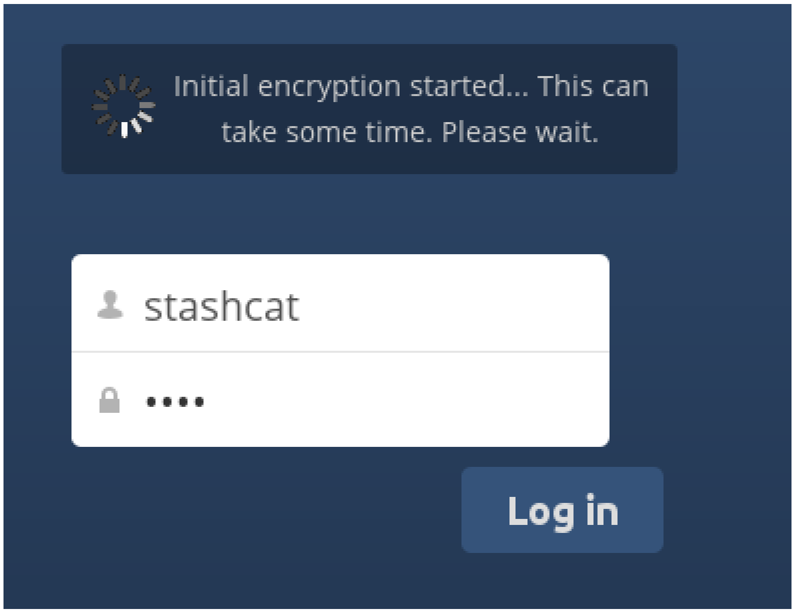
AGPL-licensed by Sam Tuke, Bjoern Schiessle, Florin Peter



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.



Initial encryption started... This can take some time. Please wait.

stashcat

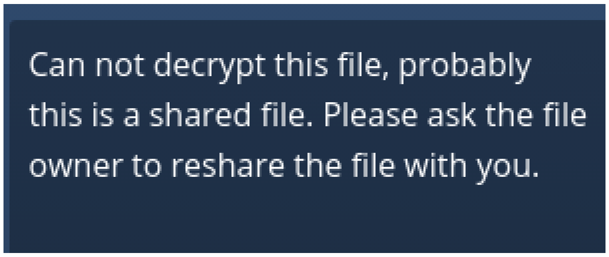
••••

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.

6.11.2 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."

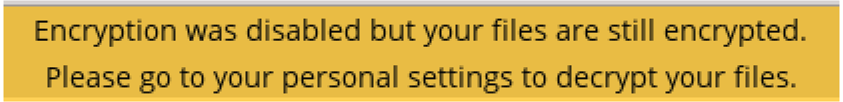
Share owners may need to re-share files after encryption is enabled; users trying to access the share will see a message advising them to ask the share owner to re-share the file with them. For individual shares, un-share and re-share the file. For group shares, share with any individuals who can't access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



Can not decrypt this file, probably this is a shared file. Please ask the file owner to reshare the file with you.

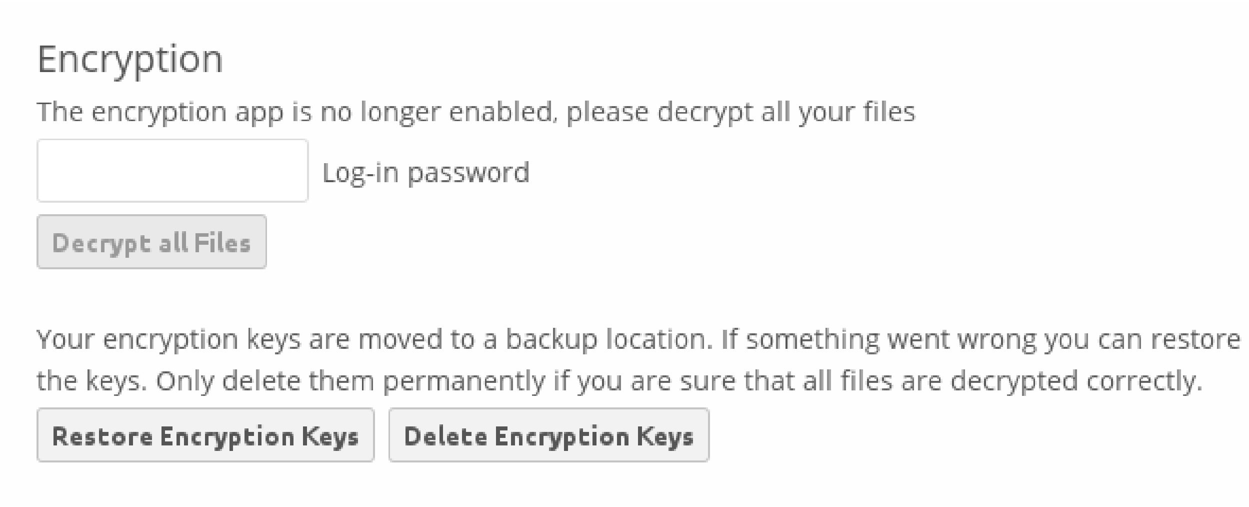
6.11.3 Decrypting Encrypted Files

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".



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.



Encryption

The encryption app is no longer enabled, please decrypt all your files

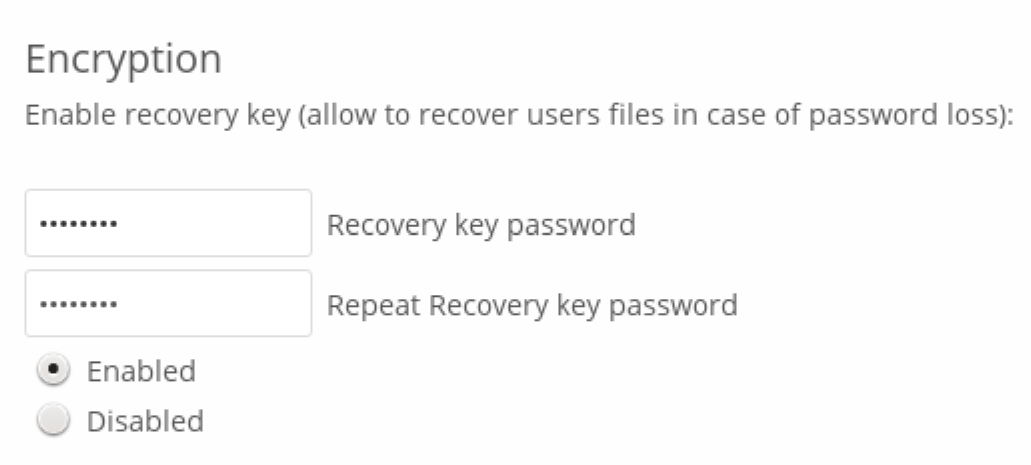
Your encryption keys are moved to a backup location. If something went wrong you can restore the keys. Only delete them permanently if you are sure that all files are decrypted correctly.

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.

6.11.4 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.



Encryption

Enable recovery key (allow to recover users files in case of password loss):

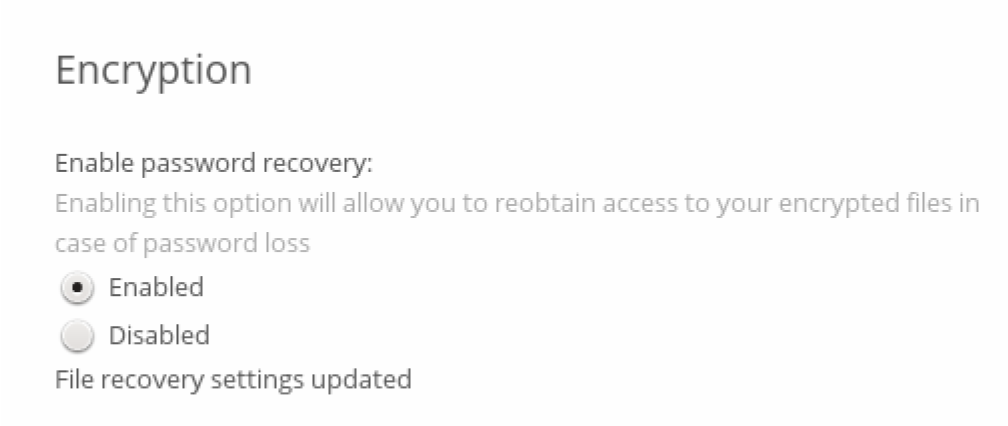
•••••••• Recovery key password

•••••••• Repeat Recovery key password

● Enabled

○ Disabled

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.



Encryption

Enable password recovery:

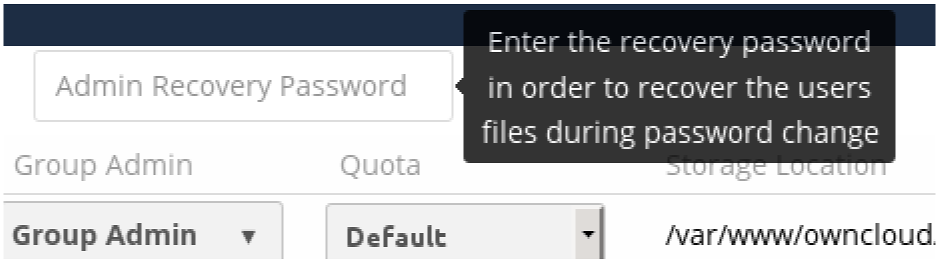
Enabling this option will allow you to reobtain access to your encrypted files in case of password loss

● Enabled

○ Disabled

File recovery settings updated

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.



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.

6.11.5 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.
- 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.

6.11.6 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.

6.11.7 "Missing requirements" Message on Windows Servers

If you get a "Missing requirements" error message when you enable encryption on a Windows server, enter the absolute location of your openSSL configuration file in `config.php`:

```
'openssl' => array(
    'config' => 'C:\path\to\openssl.cnf',
),
```

For example, in a typical installation on a 64-bit Windows 7 system it looks like this:

```
'openssl' => array(
    'config' => 'C:\OpenSSL-Win64\openssl.cnf',
),
```

There are many ways to configure OpenSSL, so be sure to verify your correct file location.

6.12 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.

6.12.1 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

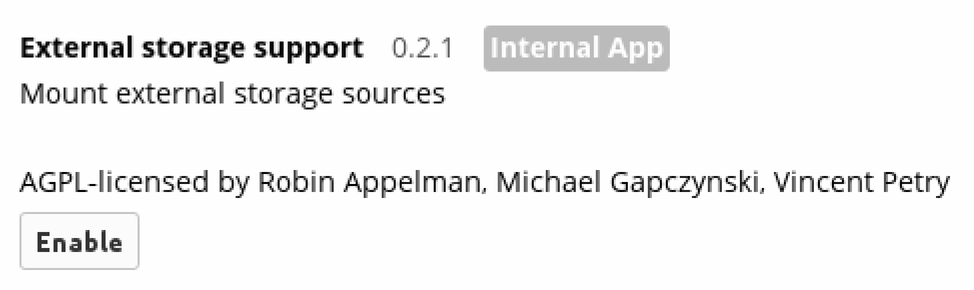
ownCloud users can be given permission to mount any of these, except local storage.

To understand how ownCloud manages passwords for external mounts, and the security implications, see the **External Storage Password Management** section of *Configuring External Storage (Configuration File)*.

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

6.12.2 Enabling External Storage Support

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



External storage support 0.2.1

Mount external storage sources

AGPL-licensed by Robin Appelman, Michael Gapczynski, Vincent Petry

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

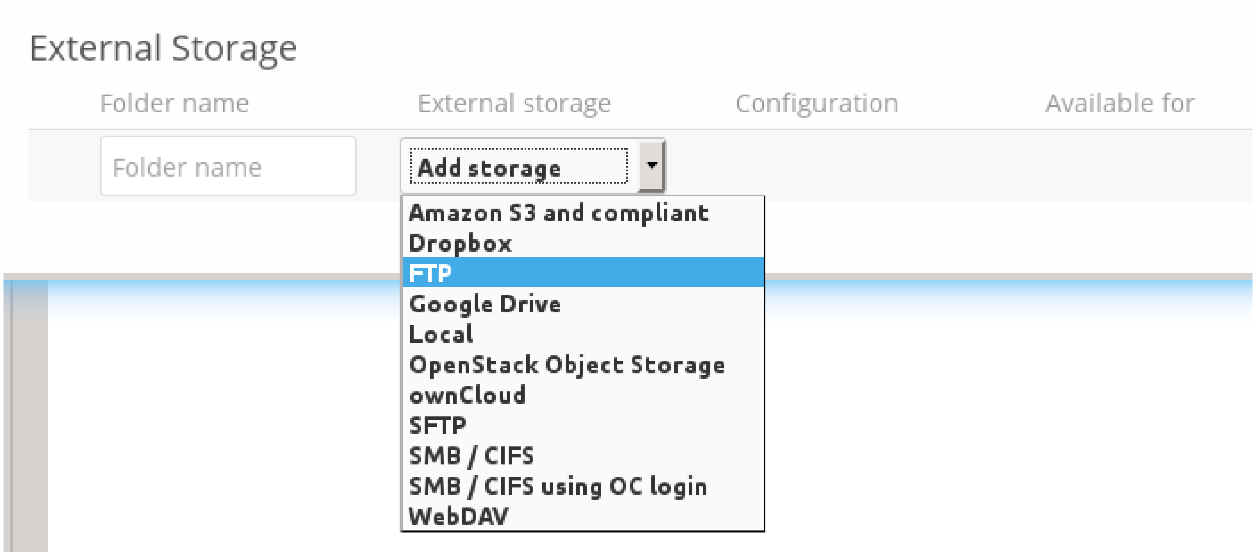
When your configuration is correct you'll see a green light at the left, and if it isn't you'll see a red light.

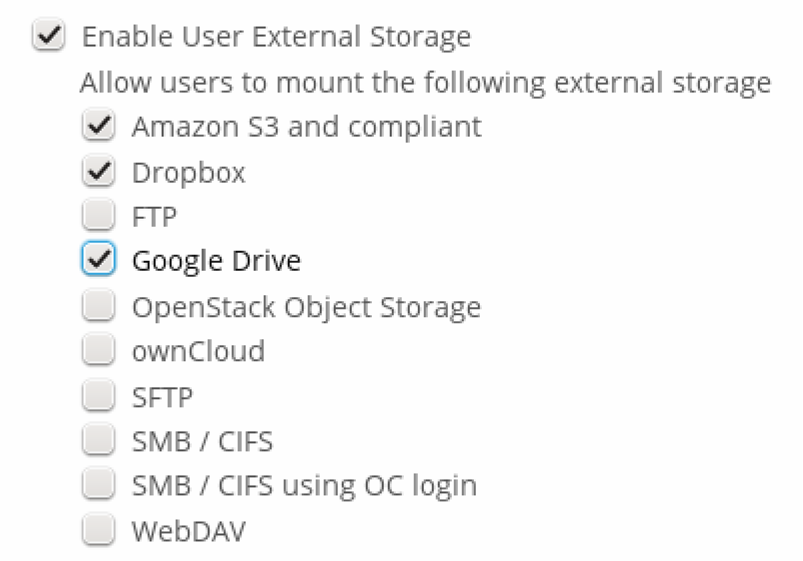
Check `Enable User External Storage` to allow your users to mount their own external storage services, and check the services you want to allow.

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

6.12.3 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 informations.




☑ Enable User External Storage

Allow users to mount the following external storage

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

6.12.4 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 that a users is browsing your ownCloud installation.

Please also be aware that ownCloud might not always be able to find out what has been changed remotely (files changes 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.

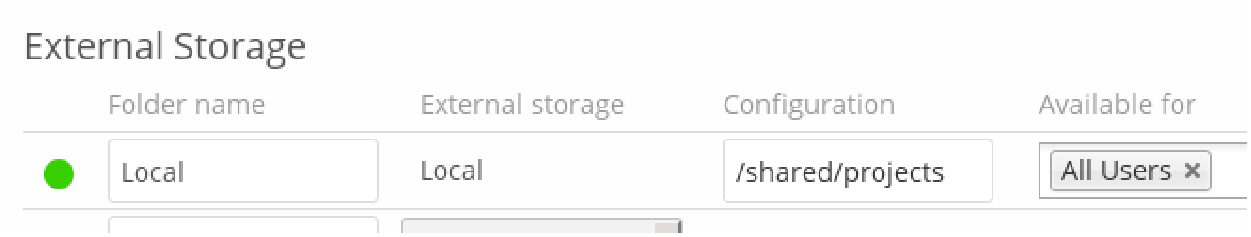
6.12.5 Local Storage

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.



External Storage

	Folder name	External storage	Configuration	Available for
●	Local	Local	/shared/projects	All Users ×

6.12.6 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.

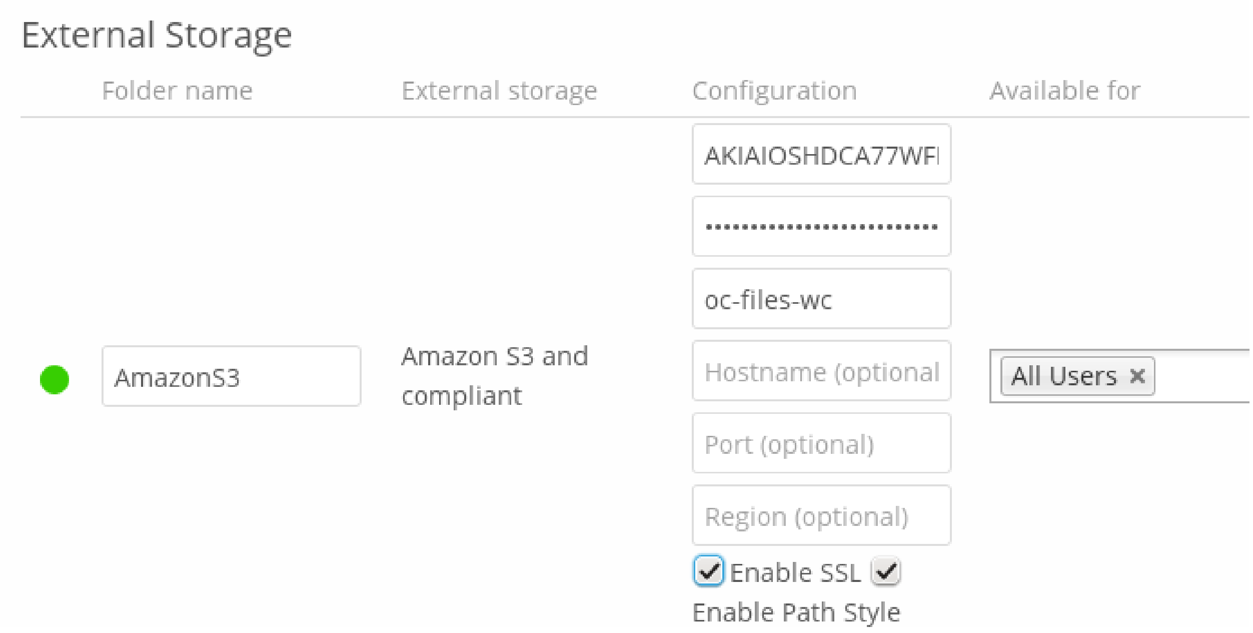
In the `Bucket` field enter the name of your S3 bucket you want to share.

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.

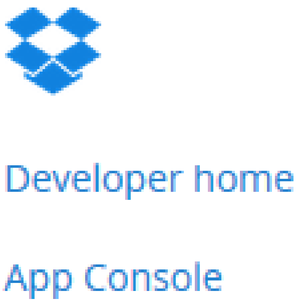
6.12.7 Dropbox

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



External Storage

	Folder name	External storage	Configuration	Available for
●	AmazonS3	Amazon S3 and compliant	AKIAIOSHDCA77WFI •••••••••••••••••••••••••• oc-files-wc Hostname (optional Port (optional) Region (optional) ☑ Enable SSL ☑ Enable Path Style	All Users ×



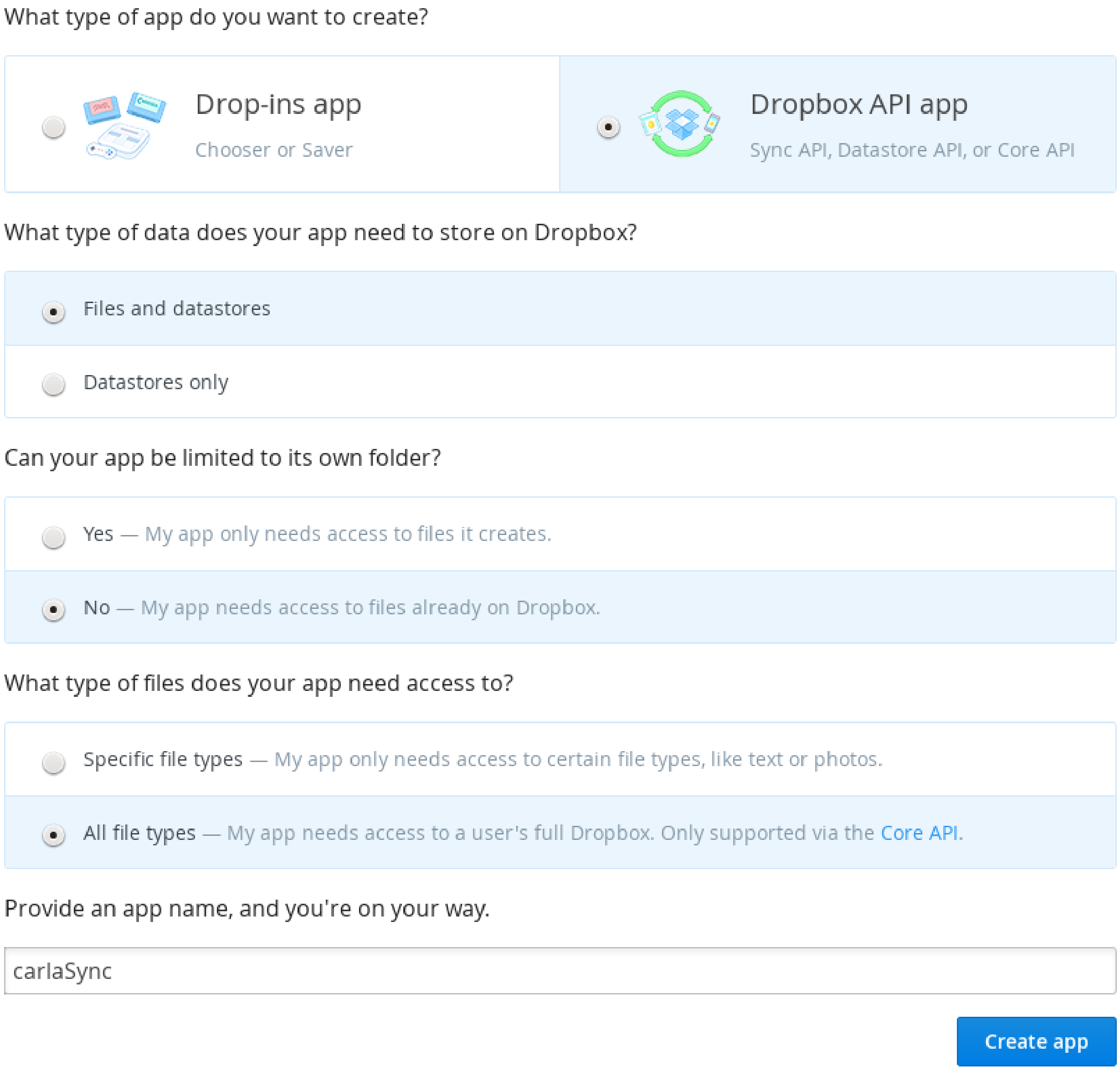
Developer home

App Console

If you have not already created any Dropbox apps it will ask you to accept their terms and conditions. Then you are presented with the choice to create either a Drop-ins App or a Dropbox API App. Click `Dropbox API App`, and then check:

- Files and datastores.
- No – My app needs access to files already on Dropbox.
- All file types – My app needs access to a user's full Dropbox. Only supported via the CoreAPI.

Then enter whatever name you want for your app.



What type of app do you want to create?

○ Drop-ins app — Chooser or Saver

● Dropbox API app — Sync API, Datastore API, or Core API

What type of data does your app need to store on Dropbox?

● Files and datastores

○ Datastores only

Can your app be limited to its own folder?

○ Yes — My app only needs access to files it creates.

● No — My app needs access to files already on Dropbox.

What type of files does your app need access to?

○ Specific file types — My app only needs access to certain file types, like text or photos.

● All file types — My app needs access to a user's full Dropbox. Only supported via the Core API.

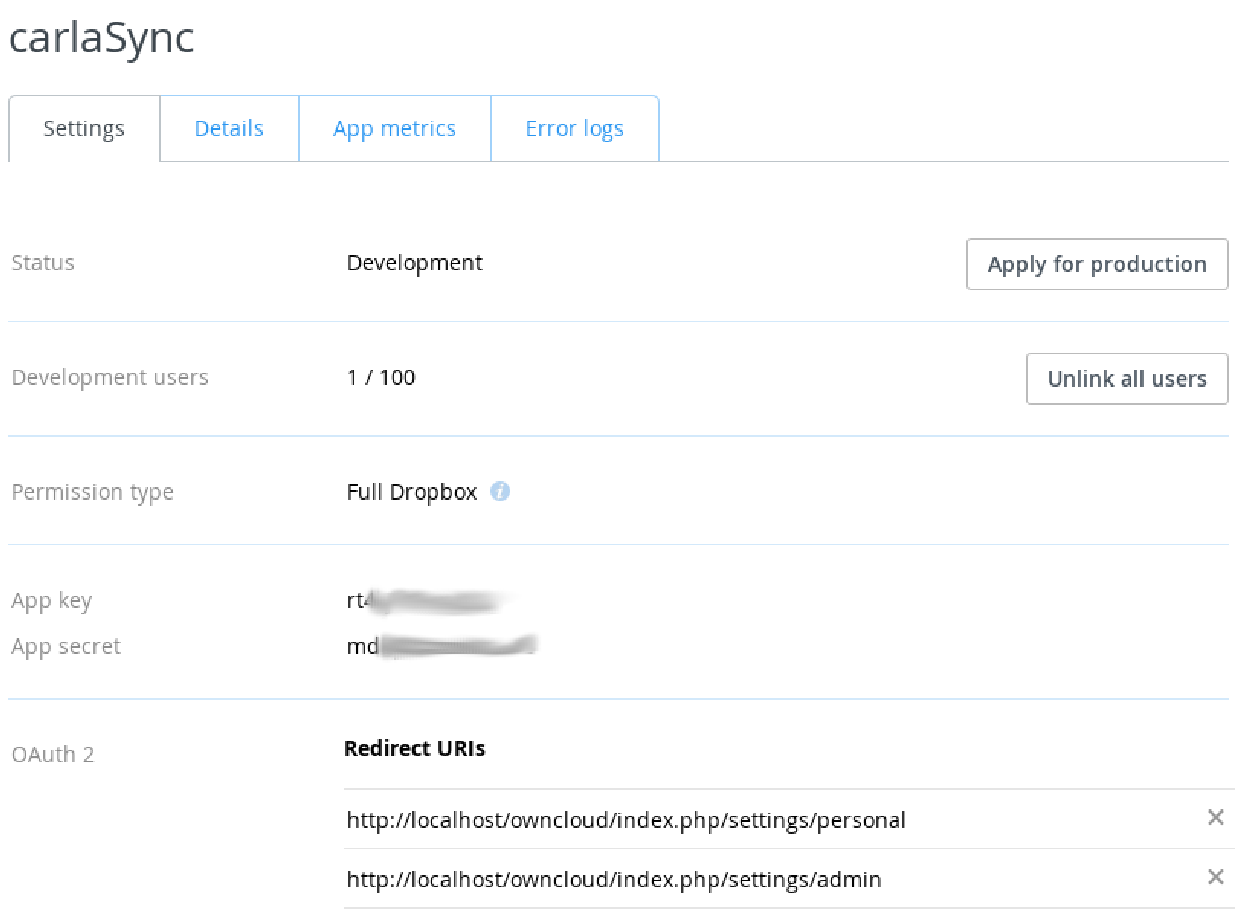
Provide an app name, and you're on your way.

carlaSync

Now click the `Create App` button. Under `Status`, do not click `Development (Apply for production status)` because that is for apps that you want to release publicly.

Click `Enable additional users` to allow multiple oC users to use your new Dropbox share.

Note your App key and App secret, which you will enter in the External Storage form on your ownCloud Admin page.



carlaSync

Status	Development	Apply for production
Development users	1 / 100	Unlink all users
Permission type	Full Dropbox	
App key	rt[redacted]	
App secret	md[redacted]	
OAuth 2	**Redirect URIs**	
	http://localhost/owncloud/index.php/settings/personal	×
	http://localhost/owncloud/index.php/settings/admin	×

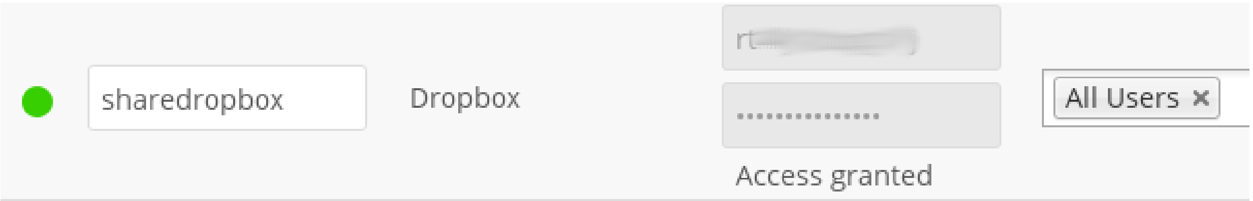
You need two `Redirect URIs`. You may use `localhost` as the hostname for testing because you don't need to use HTTPS, but this is not recommended for production use because it sends all traffic in the clear:

```
http://localhost/owncloud/index.php/settings/personal
http://localhost/owncloud/index.php/settings/admin
```

HTTPS is recommended for production use to encrypt your sessions:

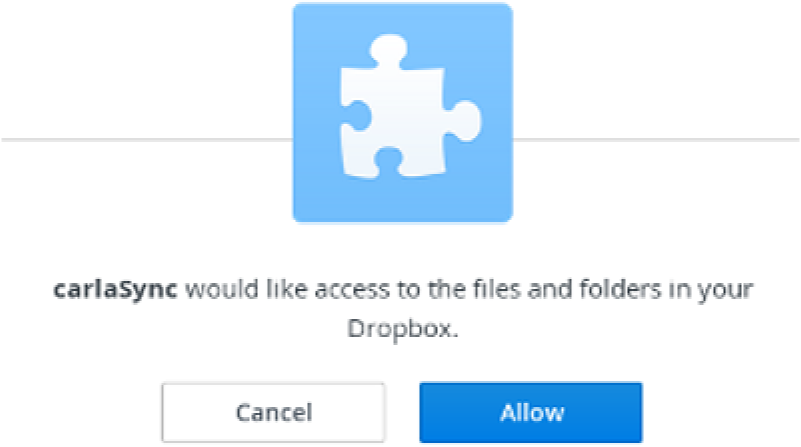
```
https://localhost/owncloud/index.php/settings/personal
https://localhost/owncloud/index.php/settings/admin

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

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.



●	sharedropbox	Dropbox	rt[redacted] ••••••••••••••• Access granted	All Users ×

You must be logged into Dropbox, and when ownCloud successfully verifies your connection Dropbox will ask for verification to connect to your Dropbox account. Click `Allow`, and you're done.

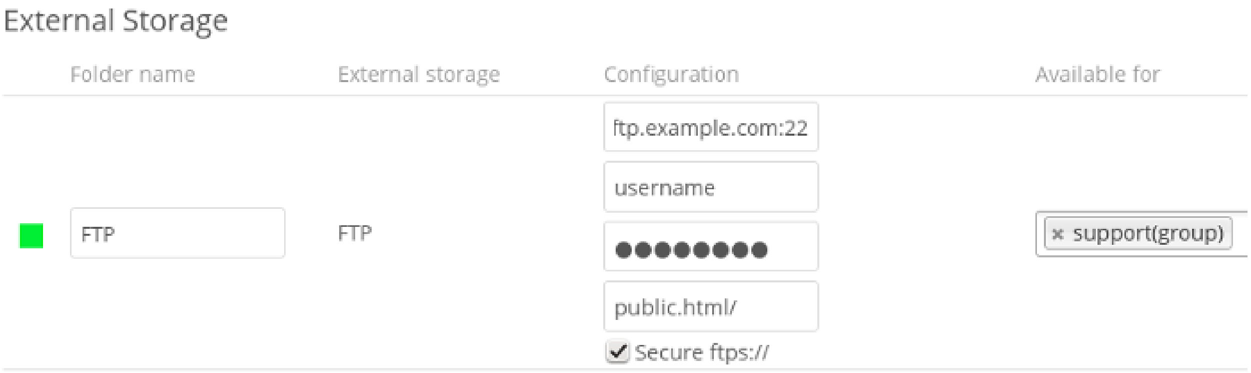


carlaSync would like access to the files and folders in your Dropbox.

6.12.8 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.
- 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.



External Storage

	Folder name	External storage	Configuration	Available for
🟩	FTP	FTP	ftp.example.com:22 username ●●●●●●●●● public.html/ ☑ Secure ftps://	× support(group)

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:

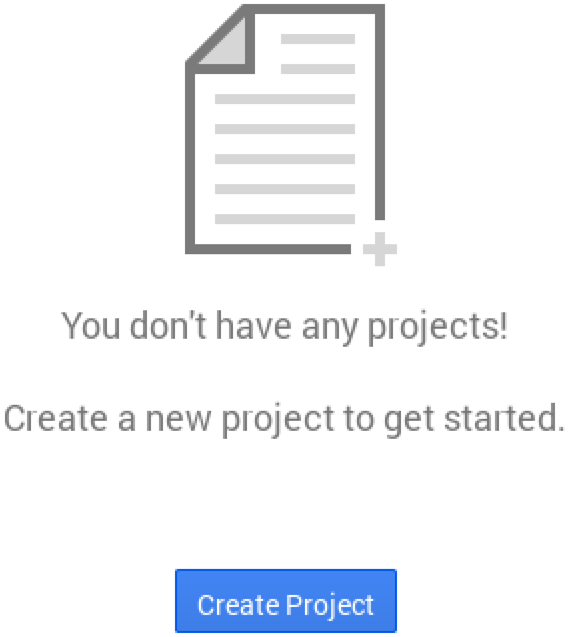
- Whatever name you want for your local mountpoint.
- The URL of your SFTP server.
- SFTP server username and password.
- The SFTP directory to mount in ownCloud.
- The ownCloud users or groups who are allowed to access the share.

6.12.9 Google Drive

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

Note: Your ownCloud server must have a registered domain name and be accessible over the Internet; Google Drive will not connect to a LAN-only server.

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 `Go to my console`, and then click the `Create Project` button. It takes a minute or two to create your new project.



You don't have any projects!

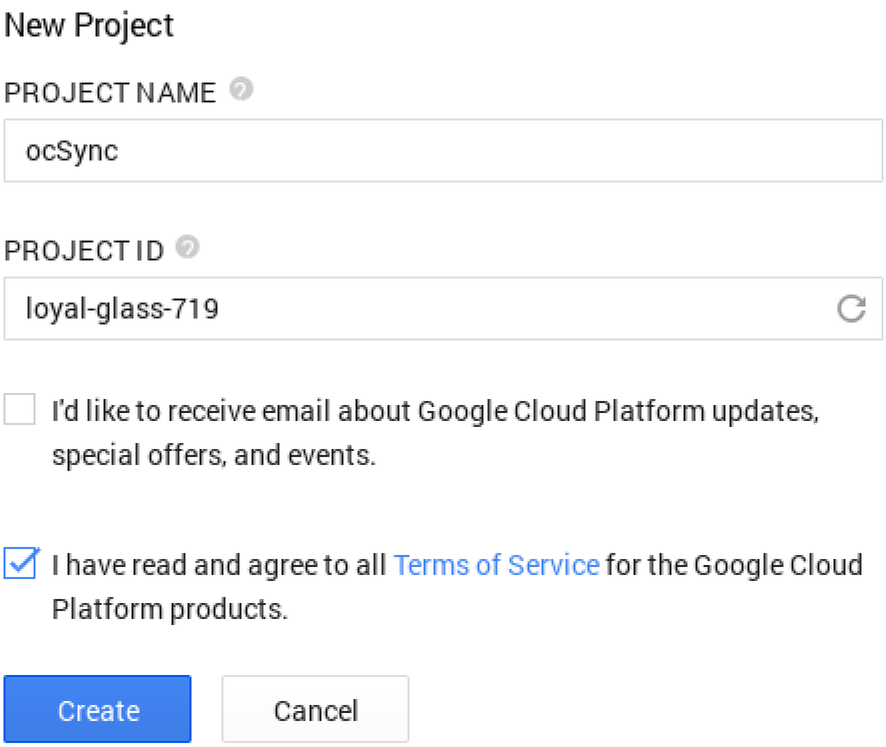
Create a new project to get started.

In the next screen give your project a name, accept the default `Project ID` or create your own, click the Terms of Service box, and click the `Create` button.

The next screen is your `Project Dashboard`. In the left sidebar click `APIs & Auth > APIs`, and then enable the `Drive API` and `Drive SDK` by toggling the boxes in the far-right `Status` column to the green On buttons.

This brings you to the `Google Drive SDK` screen. Click `API Access`.

This opens the `API Access` screen. Click the `Create a 0Auth 2.0 Client ID` button.



New Project

PROJECT NAME

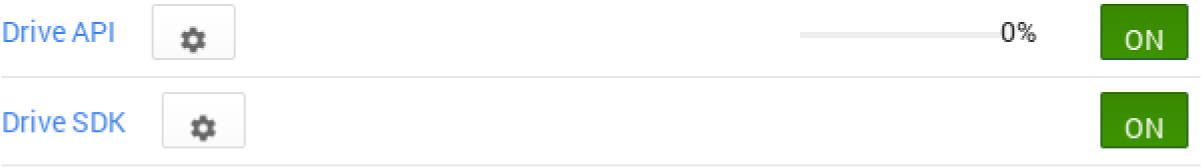
ocSync

PROJECT ID

loyal-glass-719

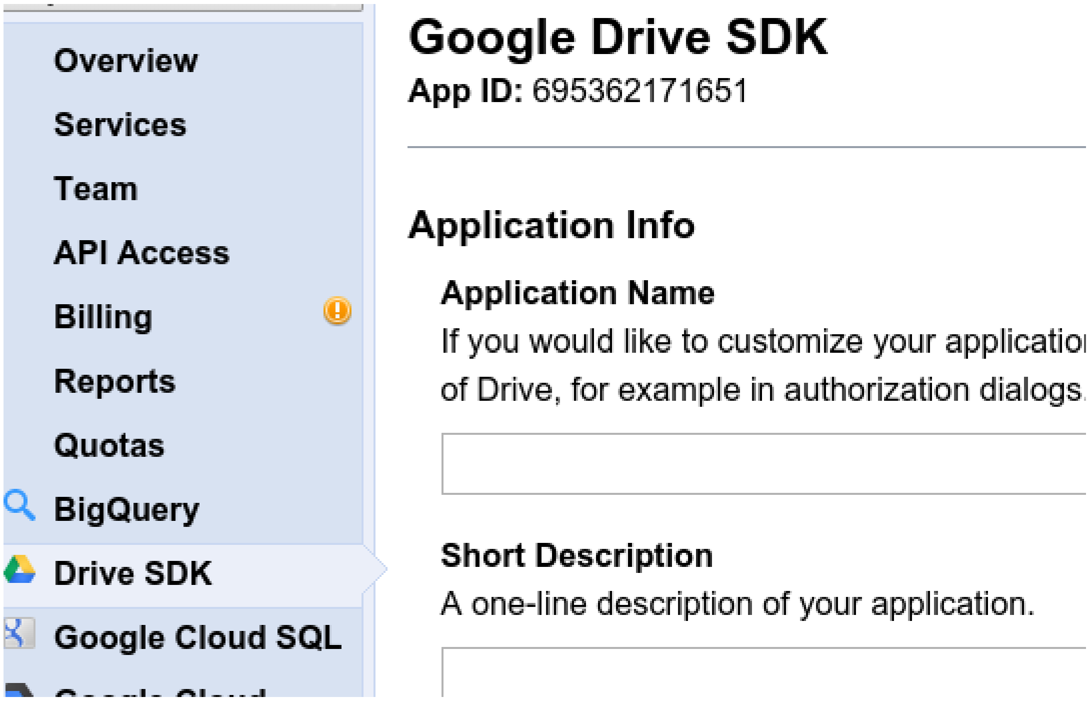
□ I'd like to receive email about Google Cloud Platform updates, special offers, and events.

☑ I have read and agree to all Terms of Service for the Google Cloud Platform products.



Drive API 0% ON

Drive SDK ON



Google Drive SDK

App ID: 695362171651

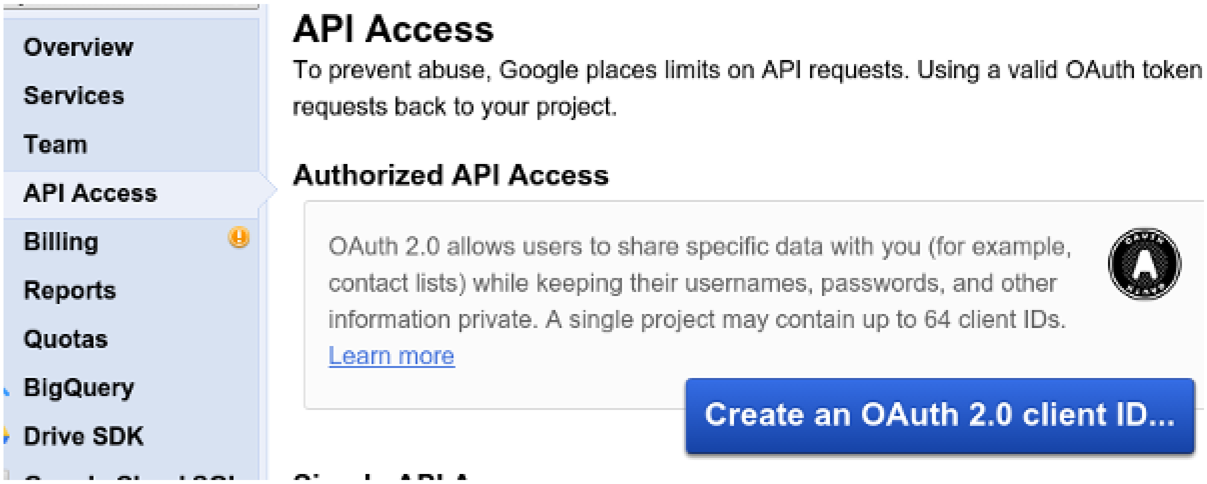
Application Info

Application Name

If you would like to customize your applicatio of Drive, for example in authorization dialogs

Short Description

A one-line description of your application.



API Access

To prevent abuse, Google places limits on API requests. Using a valid OAuth token requests back to your project.

Authorized API Access

OAuth 2.0 allows users to share specific data with you (for example, contact lists) while keeping their usernames, passwords, and other information private. A single project may contain up to 64 client IDs. Learn more

The next screen that opens is `Create Client ID: Branding Information`. Google requires to you to fill this out. When you're finished move on to the `Create Client ID: Client ID Settings` screen.

The `Application Type` is Web application.

Click `Your site or hostname (more options)` to expose `Authorized Redirect URIs`. Enter two Redirect URIs like these examples, replacing `https://example.com/owncloud/` with your own ownCloud server URL. You must use a registered domain name, and you cannot use the server's IP address.

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

Click `Create client ID` and you'll see a screen like this:

This contains your `Client ID` and `Client Secret`, which you need to set up your ownCloud connection. Go to your `Admin` page in ownCloud, create your new folder name, enter the Client ID and Client Secret, select your users and groups, and click `Grant Access`.

Google will open a dialogue asking for permission to connect to ownCloud. Click `Accept` and you're finished.

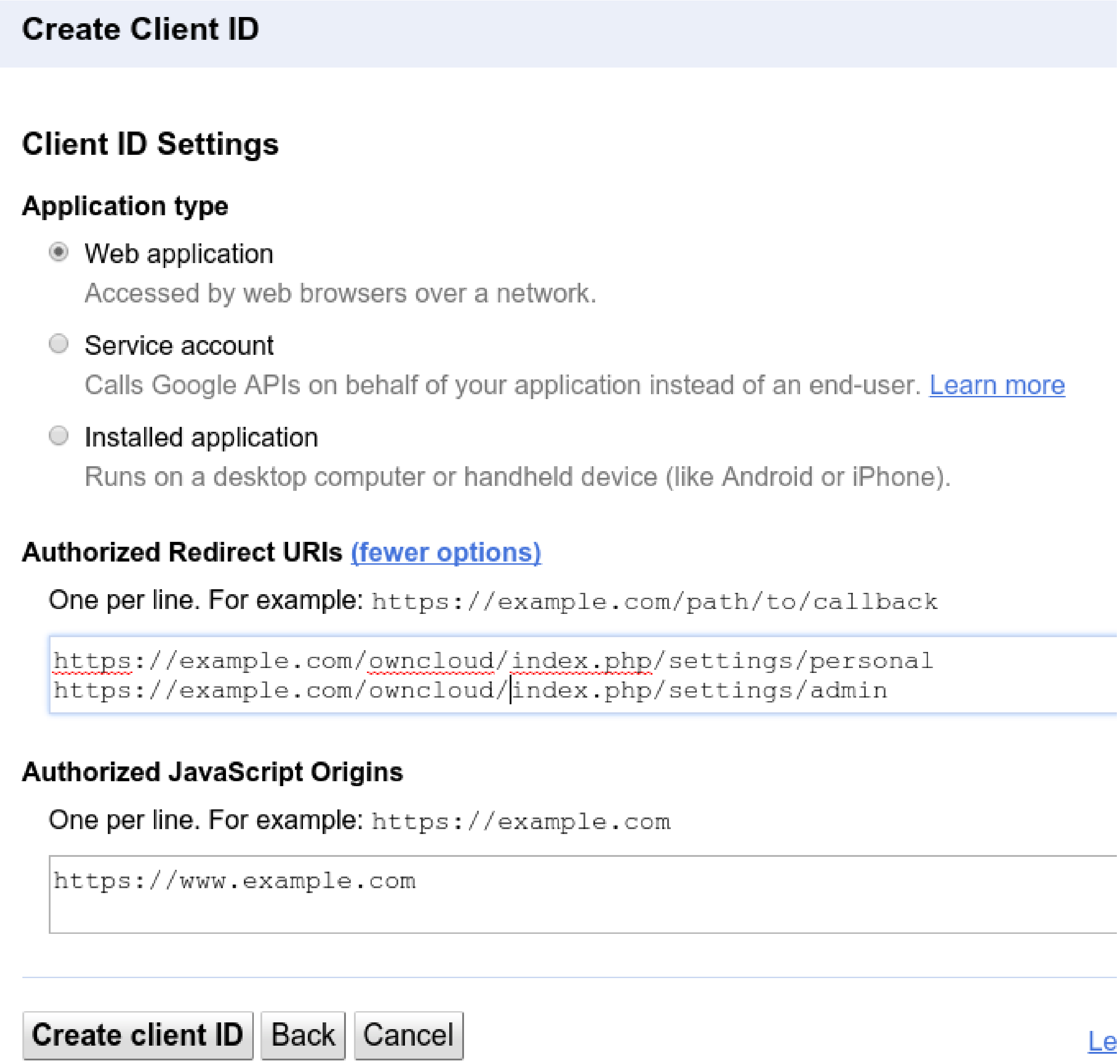
6.12.10 SMB/CIFS

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.
- Root – The folder 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.



Create Client ID

Client ID Settings

Application type

● Web application
Accessed by web browsers over a network.

○ Service account
Calls Google APIs on behalf of your application instead of an end-user. Learn more

○ Installed application
Runs on a desktop computer or handheld device (like Android or iPhone).

Authorized Redirect URIs (fewer options)

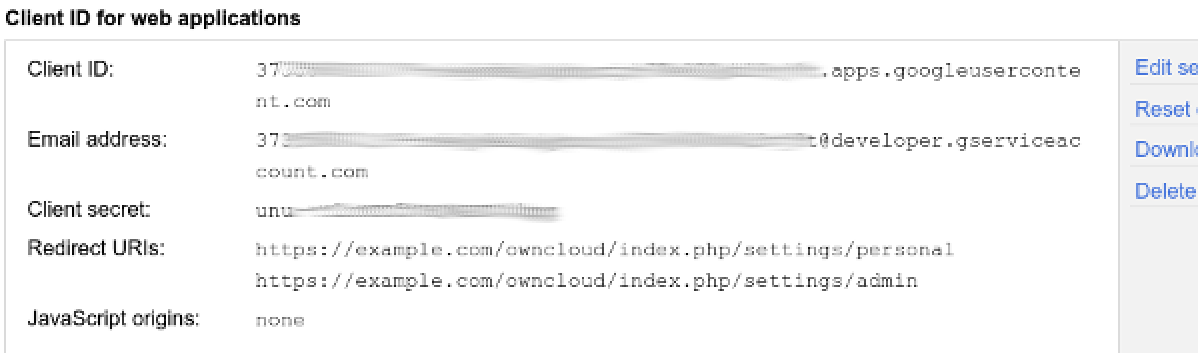
One per line. For example: `https://example.com/path/to/callback`

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

Authorized JavaScript Origins

One per line. For example: `https://example.com`

```
https://www.example.com
```



Client ID for web applications

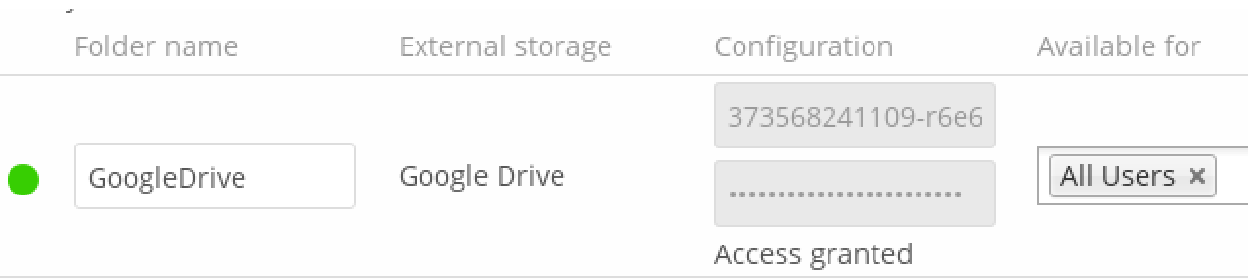
Client ID: 37[redacted].apps.googleusercontent.com

Email address: 37[redacted]@developer.gserviceaccount.com

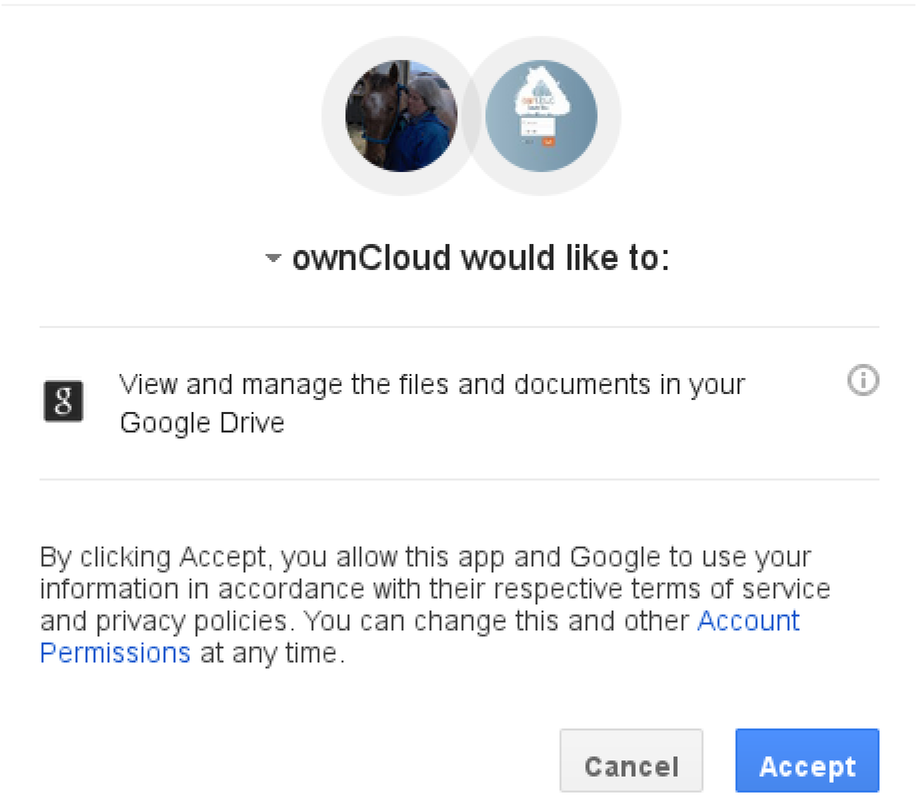
Client secret: unu[redacted]

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

JavaScript origins: none



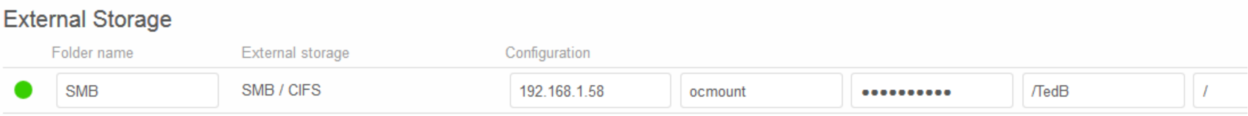
	Folder name	External storage	Configuration	Available for
●	GoogleDrive	Google Drive	373568241109-r6e6 •••••••••••••••••••••••• Access granted	All Users ×



▾ ownCloud would like to:

View and manage the files and documents in your Google Drive

By clicking Accept, 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.



External Storage

	Folder name	External storage	Configuration				
●	SMB	SMB / CIFS	192.168.1.58	ocmount	•••••••••••	/TedB	/

6.12.11 SMB/CIFS using OC login

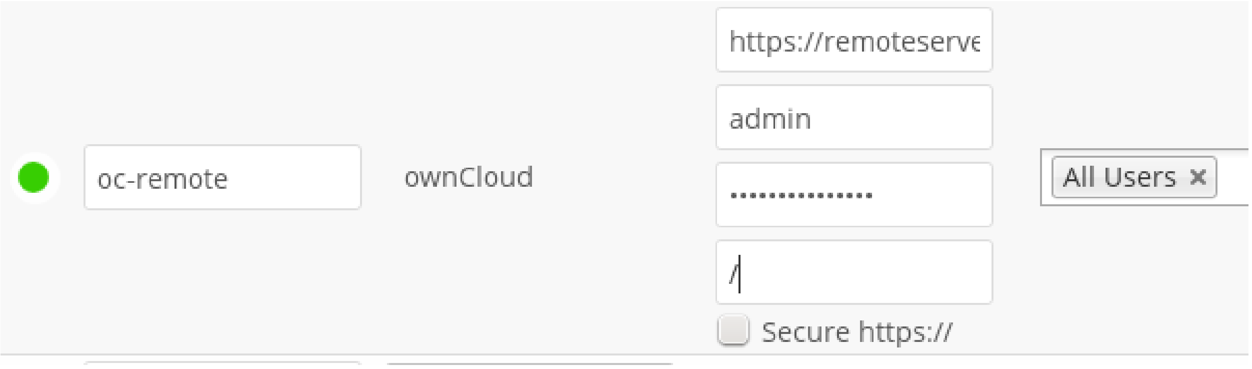
This works the same way as setting up a SMB/CIFS mount, except you can use your ownCloud logins intead of the SMB/CIFS server logins. To make this work, your ownCloud users need the same login and password as on the SMB/CIFS server.

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.

6.12.12 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.
- URL – The URL of the WebDAV or ownCloud server.
- Username and password for the remote server
- Root – The remote folder 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)



oc-remote

ownCloud

https://remoteserve

admin

....

/

☐ Secure https://

All Users ×

6.12.13 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

6.12.14 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.

6.13 Configuring External Storage (Configuration File)

Since ownCloud 4.0 it is possible to configure the filesystem to mount external storage providers into ownCloud's virtual file system. You can configure these file systems by creating and editing `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 consist 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

Note: You need to enable the *External storage support* app first before you can use the examples below. See the section *Configuring External Storage (GUI)* how to do this.

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!

6.13.1 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 informations.

6.13.2 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 that a users is browsing your ownCloud installation.

Please also be aware that ownCloud might not always be able to find out what has been changed remotely (files changes 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.

6.13.3 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
            }
        }
    }
}
```

6.13.4 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

6.13.5 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 on the ftp server
- **password**: the password to login on 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 folder 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 build 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 on the SSH server
- **root**: the folder 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 build 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 on the samba server
- **password**: the password to login on the samba server
- **share**: the share on the samba server to mount
- **root**: the folder 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 on the webdav server
- **password**: the password to login on the webdav server
- **secure**: whether to use https:// to connect to the webdav server instead of http:// (optional, defaults to false)
- **root**: the folder 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 on the swift server
- **token**: the authentication token to login on 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
}
```

6.13.6 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**

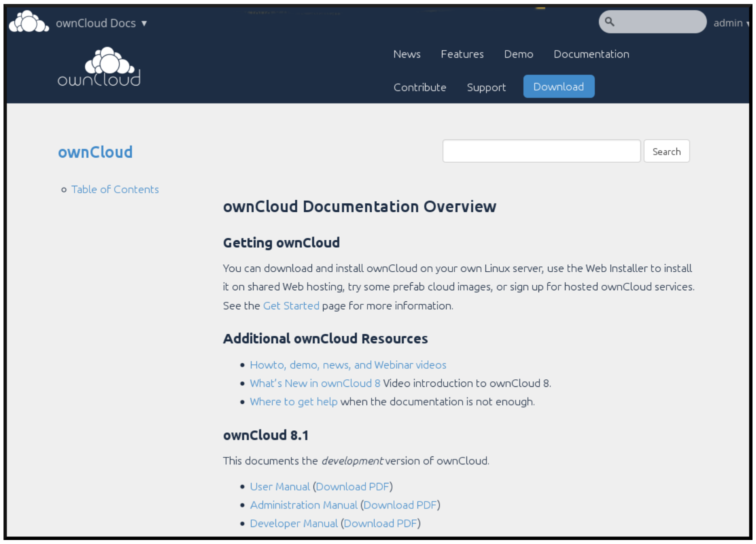
When you enter credentials into the `files_external` dialog those are stored on the filesystem and encrypted with a key stored in `config.php`. This is required since ownCloud needs access to those files and shares even when the user is not logged-in to have sharing and other key features properly working.

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.

6.14 Linking External Sites

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



ownCloud

ownCloud Documentation Overview

Getting ownCloud

You can download and install ownCloud on your own Linux server, use the Web Installer to install it on shared Web hosting, try some prefab cloud images, or sign up for hosted ownCloud services. See the Get Started page for more information.

Additional ownCloud Resources

- Howto, demo, news, and Webinar videos
- What's New in ownCloud 8 Video introduction to ownCloud 8.
- Where to get help when the documentation is not enough.

ownCloud 8.1

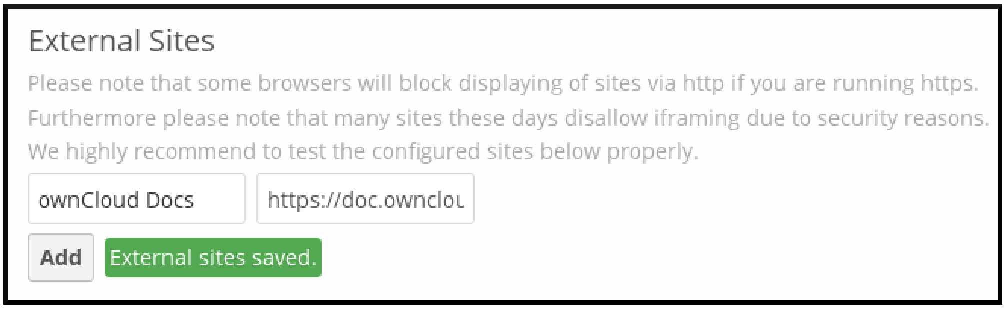
This documents the *development* version of ownCloud.

- User Manual (Download PDF)
- Administration Manual (Download PDF)
- Developer Manual (Download PDF)

Figure 6.1: *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. Hover your cursor to the right of your links to make the trashcan icon appear when you want to remove them.


External Sites

Please note that some browsers will block displaying of sites via http if you are running https. Furthermore please note that many sites these days disallow iframing due to security reasons. We highly recommend to test the configured sites below properly.

ownCloud Docs | https://doc.ownclou

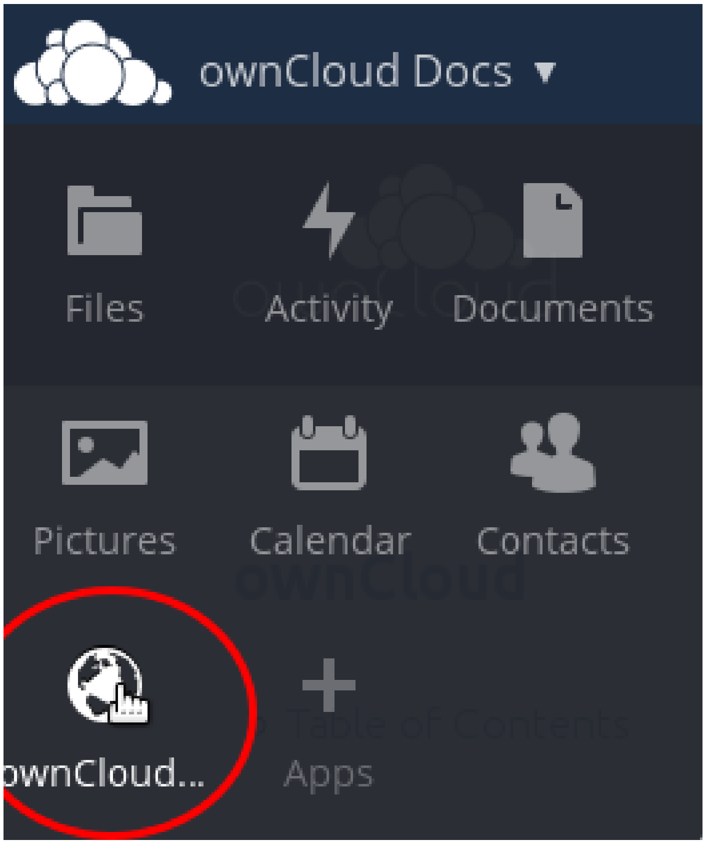
Figure 6.2: *Click to enlarge*

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.

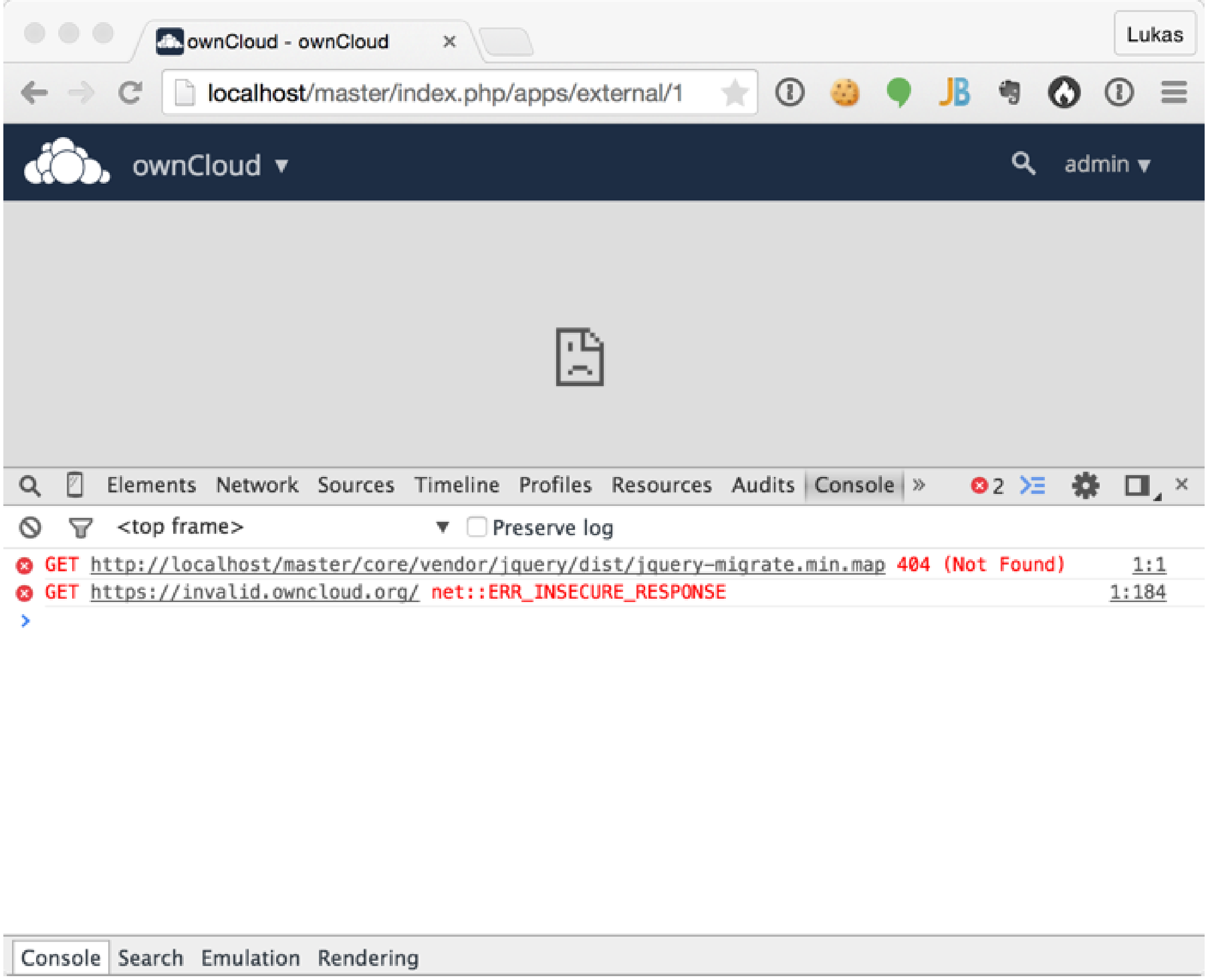
6.15 File Sharing

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:

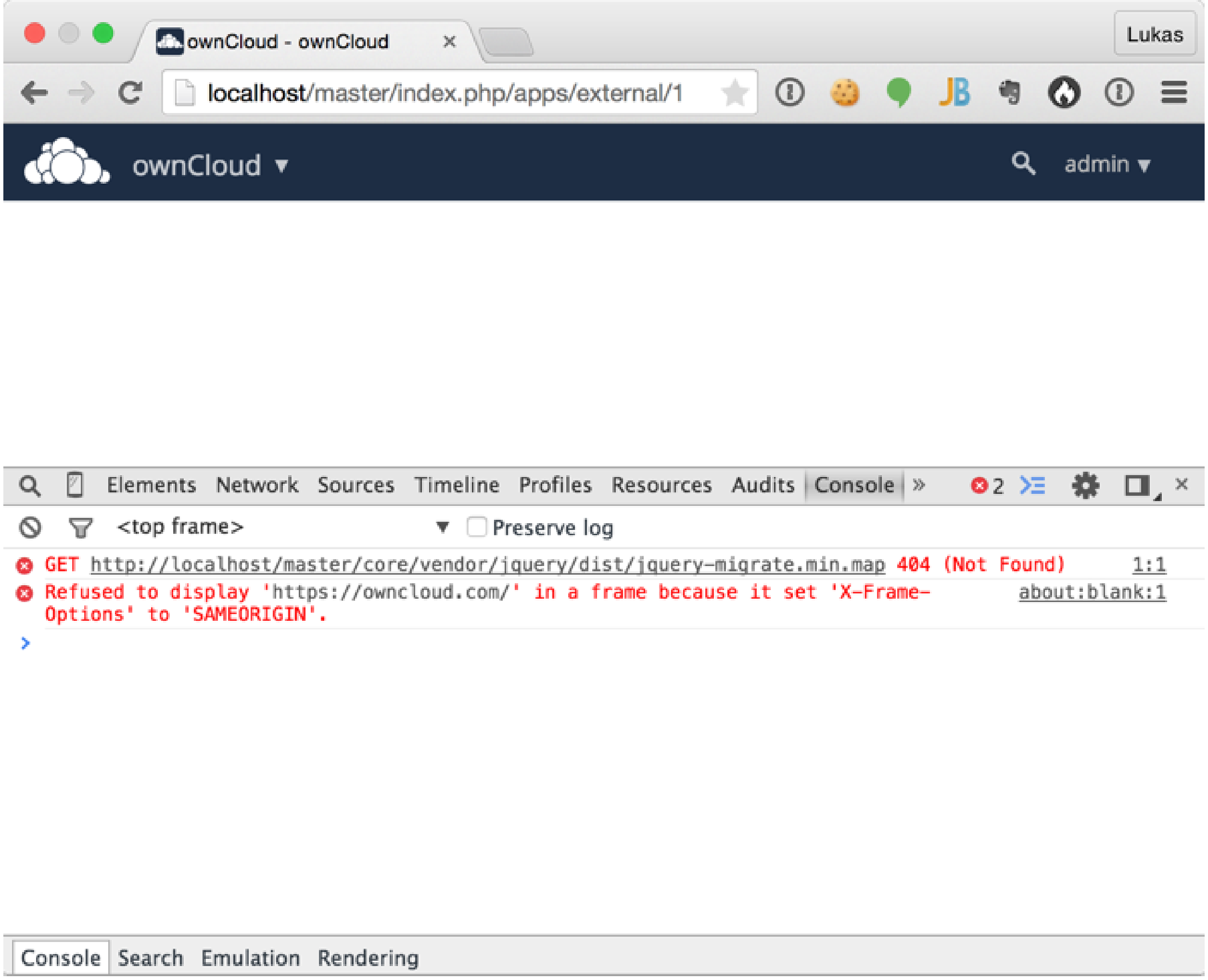
- Allowing users to share files
- Allowing users to create public shares
- Requiring a password on public shares
- Allowing public uploads to public shares
- Requiring an expiration date on public share links
- Allowing resharing
- Restricting sharing to group members only
- Allowing email notifications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

You may also allow users to create server-to-server shares (see "Configuring Server-to-Server Sharing" in the Admin manual).

Note: The Shared folder has been removed from new installations of ownCloud 7. Shares now appear in the top level of your file tree on your Files page. If you are upgrading from older ownCloud versions you will still have your


ownCloud - ownCloud
Lukas
localhost/master/index.php/apps/external/1
ownCloud ▾
admin ▾
Elements Network Sources Timeline Profiles Resources Audits Console
<top frame>
Preserve log
Console Search Emulation Rendering

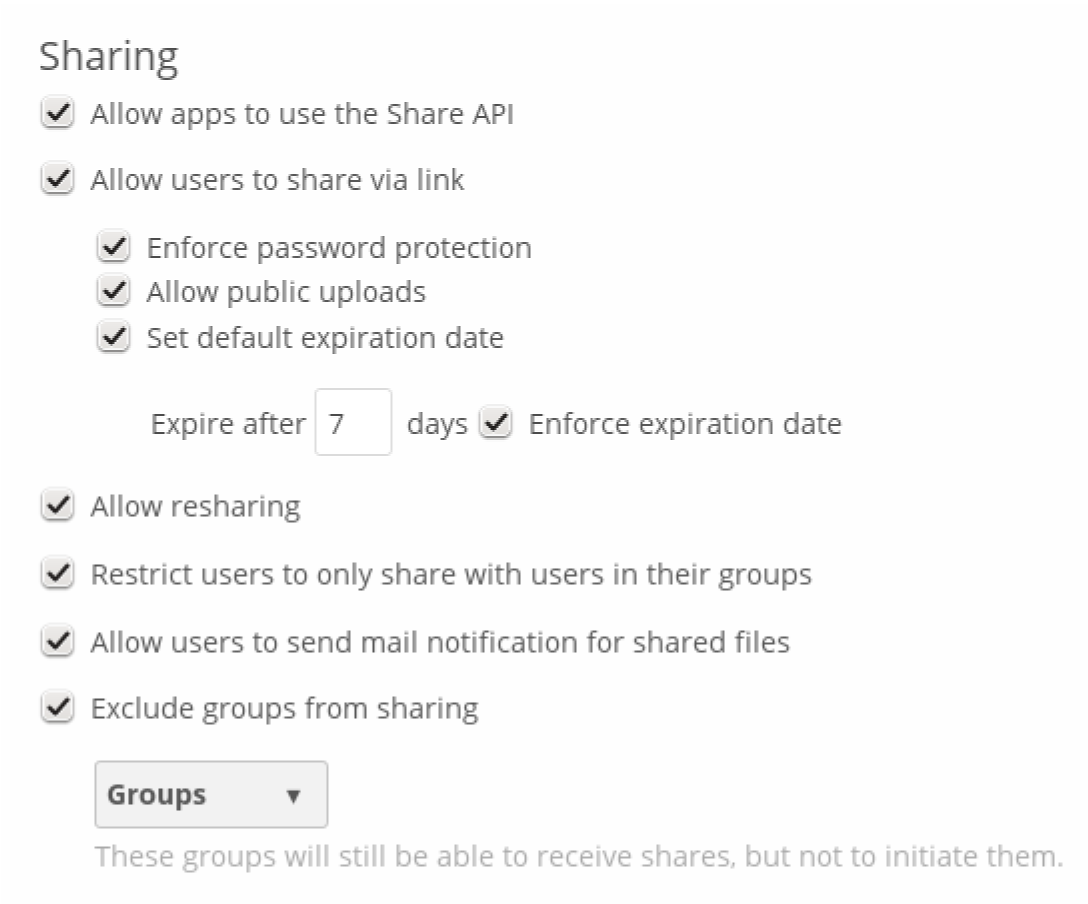
```
GET http://localhost/master/core/vendor/jquery/dist/jquery-migrate.min.map 404 (Not Found)   1:1
GET https://invalid.owncloud.org/ net::ERR_INSECURE_RESPONSE                               1:184
```


ownCloud - ownCloud
Lukas
localhost/master/index.php/apps/external/1
ownCloud ▾
admin ▾
Elements Network Sources Timeline Profiles Resources Audits Console
2
<top frame>
□ Preserve log
Console Search Emulation Rendering

```
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-          about:blank:1
Options' to 'SAMEORIGIN'.
>
```

old Shared folder, but you can change the default shared folder in `config.php` with the `'share_folder' =>` directive.

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



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

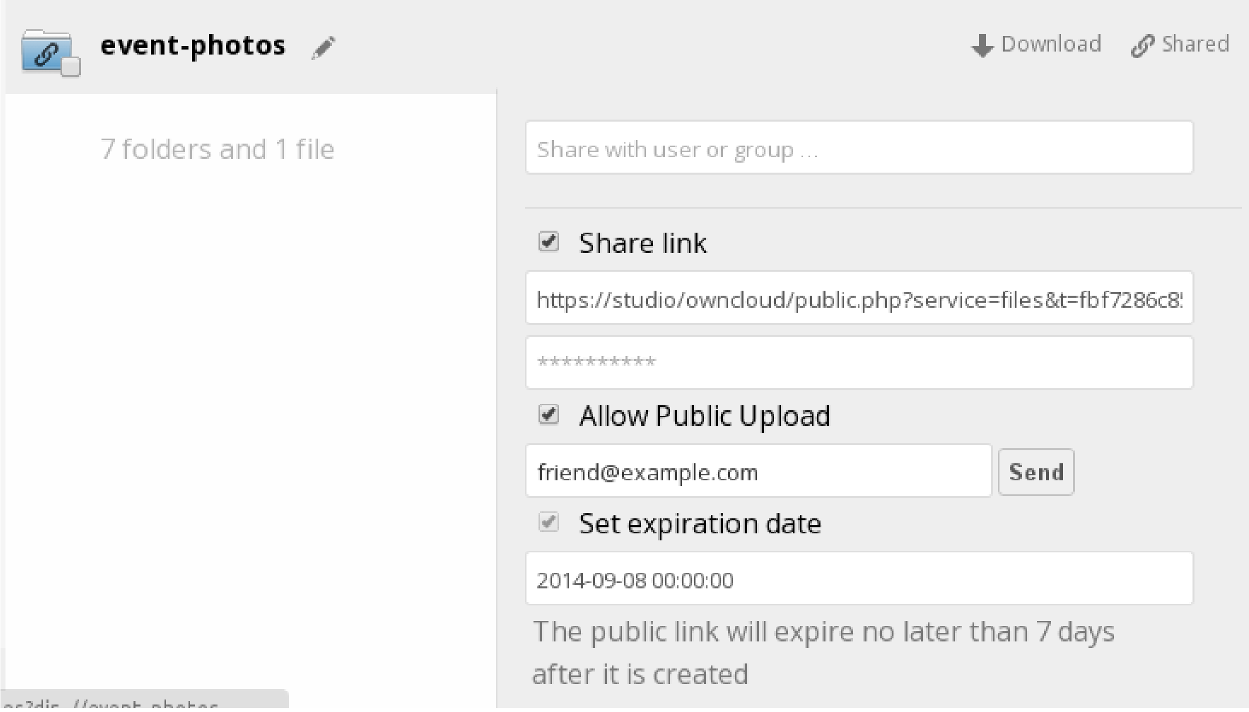
Groups ▼

These groups will still be able to receive shares, but not to initiate them.

- Check `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)

- 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:



Share with user or group ...

☑ Share link

https://studio/owncloud/public.php?service=files&t=fbf7286c8

****

☑ Allow Public Upload

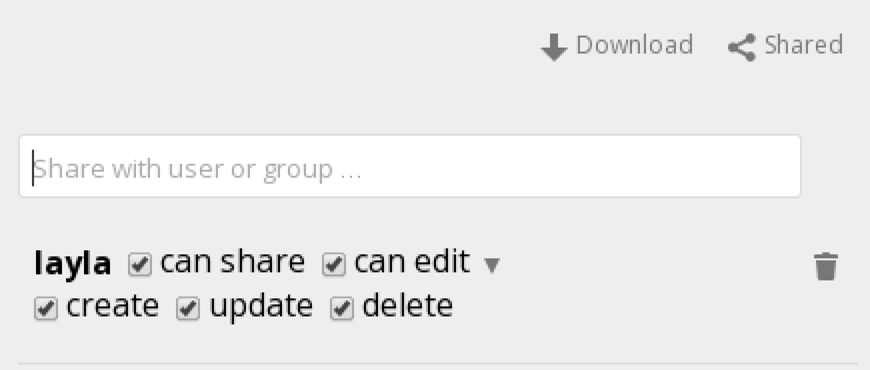
friend@example.com

☑ Set expiration date

2014-09-08 00:00:00

The public link will expire no later than 7 days after it is created

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



Share with user or group ...

layla ☑ can share ☑ can edit ▾
☑ create ☑ update ☑ delete

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.

6.15.1 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.

6.16 Files Locking App Configuration

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.

6.17 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 only be applied in scenarios were the administrator have complete control over the ownCloud instance.

This document lists some security hardenings which require manual interaction by administrators. The whole document content is based on the assumption that you run ownCloud Server on Apache2 on a Linux environment.

6.17.1 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.

6.17.2 Operating system

Give PHP read accesss 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.

Enable hardening modules such as SELinux

It is highly recommend to enable hardening modules such as SELinux where possible. See *SELinux Configuration* to learn more about SELinux.

6.17.3 Deployment

Move data directory outside of the web root

It is highly recommended to move the data directory (where ownCloud stores its data) outside of the web root (i.e. outside of `/var/www`) It is possible to do this by moving the folder manually, and then adjusting the `'datadirectory'` parameter in `config.php`.

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 vulnerable 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.

6.17.4 Use HTTPS

Using ownCloud without using an encrypted HTTPS connection might allow attackers in a man-in-the-middle (MITM) situation to intercept your users data and passwords. Thus ownCloud always recommends to setup ownCloud behind HTTPS.

How to setup HTTPS on your web server depends on your setup, we recommend to check your distribution's vendor information on how to configure and setup HTTPS.

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 already a good start it will often not completely prevent man-in-the-middle attacks for a regular user. Thus administrators are encouraged to set the HTTP Strict Transport Security header which will instruct browsers to not allow any connection to the ownCloud instance anymore using HTTPS and a invalid certificate warning will often not be able to get bypassed.

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

```
<VirtualHost *:443>
   ServerName cloud.owncloud.com
   Header always add Strict-Transport-Security "max-age=15768000"
</VirtualHost>
```

It shall be noted that this requires that the `mod_headers` extension to be installed.

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 depends completely on your environment and thus giving a generic recommendation is not really possible.

We recommend to use the Mozilla SSL Configuration Generator to generate a suitable configuration suited for your environment, furthermore the free Qualys SSL Labs Tests give a good guidance whether the SSL server was correctly configured.

6.17.5 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.

6.17.6 Serve security related Headers by the web server

Basic security headers are served by ownCloud already in a default environment. These includes:

- **`X-Content-Type-Options:  nosniff`**
    - Instructs some browsers to not sniff the mimetype of files. This is used for example to prevent browsers to interpret text files as JavaScript.
- **`X-XSS-Protection:  1; mode=block`**
    - Enforces the browsers to enable their browser side Cross-Site-Scripting filter.
- **`X-Robots-Tag:  none`**
    - Instructs search machines to not index these page.
- **`X-Frame-Options:  SAMEORIGIN`**
    - Prevents to embed the ownCloud instance within an iframe from other domains to prevent Clickjacking and other similiar attacks.

However, these headers are added by the applications code in PHP and thus not served on static resources and rely on the fact that there is no way to bypass the intended response code path.

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 as well as the following Apache modules needs 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 above mentioned security headers are shipped.

6.18 JavaScript and CSS Asset Management

In production environments, JavaScript and CSS files are delivered in a concatenated and compressed format.

ownCloud creates individual JavaScript and CSS files and saves them in a folder called 'assets' in the web root. This folder must be owned by the web server user and is used for static delivery of these files.

Note: Test this thoroughly on production systems as it should work reliably with core apps, but you may encounter problems with community/third-party apps.

6.18.1 Parameters

```
<?php

  'asset-pipeline.enabled' => true,
```

You can set this parameters in the `config/config.php`

6.19 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.

6.19.1 Parameters

If you want to disable the ownCloud help menu item you can use the **knowledgebaseenabled** parameter inside the `config/config.php`. The **knowledgebaseurl** parameter is used to set the http path to the ownCloud help page. The server should support OCS.

```
<?php

  "knowledgebaseenabled" => true,
  "knowledgebaseurl"     => "http://api.apps.owncloud.com/v1",
```

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

6.20 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.

6.20.1 Parameters

```
<?php

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

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

6.21 Logging Configuration

To get an idea of how the current status of an ownCloud system is or to solve issues log information is a good point to start with. ownCloud allows to configure the way how and which depth of information should be logged.

6.21.1 Parameters

First you need to decide in which way logging should be done. You can choose between the two options **owncloud** and **syslog**. Then you need to configure the log level which directly influences how much information will be logged. You can choose between:

- **0**: DEBUG
- **1**: INFO
- **2**: WARN
- **3**: ERROR
- **4**: FATAL

The most detailed information will be written if **0** (DEBUG) is set, the least information will be written if **3** (ERROR) is set. Keep in mind that it might slow down the whole system if a too detailed logging will has been configured. By default the log level is set to **2** (WARN).

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

ownCloud

All log information will be written to a separate log file which can be viewed using the log menu in the admin menu of ownCloud. By default a log file named **owncloud.log** will be created in the directory which has been configured by the **datadirectory** parameter.

The desired date format can optionally be defined using the **logdateformat**. 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 the date/time format will be written in the format "*January 10, 2013 15:20:25*".

```
<?php

  "log_type" => "owncloud",
  "logfile" => "owncloud.log",
  "loglevel" => "3",
  "logdateformat" => "F d, Y H:i:s",
```

syslog

All log information will be send to the default syslog daemon of a system.

```
<?php

  "log_type" => "syslog",
  "logfile" => "",
  "loglevel" => "3",
```

6.22 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
* Upgrade 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.

Note: 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

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

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

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 7.0.4
```

Query your ownCloud server status:

```
$ sudo -u www-data php occ status
  Array
  (
   [installed] => true
   [version] => 7.0.4.2
   [versionstring] => 7.0.4
   [edition] =>
  )
```

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

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

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

```
$ sudo -u www-data php occ help maintenance:mode
  Usage:
  maintenance:mode [--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.
```

6.22.1 Maintenance Commands

These three 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 config
```

6.22.2 User Commands

The `user` commands 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*). In this example the username is layla:

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

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  |
+------------------+----+
```

6.22.3 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
```

6.22.4 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. 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 a temporary directory and then performing the upgrade on them, to ensure that the upgrade will complete correctly. 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
```

6.22.5 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 installed and its PHP connector
- 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 MySQL/MariaDB:

```
$ sudo -u www-data php occ db:generate-change-script
$ 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 From SQLite to MySQL, MariaDB, or PostgreSQL*

6.22.6 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 includes “rob”:

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

You can see your whole LDAP configuration, or the configuration for a single configID:

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

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

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

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

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

6.22.7 File Scanning

The `files:scan` command scans for new files for the file cache, and isn’t intended to be run manually.

6.23 Performance Tips

The performance of ownCloud, like any LAMP application, is dependent on all components of the stack. Maximizing performance can be achieved by optimizing the operations and interactions of the underlying network, hardware, operating systems, webservers, databases, and storage.

This guide cannot cover all possible configurations and will instead cover tips that are specific to ownCloud or give the greatest benefit.

6.23.1 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.

- On Windows you can run `coreinfo` from Sysinternals
  Windows Sysinternals Download Coreinfo
  which gives you details of the processor and extensions present.
  Note: you may have to run the command shell as administrator to get an output.

- 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.

6.23.2 OPcache Extension

OPcache improves PHP performance by storing precompiled script bytecode in shared memory, thereby removing the need for PHP to load and parse scripts on each request. This extension is bundled with PHP 5.5.0 and later, and is available in PECL for PHP versions 5.2, 5.3, and 5.4.

6.23.3 Object Caching

ownCloud is written to take advantage of object caching. Object caching can be done locally with the APCu extension, or for distributed PHP environments using Memcached. Memcached servers must be specified in the "memcached_servers" array in ownCloud's config file.

6.23.4 Enable the SPDY protocol

Your webserver can be configured to use the SPDY protocol which could improve the overall performance of ownCloud. Please have a look at the documentation of your webservers module for more infos:

- mod-spdy for Apache

- ngx_http_spdy_module for NginX

Note: If you want to enable SPDY for Apache please note the Known Issues of this module to avoid problems after enabling it.

6.23.5 Serving static files via web server

See the section *Serving Static Files for Better Performance* for a description and the benefits.

6.23.6 Using cron to perform background jobs

See the section *Defining Background Jobs* for a description and the benefits.

6.23.7 Using MySQL instead of SQLite

MySQL or MariaDB should be preferred because of the performance limitations of SQLite with highly concurrent applications, like ownCloud.

On large instances you could consider running MySQLTuner to optimize the database.

See the section *Database Configuration* 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 From SQLite to MySQL, MariaDB, or PostgreSQL*.

6.23.8 Improve slow performance with MySQL on Windows

On Windows hosts running MySQL on the same system changing the parameter `dbhost` in your `config/config.php` from `localhost` to `127.0.0.1` could improve the page loading time.

See also this forum thread.

6.23.9 Nginx: caching ownCloud gallery thumbnails with fastcgi_cache_purge

One of the optimisations for ownCloud when using Nginx as webserver is to combine FastCGI caching with “Cache Purge”, a 3rdparty Nginx module that adds the ability to purge content from *FastCGI*, *proxy*, *SCGI* and *uWSGI* caches. This mechanism speeds up thumbnail presentation as it shifts requests to Nginx and minimizes php invocations which else would take place for every thumbnail presented every time.

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

Note I:

Unlike Apache, Nginx does not dynamically load modules. All modules needed, must be compiled into Nginx. This is one of the reasons for Nginx´s performance.

Note II:

It is expected to have an already running Nginx installation with a working configuration set up like described in the ownCloud documentation.

Nginx module check

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

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

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

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

1. *Preparation*

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

   Add following lines (in case, replace `{trusty}` by your distribution name):

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

   Then do a

   ```
   sudo apt-get update
   ```

   Note:

   If you’re not overly cautious and wish to install the latest and greatest Nginx packages and features, you may have to install Nginx from its mainline repository.

   From the Nginx homepage: “In general, you should deploy Nginx from its mainline branch at all times.”

   If you would like to use standard Nginx from the latest mainline branch but without compiling in any additional modules, just run `sudo apt-get install nginx`.

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

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

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

   ```
   ls -la
   ```

   Please replace `{release}` with the release downloaded

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

   If folder “modules” is not present, do:

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

   If not present, add the following line at the top under `#export DH_VERBOSE=1`:

   ```
   MODULESDIR = $(CURDIR)/debian/modules
   ```

   And the end of *every* `./configure` command add:

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

   Don’t forget to escape preceeding lines with a backslash `\`.
   The parameters may now look :

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

4. *Compile and install Nginx*

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

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

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

   It should show now: `ngx_cache_purge`

   Show Nginx version including all features compiled and installed:

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

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

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

For eyery nginx component listed do a:

```
sudo apt-mark hold <component>
```

7. *Regular checks for nginx updates*
   Do a regular visit on the Nginx news page and proceed in case of updates with item 2 to 5

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

1. *Preparation*
   Create a directory where Nginx will save the cached thumbnails. Use any path that fits to your environment. Replace `{path}` with the path used, example path below:

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

2. *Configuration*

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

   Note: the `keys_zone` / `fastcgi_cache` name and the `{path}` must be unique to each instance of ownCloud serverd with Nginx !

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

   ```
   fastcgi_cache_path {path} levels=1:2 keys_zone=OWNCLOUD:100m inactive=60m;
   ```

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

   ```
   set $skip_cache 1;

   # POST requests and urls with a query string should always go to PHP
   if ($request_uri ~* "thumbnail.php") {
               set $skip_cache 0;
           }

       fastcgi_cache_key "$scheme$request_method$host$request_uri";
       fastcgi_cache_use_stale error timeout invalid_header http_500;
       fastcgi_ignore_headers Cache-Control Expires Set-Cookie;

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

               include fastcgi_params;
               fastcgi_param SCRIPT_FILENAME
               $document_root$fastcgi_script_name;
               fastcgi_param PATH_INFO $fastcgi_path_info;
               fastcgi_param HTTPS on;
               fastcgi_pass php-handler;
   ```

```
                fastcgi_cache_bypass $skip_cache;
                fastcgi_no_cache $skip_cache;
                fastcgi_cache OWNCLOUD;
                fastcgi_cache_valid 60m;
            }
```

Note regarding the `fastcgi_pass` parameter:
Use whatever fits your configuration. In the example above, a `upstream` was defined in an Nginx global configuration file.
This then can look like:

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

3. *Test the configuration*

   ```
   sudo service nginx restart
   ```

   - Open your browser and clear your cache.
   - Logon to your ownCloud instance, open the gallery app, move thru your folders
     and watch while the thumbs are generated for the first time.
   - You may also watch with eg. `htop` your system load while the thumbnails are processed.
   - Goto another app or logout and relogon.
   - Open the gallery app again and browse to the folders you accessed before.
     Your thumbnails should appear more or less immediately.
   - `htop` will not show up additional load while processing, compared to the high load before.

6.24 Previews Configuration

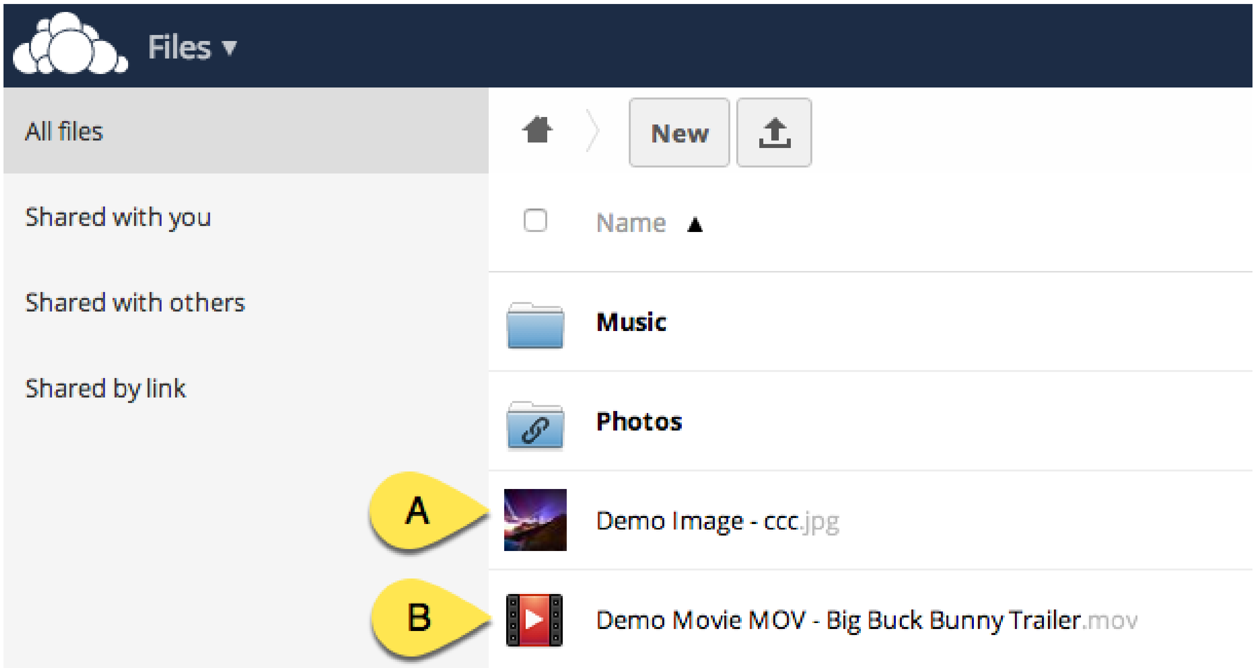
The ownCloud thumbnail system is used to generate thumbnails from various file types, which are then shown as thumbnails in the Files application of the web interface, or as a larger preview for public shared links.

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 (depending on the filetype) 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.



6.24.1 Parameters

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

Disabling previews:

Under certain circumstances, for example if the server has only very limited resources, you might want to consider disabling the generation of previews. 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,
```

6.25 Reverse Proxy Configuration

The automatic hostname, protocol or webroot detection of ownCloud can fail in certain reverse proxy situations. This configuration allows to manually override the automatic detection.

6.25.1 Parameters

If ownCloud fails to automatically detected 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.

6.25.2 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`.

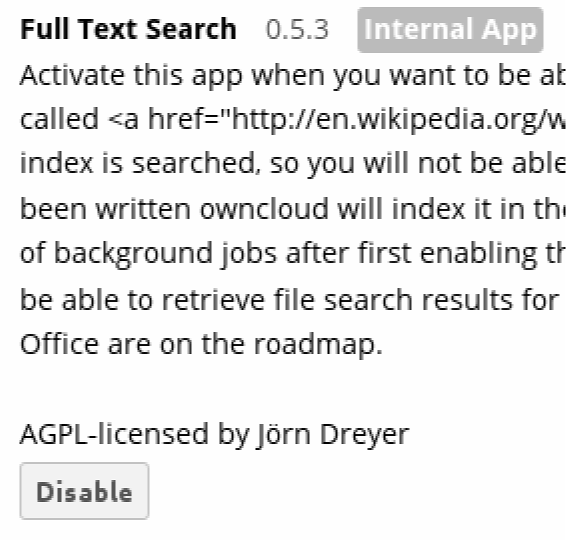
```
<?php
$CONFIG = array (
  "overwritehost"     => "ssl-proxy.tld",
  "overwriteprotocol" => "https",
  "overwritewebroot"  => "/domain.tld/owncloud",
  "overwritecondaddr" => "^10\.0\.0\.1$",
);
```

Note: If you want to use the SSL proxy during installation you have to create the `config/config.php` otherwise you have to extend to existing **$CONFIG** array.

6.26 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.

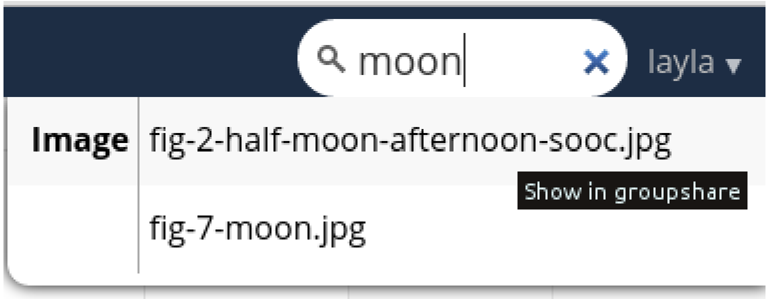


Full Text Search 0.5.3 Internal App

Activate this app when you want to be ab
called <a href="http://en.wikipedia.org/w
index is searched, so you will not be able
been written owncloud will index it in th
of background jobs after first enabling th
be able to retrieve file search results for
Office are on the roadmap.

AGPL-licensed by Jörn Dreyer

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.



moon

layla

Image	fig-2-half-moon-afternoon-sooc.jpg
	fig-7-moon.jpg

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.

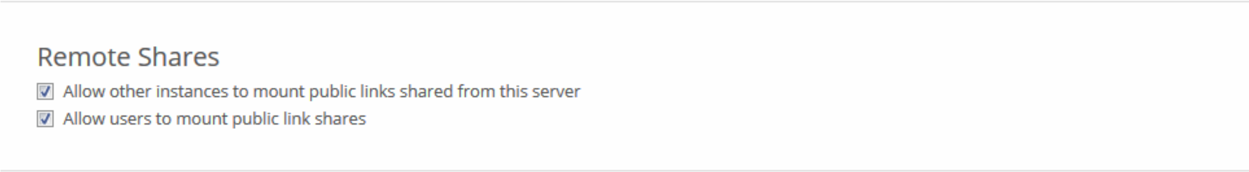
6.27 Configuring Server-to-Server Sharing

ownCloud 7 introduces a powerful new feature, server-to-server sharing. With just a few clicks you can easily and securely create public shares for sharing files and directories with other ownCloud 7 (and newer versions) servers. You can automatically send an email notification when you create the share, add password protection, allow users to upload files, and set an expiration date.

Note: This is called Federated Cloud Sharing in ownCloud 8. You may create shares with oC8 servers, following the steps below using public link shares. oC8 also supports creating the share using the `Share with user or group` form field. This is not supported in oC7 and you must use public link shares on both servers.

Follow these steps to create a new public share:

1. Go to the Admin page and scroll to the Remote Shares section.



Remote Shares

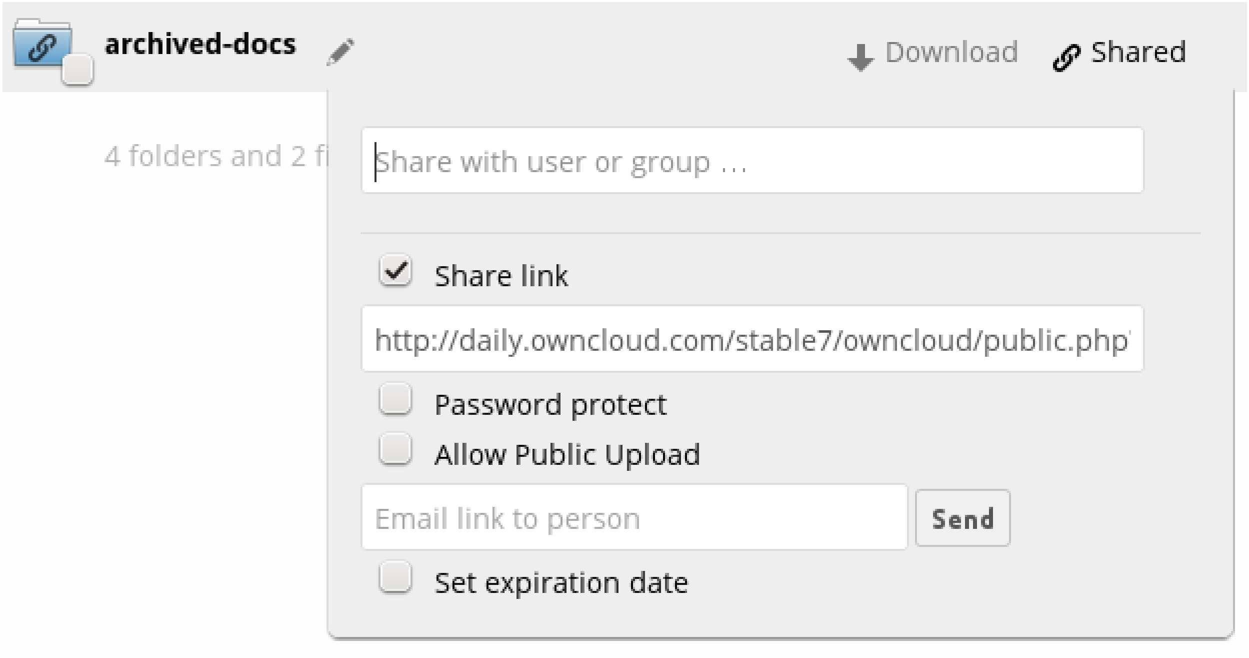
☑ Allow other instances to mount public links shared from this server

☑ Allow users to mount public link shares

2. To enable server-to-server sharing, and to allow remote users to mount your shares in their ownCloud 7 accounts, check `Allow other instances to mount public links shared from this server.` Leaving the checkbox blank disables server-to-server sharing.

3. You can enable the users on your local ownCloud server to mount public link shares by checking `Allow users to mount public link shares.` When this is not checked your users cannot mount public link shares, though they can view and download them.

4. Now go to your Files page and hover your cursor over the file or directory you want to share to expose the administration options. Check the `Share Link` checkbox to create the share, and to expose all of your sharing options.



archived-docs

4 folders and 2 fi

Share with user or group ...

☑ Share link

http://daily.owncloud.com/stable7/owncloud/public.php

☐ Password protect

☐ Allow Public Upload

Email link to person

☐ Set expiration date

Your new public share is labeled with a chain link. If you do not protect it with a password, it is visible to anyone who has the URL. Users on other ownCloud 7 servers can mount it and use it just like any ownCloud share.

Un-check the `Share Link` checkbox to disable the share.

See "Using Server-to-Server Sharing" in the Users Manual to learn how to connect to a remote public share.

6.27.1 Notes

Your Apache Web server must have `mod_rewrite` enabled, and you must have `trusted_domains` configured in `config.php`. Consider also enabling SSL to encrypt all traffic between your servers. (See *Manual Installation on Linux* to learn more about mod_rewrite, SSL, and alternative HTTP servers. See *Installation Wizard* to learn more about configuring trusted domains.)

Self-signed certificates for Server-to-Server Sharing are currently not supported.

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/public.php?service=files&t=6b6fa9a714a32ef0af8a83dde358deec`, 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`.

6.28 Serving Static Files for Better Performance

Since ownCloud 5 it is possible to let web servers handle static file serving. This should generally improve performance (web servers are optimized for this) and in some cases permits controlled file serving (i.e. pause and resume downloads).

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.

6.28.1 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 web server on path outside the specified Directory. This is needed for configured local mounts which may reside outside 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 mod_xsendfile website linked above

6.28.2 LigHTTPd (X-Sendfile2)

LigHTTPd uses similar headers to Apache2, apart from the fact that it does not handle partial downloads in the same way Apache2 does. For this reason, a different method is used for LigHTTPd.

Installation

X-Sendfile and X-Sendfile2 are supported by default in LigHTTPd and no additional operation should be needed to install it.

Configuration

Your server configuration should include the following statements:

```
fastcgi.server          = ( ".php" => ((
   ...
   "allow-x-send-file" => "enable",
   "bin-environment" => (
      "MOD_X_SENDFILE2_ENABLED" => "1",
   ),
)))
```

- **allow-x-send-file**: enables LigHTTPd to use X-Sendfile and X-Sendfile2 headers to serve files
- **bin-environment**: is used to parse MOD_X_SENDFILE2_ENABLED to the ownCloud backend, to make it use the X-Sendfile and X-Sendfile2 headers in it’s response

6.28.3 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

Configuration is similar to Apache:

```
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 mulitple 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

6.28.4 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.

6.29 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.

6.29.1 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",
```

6.30 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:

6.30.1 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}'
                              ),
    ),
),
```

6.30.2 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'
                              ),
    ),
),
```

6.30.3 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'
                              ),
    ),
),
```

6.31 User Authentication with LDAP

ownCloud ships with an LDAP application so that your existing LDAP users may have access to your ownCloud server without creating separate ownCloud user accounts.

Note: For performance reasons, we recommend using PHP 5.4 or greater to use the LDAP application with more than 500 users. The PHP LDAP module is required; this is supplied by `php5-ldap` on Debian/Ubuntu, and `php-ldap` on CentOS/Red Hat/Fedora.

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
- Read-only access to your LDAP (no edit or delete of users on your LDAP)

Note: 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*.

6.31.1 Configuration

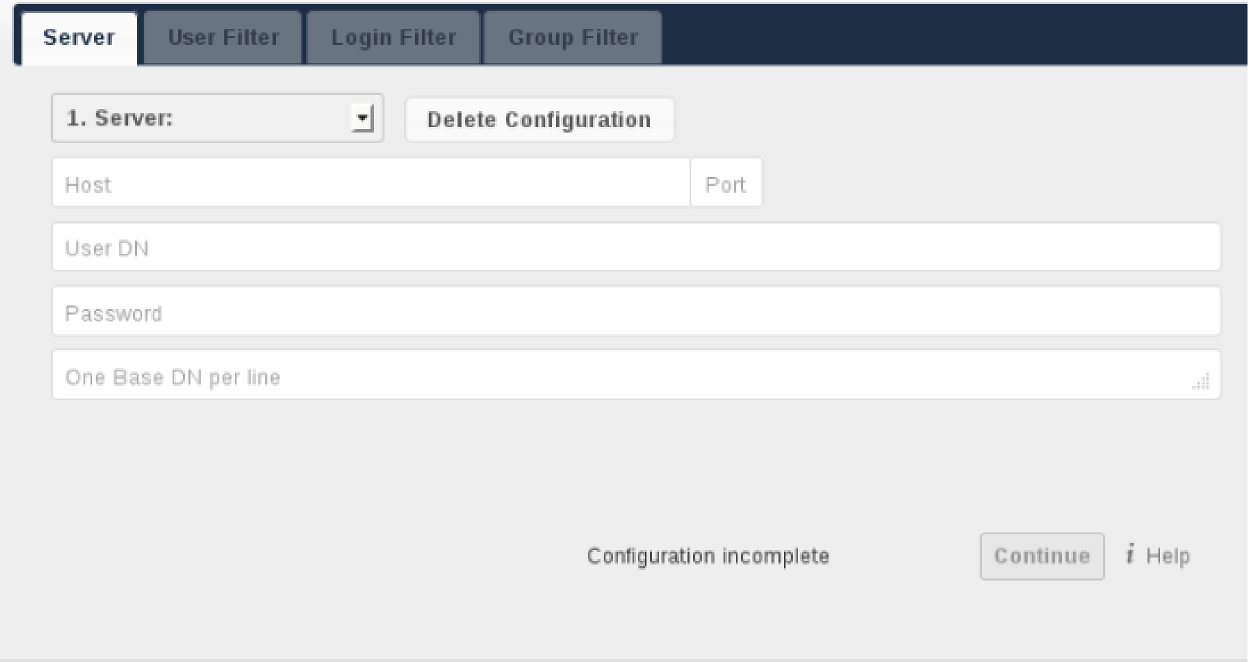
First enable the `LDAP user and group backend` app on the Apps page in ownCloud. Then go to your Admin page to configure it.

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.

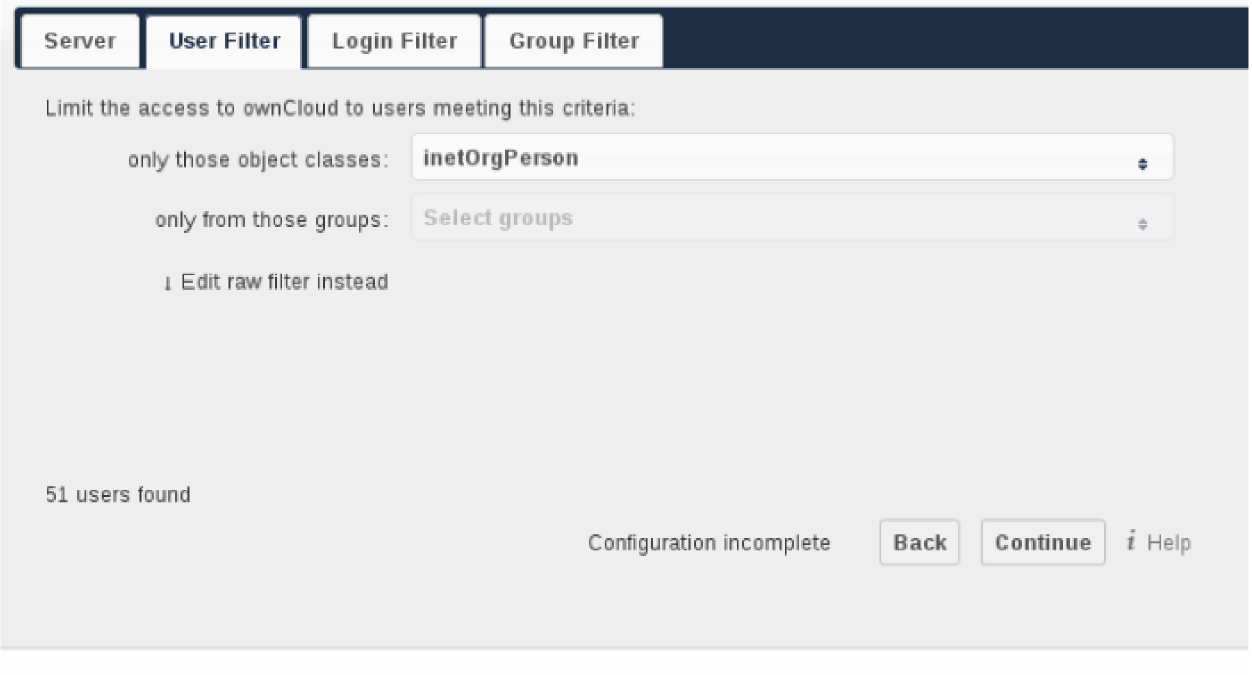
Base DN: The base DN of LDAP, from where all users and groups can be reached. You may enter multiple base DNs, one per line. (Base DNs for users and groups can be set in the Advanced tab.) This field is mandatory. ownCloud attempts to determine the Base DN according to the provided User DN or the provided 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.



Limit the access to ownCloud to users meeting this criteria:

only those object classes: inetOrgPerson

only from those groups: Select groups

↓ Edit raw filter instead

51 users found

Configuration incomplete

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:

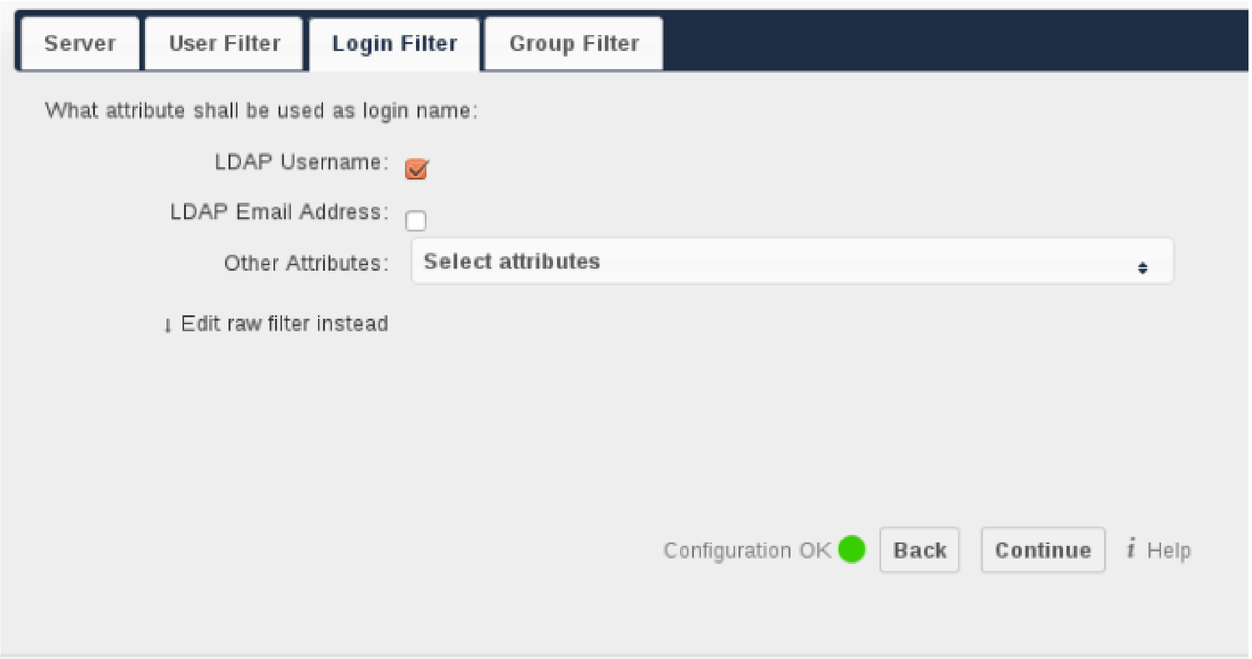
- *(&(objectClass=inetOrgPerson)(memberOf=cn=owncloudusers,ou=groups,dc=example,dc=com))*

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.



What attribute shall be used as login name:

LDAP Username: ☑

LDAP Email Address: □

Other Attributes: Select attributes

↓ Edit raw filter instead

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

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.

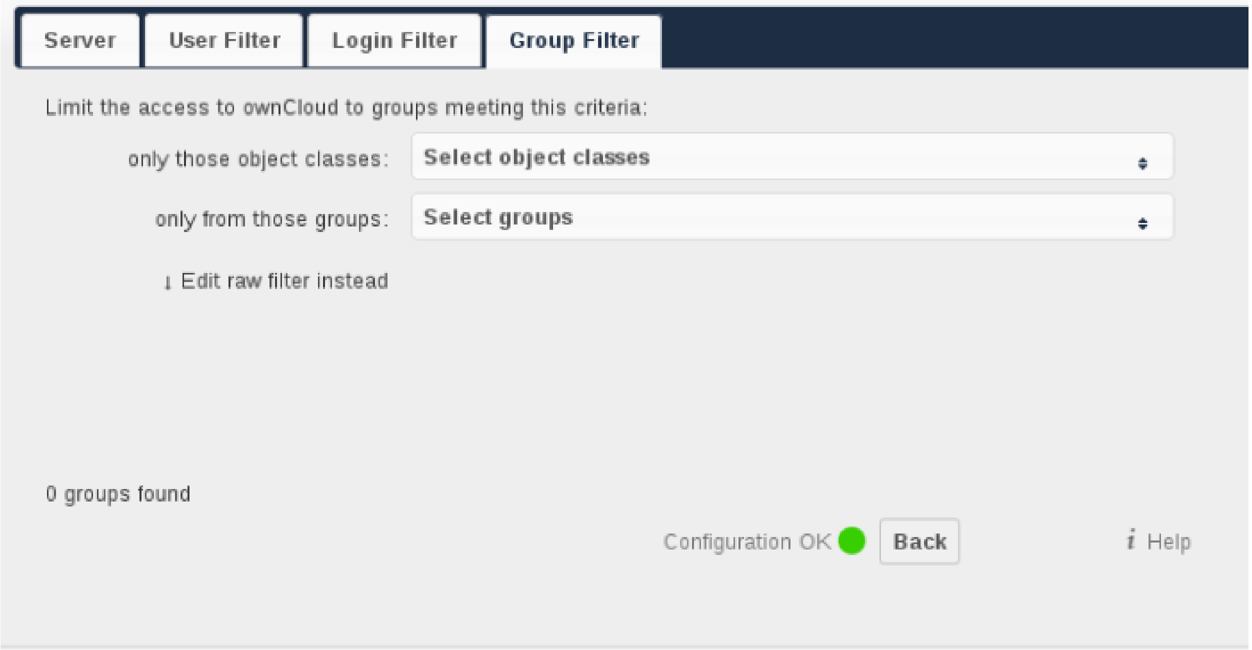
only those object classes: ownCloud will determine the object classes that are typically available for group objects in your LDAP server. ownCloud will only list object classes that return at least one group object. You can select multiple object classes. A typical object class is “group”, or “posixGroup”.

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*



Limit the access to ownCloud to groups meeting this criteria:

only those object classes: Select object classes

only from those groups: Select groups

0 groups found

- *objectClass=posixGroup*

y groups found: This tells you approximately how many groups will be available in ownCloud. The number updates automatically after any change.

6.31.2 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

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.

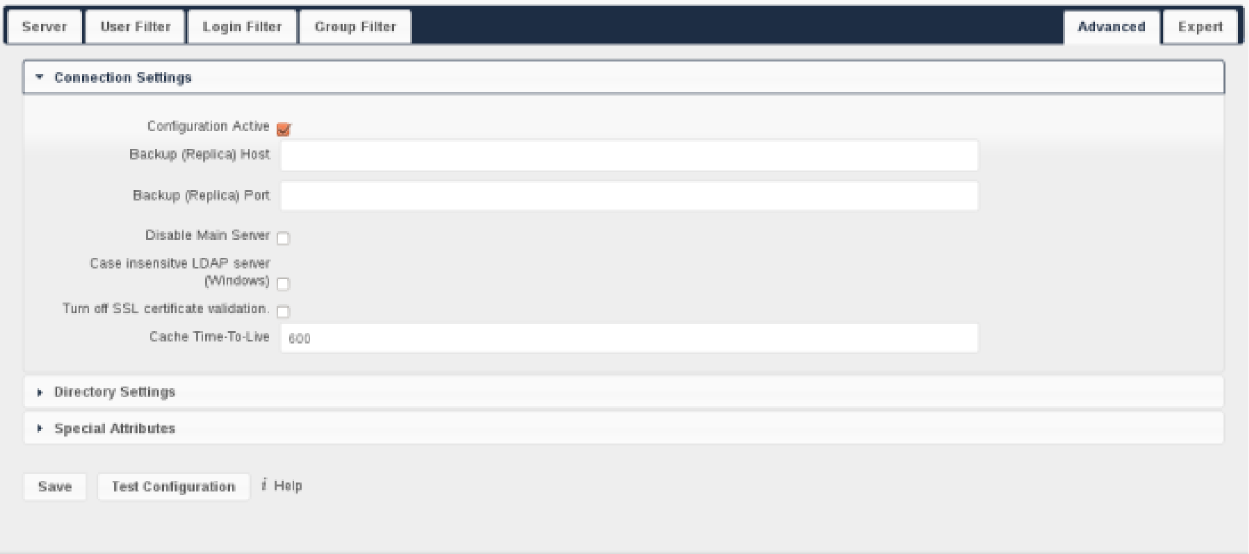
Backup (Replica) Host: If you have a backup LDAP server, enter the connection settings here. ownCloud will then automatically connect to the backup when the main server cannot be reached. The backup server must be a replica of the main server so that the object UUIDs match.

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:



▾ Connection Settings

Configuration Active ☑

Backup (Replica) Host

Backup (Replica) Port

Disable Main Server □

Case insensitive LDAP server (Windows) □

Turn off SSL certificate validation. □

Cache Time-To-Live 600

▸ Directory Settings

▸ Special Attributes

Figure 6.3: LDAP Advanced Settings, section Connection Settings

- *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

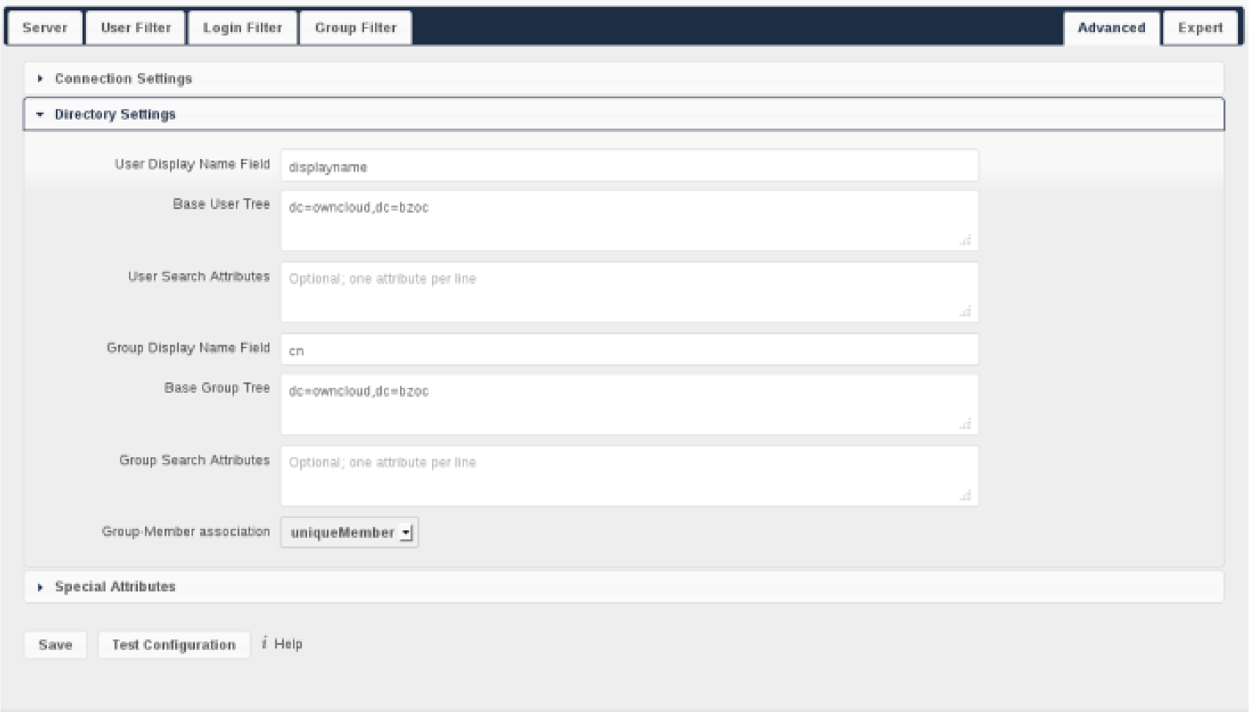
User Display Name Field: The attribute that should be used as display name in ownCloud.

- 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



User Display Name Field: displayname

Base User Tree: dc=owncloud,dc=bzoc

User Search Attributes: Optional; one attribute per line

Group Display Name Field: cn

Base Group Tree: dc=owncloud,dc=bzoc

Group Search Attributes: Optional; one attribute per line

Group-Member association: uniqueMember

Figure 6.4: LDAP Advanced Settings, section Directory Settings

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

User Search Attributes: These attributes are used when searches for users are performed, for example in the 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 being 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

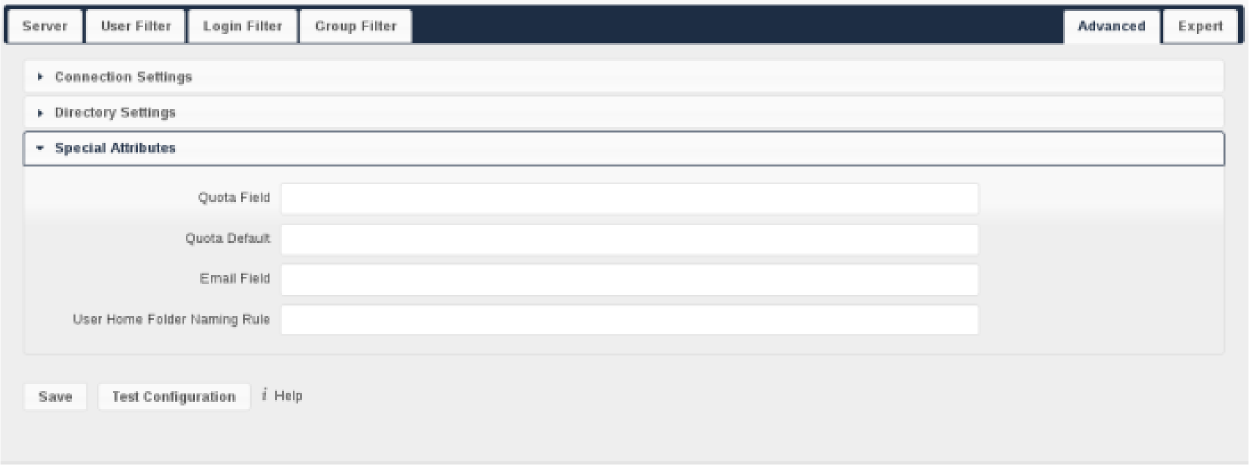


Figure 6.5: LDAP Advanced Settings, section Special Attributes

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".

- 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*

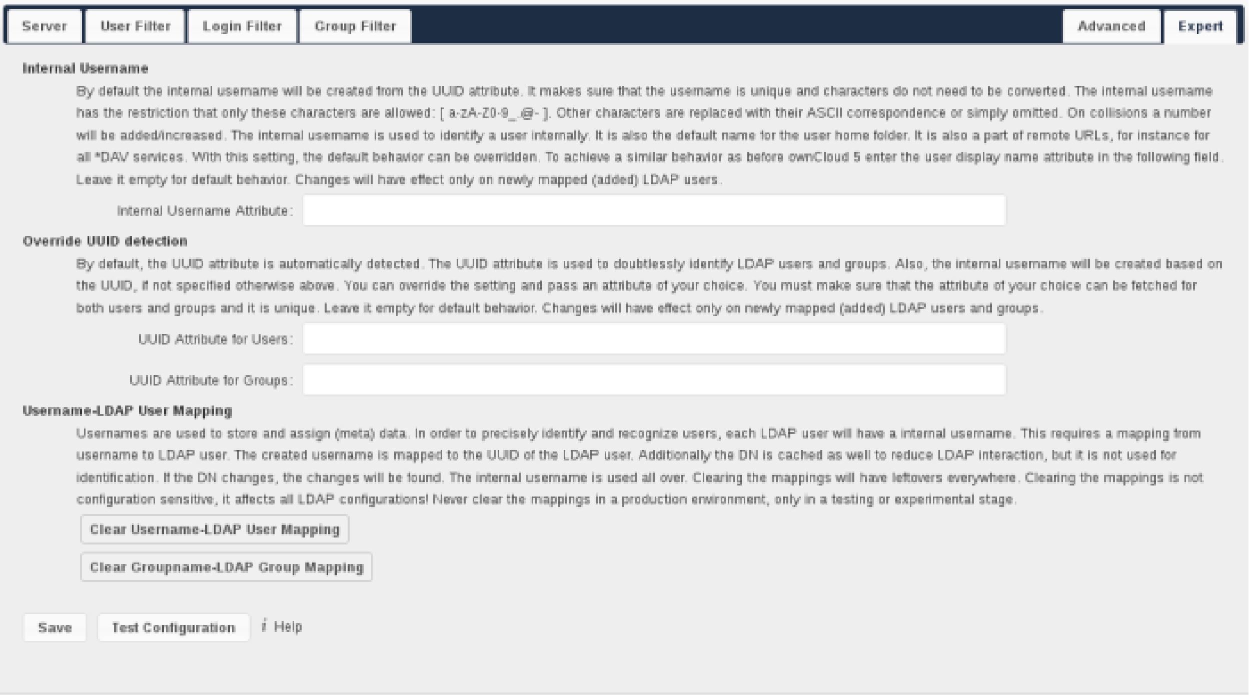
Email Field: Set the user's email from their LDAP attribute. Leave it empty for default behavior.

- Example: *mail*

User Home Folder Naming Rule: By default, the ownCloud server creates the user directory in your ownCloud data directory. You may want to override this setting and name it after an attribute value. The attribute given can also return an absolute path, e.g. `/mnt/storage43/alice`. Leave it empty for default behavior.

- Example: *cn*

6.31.3 Expert Settings



Internal Username

By default the internal username will be created from the UUID attribute. It makes sure that the username is unique and characters do not need to be converted. The internal username has the restriction that only these characters are allowed: [a-zA-Z0-9_.@-]. Other characters are replaced with their ASCII correspondence or simply omitted. On collisions a number will be added/increased. The internal username is used to identify a user internally. It is also the default name for the user home folder. It is also a part of remote URLs, for instance for all *DAV services. With this setting, the default behavior can be overridden. To achieve a similar behavior as before ownCloud 5 enter the user display name attribute in the following field. Leave it empty for default behavior. Changes will have effect only on newly mapped (added) LDAP users.

Internal Username Attribute:

Override UUID detection

By default, the UUID attribute is automatically detected. The UUID attribute is used to doubtlessly identify LDAP users and groups. Also, the internal username will be created based on the UUID, if not specified otherwise above. You can override the setting and pass an attribute of your choice. You must make sure that the attribute of your choice can be fetched for both users and groups and it is unique. Leave it empty for default behavior. Changes will have effect only on newly mapped (added) LDAP users and groups.

UUID Attribute for Users:

UUID Attribute for Groups:

Username-LDAP User Mapping

Usernames are used to store and assign (meta) data. In order to precisely identify and recognize users, each LDAP user will have a internal username. This requires a mapping from 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 changes will be found. The internal username is used all over. Clearing the mappings will have leftovers everywhere. Clearing the mappings is not configuration sensitive, it affects all LDAP configurations! Never clear the mappings in a production environment, only in a testing or experimental stage.

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.

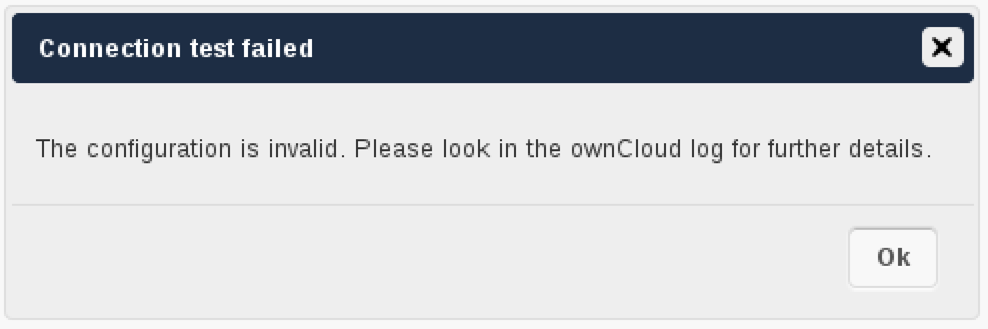
The same is valid for groups.

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!

6.31.4 Testing the configuration

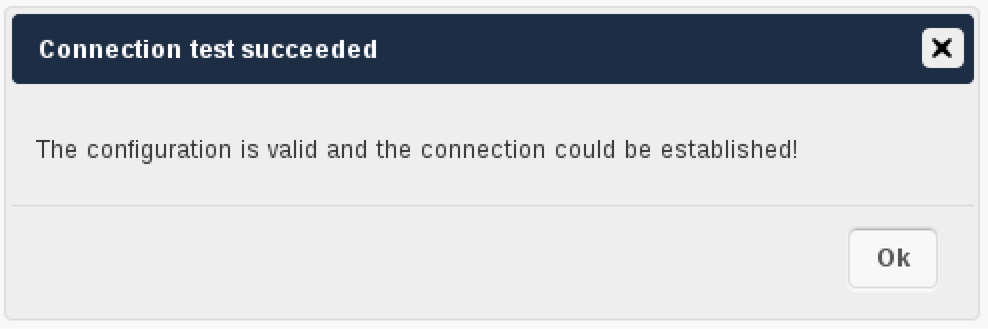
The **Test Configuration** button checks the values as currently given in the input fields. You do not need to save before testing. By clicking on the button, ownCloud will try to bind to the ownCloud server using the settings currently given in the input fields. The response will look like this:


Connection test failed

The configuration is invalid. Please look in the ownCloud log for further details.

Figure 6.6: 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.


Connection test succeeded

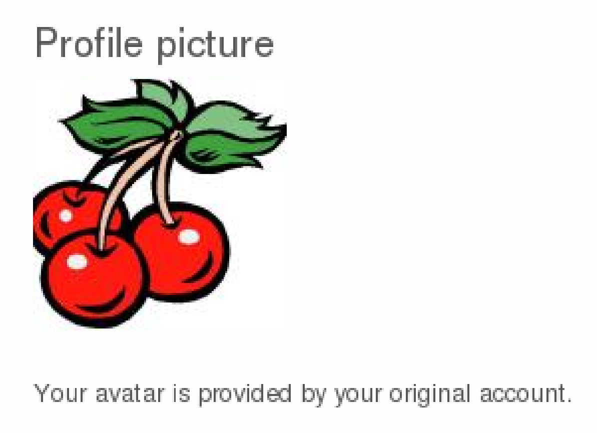
The configuration is valid and the connection could be established!

Figure 6.7: Success

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

6.31.5 ownCloud Avatar integration

ownCloud support 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*.



Profile picture

Your avatar is provided by your original account.

Figure 6.8: Profile picture fetched from LDAP

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 it 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.

6.31.6 Troubleshooting, Tips and Tricks

6.31.7 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)

6.31.8 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.

6.31.9 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.

6.31.10 ownCloud LDAP Internals

Some parts of how the LDAP backend works are described here.

6.31.11 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 something else is specified in *User Home Folder Naming Rule*), which contains files and meta data.

As of ownCloud 5 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.

6.31.12 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.

6.31.13 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 instead. When you have a scheduled downtime, check **Disable Main Server** to avoid unnecessary connection attempts.

6.32 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 prequisites 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   |
+-----------------+-----------------+------------------+--------------------------------------+
```

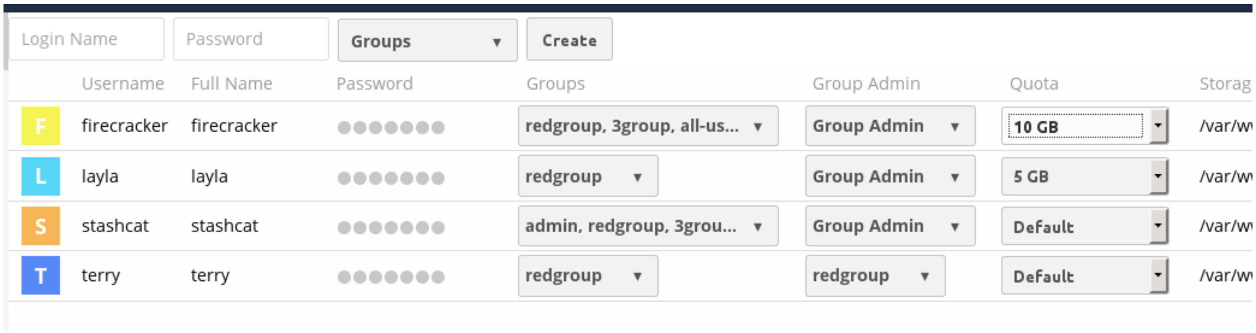
Then you can run `sudo -u www-data php occ user:delete aaliyah_brown` to delete user aaliyah_brown. You must use the user's ownCloud name.

6.32.1 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.

6.33 User Management

In ownCloud 7, the Users management page has been streamlined and improved. 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, and, if you so desire, delete them with a single click.



	Username	Full Name	Password	Groups	Group Admin	Quota	Storag
F	firecracker	firecracker	●●●●●●●	redgroup, 3group, all-us... ▾	Group Admin ▾	10 GB	/var/w
L	layla	layla	●●●●●●●	redgroup ▾	Group Admin ▾	5 GB	/var/w
S	stashcat	stashcat	●●●●●●●	admin, redgroup, 3grou... ▾	Group Admin ▾	Default	/var/w
T	terry	terry	●●●●●●●	redgroup ▾	redgroup ▾	Default	/var/w

User accounts have the following properties:

Login Name (Username) This is 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.

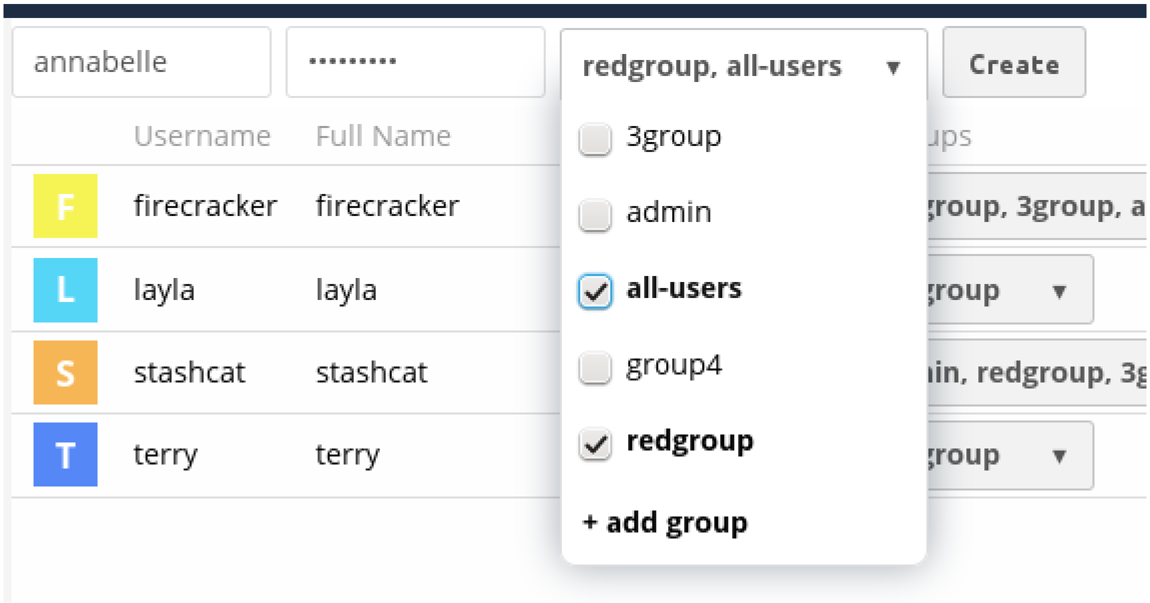
Group Admin Group admins are granted administrative privileges on specific groups, and can add and remove users from their groups.

Quota The maximum disk space assigned to each user. Any user that exceeds the quota cannot upload or sync data. ownCloud 7 introduces a new feature, and that is the option to include external storage in user quotas.

6.33.1 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



annabelle

•••••••••

redgroup, all-users ▾

	Username	Full Name	ups
F	firecracker	firecracker	roup, 3group, a
L	layla	layla	roup ▾
S	stashcat	stashcat	in, redgroup, 3g
T	terry	terry	roup ▾

Login names may contain letters (a-z, A-Z), numbers (0-9), dashes (-), underscores (_), periods (.) and ampersands (@). 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.

Remember to give your new users their logins and passwords.

6.33.2 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*.

6.33.3 Renaming a User

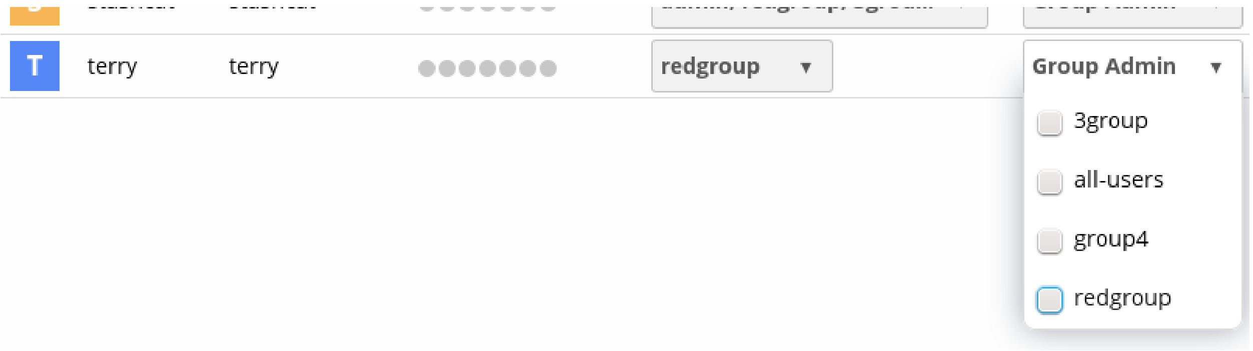
Each ownCloud user has two names: a unique **Login Name** used for authentication, and a **Full Name**, which is their display name. You can edit the display name of a user, but you cannot change the login name of any user.

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

6.33.4 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.

6.33.5 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.

6.33.6 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.

6.33.7 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 the "File Sharing" section of the Admin Manual to learn how to create persistent file shares that survive user deletions.

6.34 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.

CHAPTER

SEVEN

MAINTENANCE

7.1 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.

7.2 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

7.2.1 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"`/
```

7.2.2 Backup Database

MySQL

MySQL is the recommended database engine. To backup MySQL:

```
mysqldump --lock-tables -h [server] -u [username] -p[password] [db_name] > owncloud-sqlbkp_`date +"%
```

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"
```

7.3 Updating 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.

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, and for installations with a smaller number of users and data.

If you installed ownCloud from the openSUSE Build Service repositories or your own Linux distribution repositories, then it is best to upgrade ownCloud using your package manager rather than using the Updater app or upgrading manually. See the *Upgrading Your ownCloud Server* for instructions on maintaining your ownCloud server from packages and upgrading manually.

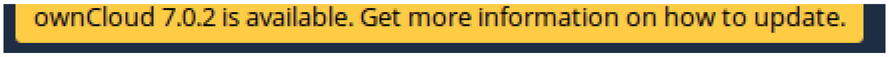
You should always maintain regular backups (see *Backing up ownCloud*), and make a backup before every update/upgrade. The Updater app does not backup your database or data directory.

The Updater app performs these operations:

- Creates a `backup` directory under your ownCloud data directory
- Download and extracts updated package content into the `backup/packageVersion` directory
- Makes a copy of your current ownCloud instance, except for your data directory, to `backup/currentVersion-randomstring`
- Moves all directories except `data`, `config` and `themes` from the current instance to `backup/tmp`
- Moves all directories from `backup/packageVersion` to the current version
- Copies your old `config.php` to the new `config/` directory

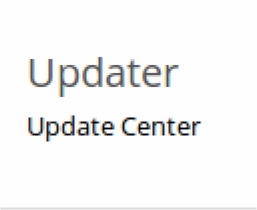
Using the Updater app to upgrade your ownCloud installation is just a few steps:

1. You should see a notification at the top of any ownCloud page when there is a new update available:

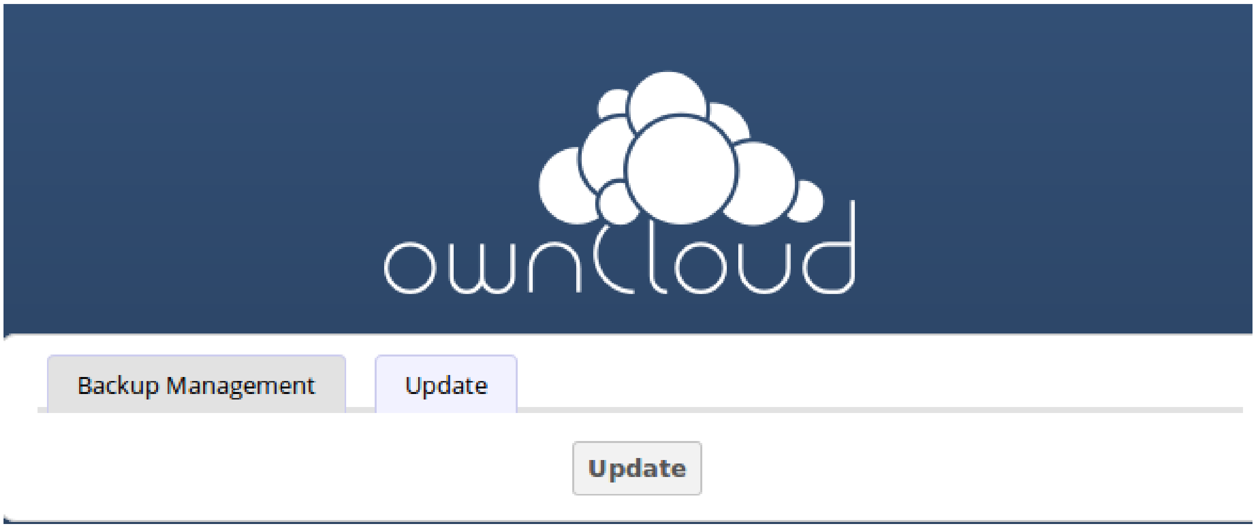


ownCloud 7.0.2 is available. Get more information on how to update.

2. Even though the Update 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 *Setting Strong Permissions*.
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; see *Setting Strong Permissions*. Otherwise you will see a message about checking your installation, making a backup, and moving files:

8. Click Proceed, and then it downloads the updates, which may take a few minutes:
7. The Update app wants you to be very sure you want to update, and so you must click one more button, the Start Update button:

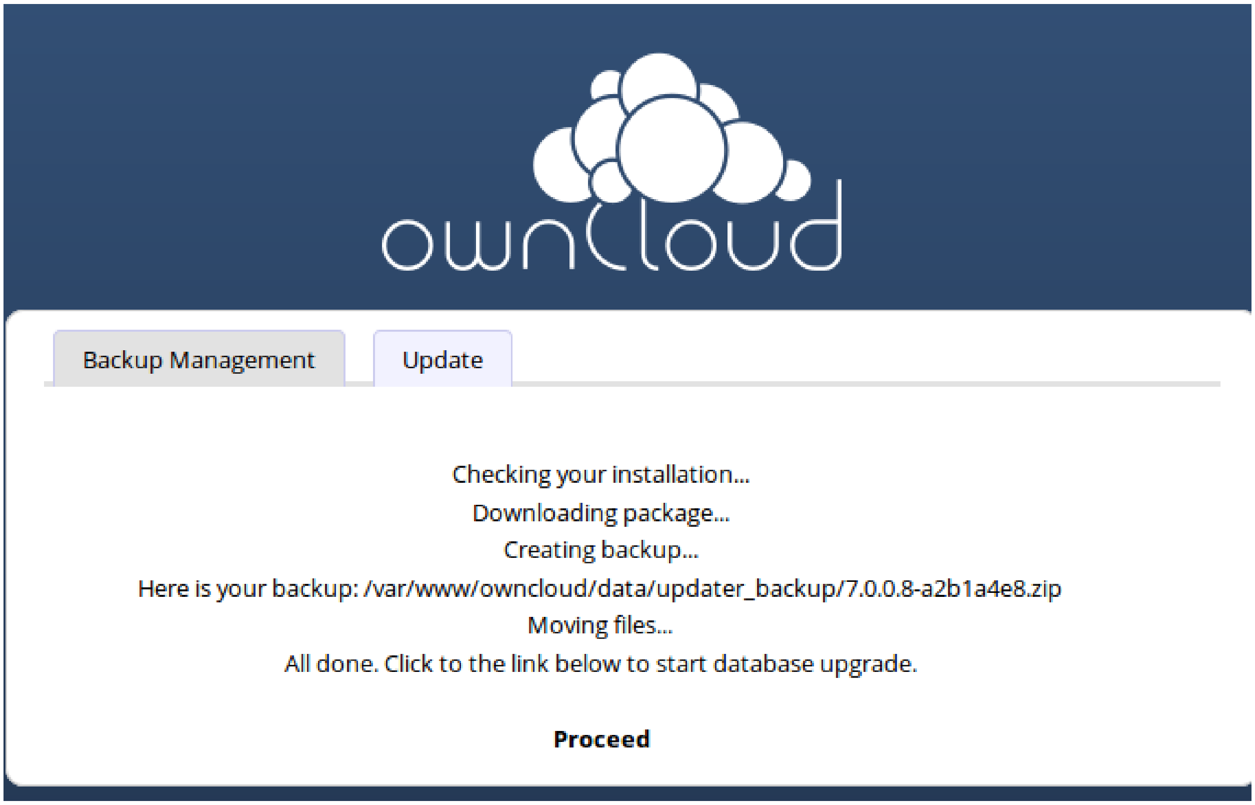
Note: If you have a large ownCloud installation, at this point you should use the `occ upgrade` command, running it as your HTTP user, instead of clicking Start Update, in order to avoid PHP timeouts. This example is for Ubuntu Linux:

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

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

8. It works 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.



Checking your installation...

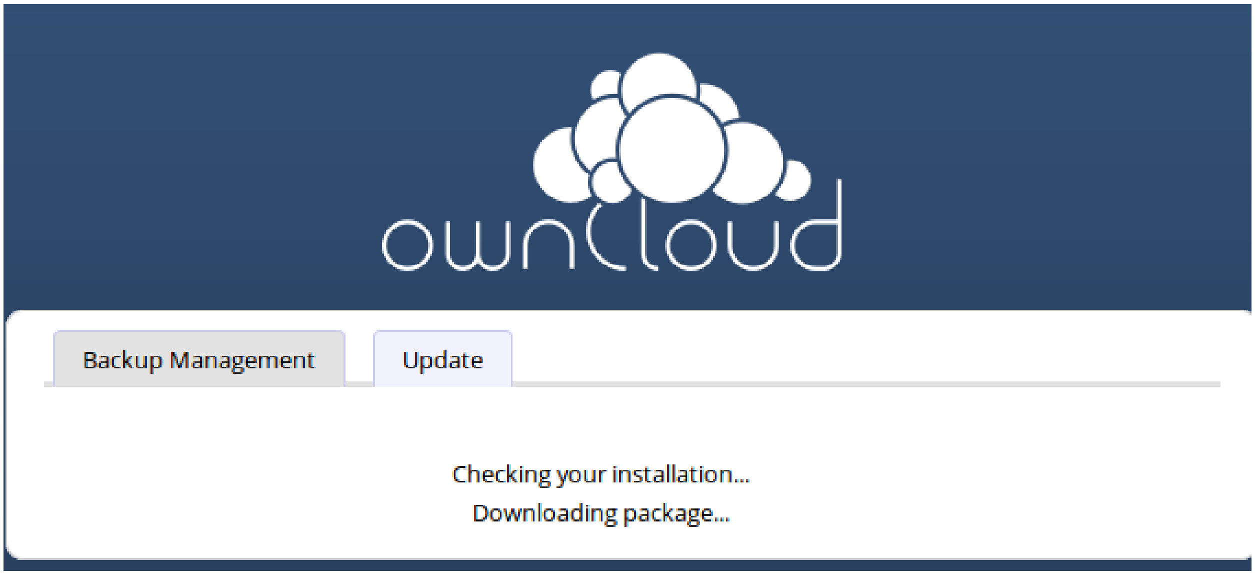
Downloading package...

Creating backup...

Here is your backup: /var/www/owncloud/data/updater_backup/7.0.0.8-a2b1a4e8.zip

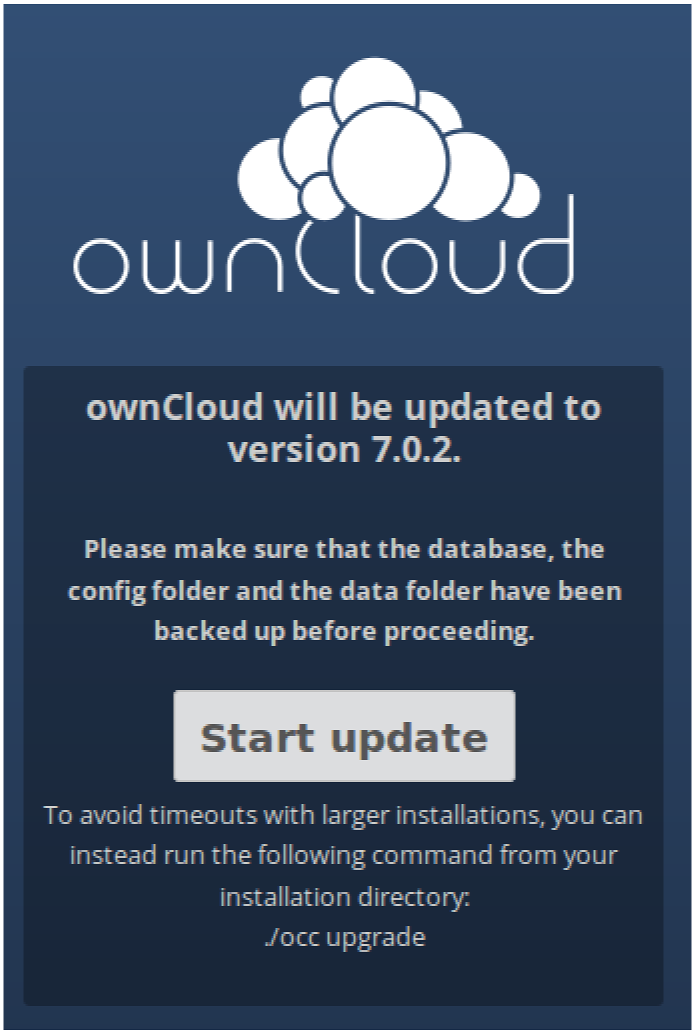
Moving files...

All done. Click to the link below to start database upgrade.



Checking your installation...

Downloading package...

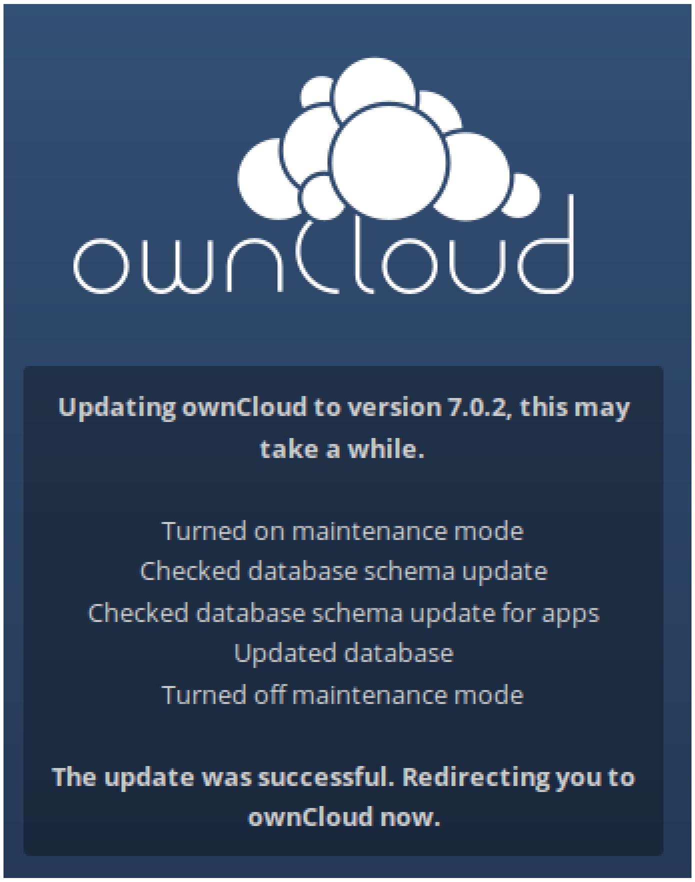


ownCloud will be updated to version 7.0.2.

Please make sure that the database, the config folder and the data folder have been backed up before proceeding.

To avoid timeouts with larger installations, you can instead run the following command from your installation directory:

```
./occ upgrade
```



Updating ownCloud to version 7.0.2, 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.

If the Updater app fails, then you must update manually. See *Upgrading Your ownCloud Server* to learn how to upgrade manually.

7.3.1 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.

7.4 Upgrading Your ownCloud Server

It is best to keep your ownCloud server upgraded regularly, and to install all point releases and major releases without skipping any of them. Major releases are 6.0 and 7.0, and point releases are intermediate releases for each major release. For example, 7.0.1 and 7.0.2 are point releases.

There are multiple ways to keep your ownCloud server upgraded: with the Updater App, 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.

(See *Updating ownCloud with the Updater App* to learn about the Updater App.)

Note: Before upgrading to a new major release, always first 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.

7.4.1 Preferred Upgrade Method

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.

You should always maintain regular backups (see *Backing up ownCloud*), and make a backup before every upgrade.

When a new ownCloud release is available you will see a yellow banner in your ownCloud Web interface.

ownCloud 7.0.2 is available. Get more information on how to update.

Upgrading is disruptive. When you upgrade ownCloud with your Linux package manager, that is just the first step to applying the upgrade. After downloading the new ownCloud packages your session will be interrupted, and you must run the upgrade wizard to complete the upgrade, which is discussed in the next section.

7.4.2 Upgrading With Your Linux Package Manager

When an ownCloud upgrade is available from the openSUSE Build Service repository, you can 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. There is one more step, and that is to run the upgrade wizard to perform the final steps of updating the database and turning off maintenance mode. After using your package manager to install the current ownCloud release, you will see two screens. On the first screen, click the Start Upgrade button, or optionally run the `occ upgrade` command instead of clicking the button.

`occ upgrade` is more reliable, especially on installations with large datasets and large numbers of users because it avoids the risk of PHP timeouts.

Note: The `occ` command does not download ownCloud updates. You must first download the updated code, and then `occ` performs the final upgrade steps.

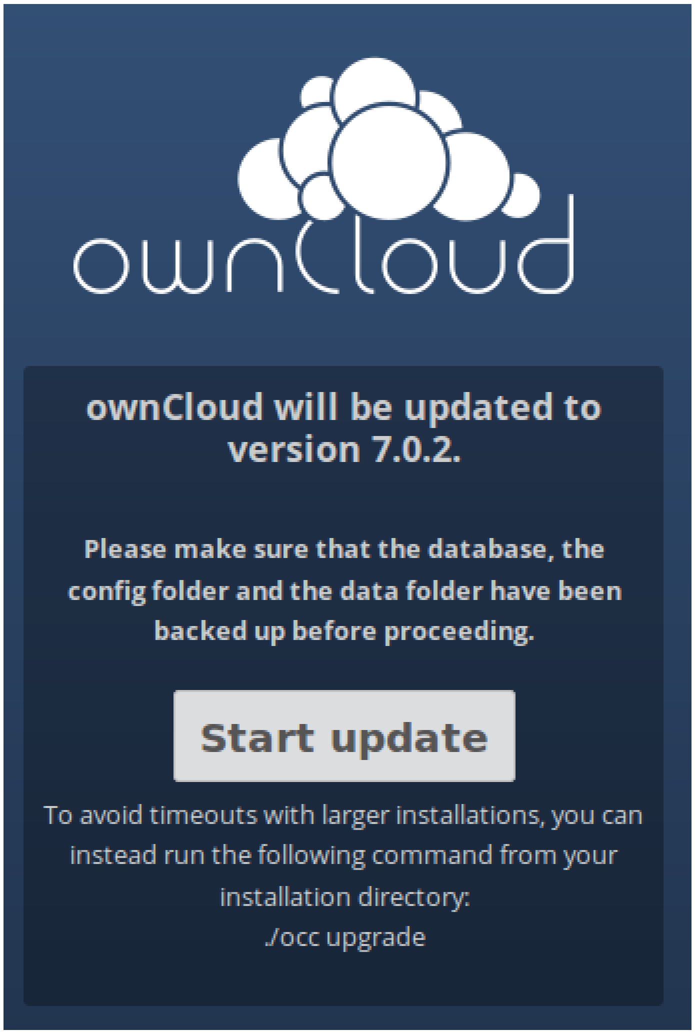
The `occ` command is in your `owncloud/` directory. You must run it as your HTTP user. This example is for Debian/Ubuntu:

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

This example is for Fedora, CentOS, and Red Hat Linux:

```
$ sudo -u apache php occ upgrade
```

- The HTTP user and group in Debian/Ubuntu is `www-data`.
- The HTTP user and group in Fedora/CentOS/RHEL is `apache`.
- The HTTP user and group in Arch Linux is `http`.



ownCloud will be updated to version 7.0.2.

Please make sure that the database, the config folder and the data folder have been backed up before proceeding.

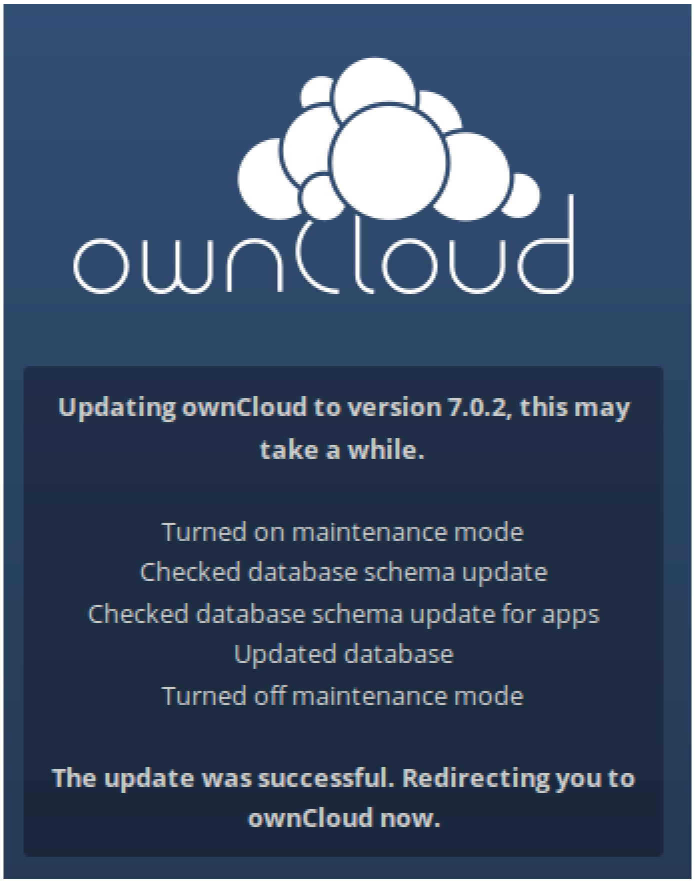
To avoid timeouts with larger installations, you can instead run the following command from your installation directory:

```
./occ upgrade
```

- The HTTP user in openSUSE is `wwwrun`, and the HTTP group is `www`.

See *Using the occ Command* to learn more about using the `occ` command, and see the **Setting Strong Directory Permissions** section of *Installation Wizard* to learn how to find your HTTP user.

When the upgrade is successful you will see the following screen:



Updating ownCloud to version 7.0.2, 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.

If the upgrade fails, then you must try a manual upgrade.

7.4.3 Manual Upgrade Procedure

Start by putting 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. This example is for Ubuntu Linux:

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

Please see *Using the occ Command* to learn more about `occ`.

The other way is by entering your `config.php` file and changing `'maintenance' => false,` to `'maintenance' => true,`. When you're finished upgrading, remember to change `true` to `false`.

Then:

1. Ensure that you are running the latest point release of your current major ownCloud version.
2. Deactivate all third party applications (not core apps), and review them for compatibility with your new ownCloud version.

3. Back up your existing ownCloud Server database, data directory, and `config.php` file. (See *Backing up ownCloud*.)
4. Download the latest ownCloud Server version 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/`

On Linux operating systems, change to your new directory and download the current ownCloud tarball with `wget`:

```
wget http://download.owncloud.org/community/owncloud-latest.tar.bz2
```

For Windows operating systems see the installation instruction in *Windows 7 and Windows Server 2008*.

5. Stop your web server.

Depending on your environment, you will be running either an Apache server or a Windows IIS server. To stop an Apache server, refer to the following table for specific commands to use in different Linux operating systems:

Operating System	Command (as root)
CentOS/ Red Hat	`apachectl stop`
Debian or Ubuntu	`/etc/init.d/apache2 stop`
SUSE Enterprise Linux 11	`/usr/sbin/rcapache2 stop`
openSUSE 12.3 and up	`systemctl stop apache2`

To stop the Windows IIS web server, you can use either the user interface (UI) or command line method as follows:

Method	Procedure
User Interface (UI)	1. Open IIS Manager and navigate to the web server node in the tree. 2. In the **Actions** pane, click **Stop**.
Command Line	1. Open a command line window as administrator. 2. At the command prompt, type **net stop WAS** and press **ENTER**. 3. (Optional) To stop W3SVC, type **Y** and then press **ENTER**.

6. Rename or move your current ownCloud directory (named `owncloud/` if installed using defaults) to another location.
7. Unpack your new tarball:

   ```
   tar xjf owncloud-latest.tar.bz2
   ```

   In Microsoft Windows environments, you can unpack the release tarball using WinZip or a similar tool (for example, Peazip). Always unpack server code into an empty directory. Unpacking the server code into an existing, populated directory is not supported and will cause all kinds of errors.
8. 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.`
9. Copy and paste the `config.php` file from your old version of ownCloud to your new ownCloud version.
10. 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.

Note: We recommend storing your `data/` directory in a location other than your `owncloud/` directory.

11. Restart your web server.

Depending on your environment, you will be running either an Apache server or a Windows IIS server. In addition, when running your server in a Linux environment, the necessary commands for stopping the Apache server might differ from one Linux operating system to another.

To start an Apache server, refer to the following table for specific commands to use in different Linux operating systems:

Operating System	Command (as root)
CentOS/ Red Hat	`apachectl start`
Debian or Ubuntu	`/etc/init.d/apache2 start`

SUSE Enterprise Linux 11
openSUSE 12.3 and up
	`/usr/sbin/rcapache2 start`
	`systemctl start apache2`

To start the Windows IIS web server, you can use either the user interface (UI) or command line method as follows:

Method	Procedure
User Interface (UI)	1. Open IIS Manager and navigate to the web server node in the tree. 2. In the **Actions** pane, click **Stop**.
Command Line	1. Open a command line window as administrator. 2. At the command prompt, type **net stop WAS** and press **ENTER**. 3. (Optional) To stop W3SVC, type **Y** and then press **ENTER**.

12. 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 previous section. Review the prequisites, 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 update from the command line using `occ` to avoid 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, and then `occ` performs the final upgrade steps.

    Please see *Using the occ Command* to learn more about `occ`.

13. The upgrade operation takes a few minutes, 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 the 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.

Now you can enable your third-party apps.

7.4.4 Setting Strong Permissions

For hardened security we highly recommend setting the permissions on your ownCloud directory as strictly as possible. After upgrading, verify that your ownCloud directory permissions are set according to the Setting Strong Directory Permissions section of *Installation Wizard*.

7.5 Restoring ownCloud

To restore an ownCloud installation there are three main things you need to restore:

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

When you have completed your restoration, see the `Setting Strong Directory Permissions` section of *Installation Wizard*.

7.5.1 Restore Folders

Note: This guide assumes that your previous backup is called "owncloud-dirbkp"

Simply copy your config 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/
```

7.5.2 Restore Database

Note: This guide assumes that your previous backup is called "owncloud-sqlbkp.bak"

MySQL

MySQL is the recommended database engine. To backup 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
```

7.6 Migrating ownCloud Installations

To migrate an ownCloud install, follow those steps:

1. Backup data/config folders and your database (see *Backing up ownCloud*)
2. Move your data
3. Restore your data/config folders and your database (see *Restoring ownCloud*)
4. Update config.php of any changes to your database connection

7.7 Converting From SQLite to MySQL, MariaDB, or PostgreSQL

You can convert a SQLite database to a more performing MySQL, MariaDB or PostgreSQL database with the ownCloud command line tool `occ`, which first appeared in ownCloud version 7.0.0. You must have ownCloud 7 to perform this conversion. SQLite is sufficient for testing and for very small installations, but for production servers with multiple users it is better to use MySQL, MariaDB or PostgreSQL.

Please see *Using the occ Command* for more information on using the `occ` command.

7.7.1 Running the Conversion

First set up the new database, in these examples called "new_db_name". In your ownCloud root folder call:

```
php occ db:convert-type [options] type username hostname database
```

The available values for the `type` parameter are:

- mysql (for MySQL or MariaDB)
- pgsql (for PostgreSQL)

7.7.2 Conversion Options

- `--port="3306"` Your database port (optional, specify the port if it is a non-standard port).
- `username` A database admin user.
- `--password="mysql_user_password"` The database admin password, if there is one.
- `--clear-schema` Clear the schema in the new DB (optional).
- `--all-apps` By default, tables for enabled apps are converted. Use this option to convert tables of deactivated apps.

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 the tables of removed apps will not be converted even with the option `--all-apps`

This example converts the SQLite DB and tables for all installed apps to MySQL/MariaDB:

```
php occ db:convert-type --all-apps mysql oc_mysql_user 127.0.0.1 new_db_name
```

To complete the conversion, type `yes` when prompted `Continue with the conversion?` On success the converter will automatically configure the new database in your ownCloud configuration in `config.php`.

7.7.3 Unconvertible Tables

After conversion some obsolete database tables may be left over. The converter will tell you what these are:

```
The following tables will not be converted:
oc_permissions
...
```

You can ignore these tables. Here is a list of known old tables:

- oc_calendar_calendars
- oc_calendar_objects
- oc_calendar_share_calendar
- oc_calendar_share_event
- oc_fscache
- oc_log
- oc_media_albums
- oc_media_artists
- oc_media_sessions
- oc_media_songs
- oc_media_users
- oc_permissions
- oc_queuedtasks
- oc_sharing

CHAPTER

EIGHT

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 Edition with commercial support options.

8.1 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.

8.2 General Troubleshooting

8.2.1 Debugging the issue

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` from the Admin page. Please see *Logging Configuration* for more informations on this 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 decent developer tools for viewing the console, and you usually access them by pressing F-12. For Firefox it is recommended to install the Firebug extension.

Note: The logfile of ownCloud is located in the datadirectory `/var/www/owncloud/data/owncloud.log`.

8.2.2 Debugging Sync-Issues

Note: 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

8.2.3 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 informations.
- `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 From SQLite to MySQL, MariaDB, or PostgreSQL*.
- `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 too large packet. Please refer to the manual of your database 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 datadirectory (`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`.

8.3 Troubleshooting Webserver and PHP problems

8.3.1 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 datadirectory `/var/www/owncloud/data/owncloud.log`.

8.3.2 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 over this 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. LigHTTPd

- ModWebDAV
- X-Sendfile2 (causing broken downloads if not configured correctly)

5. PHP

- eAccelerator

8.4 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 informations about WebDAV problems.

8.5 Troubleshooting Contacts & Calendar

8.5.1 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.

8.5.2 Apple iOS

Below is what have proven to work with iOS including iOS 7.

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-know" 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
```

If you use lighttpd as web server, the setting looks something like:

```
url.redirect = (
    "^/.well-known/carddav" => "/owncloud/remote.php/carddav",
    "^/.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.

8.5.3 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:

Outdated lighttpd web server lighttpd in debian wheezy (1.4.31) doesn't support the PATCH HTTP verb. Upgrade to lighttpd >= 1.4.33.

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.