



ownCloud Administrators Manual

Release 7.0

The ownCloud developers

November 11, 2014

CONTENTS

1	Introduction	1
1.1	Target Audience	1
1.2	Document Structure	1
2	What's New for Admins in ownCloud 7	5
2.1	New User Management	5
2.2	External Storage	5
2.3	Object Stores as Primary Storage	5
2.4	Server to Server Sharing	5
2.5	SharePoint Integration (Enterprise Edition only)	5
2.6	Windows Network Drive Integration (Enterprise Edition only)	6
2.7	Sharing	6
2.8	Email Configuration Wizard	6
2.9	Editable Email Templates	6
2.10	Active Directory and LDAP Enhancements	6
3	Installation	7
3.1	Appliances	7
3.2	Linux Distributions	7
3.3	Mac OS X	8
3.4	Windows 7 and Windows Server 2008	8
3.5	Univention Corporate Server	15
3.6	Manual Installation	21
3.7	Other Installation Methods	27
3.8	Installation Wizard	27
3.9	Nginx Configuration	31
3.10	Lighttpd Configuration	32
3.11	Yaws Configuration	33
3.12	Hiawatha Configuration	34
4	Configuration	35
4.1	Apps Configuration	35
4.2	User Management	37
4.3	User Authentication with LDAP	40
4.4	Configuring Server-to-Server Sharing	53
4.5	Defining Background Jobs	54
4.6	Configuring the Collaborative Documents App	55
4.7	Enabling the Documents App	56
4.8	Asset Management	56
4.9	Using Third Party Components	57
4.10	Configuring the ClamAV Antivirus Scanner	57

4.11	Defining Automatic Configuration	60
4.12	Custom Client Configuration	62
4.13	Database Configuration	63
4.14	Using Server-Side Encryption	69
4.15	Knowledge Base Configuration	74
4.16	Language Configuration	74
4.17	Logging Configuration	75
4.18	Mail Configuration	76
4.19	Preview Configuration	83
4.20	Reverse Proxy Configuration	85
4.21	File Sharing	85
4.22	Uploading big files > 512MB (as set by default)	88
4.23	Enabling uploading big files	88
4.24	Configuring External Storage (GUI)	89
4.25	Configuring External Storage (Configuration File)	102
4.26	Custom User Backend Configuration	108
4.27	Serving static files via web server	109
4.28	Enabling Full-Text Search	112
4.29	Config.php Parameters	113
5	Maintenance	125
5.1	Maintenance Mode Configuration	125
5.2	Backing up ownCloud	125
5.3	Updating ownCloud with the Updater App	126
5.4	Upgrading Your ownCloud Server	131
5.5	Restoring ownCloud	133
5.6	Migrating ownCloud Installations	134
5.7	Converting Database Type	134
6	Issues	137
7	Indices and tables	139

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 Word 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 Document Structure

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

1.2.1 Installation

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

- [Linux Distributions](#) (recommended)
- [Windows 7 and Windows Server 2008](#)
- [Manual Installation](#)
- [Other Installation Methods](#)
- [Univention Corporate Server](#)
- [Mac OS X](#) (not supported)
- [Appliances](#)

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

1.2.2 Configuration

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

- *Apps Configuration*
- *File Sharing*
- *User Management*
- *Database Configuration*
- *User Authentication with LDAP*
- *Configuring Server-to-Server Sharing*
- *Configuring External Storage (GUI)*
- *Configuring External Storage (Configuration File)*
- *Defining Background Jobs*
- *Mail Configuration*
- *Defining Automatic Configuration*
- *Using Server-Side Encryption*
- *Uploading big files > 512MB (as set by default)*
- *Reverse Proxy Configuration*
- *Serving static files via web server*
- *Using Third Party Components*
- *Custom User Backend Configuration*
- *Custom Client Configuration*
- *Knowledge Base Configuration*
- *Logging Configuration*
- *Language Configuration*
- *Configuring the ClamAV Antivirus Scanner*
- *Preview Configuration*
- *Enabling Full-Text Search*
- *Config.php Parameters*

1.2.3 Maintenance

This section 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:

- *Maintenance Mode Configuration*
- *Backing up ownCloud*
- *Updating ownCloud with the Updater App*
- *Upgrading Your ownCloud Server*
- *Restoring ownCloud*

- *Migrating ownCloud Installations*
- *Converting Database Type*

WHAT'S NEW FOR ADMINS IN OWNCLOUD 7

2.1 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, including 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.

2.2 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.

2.3 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.

2.4 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.

2.5 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.

2.6 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.

2.7 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.

2.8 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.

2.9 Editable Email Templates

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

2.10 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.

INSTALLATION

3.1 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.).

3.1.1 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](#)

3.1.2 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.

3.2 Linux Distributions

3.2.1 Supported Distribution Packages

Ready-to-use packages are available at [openSUSE Build Service](#) for a variety of Linux distributions.

If your distribution is not listed please follow *Manual Installation*.

Additional installation guides and notes

Fedora: Make sure SELinux is disabled or else the installation process might fail.

Archlinux: There 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.

Follow the wizard to complete your installation

For setting up your ownCloud instance after installation, please refer to the [Installation Wizard](#) section.

3.3 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.

3.4 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.

3.4.1 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*.

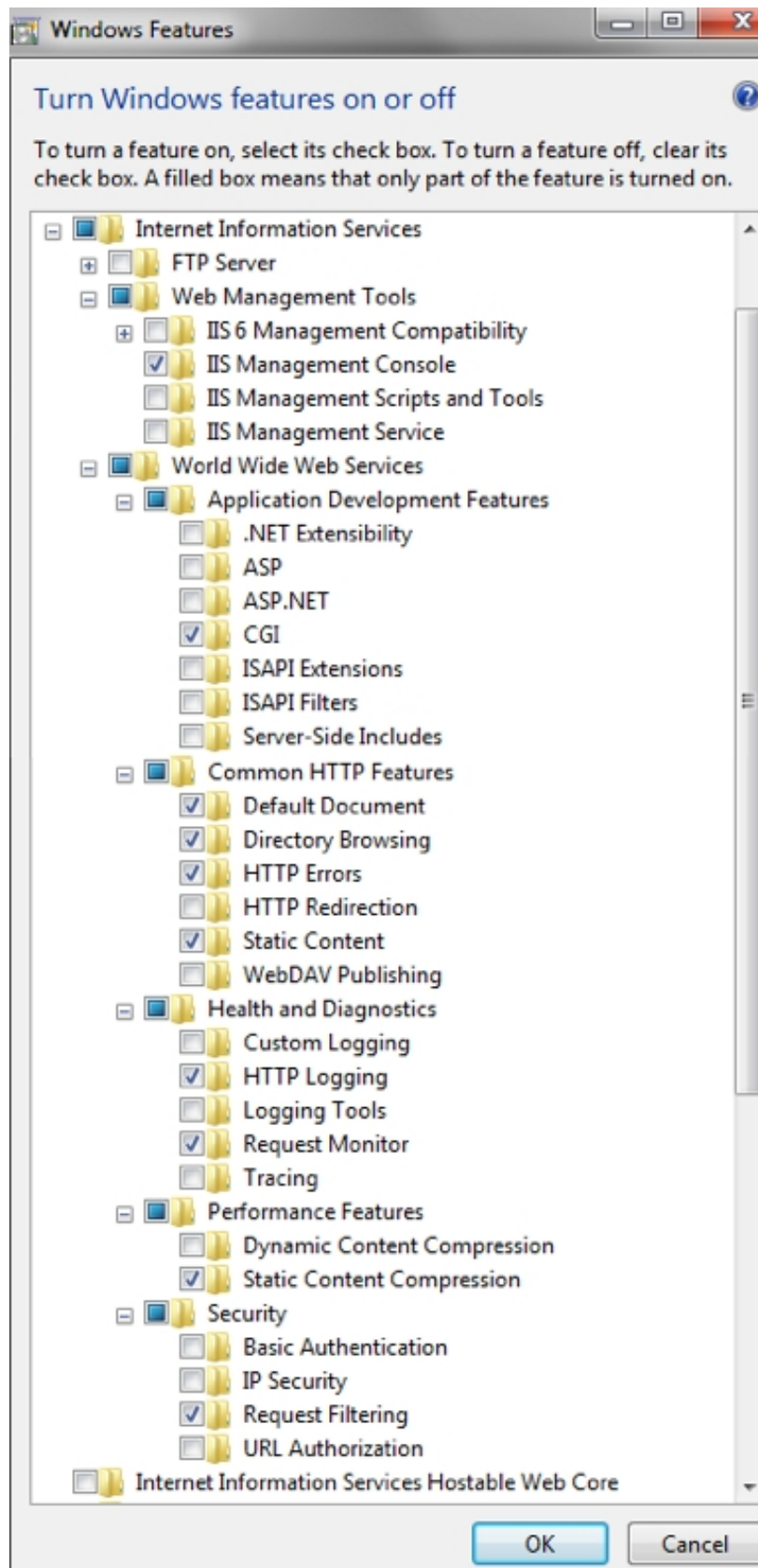


Figure 3.1: Windows Features required for ownCloud on Windows 7

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](#).

3.4.2 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](#).

3.4.3 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.

 **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

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/). <http://learn.iis.net/page.aspx/353/install-and-configure-mysql-for-php-applications-on-iis-7-and-above/>

3.4.4 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.

3.4.5 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 synchronizing 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.

3.4.6 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.

3.5 Univention Corporate Server

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

3.5.1 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 3.1 – continued from previous page

Key	Default	Description	Introduced
owncloud/ldap/loginFilter	(&((&(objectClass=posixAccount) (objectClass=shadowAccount)) (objectClass=univentionMail) (ob- jectClass=sambaSamAccount) (ob- jectClass=simpleSecurityObject) (&(objectClass=person) (ob- jectClass=organizationalPerson) (objectClass=inetOrgPerson))) (!(uidNumber=0)) (!(uid=*\$)) (&(uid=%uid) (ownCloudEn- abled=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) (ob- jectClass=sambaSamAccount) (ob- jectClass=simpleSecurityObject) (&(objectClass=person) (ob- jectClass=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 re- trieved (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, mailPrimaryAd- dress	Attributes taken into consideration when searching for groups (comma separated)	5.0.9
owncloud/join/users/update	yes	Whether ownCloud LDAP schema should be applied to existing users	2012.0.1
owncloud/group/enableDomainUsers	1	Whether the group "Domain Users" shall be enabled for ownCloud on install	2012.4.0.4

Continued on next page

Table 3.1 – continued from previous page

Key	Default	Description	Introduced
owncloud/join/users/filter	(&(!(&(objectClass=posixAccount) (objectClass=shadowAccount)) (objectClass=univentionMail) (ob- jectClass=sambaSamAccount) (ob- jectClass=simpleSecurityObject) (&(objectClass=person) (ob- jectClass=organizationalPerson) (objectClass=inetOrgPerson))) (!(uidNumber=0)) (!(uid=*\$) (uid=owncloudsystemuser) (uid=join-backup) (uid=join- slave))) (!(object- Class=ownCloudUser)))	Filters, on which LDAP users the ownCloud schema should be ap- plied to. The default excludes system users and already own- CloudUsers.	2012.0.1
owncloud/join/groups/filter	(empty)	Filters which LDAP groups will be en/disabled for own- Cloud 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 unusable system!

Entries

Category

Search attribute

Keyword

Search

All

All

owncloud*

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

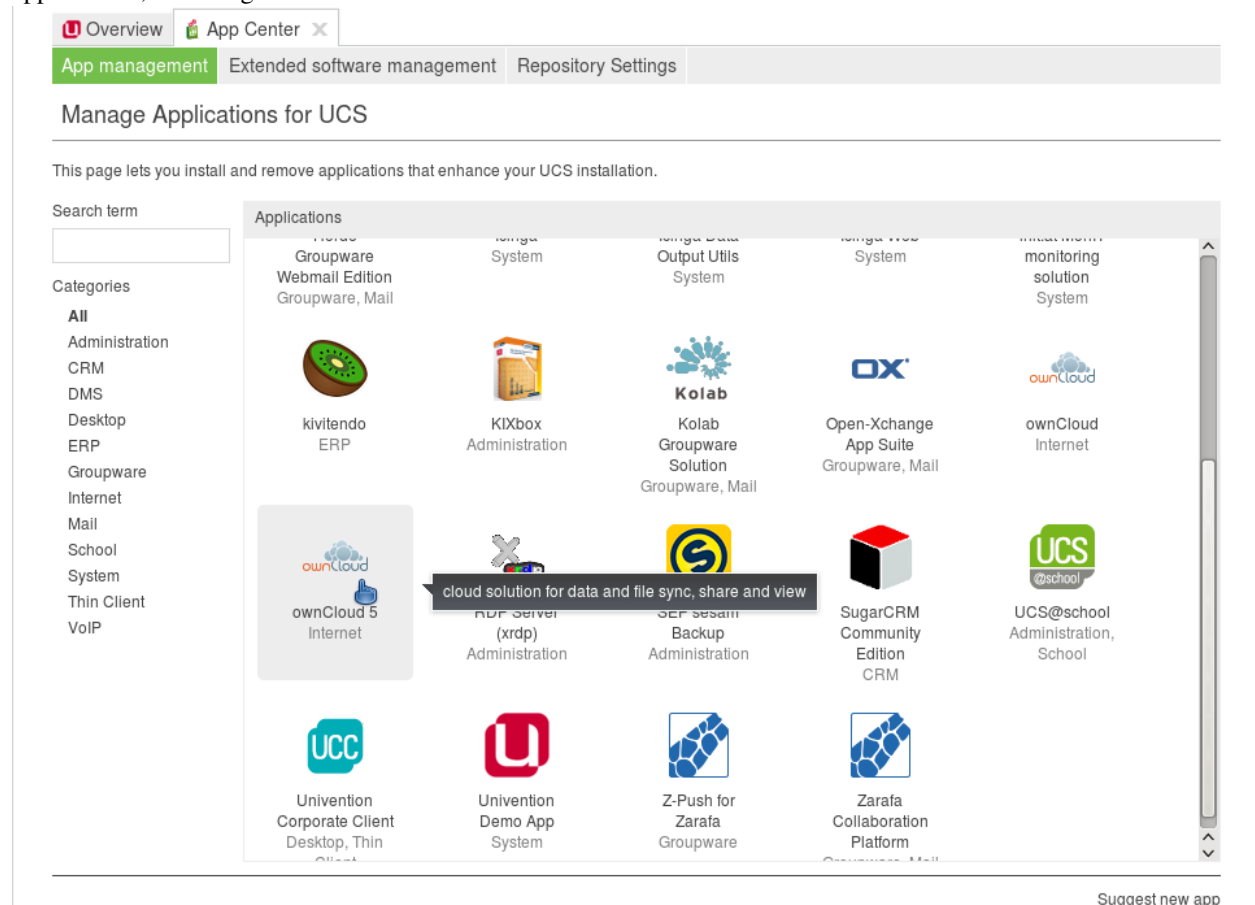
Add

3.5.2 Installation

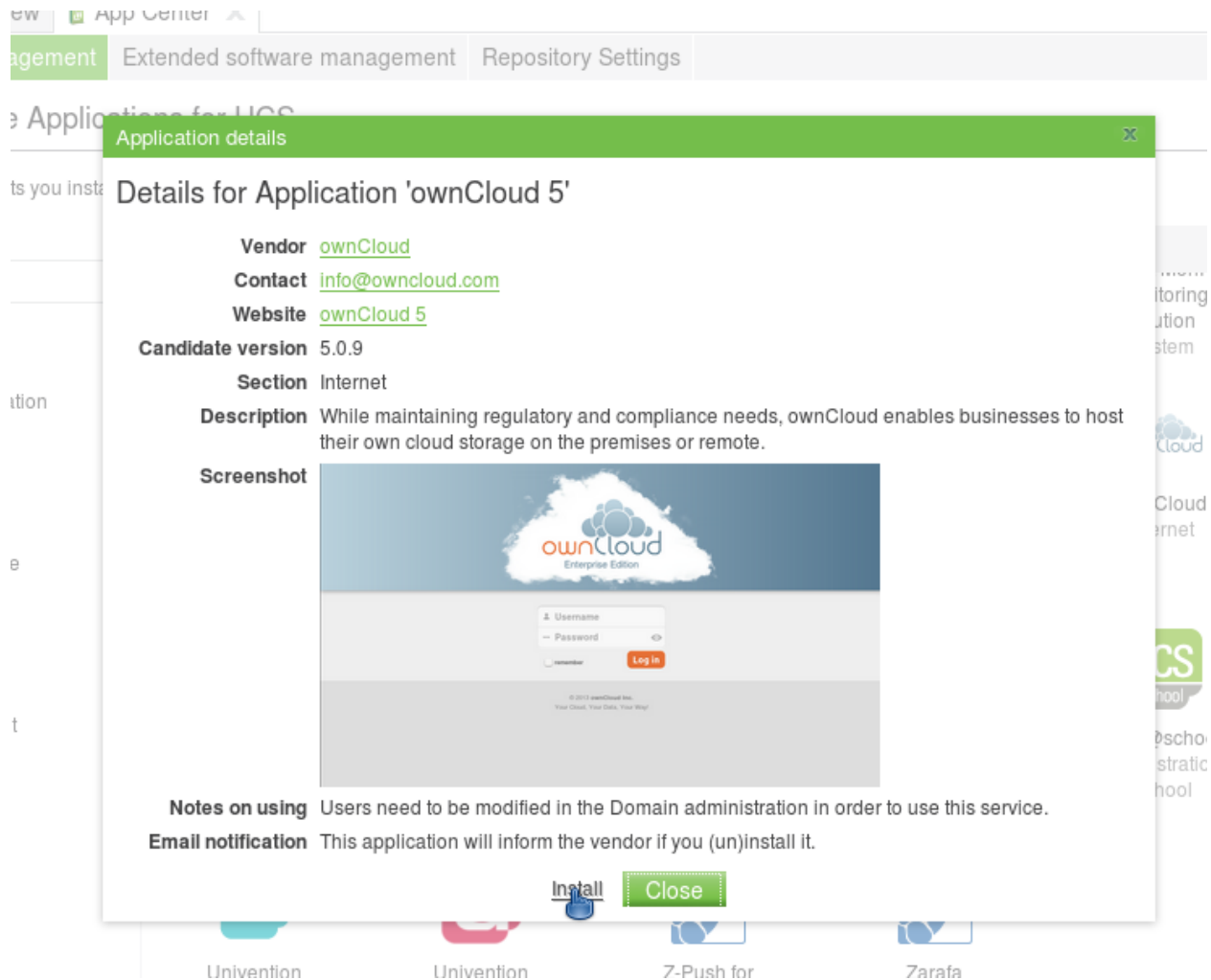
Now, we are ready to install ownCloud. This can be either done through the UCS App Center (recommended) or by downloading the packages.

UCS App Center

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



Click on ownCloud 5 and follow the instructions.



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.

3.5.3 Postconfiguration (optional)

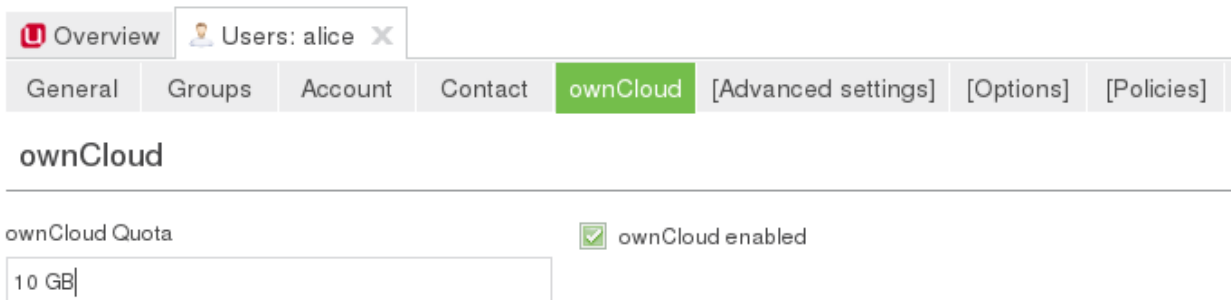
There is only one local admin user “owncloudadmin”, you can find his password in `/etc/owncloudadmin.secret`. Use this account, if you want to change basic ownCloud settings.

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.

3.5.4 Using ownCloud

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

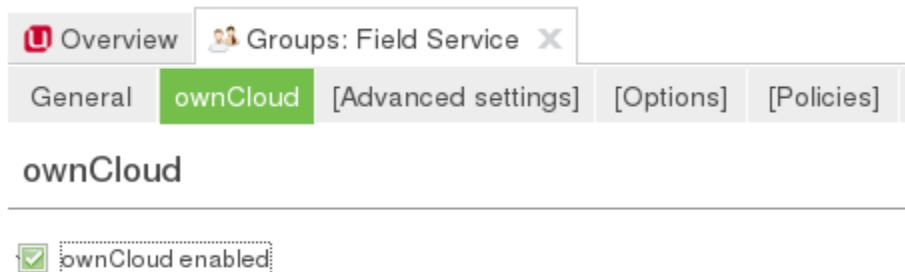
If you did not, go to the UMC and enable the users who shall have access (see picture below). Then, login at <http://myserver.com/owncloud/> with your LDAP credentials.



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

Groups 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.



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.

3.6 Manual Installation

If you do not want to use packages, here is how you setup ownCloud from scratch using a classic LAMP (Linux, Apache, MySQL, PHP) setup:

This document provides a complete walk-through for installing ownCloud on Ubuntu 14.04 LTS Server with Apache and MySQL.

3.6.1 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 highly likely to require contacting your hoster (e.g. for installing required modules).

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

- php5 ($\geq 5.3.8$, 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 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 (if you use the mentioned app, you must install that package):

- PHP module ldap (for ldap integration)
- smbclient (for SMB storage)
- PHP module ftp (for FTP storage)

Recommended for specific apps (*optional*):

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

For enhanced 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/enable these modules.
- Make sure your distribution's php version fulfils 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 any WebDAV support 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.

3.6.2 Example installation on Ubuntu 14.04 LTS Server

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

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

Remarks:

- 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 for 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:

- Navigate 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://owncloud.org/owncloud.asc
gpg --import owncloud.asc
gpg 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. Typically, on Ubuntu systems this `/var/www/owncloud`, so your copying command is:

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

3.6.3 Setting Secure Directory Permissions

Your HTTP user must own at least 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. We recommend setting the directory permissions as strictly as possible for stronger security.

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 a single line in it:

```
<?php echo exec('whoami'); ?>
```

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.

Note: When using an NFS mount for the data directory, do not change ownership as above. 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 generic command to change ownership of all files and subdirectories in a directory is:

```
chown -R <http-user>:<http-user> /path/to/owncloud/
```

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:

```
chown -R root:root /path/to/owncloud/  
chown <http-user>:<http-user> /path/to/owncloud/config/config.php  
chown -R <http-user>:<http-user> /path/to/owncloud/data/  
chown root:root /path/to/owncloud/data/.htaccess  
chown <http-user>:<http-user> /path/to/owncloud/apps/
```

These strict permissions will prevent the Updater app from working. If you use the Updater app, it needs your whole ownCloud directory to be owned by the http-user, like these examples:

- 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 Update app has run, you should re-apply the strict permissions.

Apache is the recommended Web server.

3.6.4 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.

3.6.5 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.

3.6.6 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 MultiViews
    AllowOverride All
    Order allow,deny
    allow from all
</Directory>
```

Example config for Apache 2.4:

```
<Directory /path/to/owncloud>
    Options Indexes FollowSymLinks MultiViews
    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 MultiViews
        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 MultiViews
    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
```

3.6.7 Install Wizard

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

3.6.8 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*

3.7 Other Installation Methods

3.7.1 PageKite Configuration

You can use this [PageKite how to](#) to make your local ownCloud accessible from the internet using PageKite.

3.8 Installation Wizard

When ownCloud prerequisites are fulfilled and all ownCloud files are installed on the server, the last thing left to do for finishing the installation is running the Installation Wizard.

- Open your web browser
- Navigate to your ownCloud instance.
 - If you are installing ownCloud on the same machine as you are accessing the install wizard from, the url will be <https://localhost/owncloud>
 - If you are installing ownCloud on a different machine, you'll have to access it by its hostname or IP address, e.g. <https://example.com/owncloud>

- Please take notice of the note at the end of the site if you’re accessing your ownCloud instance via a different IP address or host name during setup than your users are going to use later on (or if your ownCloud instance should be accessible via more than one host name or IP address).
- 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:

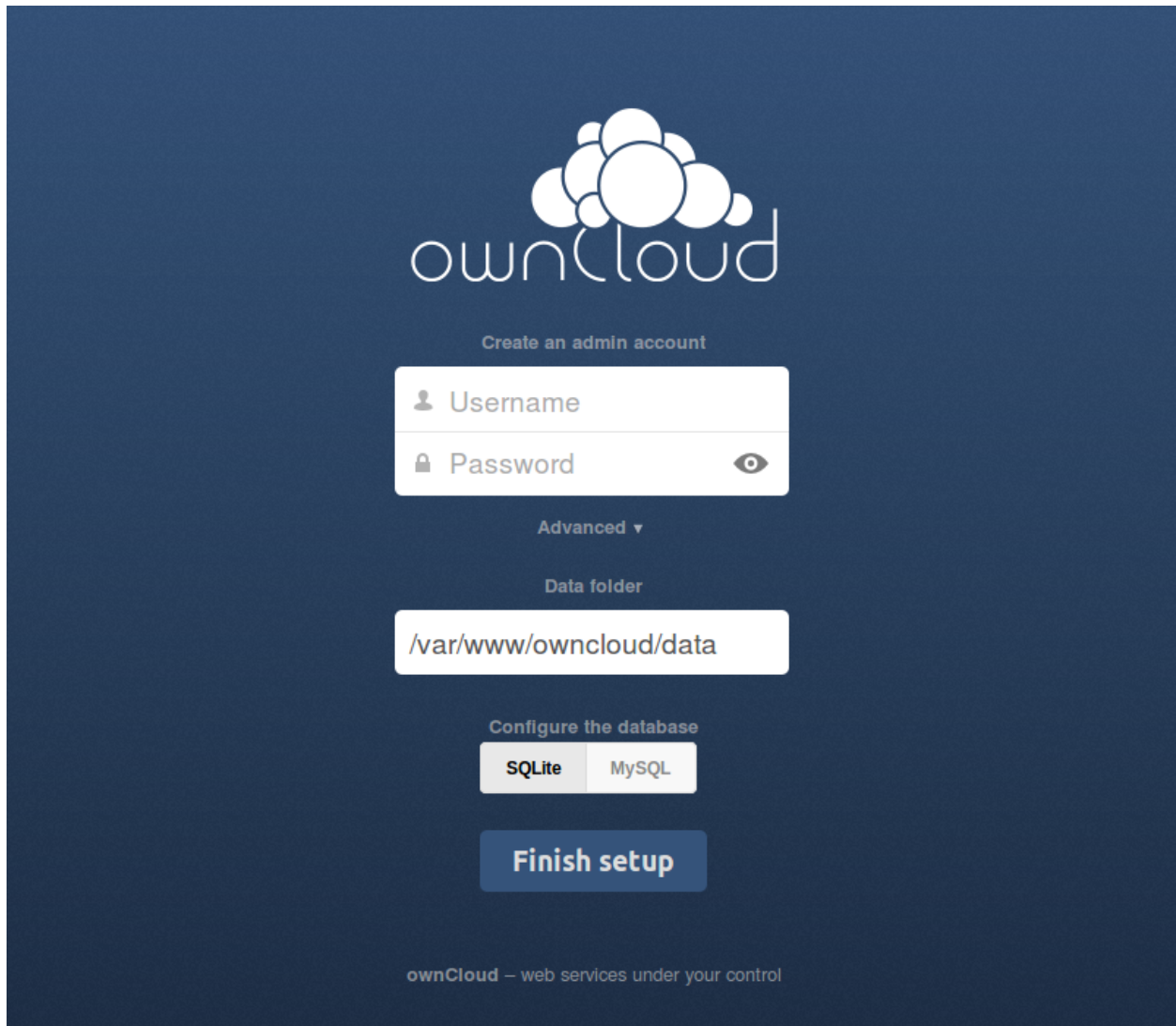


3.8.1 Required Settings

Under “create an admin account” you are requested to enter a username and password for the administrative user account. You can choose any username and password as you see fit, just make sure to remember it, you will need it later whenever you want to configure something for your ownCloud instance.

3.8.2 Advanced Options

- Advanced settings are available for configuring a different database or data directory than the default ones.
- 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!
- Show these additional options by clicking on “Advanced”:



The image shows the ownCloud installation wizard interface. At the top is the ownCloud logo. Below it, the text "Create an admin account" is displayed. There are two input fields: "Username" with a person icon and "Password" with a lock icon and a toggle eye icon. Below these fields is a link "Advanced" with a downward arrow. Then, the text "Data folder" is shown above a text input field containing the path "/var/www/owncloud/data". Below this is the section "Configure the database" with two buttons: "SQLite" (highlighted) and "MySQL". At the bottom of the configuration section is a large blue button labeled "Finish setup". At the very bottom of the screen, the text "ownCloud – web services under your control" is displayed.

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 or PostgreSQL you have two options regarding the database name and user account you specify:
 - You can specify either an admin/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.
 - * Beware that there are restrictions as to what characters a database name may or may not contain, see

- the [MySQL Schema Object Names](#) documentation for details);
- * Make sure to choose a name under which no database exists yet
- * ownCloud will use the provided credentials and create its own user with permissions only on its own database.
- You can enter the name of an existing database and the username/password of a user with permissions restricted to this one database only
 - * You can create such a user yourself, e.g. via phpmyadmin.
 - * This user shouldn't have permission to create a database.
 - * It should have full permissions on the (existing) database with the name you specify.

3.8.3 Finish Installation

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

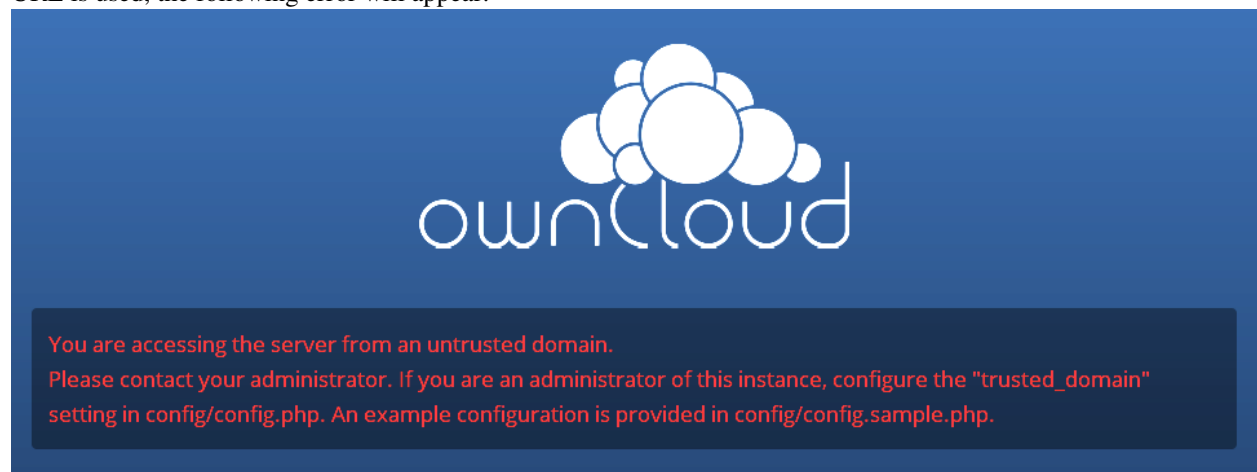
3.8.4 Note

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. No domain names of clients go there.

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.

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.

It should be noted that 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:



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

3.9 Nginx Configuration

- You need to insert the following code into **your nginx config file**.
- 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](#)).

```

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/;
    # set max upload size
    client_max_body_size 10G;
    fastcgi_buffers 64 4K;

    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.

3.10 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.

3.11 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.

3.12 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  
}
```

CONFIGURATION

4.1 Apps Configuration

After installing ownCloud, you might want to provide added functionality on top of the core functionality that is installed by default. ownCloud enables you to enhance your experience and your user's experiences by installing various *apps*.

4.1.1 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.

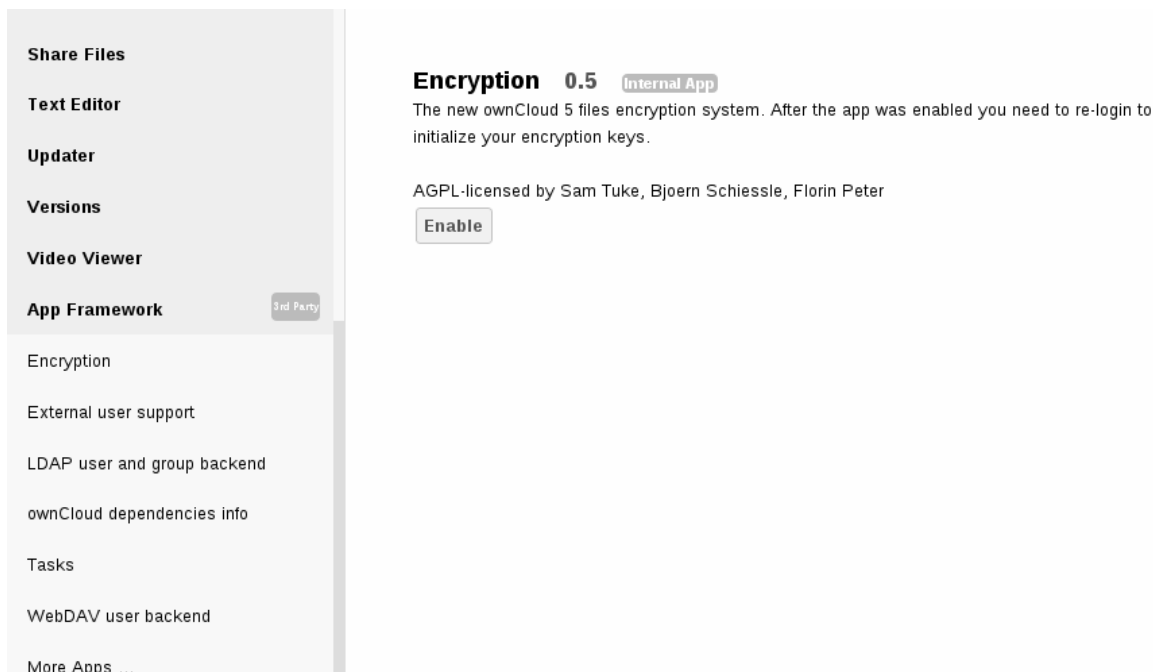


Figure 4.1: Administrator application page

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

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

4.1.2 Managing Apps

In the Apps page, you can enable or disable applications, as well as modify any app settings, by clicking the app name. 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.

To modify an app:

1. Click the app that you want to modify.

The app is highlighted and any available settings for the app appear in the Application View window.

2. Make any desired changes to the app.

All app changes are applied dynamically as soon as you select the change. As there are many apps available for use with ownCloud, we do not list all apps or apps settings that are available to you. However, we have documented several standard apps (for example, the Contacts and Calendar apps) that are directly supported.

4.1.3 Adding Third Party Apps

As mentioned earlier, ownCloud supports a number of different 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.

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.

Though ownCloud provides many apps in the server installation, you can view many more apps at 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 install any that you like.

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.

4.1.4 Setting App Parameters

Parameters are set in the `config/config.php` inside the `$CONFIG` array.

Use custom app directories

Use the `apps_paths` array to set the apps folders which should be scanned for available apps and/or where user specific apps should be installed. 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: If you want to make sure that the default `/apps/` folder only contains apps shipped with ownCloud, you should

follow the example and set-up a **/apps2/** folder which will be used to store all apps downloaded by users

```
<?php

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

Using Your Own Appstore

You can enable the installation of apps from your own apps store. However, 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",
```

Guarding Against Malicious Code

You can enable checks for malicious code fragments in third party apps by setting the **appcodechecker** parameter.

```
<?php

"appcodechecker" => false,
```

4.2 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.

User accounts have the following properties:

Login Name	Password	Groups	Create				
Username	Full Name	Password	Groups	Group Admin	Quota	Storage	
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	

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.

4.2.1 Creating a New User

To create a user account:

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

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

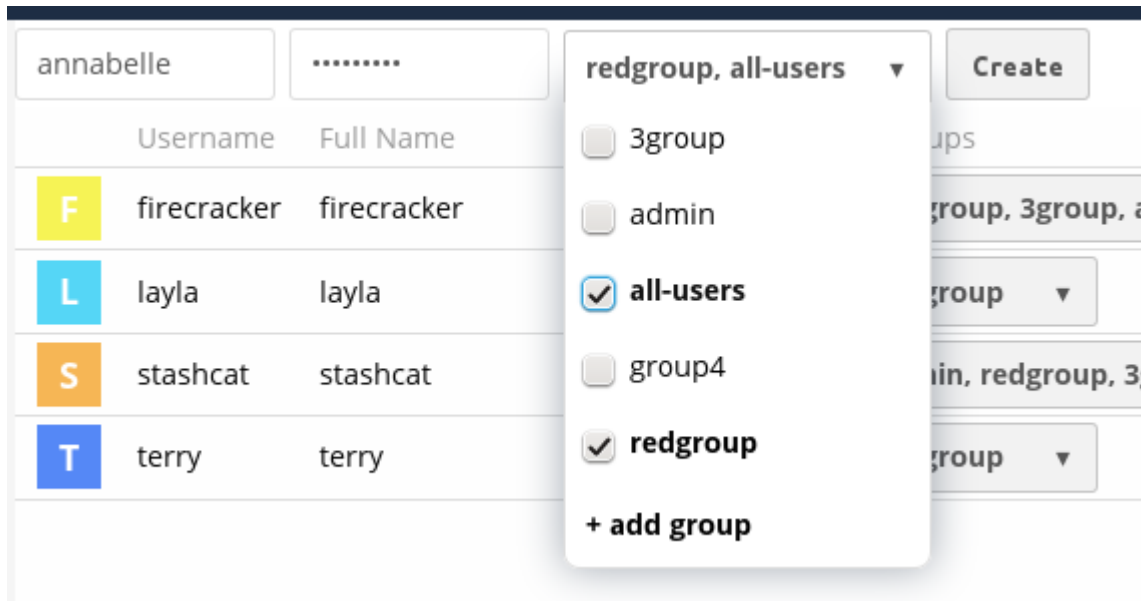
Remember to give your new users their logins and passwords.

4.2.2 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 [Using Server-Side Encryption](#).



4.2.3 Renaming a User

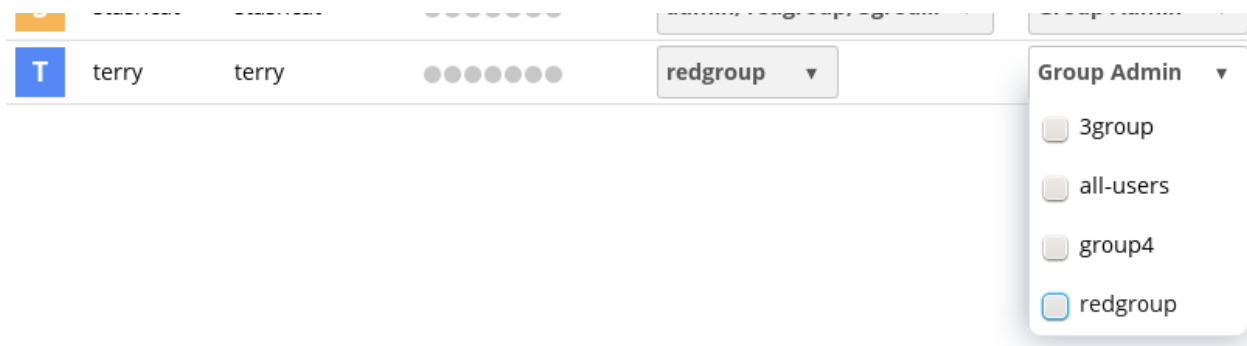
Each ownCloud user has two names: a unique **Login Name** used for authentication, and a **Full Name**, which is their display name. You can edit the display name of a user, but you cannot change the login name of any user.

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

4.2.4 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.

4.2.5 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.

4.2.6 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.

4.2.7 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.

4.3 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 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

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

4.3.1 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:

The screenshot shows the 'Server' configuration tab in the ownCloud administrator interface. It includes a dropdown menu labeled '1. Server:', a 'Delete Configuration' button, and input fields for 'Host', 'Port', 'User DN', 'Password', and a multi-line text area for 'One Base DN per line'. At the bottom, a status message 'Configuration incomplete' is shown next to 'Continue' and 'Help' buttons.

- `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 have access to your ownCloud server. You may bypass the form fields and enter a raw LDAP filter if you prefer.

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

only from those groups: If your LDAP server supports the `member-of-overlay` in LDAP filters, you can define that only users from one or more certain groups are allowed to appear and log in into 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`

x users found: This is an indicator that tells you approximately how many users will be allowed to access ownCloud. The number updates automatically after any changes.

Login Filter

The settings in the Login Filter tab determine what the user's login will be, for example an LDAP username, or an email address. You may select multiple user details. (You may bypass the form fields and enter a raw LDAP filter if you prefer.)

You may override your User Filter settings on the User Filter tab by using a raw LDAP filter.

LDAP Username: If this value is checked, the login value will be compared to the username in the LDAP directory. The corresponding attribute, usually *uid* or *samaccountname* will be detected automatically by ownCloud.

LDAP Email Address: If this value is checked, the login value will be compared to an email address in the LDAP directory; specifically, the *mailPrimaryAddress* and *mail* attributes.

Other Attributes: This multi-select box allows you to select other attributes for the comparison. The list is generated automatically from the user object attributes in your LDAP server.

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

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

Examples:

- only username: *uid=%uid*
- username or email address: *(!(uid=%uid)(mail=\$uid))*

Group Filter

By default, no LDAP groups will be available in ownCloud. The settings in the group filter tab determine which groups will be available in ownCloud. You may also elect to enter a raw LDAP filter instead.

Server

User Filter

Login Filter

Group Filter

What attribute shall be used as login name:

LDAP Username: ☒

LDAP Email Address: ☐

Other Attributes:

[⌵ Edit raw filter instead](#)

Configuration OK ● [Back](#) [Continue](#) [i Help](#)

Server

User Filter

Login Filter

Group Filter

Limit the access to ownCloud to groups meeting this criteria:

only those object classes:

only from those groups:

[⌵ Edit raw filter instead](#)

0 groups found

Configuration OK ● [Back](#) [i Help](#)

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

only from those groups: ownCloud will generate a list of available groups found in your LDAP server. and then you select the group or groups that get access to your ownCloud server.

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

Example:

- *objectClass=group*
- *objectClass=posixGroup*

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

4.3.2 Advanced Settings

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

The Advanced Settings are structured into three parts:

- Connection Settings
- Directory Settings
- Special Attributes

Connection Settings

Figure 4.2: LDAP Advanced Settings, section Connection Settings

Configuration Active: Enables or Disables the current configuration. By default, it is turned off. When ownCloud makes a successful test connection it is automatically turned on.

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

Example:

- *directory2.my-company.com*

Backup (Replica) Port: The connection port of the backup LDAP server. If no port is given, but only a host, then the main port (as specified above) will be used.

Example:

- *389*

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

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

Turn off SSL certificate validation: Turns off SSL certificate checking. Use it for testing only!

Cache Time-To-Live: A cache is introduced to avoid unnecessary LDAP traffic, for example caching usernames so they don't have to be looked up for every page, and speeding up loading of the Users page. Saving the configuration empties the cache. The time is given in seconds.

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

Examples:

- ten minutes: *600*
- one hour: *3600*

See the Caching section below for detailed information on how the cache operates.

Directory Settings

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

- Example: *displayName*

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

- Example:

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

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

The screenshot shows the 'Directory Settings' section of the LDAP Advanced Settings. It contains the following fields and values:

- 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`

At the bottom, there are buttons for 'Save', 'Test Configuration', and 'Help'.

Figure 4.3: LDAP Advanced Settings, section Directory Settings

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



The screenshot displays the 'Special Attributes' section of the LDAP Advanced Settings. It features four input fields: 'Quota Field', 'Quota Default', 'Email Field', and 'User Home Folder Naming Rule'. Below these fields are three buttons: 'Save', 'Test Configuration', and 'Help'.

Figure 4.4: 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*

4.3.3 Expert Settings

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

Server	User Filter	Login Filter	Group Filter	Advanced	Expert
--------	-------------	--------------	--------------	----------	--------

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.

[? Help](#)

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!

4.3.4 Testing the configuration

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



Figure 4.5: Failure

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

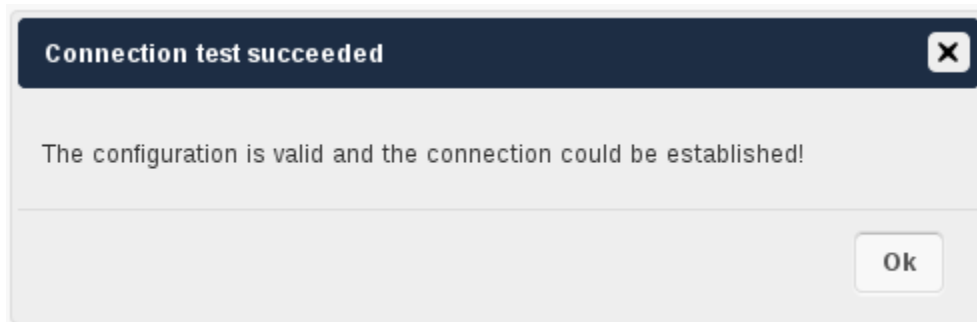


Figure 4.6: Success

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

4.3.5 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 4.7: 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.

4.3.6 Troubleshooting, Tips and Tricks

4.3.7 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)

4.3.8 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.

4.3.9 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.

4.3.10 ownCloud LDAP Internals

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

4.3.11 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.

4.3.12 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.

4.3.13 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.

4.4 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 servers. (Currently, this works only with OC7 and not older versions.) 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.

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.

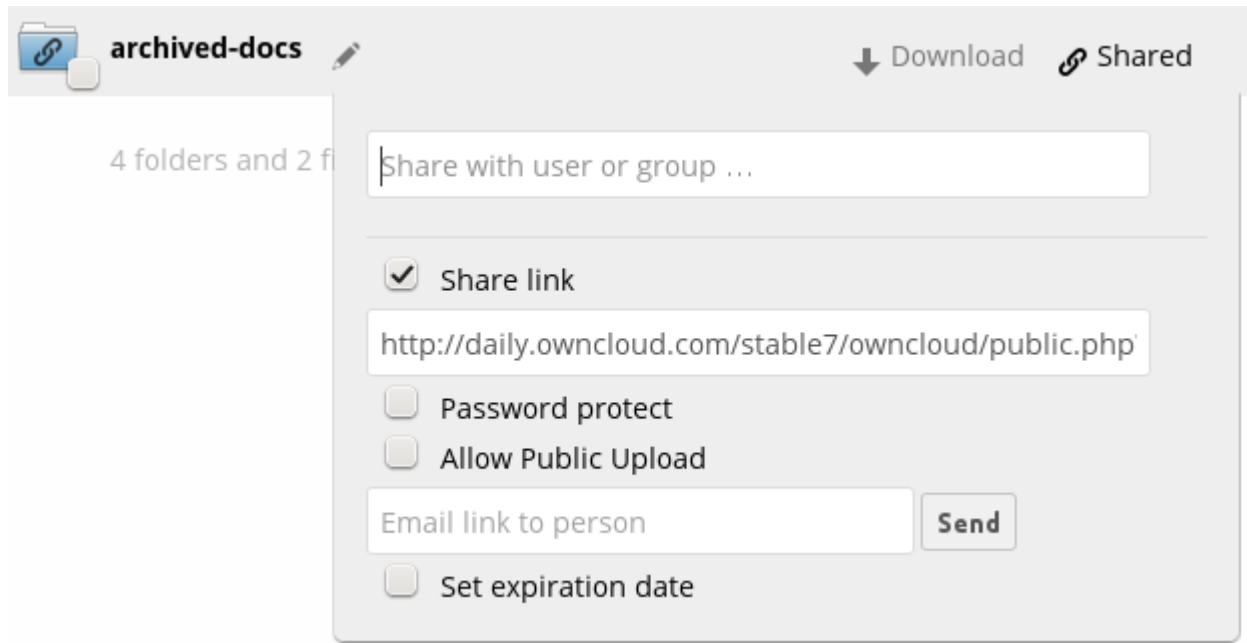
Your new public share is labeled with a chain link. If you do not protect it with a password, it is visible to anyone who has the URL. Users on other ownCloud 7 servers can mount it and use it just like any ownCloud share.

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.

4.4.1 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” in the Administrators Manual to learn more about `mod_rewrite`, SSL, and alternative HTTP servers. See “Installation Wizard” in the Administrators Manual to learn more about configuring trusted domains.)



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`.

4.5 Defining Background Jobs

A system like ownCloud sometimes requires tasks to be done on a regular basis without the need for user interaction or hindering ownCloud performance. For that purpose, as a system administrator, you can define background jobs (for example, database clean-ups) which are executed without any need for user interaction.

These jobs are typically referred to as *cron jobs*. Cron jobs are commands or shell-based scripts that are scheduled to run periodically at fixed times, dates, or intervals. `cron.php` is an ownCloud internal process that runs such background jobs on demand.

ownCloud plug-in applications register actions with `cron.php` automatically to take care of typical housekeeping operations, such as garbage collecting of temporary files or checking for newly updated files using `files_scan()` for externally mounted file systems.

4.5.1 Parameters

In the admin settings menu you can configure how cron-jobs should be executed. You can choose between the following options:

- AJAX
- Webcron
- Cron

4.5.2 Cron Jobs

You can schedule cron jobs in three ways – using AJAX, Webcron, or cron. The default method is to use AJAX. However, the recommended method is to use cron. The following sections describe the differences between each method.

AJAX

The AJAX scheduling method is the default option. Unfortunately, however, it is also the least reliable. Each time a user visits the ownCloud page, a single background job is executed. The advantage of this mechanism is that it does not require access to the system nor registration with a third party service. The disadvantage of this mechanism, when compared to the Webcron service, is that it requires regular visits to the page for it to be triggered.

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 must be able to access 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. For example:

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
```

Note: Please refer to the `crontab` man page for the exact command syntax.

4.6 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.

4.7 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

`Disable`

☒ 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.

4.8 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.

4.8.1 Parameters

```
<?php
```

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

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

4.9 Using Third Party 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.

4.9.1 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",
```

4.10 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.

4.10.1 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.

4.10.2 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

Enable

4.10.3 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

Stream Length bytes

Action for infected files found while scanning **Only log**

Save

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/clamdctl
```

Antivirus Configuration

Mode **Daemon (Socket)**

Socket

Stream Length bytes

Action for infected files found while scanning **Only log**

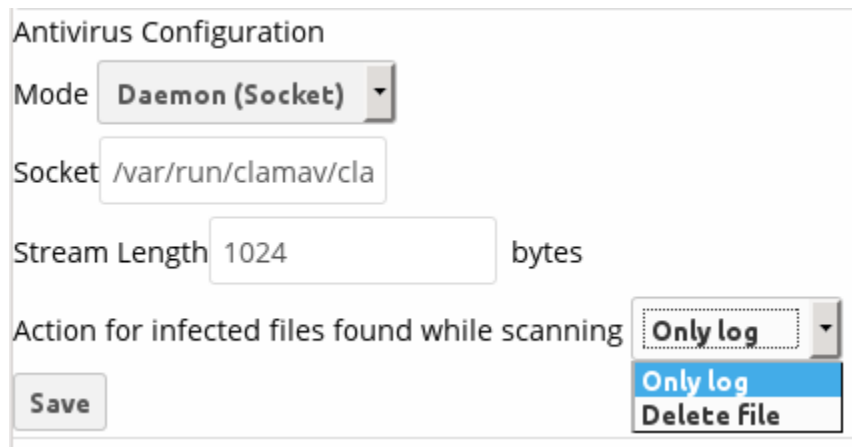
Save

Only log
Delete File

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

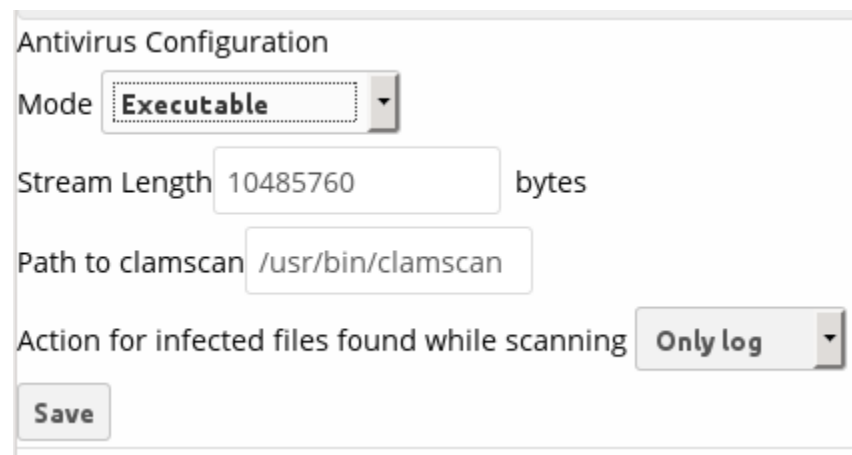
Mode **Daemon (Socket)**

Socket

Stream Length 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.



Antivirus Configuration

Mode **Executable**

Stream Length bytes

Path to clamscan

Action for infected files found while scanning **Only log**

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.

4.11 Defining Automatic Configuration

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.

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.

4.11.1 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

4.11.2 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*.

4.12 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

4.12.1 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",
```

4.13 Database Configuration

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

- MySQL / MariaDB <<https://mariadb.org/>>
- SQLite <<http://www.sqlite.org/>>
- 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 has some difficulties handling multiple users, we recommend that it be used only for single user ownCloud installations.

4.13.1 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.

4.13.2 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
```

[PostgreSQL]

```
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`.

4.13.3 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 system you're running, the command from, use:

```
mysql -uUSERNAME -p
```

To access a MySQL installation on a different machine, add the -h option with the respective host name:

```
mysql -uUSERNAME -p -h HOSTNAME
```

```
mysql> SHOW VARIABLES LIKE "version";  
+-----+-----+  
| Variable_name | Value |  
+-----+-----+  
| version       | 5.1.67 |  
+-----+-----+  
1 row in set (0.00 sec)  
mysql> quit
```

PostgreSQL:

Assuming the database server is installed on the same system you're running the command from, use:

```
psql -Uusername -downcloud
```

To access 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
```

4.14 Using Server-Side Encryption

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.

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.

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 protect users from snoopers ownCloud admins. 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 details.

4.14.1 Enabling the Encryption App

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

Encryption 0.6.1 **Internal App**

The ownCloud files encryption system provides server side-encryption keys. Please note that server side encryption requires the encryption of files that are stored on externally mounted storage.

Documentation: [User Documentation](#), [Admin Documentation](#)

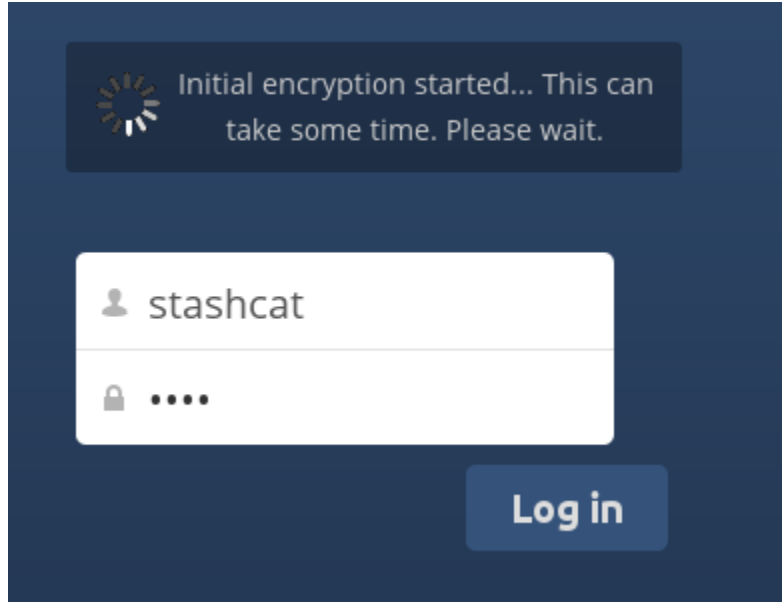
AGPL-licensed by Sam Tuke, Bjoern Schiessle, Florin Peter

Enable

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 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.



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 still change their passwords whenever they want on their Personal pages.

4.14.2 Sharing Encrypted Files

Only users who have private encryption keys (via logging out and logging back in to create their 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.

4.14.3 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".

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

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

 Log-in password

Decrypt all 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.

Restore Encryption Keys

Delete Encryption Keys

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. Though it would be very unusual for the decryption to fail as decryption is routine and reliable.

4.14.4 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. Obviously, do not lose this password.

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

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

Encryption

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

<input type="password" value="....."/>	Recovery key password
<input type="password" value="....."/>	Repeat Recovery key password
<input checked="" type="radio"/> Enabled	
<input type="radio"/> Disabled	

Encryption

Enable password recovery:

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

- ☒ Enabled
- ☐ Disabled

File recovery settings updated

Admin Recovery Password		
Group Admin	Quota	Storage Location
Group Admin ▼	Default ▼	/var/www/owncloud.

Enter the recovery password in order to recover the users files during password change

4.14.5 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.

4.14.6 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.

4.15 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.

4.15.1 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

4.16 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.

4.16.1 Parameters

```
<?php
```

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

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

4.17 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.

4.17.1 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",
```

4.18 Mail 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...

The screenshot shows the 'Email Server' configuration wizard. It includes the following fields and options:

- Send mode:** A dropdown menu with 'smtp' selected.
- Encryption:** A dropdown menu with 'TLS' selected.
- From address:** A dropdown menu with 'php', 'smtp', and 'sendmail' options. 'smtp' is currently selected.
- Authentication method:** A dropdown menu with 'Login' selected.
- Authentication required:** A checkbox that is checked.
- Server address:** A text input field containing 'smtp.alrac.net'.
- Port:** A text input field.
- Credentials:** A text input field containing 'login'.
- Test email settings:** A button labeled 'Send email'.

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.

4.18.1 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

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 **None** : Port

Credentials **Login** **Plain** **NT LAN Manager**

Test email settings **Send email**

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

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

```
--
ownCloud
web services under your control
```

4.18.2 Configuring PHP and Sendmail

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

How do you decide which one to use? PHP mode uses your local `sendmail` binary. Use this if you want to use `php.ini` to control some of your mail server functions, such as setting paths, headers, or passing extra command options to the `sendmail` binary. These vary according to which server you are using, so consult your server's documentation to see what your options are.

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

Email Server

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

Send mode

From address @

Test email settings

4.18.3 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.

4.18.4 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
```

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

or

```
<?php
```

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

If a malware or SPAM scanner is running on the SMTP server it might be necessary that you increase the SMTP timeout to e.g. 30s:

```
<?php
```

```
"mail_smtptimeout" => 30,
```

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

```
<?php
```

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

If the SMTP server only accepts secure connections you can choose between the following two variants:

SSL

A secure connection will be initiated using the outdated SMTPS protocol which uses the port 465/tcp:

```
<?php
```

```
"mail_smtphost"    => "smtp.server.dom:465",
"mail_smtpsecure"  => 'ssl',
```

TLS

A secure connection will be initiated using the STARTTLS protocol which uses the default port 25/tcp:

```
<?php
```

```
"mail_smtphost"    => "smtp.server.dom",  
"mail_smtpsecure"  => 'tls',
```

And finally it is necessary to configure if the SMTP server requires authentication, if not, the default values can be taken as is.

```
<?php
```

```
"mail_smtpauth"    => false,  
"mail_smtpname"    => "",  
"mail_smtppassword" => "",
```

If SMTP authentication is required you have to set the required username and password and can optionally choose between the authentication types **LOGIN** (default) or **PLAIN**.

```
<?php
```

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

PHP mail

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

```
<?php
```

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

Sendmail

If you want to use the well known Sendmail program to send email, it is necessary to have an installed and working email system on your *nix server. The sendmail binary (**/usr/sbin/sendmail**) is usually part of that system. ownCloud should be able to send email out of the box.

```
<?php
```

```
"mail_smtpmode"    => "sendmail",  
"mail_smtphost"    => "127.0.0.1",  
"mail_smtpport"    => 25,  
"mail_smtptimeout" => 10,  
"mail_smtpsecure"  => "",  
"mail_smtpauth"    => false,
```

```
"mail_smtpauthtype" => "LOGIN",
"mail_smtpname"      => "",
"mail_smtppassword" => "",
```

qmail

If you want to use the qmail program to send email, it is necessary to have an installed and working qmail email system on your server. The sendmail binary (`/var/qmail/bin/sendmail`) will then be used to send email. ownCloud should be able to send email out of the box.

```
<?php
```

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

4.18.5 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.

4.18.6 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 smtp
- 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.

```

4.18.7 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
```

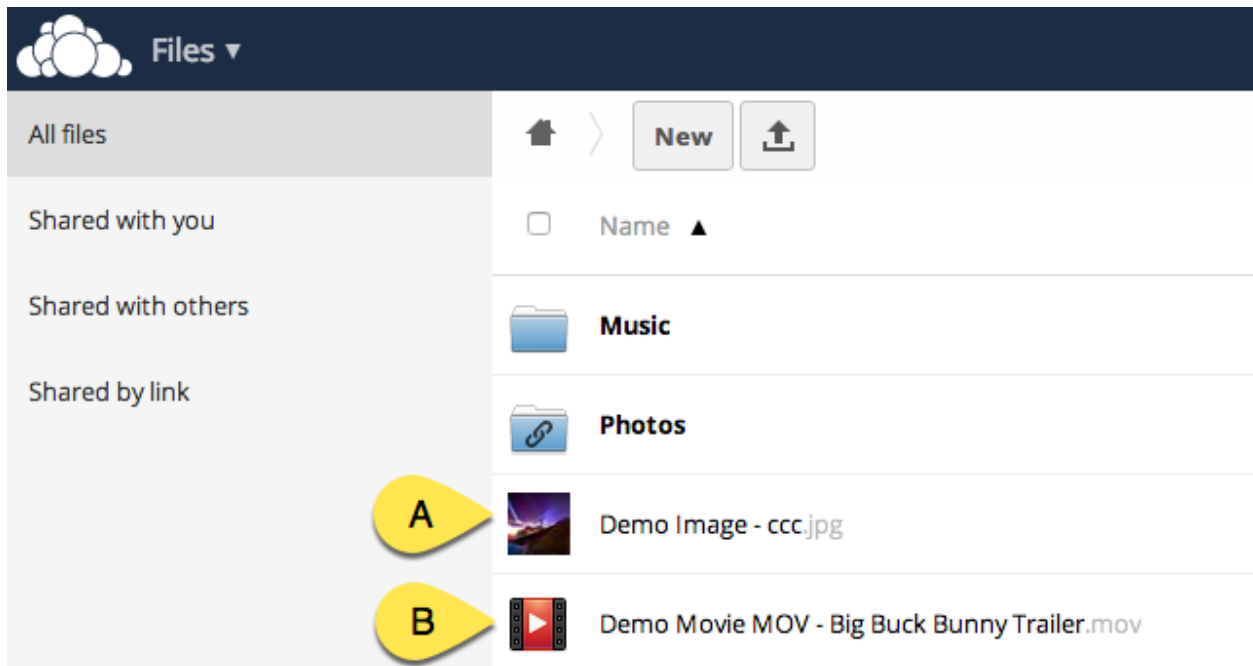
```
"mail_smtpdebug" => true,
```

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.

4.19 Preview 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.

4.19.1 Parameters

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

Disabling previews:

Under certain circumstances, for example if the server has 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,
```


4.20 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.

4.20.1 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.

4.20.2 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.

4.21 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 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 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

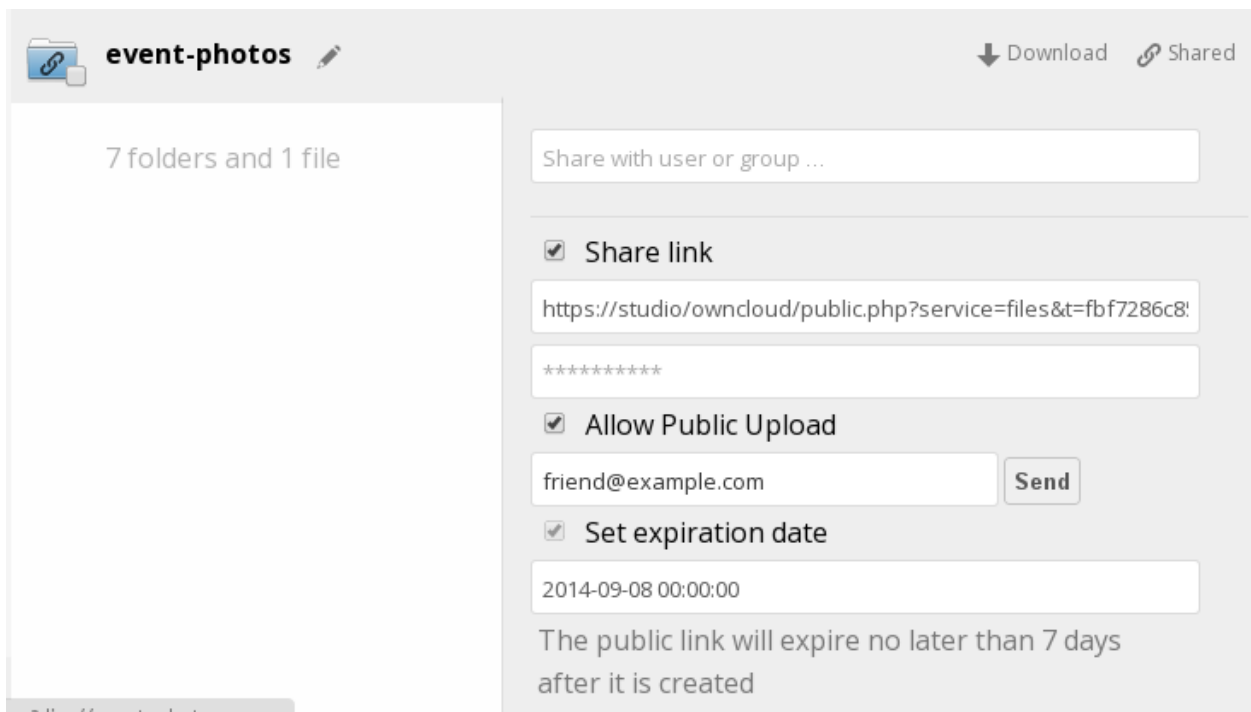
☒ Exclude groups from sharing

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

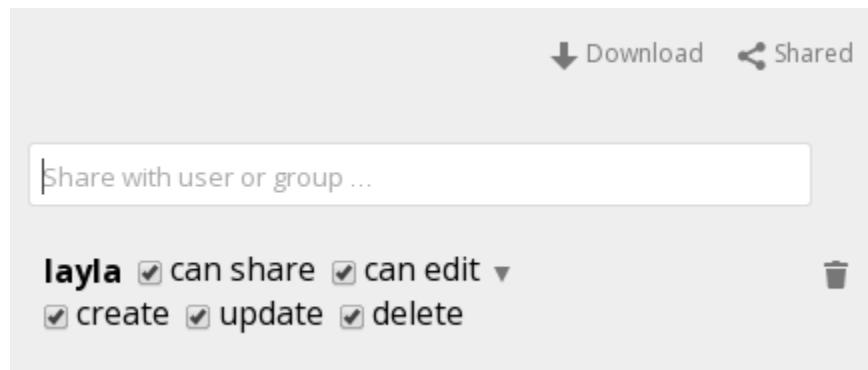
- Check `Allow apps to use the Share API` to enable users to share files. If this is not checked, no users can create file shares
- Check `Allow users to share via link` to enable creating public shares for people who are not ownCloud users. This creates a hyperlink, just like a Web page, so your ownCloud 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:



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



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

4.21.1 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 minimum `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.

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

It’s useful to know limiting factors that make it impossible to exceed the values given by the ownCloud-system:

4.22.1 Not outnumberable upload limits:

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

4.22.2 Other recommendable preconditions:

- Make sure that the latest version of PHP (at least 5.4.9) is installed
- Disable user quota. This means: set the user quota of the account, you are currently logged in, to “unlimited”.

This is important, because you possibly could not watch otherwise whether the desired changes take effect.

4.23 Enabling uploading big files

Configuring your webserver

ownCloud comes with a `.htaccess` - file which propagates all config to your webserver. To adapt those settings go to the ownCloud - Folder on your server and set the following two parameters inside the `.htaccess` file:

- `upload_max_filesize = 16G` (e.g., to stay consistent with the example value above)
- `post_max_size = 16G` (e.g., to stay consistent with the example value above)

If you don't want to use the shipped `.htaccess` - file, outcomment those options there and edit them in your global `php.ini` file:

You can easily learn the loaded configuration file by saving `<?php phpinfo(); ?>` code piece into a `php` file and calling it with your browser. Then look for the **Loaded Configuration File** value.

Alternatively:

- Under Debian or SUSE and their derivatives this file lies at `/etc/php5/apache2/php.ini`
- On Windows, you can find this file within `C:/Program Files (x86)/PHP/PHP.ini`

Set the following two parameters inside the `php.ini` to the same value as chosen inside the admin-section one step before:

- `upload_max_filesize = 16G` (e.g., to stay consistent with the example value above)
- `post_max_size = 16G` (e.g., to stay consistent with the example value above)

Output Buffering must be turned off in `.htaccess` or `php.ini`, or PHP will return memory-related errors.

- `output_buffering = 0`

These client configurations have been proven by testing maximum file sizes of 16 GB:

- Linux 32 Bit: Ubuntu, Firefox => 16GB
- Windows 8 64 Bit: Google Chrome => 8GB

Note: You will need a minimum of 16GB (e.g., to stay consistent with the example value above), in your `upload_tmp_dir`. Normally this points to `/tmp`. If your `/tmp` has not enough space, you can change the value of `upload_tmp_dir` in your `php.ini`

4.24 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.

4.24.1 Supported mounts

ownCloud admins may mount these external storage services and devices:

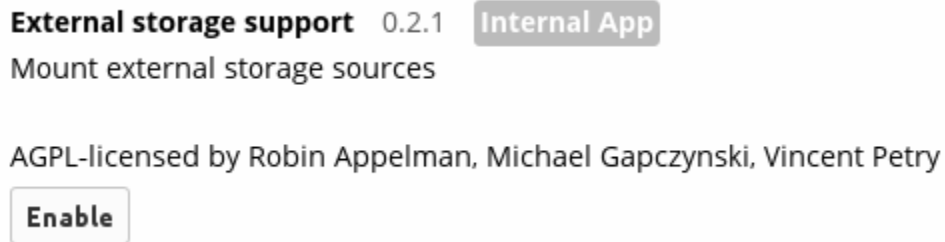
- Local
- Amazon S3 and S3 compliant
- Dropbox
- FTP/SFTP
- Google Drive
- OpenStack Object Storage
- SMB/CIFS
- SMB/CIFS using OC login
- ownCloud

- WebDAV

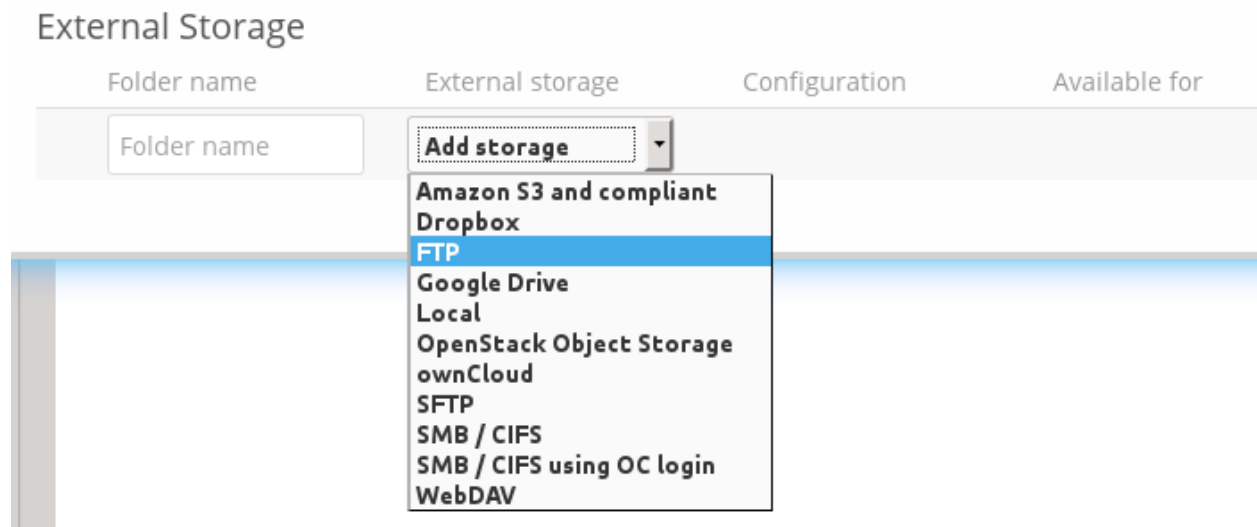
ownCloud users can be given permission to mount any of these, except local storage.

4.24.2 Enabling External Storage Support

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



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



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

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

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

4.24.3 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.

- ☒ 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

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
	<input type="text" value="Local"/>	Local	<input type="text" value="/shared/projects"/>	<input type="text" value="All Users x"/>

4.24.4 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-AWS compatible servers.

4.24.5 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
		AKIAIOSHDCA77WFI	
		
		oc-files-wc	
<div><div></div><div>AmazonS3</div></div>	Amazon S3 and compliant	Hostname (optional)	All Users x
		Port (optional)	
		Region (optional)	
		<input checked="" type="checkbox"/> Enable SSL <input checked="" type="checkbox"/>	
		Enable Path Style	



[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?

 <p>Drop-ins app Chooser or Saver</p>	 <p>Dropbox API app Sync API, Datastore API, or Core API</p>
---	--

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.

Create app

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

Settings	Details	App metrics	Error logs				
Status	Development	<button>Apply for production</button>					
Development users	1 / 100	<button>Unlink all users</button>					
Permission type	Full Dropbox						
App key	rt4[redacted]						
App secret	md[redacted]						
OAuth 2	Redirect URIs <table border="1"> <tbody> <tr> <td>http://localhost/owncloud/index.php/settings/personal</td> <td>×</td> </tr> <tr> <td>http://localhost/owncloud/index.php/settings/admin</td> <td>×</td> </tr> </tbody> </table>			http://localhost/owncloud/index.php/settings/personal	×	http://localhost/owncloud/index.php/settings/admin	×
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:

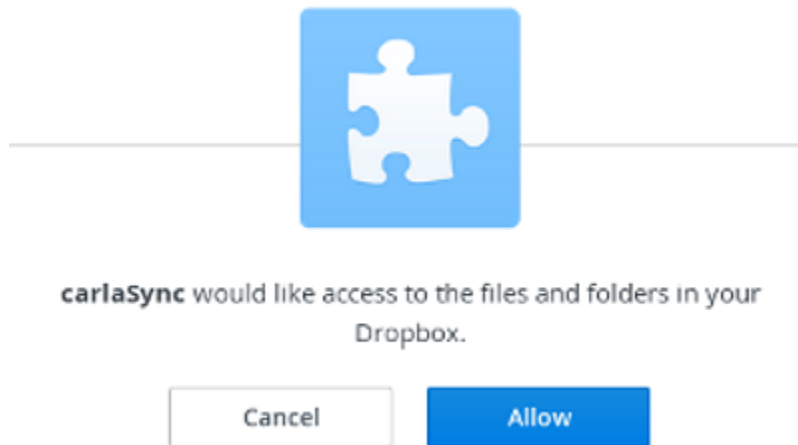
```
https://localhost/owncloud/index.php/settings/personal
https://localhost/owncloud/index.php/settings/admin

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	rt4[redacted]		<button>All Users ×</button>
			Access granted		

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.



4.24.6 FTP/FTPS/SFTP

Connecting to an FTP server requires:

- Whatever name you want for your local mountpoint.
- The URL of your FTP server.
- 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.

 A screenshot of the ownCloud configuration interface for an external storage. On the left is a green circular status indicator. Next to it is a text input field containing "FTP". To the right of this field is the label "FTP". Further right is a large light-grey area containing several input fields: "ftp.example.com", "username", a password field with ".....", and "public_html/". Below these is a checkbox labeled "Secure ftps://" which is checked. To the right of the password field is a button labeled "All Users x".

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.

4.24.7 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.

Create Project

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 OAuth 2.0 Client ID` button.

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`.

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.

Create

Cancel

Drive API



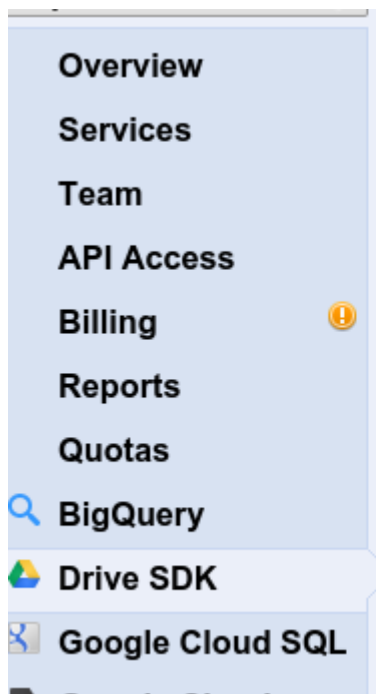
0%

ON

Drive SDK



ON



Google Drive SDK

App ID: 695362171651

Application Info

Application Name

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

Short Description

A one-line description of your application.

Overview
Services
Team
API Access
Billing
Reports
Quotas
BigQuery
Drive SDK

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](#)

Create an OAuth 2.0 client ID...

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
```

Create client ID Back Cancel

[Le](#)

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:

Client ID for web applications

Client ID:	373568241109-r6e6.apps.googleusercontent.com	Edit settings
Email address:	373568241109-developer@developer.gserviceaccount.com	Reset email address
Client secret:	unusual-phantom	Download client secret
Redirect URIs:	https://example.com/owncloud/index.php/settings/personal https://example.com/owncloud/index.php/settings/admin	
JavaScript origins:	none	

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`.

Folder name	External storage	Configuration	Available for
 <input type="text" value="GoogleDrive"/>	Google Drive	<input type="text" value="373568241109-r6e6"/> <input type="text" value="....."/> Access granted	<input type="text" value="All Users x"/>

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

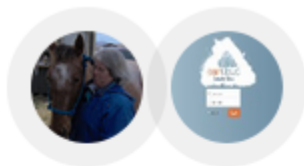
4.24.8 SMB/CIFS

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 `smclient` 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 user name 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 `/`)

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



▼ 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.

Cancel

Accept

External Storage

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

4.24.9 SMB/CIFS using OC login

This works the same way as setting up a SMB/CIFS mount, except you can use your ownCloud logins instead 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.

4.24.10 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)

The screenshot shows a configuration window for mounting a WebDAV server. On the left, there is a green status indicator and a text input field containing 'oc-remote'. To its right is the label 'ownCloud'. Further right, there are four stacked text input fields: the first contains 'https://remoteserve', the second contains 'admin', the third contains a series of dots representing a password, and the fourth contains a single character. To the right of these fields is a button labeled 'All Users x'. Below the input fields is a checkbox labeled 'Secure https://' which is currently unchecked.

4.24.11 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

4.24.12 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.

4.25 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

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!

4.25.1 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
      }
    }
  }
}
```

4.25.2 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

4.25.3 Backends

Local Filesystem

The local filesystem backend mounts a folder on the server into the virtual filesystem, the class to be used is `\OCFilesStorageLocal` 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
- **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",
    "user": "johndoe",
    "password": "secret",
    "root": "\\Videos",
    "secure": "false"
  },
  "priority": 100
}
```

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

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.

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 user name used 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 ‘/’)

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

Example

```
{
  "class": "\\OC\\Files\\Storage\\SMB",
  "options": {
    "host": "myhost.com",
    "user": "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
}
```

4.26 Custom User Backend Configuration

Starting with ownCloud 4.5 is possible to 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 => ...
        ),
    ),
),
```

Currently the “External user support” (`user_external`) app provides the following user backends:

4.26.1 IMAP

Provides authentication against IMAP servers

- **Class:** `OC_User_IMAP`
- **Arguments:** a mailbox string as defined [in the PHP documentation](#)
- **Example:**

```
<?php
```

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

4.26.2 SMB

Provides authentication against Samba servers

- **Class:** OC_User_SMB
- **Arguments:** the samba server to authenticate against
- **Example:**

```
<?php

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

FTP

Provides authentication against FTP servers

- **Class:** OC_User_FTP
- **Arguments:** the FTP server to authenticate against
- **Example:**

```
<?php

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

4.27 Serving static files via web server

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.

4.27.1 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

4.27.2 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

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

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

```

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

4.27.4 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.

4.28 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**

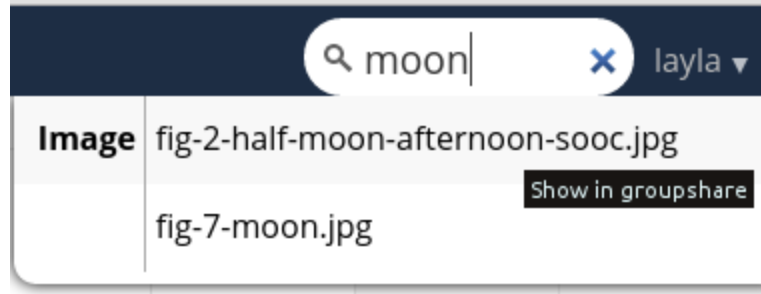
Activate this app when you want to be able to search for documents in your ownCloud. The index is searched, so you will not be able to search for documents that have been written before ownCloud will index it in the background. After enabling the app, you will be able to retrieve file search results for Office documents. Office support is on the roadmap.

AGPL-licensed by Jörn Dreyer

Disable

When you want to find a document, enter your search term in the search field at the upper right of your ownCloud Web interface. You can run a search from any ownCloud page. Hover your cursor over any of your search results to see what folder it is in, or click on the filename and it takes you to the folder.

Known limitations



It does not work with the Encryption app, because the background indexing process does not have access to the key needed to decrypt files when the user is not logged in.

Not all PDF versions can be indexed because its text extraction may be incompatible with newer PDF versions.

4.29 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`.

4.29.1 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:8080',
    ),
```

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.

```
'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: `sqlite`, `mysql`, `pgsql`, `oci`, or `mssql`.

```
'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.

4.29.2 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 PostgreSQL.

```
<?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' => '67336bcd7630dd80b2b81a413d07',
    'installed' => true,
);
```

4.29.3 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' => '',
```

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 allows you to configure alternate authentication backends. Supported backends are IMAP (OC_User_IMAP), SMB (OC_User_SMB), and FTP (OC_User_FTP).

4.29.4 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_smtppsecure' => '',
```

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

```
'mail_smtppauth' => false,
```

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

```
'mail_smtppauthtype' => '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_smtppauth`. Specify the username for authenticating to the SMTP server.

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

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

4.29.5 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`.

4.29.6 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.

4.29.7 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.

4.29.8 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_size` value.

4.29.9 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.

4.29.10 Apps

Options for the Apps folder, Apps store, and App code checker.

```
'appstoreenabled' => true,
```

When enabled, admins may install apps from the ownCloud app store.

```
'appstoreurl' => 'https://api.owncloud.com/v1',
```

The URL of the appstore to use.

```
'apps_paths' => array(
    array(
        'path' => '/var/www/owncloud/apps',
        'url' => '/apps',
        'writable' => true,
    ),
),
```

Use the `apps_paths` parameter to set the location of the Apps directory, which should be scanned for available apps, and where user-specific apps should be installed from the Apps store. The `path` defines the absolute file system path to the app folder. The key `url` defines the HTTP web path to that folder, starting from the ownCloud web root. The key `writable` indicates if a web server can write files to that folder.

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

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.

4.29.11 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_libreoffice_path' => '/usr/bin/libreoffice',
```

custom path for LibreOffice/OpenOffice binary

```
'preview_office_cli_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:

- OCPreviewImage
- OCPreviewMarkDown
- OCPreviewMP3
- OCPreviewTXT

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

- OCPreviewMovies
- OCPreviewMSOffice2003
- OCPreviewMSOffice2007
- OCPreviewMSOfficeDoc
- OCPreviewOpenDocument
- OCPreviewPDF
- OCPreviewStarOffice
- OCPreviewSVG

4.29.12 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.

4.29.13 SSL

```
'forcesssl' => 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.

4.29.14 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.

```
'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 use 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)
- mysql (MySQL)
- pgsql (PostgreSQL)
- oci (Oracle)
- mssql (Microsoft SQL Server)

```
'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

MAINTENANCE

5.1 Maintenance Mode Configuration

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.

5.1.1 Parameters

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

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

5.2 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

5.2.1 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``/
```

5.2.2 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 +%Y%m%d`.bak
```

SQLite

```
sqlite3 data/owncloud.db .dump > owncloud-sqlbkp_`date +%Y%m%d`.bak
```

PostgreSQL

```
PGPASSWORD="password" pg_dump owncloud -h [server] -U [username] -f owncloud-sqlbkp_`date +%Y%m%d`.bak
```

5.3 Updating ownCloud with the Updater App

The Updater app automates many of the steps of updating an ownCloud installation to the next point release. The Updater app should be enabled in your ownCloud instance by default, which you can easily confirm by looking on your Apps page.

Updating and upgrading your ownCloud installation are two different tasks. Updating means updating to the next point release, which is indicated by third digit of the version number. For example, 4.5.1, 5.0.17, 6.0.4 and 7.0.1 are point releases. (Look at the bottom of your Admin page to see your version number.)

Major releases are indicated by the first and second digits. So 4.5.0, 5.0.0, 6.0.0, and 7.0.0 are major releases. The Updater app is not for upgrades; please see [Upgrading Your ownCloud Server](#) for instructions on upgrading to a major release.

If you installed ownCloud from your Linux distribution repositories using your package manager, then it is best to update/upgrade ownCloud using your package manager and staying in sync with your distro updates, rather than using the Updater app or upgrading manually. You should still maintain regular backups (see [Backing up ownCloud](#)), and make a backup before every update/upgrade.

Note: If you have a large ownCloud installation you should not use the Updater app, in order to avoid PHP timeouts. The Updater app is better for smaller installations that have less data and fewer users, and for admins who do not have shell access, for example on shared hosting. Larger installations should update ownCloud with their Linux package managers or manually upgrade, and then complete the update with the `occ upgrade` command, which is in the `owncloud` directory. Run `occ help` to see complete command options. ownCloud Enterprise Edition does not include the Updater app.

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; on a stock Linux installation this is the `www-data` or `apache` user on systems that are running the Apache HTTP server. 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 this single line in it:

```
<?php echo exec('whoami'); ?>
```

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

4. Navigate to your 'Admin' page and click the 'Update Center' button under Updater:

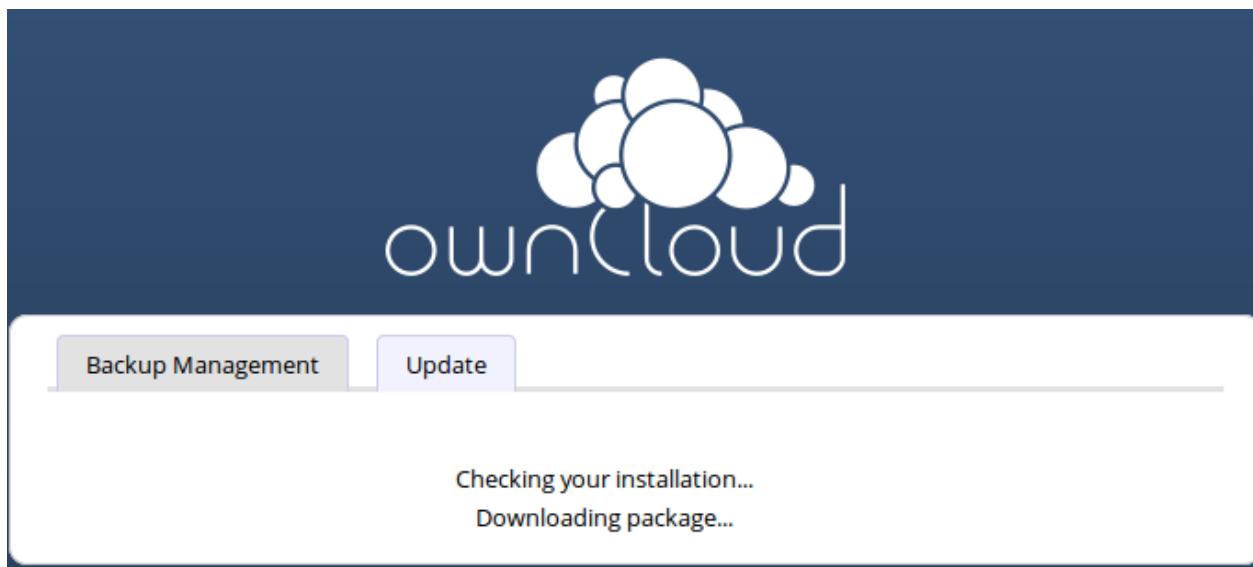
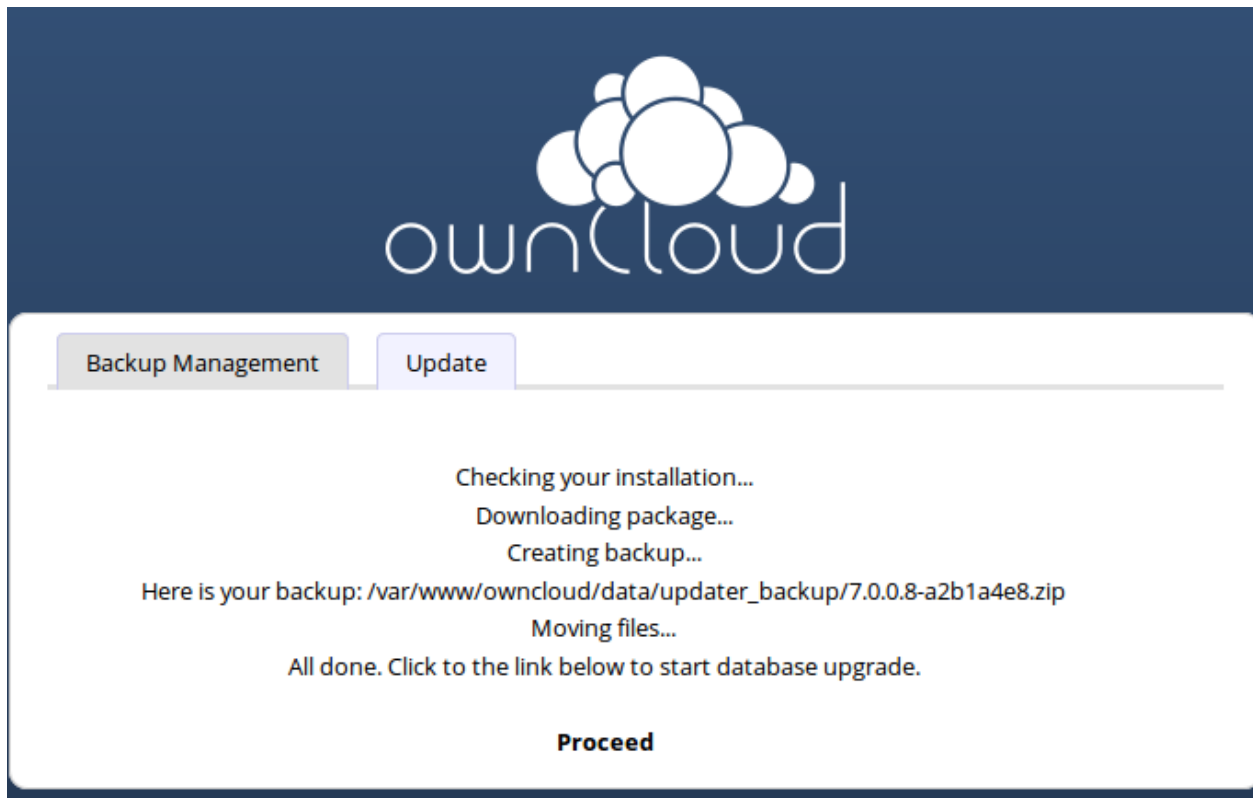
Updater
Update Center

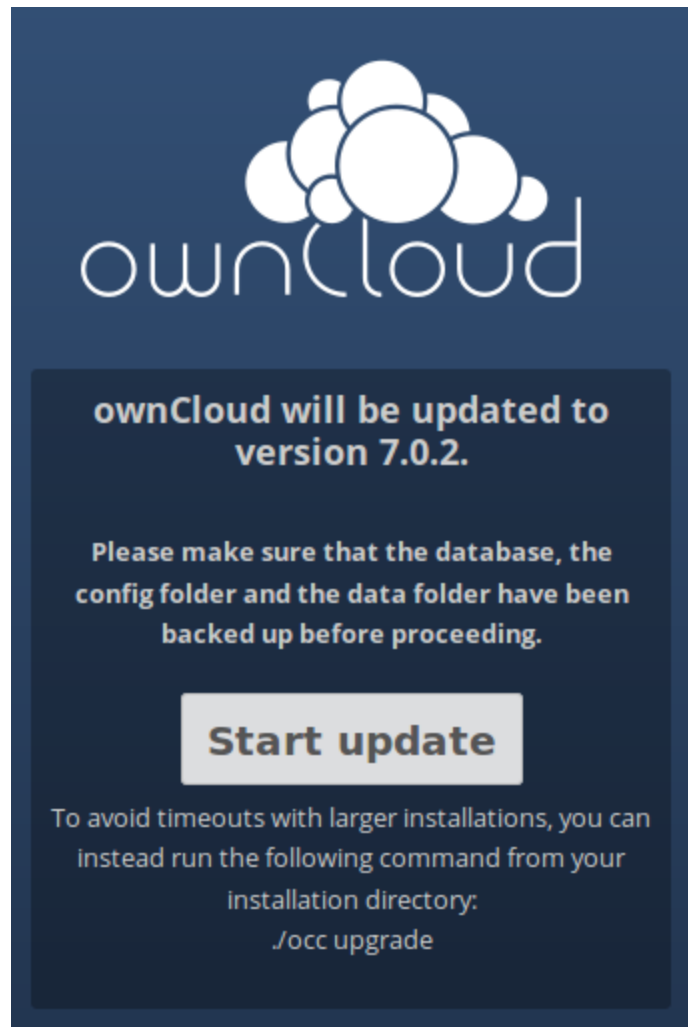
5. This takes you to the Updater control panel.

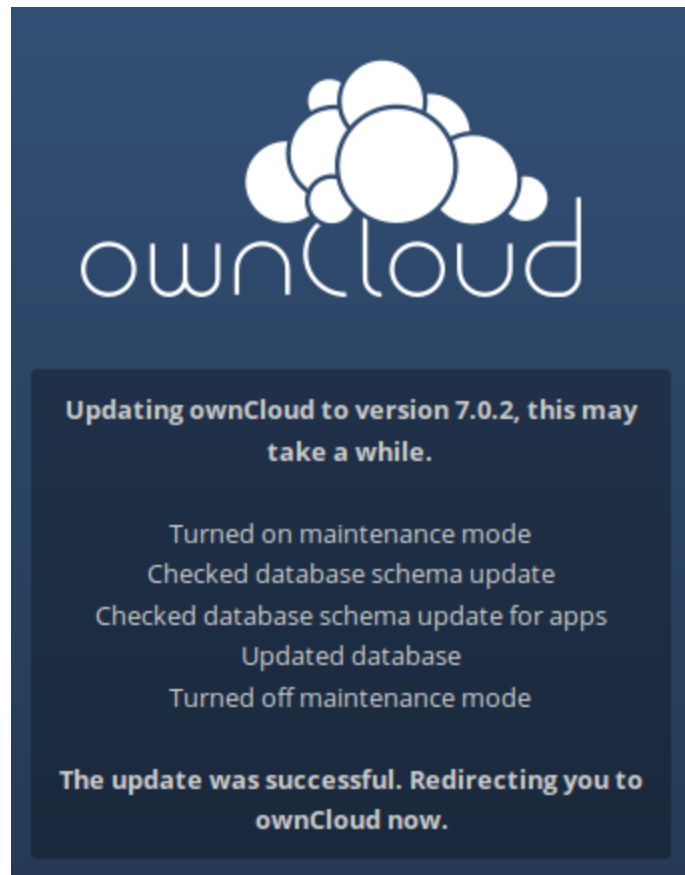


6. Click Update, and carefully read the messages. If there are any problems it will tell you, otherwise you will see a message about checking your installation, making a backup, and moving files:
8. Click Proceed, and then it downloads the updates, which may take a few minutes:
7. The Update app wants you to be very sure you want to update, and so you must click one more button, the Start Update button:
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.







If the Update app fails, then you must update manually. See [Upgrading Your ownCloud Server](#) to learn how to upgrade manually.

5.4 Upgrading Your ownCloud Server

Updating and upgrading your ownCloud installation are two different tasks. Updating means updating to the next point release, which is indicated by the third digit of the version number. For example, 4.5.1, 5.0.17, 6.0.4 and 7.0.1 are point releases. (Look at the bottom of your Admin page to see your version number.)

Major releases are indicated by the first and second digits. So 4.5.0, 5.0.0, 6.0.0, and 7.0.0 are major releases. You may use the Updater app for staying current with new point releases (Community Edition only), but not for upgrading to a major release. Please see [Upgrading ownCloud with the Updater App](#) for instructions on using the Updater app.

You cannot skip major releases; for example, upgrading from 5.0 to 7.0. This is unsupported, and you'll likely experience unpredictable results. It is best to install all upgrades and updates in order.

Note: If you installed ownCloud from your Linux distribution repositories using your package manager, then it is best to update/upgrade ownCloud using your package manager and staying in sync with your distro updates, rather than using the Updater app or upgrading manually. You should still maintain regular backups (see [Backing up ownCloud](#)), and make a backup before every update/upgrade.

5.4.1 Manual Upgrade Procedure

Start by putting your server in maintenance mode. Do this by entering your `config.php` file and changing `'maintenance' => false`, to `'maintenance' => true`,. This prevents new logins, and logged-in users can't make any further requests.

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	<code>apachectl stop</code>
Debian or Ubuntu	<code>/etc/init.d/apache2 stop</code>
SUSE Enterprise Linux 11 openSUSE 12.3 and up	<code>/usr/sbin/rcapache2 stop</code> <code>systemctl stop apache2</code>

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)	<ol style="list-style-type: none">1. Open IIS Manager and navigate to the web server node in the tree.2. In the Actions pane, click Stop.
Command Line	<ol style="list-style-type: none">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. If you have your `data/` directory already stored in another location, you can skip this step. If you want to do so, now is a good time to change the location of your `data/` directory. See the “Advanced Options” chapter in *Installation Wizard* for more information about changing the default database or data 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	<code>apachectl start</code>
Debian or Ubuntu	<code>/etc/init.d/apache2 start</code>
SUSE Enterprise Linux 11 openSUSE 12.3 and up	<code>/usr/sbin/rcapache2 start</code> <code>systemctl start apache2</code>

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)	<ol style="list-style-type: none"> 1. Open IIS Manager and navigate to the web server node in the tree. 2. In the Actions pane, click Stop.
Command Line	<ol style="list-style-type: none"> 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. Review the prerequisites, and if you have followed all the steps click the **Start Update** button.

If you are an enterprise customer, or are running a large installation with a lot of files and users, you should launch the update from the command line. The `occ` command is in your `owncloud/` directory, so on a typical Linux installation you could run this command:

```
php /var/www/owncloud/occ upgrade
```

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 review your third-party apps, and upgrade and enable them.

5.5 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

5.5.1 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/
```

5.5.2 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

```
sqlite3 data/owncloud.db .dump < owncloud-sqlbkp.bak
```

PostgreSQL

```
PGPASSWORD="password" pg_restore -c -d owncloud -h [server] -U [username] owncloud-sqlbkp.bak
```

5.6 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

5.7 Converting Database Type

You can convert a sqlite database to a more performing MySQL, MariaDB or PostgreSQL database with the ownCloud command line tool. Available since ownCloud version 7.0.0. To convert former ownCloud installations, first update to at least version 7.0.0.

5.7.1 Run the conversion

First setup the new database, here called “new_db_name”. In ownCloud root folder call

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

The Options

- `--port="3306"` the database port (optional)
- `--password="mysql_user_password"` password for the new database. If omitted the tool will ask you (optional)
- `--clear-schema` clear schema (optional)
- `--all-apps` by default, tables for enabled apps are converted, use to convert also tables of deactivated apps (optional)

Note: The converter searches for apps in your configured app folders and uses the schema definitions in the apps to create the new table. So tables of removed apps will not be converted even with option `--all-apps`

For example

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

To successfully proceed with the conversion, you must type `yes` when prompted with the question `Continue with the conversion?`

On success the converter will automatically configure the new database in your ownCloud config `config.php`.

5.7.2 Unconvertible Tables

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

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

ISSUES

If you think you have found a bug in ownCloud, please:

- Search for a solution
- Double check your configuration

If you can't find a solution, please file an issue:

- If the issue is with the ownCloud server, report it to the [GitHub core repository](#)
- If the issue is with the ownCloud client, report it to the [GitHub mirall repository](#)
- If the issue with with an ownCloud app, report it to where that app is developed
 - If the app is listed [here](#) report it to the correct repository
 - If the app is listed [in the apps repository](#) report it there

Please note that the mailing list should not be used for bug reports, as it is hard to track them there.

INDICES AND TABLES

- *genindex*