# ownCloud Administrators Manual

***Release 7.0***

**The ownCloud developers**

August 25, 2014

CONTENTS

CHAPTER

ONE

# INTRODUCTION

Welcome to the ownCloud Administrator Guide. This guide describes administrator tasks for ownCloud; a flexible, open source, file synchronization and sharing solution. ownCloud is comprised of a server running on either a Linux or Microsoft 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*
- *User Management*
- *Database Configuration*
- *User Authentication with LDAP*
- *Configuring Server-to-Server Sharing*
- *Custom Mount Configuration (GUI)*
- *Custom Mount Configuration*
- *Defining Background Jobs*
- *Mail Configuration*
- *Defining Automatic Configuration*
- *Use 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*

### 1.2.3 Maintenance

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

- *Maintenance Mode Configuration*
- *Migrating ownCloud Installations*
- *Updating ownCloud*

CHAPTER

TWO

# INSTALLATION

## 2.1 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.).

### 2.1.1 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

### 2.1.2 ownCloud on Hardware Appliances

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

- 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.

## 2.2 Linux Distributions

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

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

### Follow the wizard to complete your installation

For setting up your ownCloud instance after installation, please refer to the *Installation Wizard* section.

## 2.3 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.

## 2.4 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. For this reason, while ownCloud server will run on Windows, is not recommended at this time.

**Note:** You must move the data directory outside of your public root (See advanced install 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 support.

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.

### 2.4.1 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 you server. This console also enables you to change certificate options and manage items like file upload size.
8. Check the CGI checkbox under `Application Development Features` in order to enable PHP on IIS.
9. Turn off WebDAV publishing to avoid conflicts between the Windows WebDAV and the ownCloud WebDAV interface.

   ---

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

   ---

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

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

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

Continue by installing PHP.

#### Windows Server 2008

1. Navigate to *Start –> Control Panel –> Programs*.
2. Under Programs and Features, click the link titled *Turn Windows Features on and Off*.

   The Server Manager starts.
3. In the Server Manager, click `Roles`
4. Click `Add Roles`.
5. Use the *Add Roles Wizard* to add the web server role.

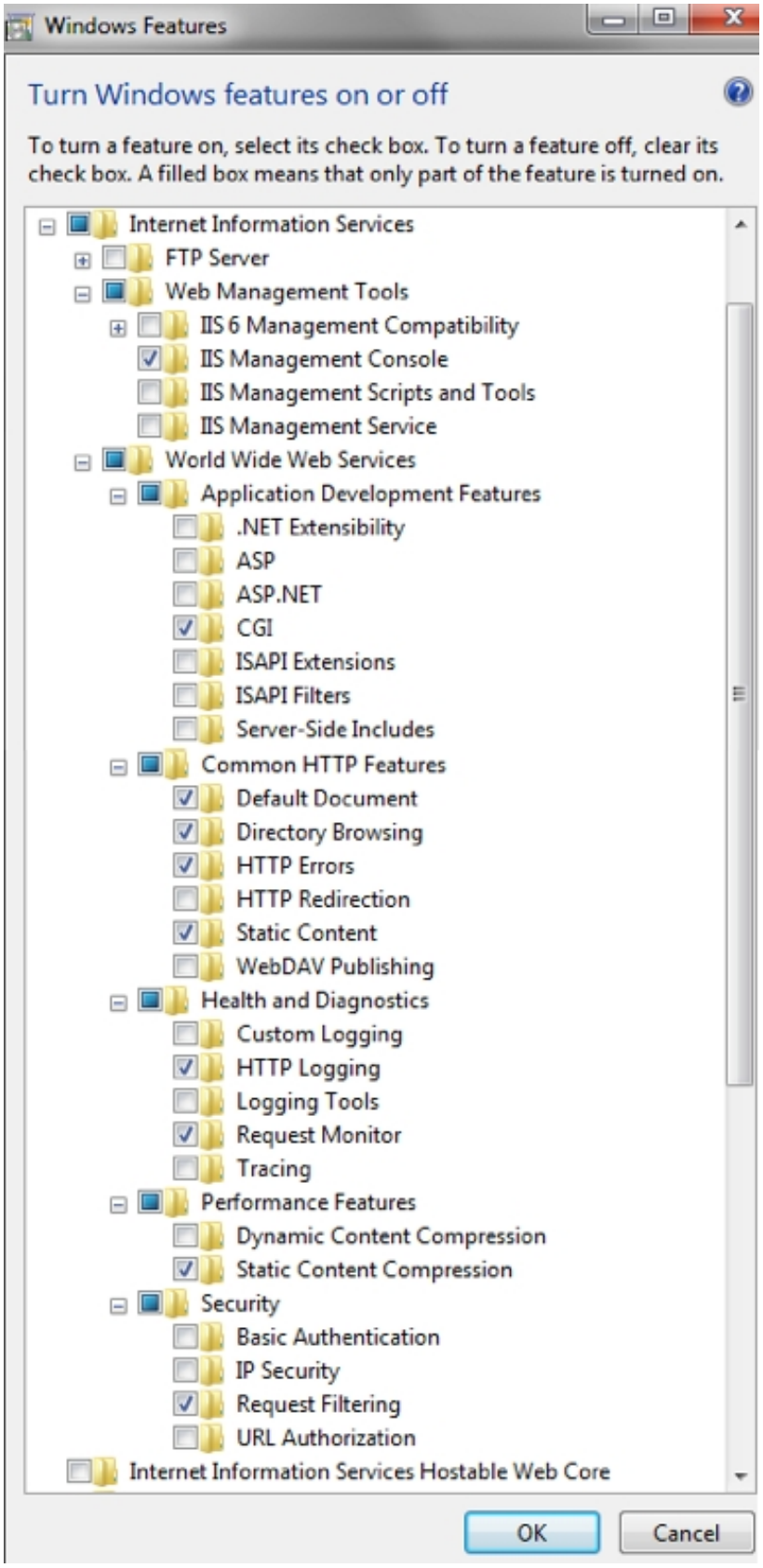



Figure 2.1: **Windows Features required for ownCloud on Windows 7**

**Role Services:** 40 installed

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

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.

### 2.4.2 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.

### 2.4.3 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.

> **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, this 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.

### 2.4.4 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.

### 2.4.5 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 syncrhonizing with the Desktop and Mobile Clients will fail.

To ensure your configuration is correct:

1. Open IIS Manager7.

2. In the *Connections* bar, select your site below *Sites*, or choose the top level entry if you want to modify the machine-wide settings.
3. Choose the *Handler Mappings* feature.
4. Click *PHP_via_fastCGI*.
5. Choose *Request Restrictions* and locate the *Verbs* tab.
6. Ensure *All Verbs* is checked.
7. Click *OK*.
7. Choose the *Request Filtering* feature from the IIS Manager.
8. Ensure that all verbs are permitted (or none are forbidden) in the *Verbs* tab.

> **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.

### 2.4.6 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.

## 2.5 Univention Corporate Server

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

### 2.5.1 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 2.1 – continued from previous page

| Key | Default | Description | Introduced |
|---|---|---|---|
| owncloud/ldap/loginFilter | (&(\|(&(objectClass=posixAccount) (objectClass=shadowAccount)) (objectClass=univentionMail) (objectClass=sambaSamAccount) (objectClass=simpleSecurityObject) (&(objectClass=person) (objectClass=organizationalPerson) (objectClass=inetOrgPerson))) (!(uidNumber=0)) (!(uid=*$)) (&(uid=%uid) (ownCloudEnabled=1))) | The LDAP filter that shall be used when a user tries to log in. | 2012.0.1 |
| owncloud/ldap/userlistFilter | (&(\|(&(objectClass=posixAccount) (objectClass=shadowAccount)) (objectClass=univentionMail) (objectClass=sambaSamAccount) (objectClass=simpleSecurityObject) (&(objectClass=person) (objectClass=organizationalPerson) (objectClass=inetOrgPerson))) (!(uidNumber=0))(!(uid=*$)) (&(ownCloudEnabled=1))) | The LDAP filter that shall be used when the user list is being retrieved (e.g. for sharing) | 2012.0.1 |
| owncloud/ldap/groupFilter | (&(objectClass=posixGroup) (ownCloudEnabled=1)) | The LDAP filter that shall be used when the group list is being retrieved (e.g. for sharing) | 2012.4.0.4 |
| owncloud/ldap/internalNameAttribute | uid | Attribute that should be used to create the user's owncloud internal name | 5.0.9 |
| owncloud/ldap/displayName | uid | The LDAP attribute that should be displayed as name in ownCloud | 2012.0.1 |
| owncloud/ldap/user/searchAttributes | uid,givenName,sn,description,employeeNumber,mailPrimaryAddress | Attributes taken into consideration when searching for users (comma separated) | 5.0.9 |
| owncloud/ldap/user/quotaAttribute | ownCloudQuota | Name of the quota attribute. The default attribute is provided by owncloud-schema. | 5.0.9 |
| owncloud/ldap/user/homeAttribute | (empty) | Attribute that should be used to create the user's owncloud internal home folder | 5.0.9 |
| owncloud/ldap/group/displayName | cn | The LDAP attribute that should be used as groupname in ownCloud | 2012.4.0.4 |
| owncloud/ldap/group/searchAttributes | cn,description, mailPrimaryAddress | Attributes taken into consideration when searching for groups (comma separated) | 5.0.9 |
| owncloud/join/users/update | yes | Wether ownCloud LDAP schema should be applied to existing users | 2012.0.1 |
| owncloud/group/enableDomainUsers | 1 | Wether the group "Domain Users" shall be enabled for ownCloud on install | 2012.4.0.4 |

Continued on next page

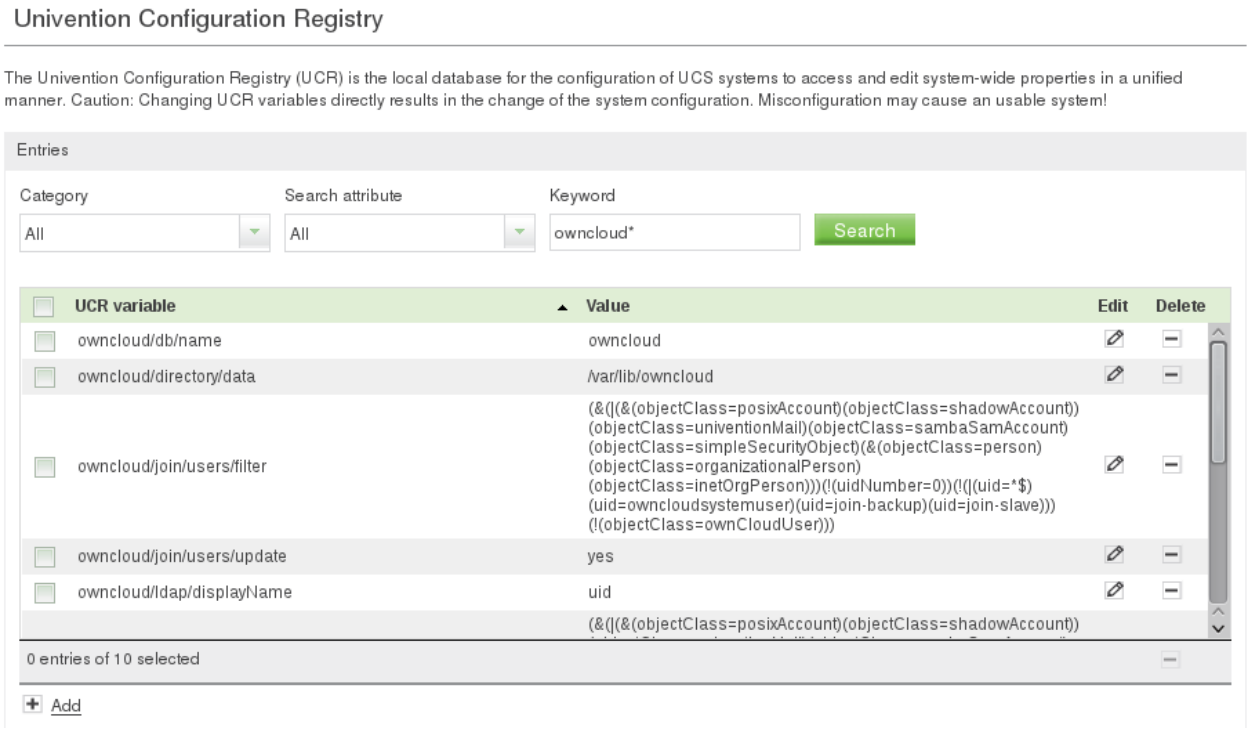
Table 2.1 – continued from previous page

| Key | Default | Description | Introduced |
|---|---|---|---|
| owncloud/join/users/filter | (&(\|(&(objectClass=posixAccount) (objectClass=shadowAccount)) (objectClass=univentionMail) (objectClass=sambaSamAccount) (objectClass=simpleSecurityObject) (&(objectClass=person) (objectClass=organizationalPerson) (objectClass=inetOrgPerson))) (!(uidNumber=0)) (!(\|(uid=*$) (uid=owncloudsystemuser) (uid=join-backup) (uid=join-slave))) (!(objectClass=ownCloudUser))) | Filters, on which LDAP users the ownCloud schema should be applied to. The default excludes system users and already ownCloudUsers. | 2012.0.1 |
| owncloud/join/groups/filter | (empty) | Filters which LDAP groups will be en/disabled for ownCloud when running the script /usr/share/owncloud/update-groups.sh | 2012.4.0.4 |

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

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

or via UMC:

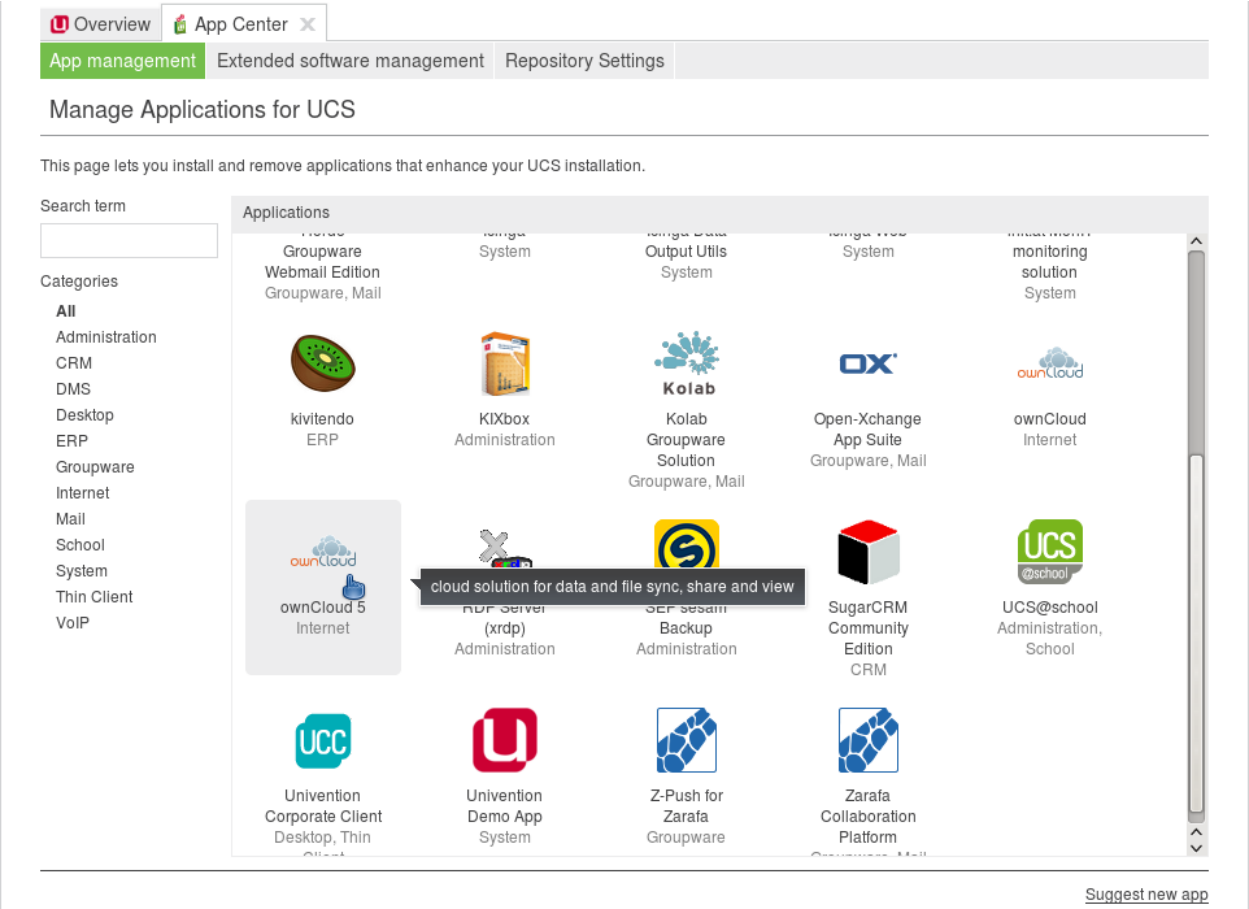


### 2.5.2 Installation

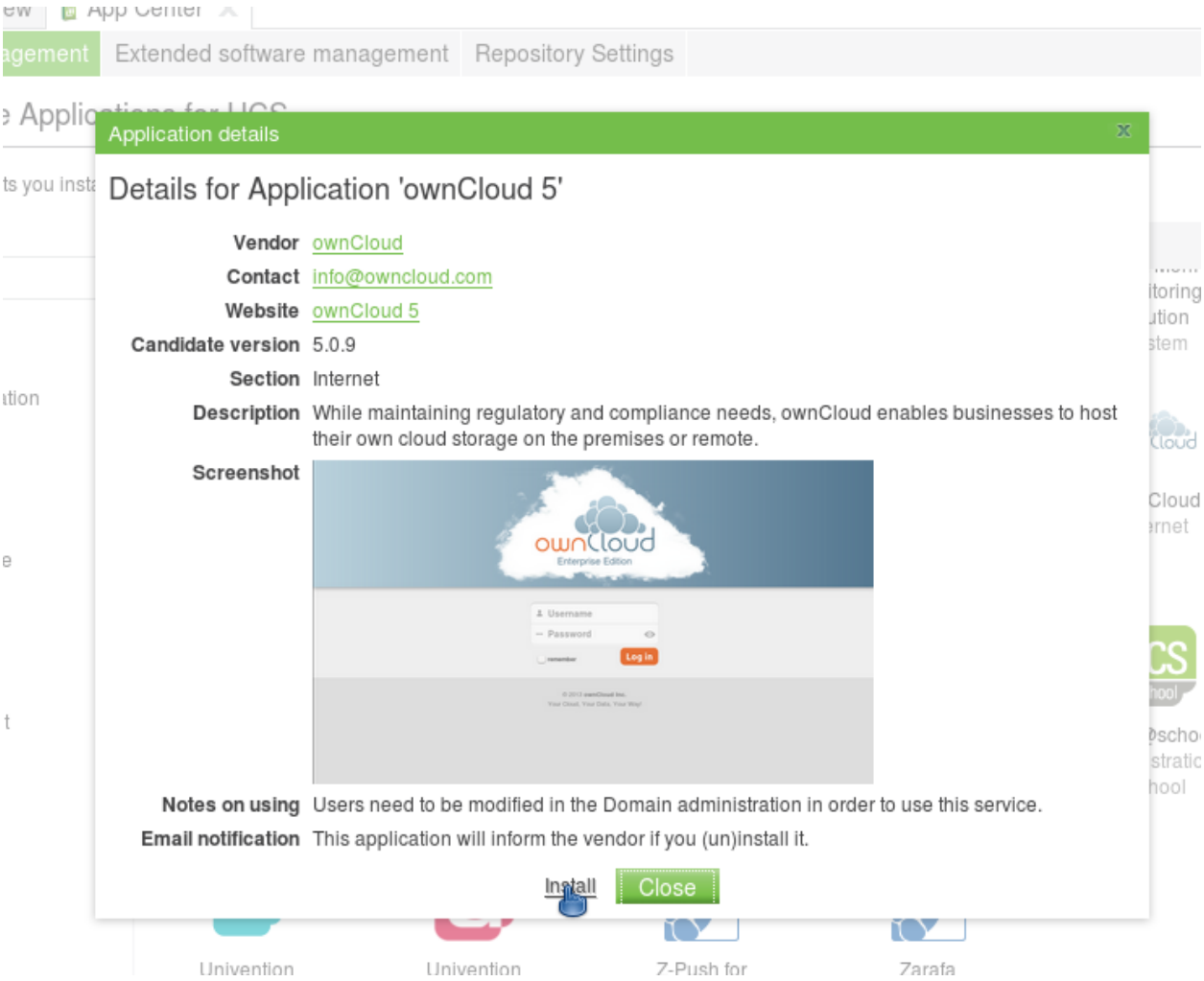
Now, we are ready to install ownCloud. This can be either done through the UCS App Center (recommended) or by downloading the packages.

#### 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.

### 2.5.3 Postconfiguration (optional)

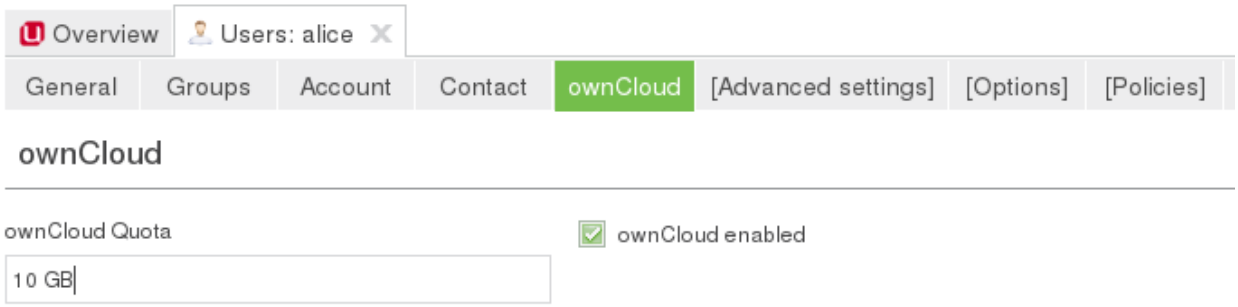
There is only one local admin user "owncloudadmin", you can find his password in */etc/owncloudadmin.secret*. Use this account, if you want to change basic ownCloud settings.

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.

### 2.5.4 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.

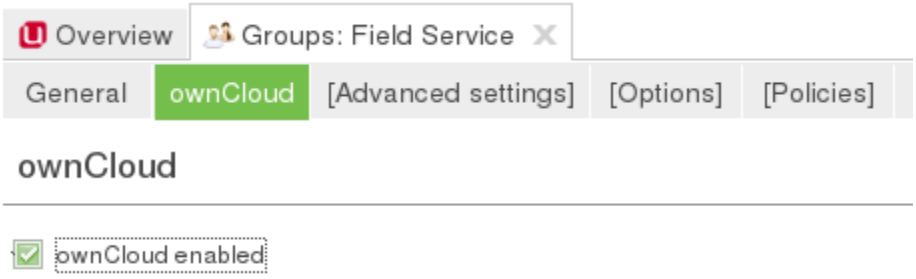
If you did not, go to the UMC and enable the users who shall have access (see picture below). Then, login at http://myserver.com/owncloud/ with your LDAP credentials.




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.

## 2.6 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 12.04 LTS Server with Apache and MySQL. It also provides guidelines for installing it on other distributions, web servers and database systems.

### 2.6.1 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 (>= 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 recommendable 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)
- 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.

### Example installation on Ubuntu 12.04.4 LTS Server

On a machine running a pristine Ubuntu 12.04.4 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:

```
sudo apt-get install apache2 mysql-server libapache2-mod-php5
sudo apt-get install php5-gd php5-json php5-mysql php5-curl
sudo 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 list 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 (you will need it during ownCloud database setup).

First, download the archive of the latest ownCloud version:

- Navigate to the ownCloud Installation Page
- Click "Tar or Zip file"
- In the opening dialog, chose the "Linux" link.
- This will start the download of 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.
- If that's a different machine than the one you are currently working on, use e.g. FTP to transfer the downloaded archive file there.
- Extract the archive contents. Open a terminal and run:

  ```
  cd path/to/downloaded/archive
  tar -xjf owncloud-x.y.z.tar.bz2
  ```

  where `path/to/downloaded/archive` is to be replaced by the path where you put the downloaded archive, and x.y.z of course has to be replaced by the actual version number as in the file you have downloaded.
- Copy the ownCloud files to their final destination in the document root of your web server (you can skip this step if you already downloaded and extracted the files there):

  ```
  sudo cp -r owncloud /path/to/your/webservers/document-root
  ```

  where `/path/to/your/webservers/document-root`, needs to be replaced by the actual path where the document root of your web server is configured to be.
  - If you don't know where your web server's document root is located, consult its documentation. For Apache on Ubuntu 12.04 LTS for example, this would usually be `/var/www`. So the concrete command to run would be:

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

  - The above assumes you want to install ownCloud into a subdirectory "owncloud" on your web server. For installing it anywhere else, you'll have to adapt the above command accordingly.

### 2.6.2 Set the Directory Permissions

The user running your web server must own at least the config/, data/ and apps/ directories in your ownCloud installation folder so that you can configure ownCloud, create/modify and delete your data files through ownCloud and install apps through the web interface. If you are planning on also using the automatic updater app for updating, the whole `owncloud` folder must be owned by (or at least be writable to) the user running php on your system.

---

**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 run is:

  ```
  sudo chown -R <php-user>:<php-user> /path/to/your/webservers/document-root/owncloud
  ```

where `<php-user>` is to be replaced by the user running php scripts, and `/path/to/your/webservers/document-root/owncloud` by the folder where the extracted ownCloud files are located.

- For Ubuntu 12.04 LTS server, where the `owncloud` folder was copied into the Apache document root at `/var/www`, and the user running Apache and php scripts is called `www-data`, this would mean you need to run:

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

- For Arch Linux should run (as root):

  ```
  chown -R http:http /path/to/your/owncloud
  ```

- Fedora users should run (as root):

  ```
  chown -R apache:apache /path/to/your/owncloud
  ```

### 2.6.3 Web Server Configuration

**Note:** You can use ownCloud over plain http, but we strongly encourage you to use SSL/TLS. If you don't use it, and you for example access your ownCloud over an unsecured WiFi, everyone in the same WiFi can grab your authentication data or the content of files synchronized while you are on the WiFi.

Apache is the recommended web server.

#### Apache Configuration

##### 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:

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

If you are using a different distribution, check their 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 an official signing authority. SSLShopper for example has an article on your options for free SSL certificates.

##### 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:

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

Example output:

```
Server version: Apache/2.2.22 (Ubuntu)
Server built:   Jul 12 2013 13:37:10
```

This indicates an Apache of the 2.2 version branch (as e.g. you will find on Ubuntu 12.04 LTS).

Example config for Apache 2.2:

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

Example config for Apache 2.4:

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

- This config 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`).
- Edit the site file with your favorite editor (note that you'll need root permissions to modify that file). For Ubuntu 12.04 LTS, you could for example run the following command in a Terminal:

  ```
  sudo nano /etc/apache2/sites-available/default-ssl
  ```

- 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 12.04 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
                Order allow,deny
                Allow from all
                # add any possibly required additional directives here
                # e.g. the Satisfy directive (see below for details):
                Satisfy Any
        </Directory>
</VirtualHost>
</IfModule>
```

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

  ```
  sudo 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 distributions 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 configuration of your ownCloud. In above "<Directory ..." code, add the following line directly after the `allow from all` / `Require all granted` line):

  ```
  Dav Off
  ```

- Furthermore, you need to disable any server-configured authentication for ownCloud, as it's internally using Basic authentication for its *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; to do so, in above "<Directory ..." code, add the following line directly after the `allow from all` / `Require all granted` line):

  ```
  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 these to the domain you want to reach your ownCloud under.

---

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

---

- Finally, restart Apache.

- For Ubuntu systems (or distributions using upstartd), run:

```
sudo service apache2 restart
```

- For systemd systems (Fedora, ArchLinux, OpenSUSE), run:

```
systemctl restart httpd.service
```

### 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;
        return 301 https://$server_name$request_uri;  # enforce https
}

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/;

        client_max_body_size 10G; # set max upload size
        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 without SSL/TLS support, but we strongly encourage you not to do that:

- 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.

### Lighttpd Configuration

This assumes that you are familiar with installing PHP application on lighttpd.

It is important to note that the `.htaccess` used by ownCloud to protect the `data` folder are 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 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 chan
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.

### 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.

#### 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
}
```

#### Microsoft Internet Information Server (IIS)

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

### 2.6.4 Install Wizard

The last thing to do is to click through the installation wizard.

Here are some guidelines for the values to enter there if following the Ubuntu-Apache-MySQL walk-through:

- Make sure to click the "Advanced" Button to see the database settings
- Choose MySQL as Database backend (you might not be presented with any other choice if you haven't installed any other database systems).
- As Database host, enter `localhost`.
- As Database user enter `root`.
- As Database password, enter the password you entered during installation of the MySQL server package.
- As Database name, enter an arbitrary name as you see fit

Continue by following the *Installation Wizard*.

## 2.7 Other Installation Methods

### 2.7.1 Amahi Home Server Configuration

Instructions to install ownCloud into an Amahi server can be found here.

### 2.7.2 Open Wrt

Here you can find a tutorial for open Wrt

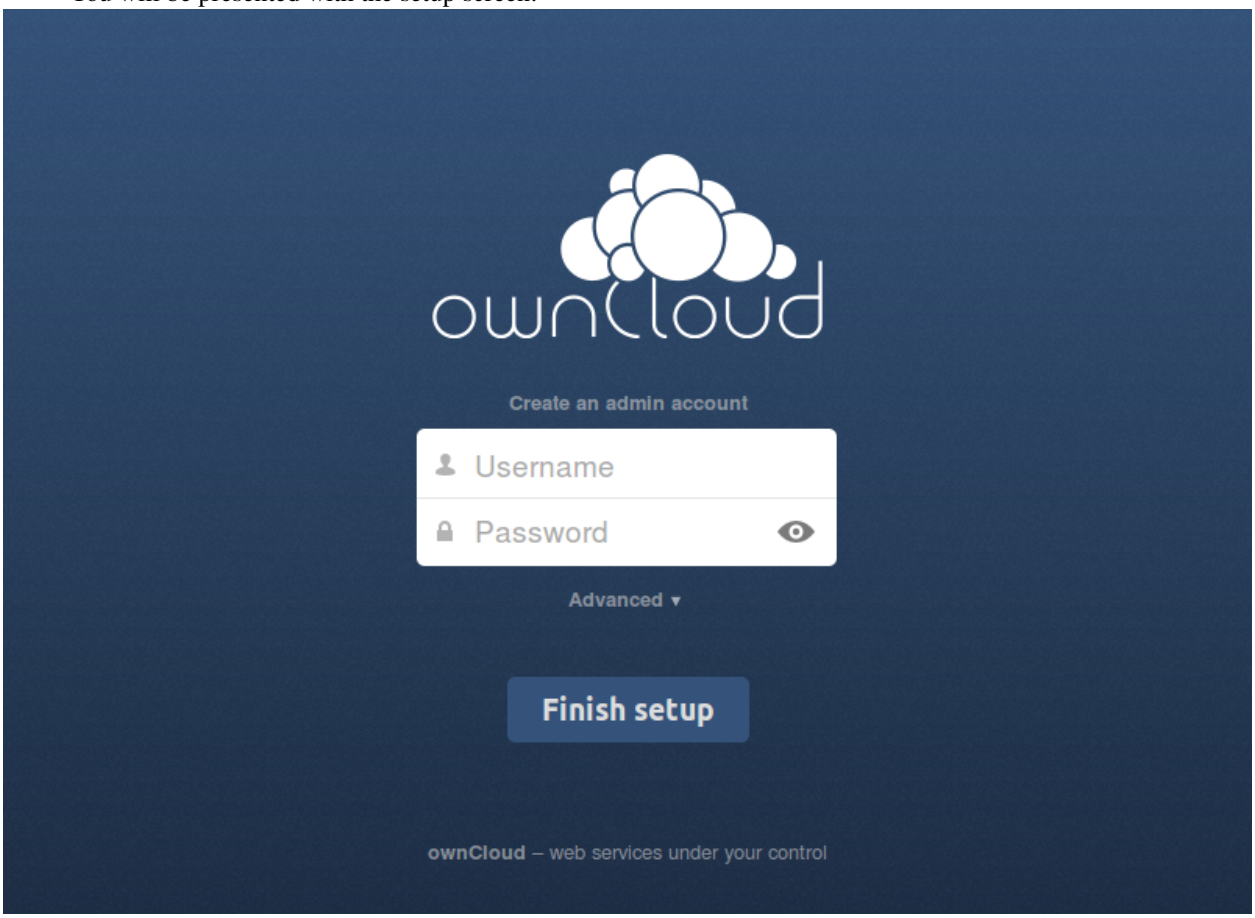
### 2.7.3 PageKite Configuration

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

## 2.8 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:

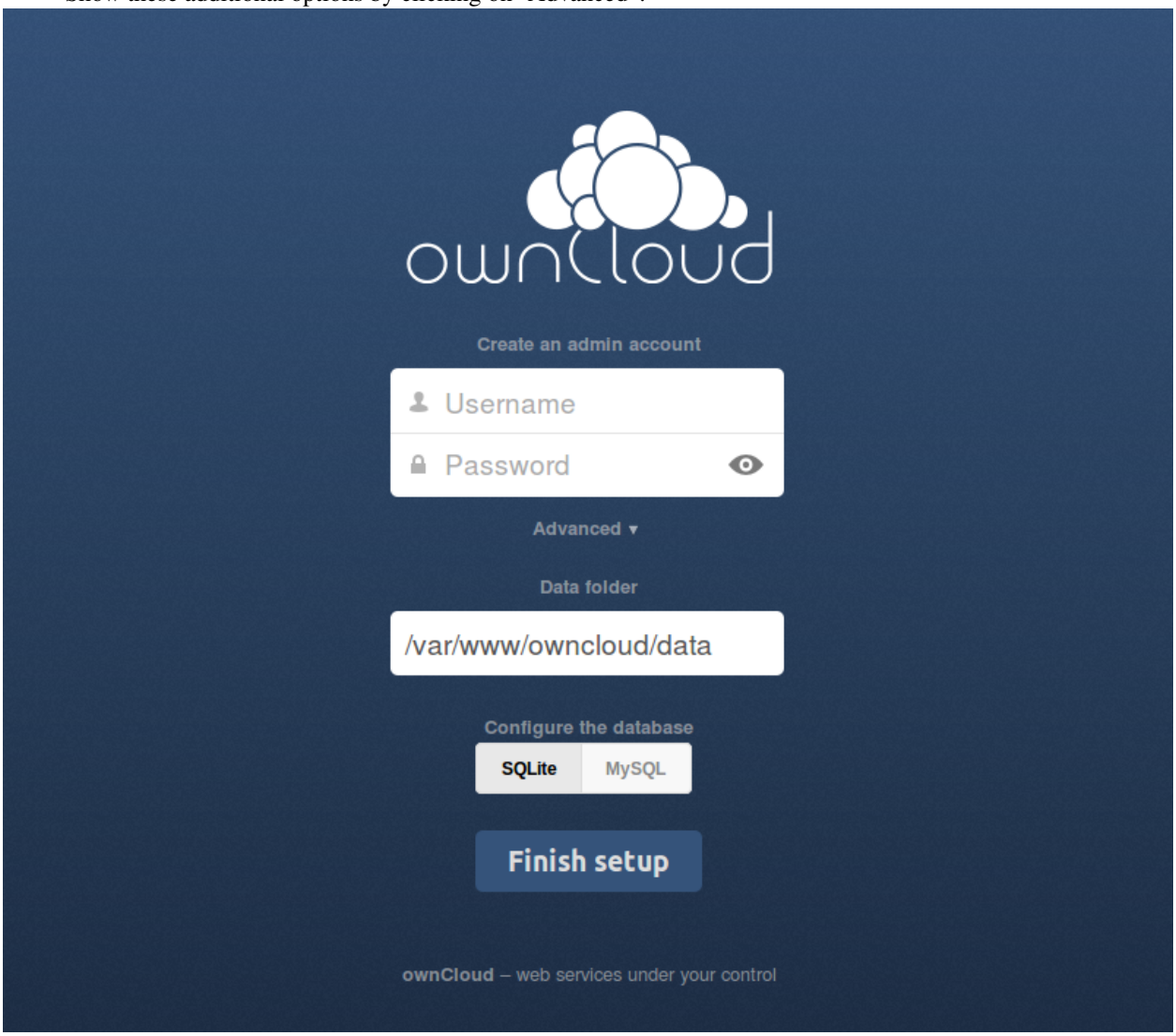


### 2.8.1 Required Settings

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.

### 2.8.2 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":




#### 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.

### 2.8.3 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.

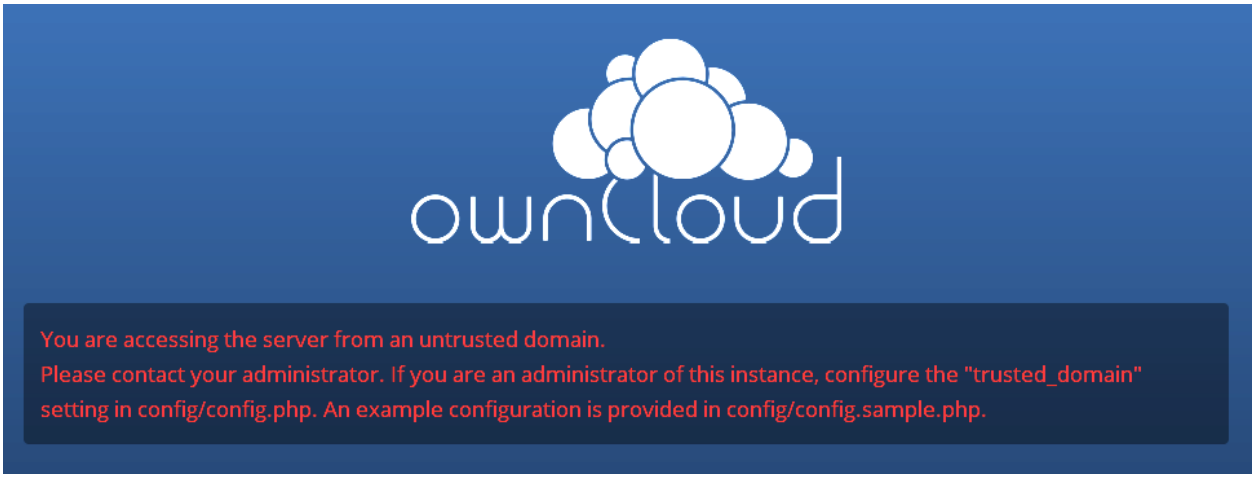
### 2.8.4 Note

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.

CHAPTER

THREE

# CONFIGURATION

## 3.1 Apps Configuration

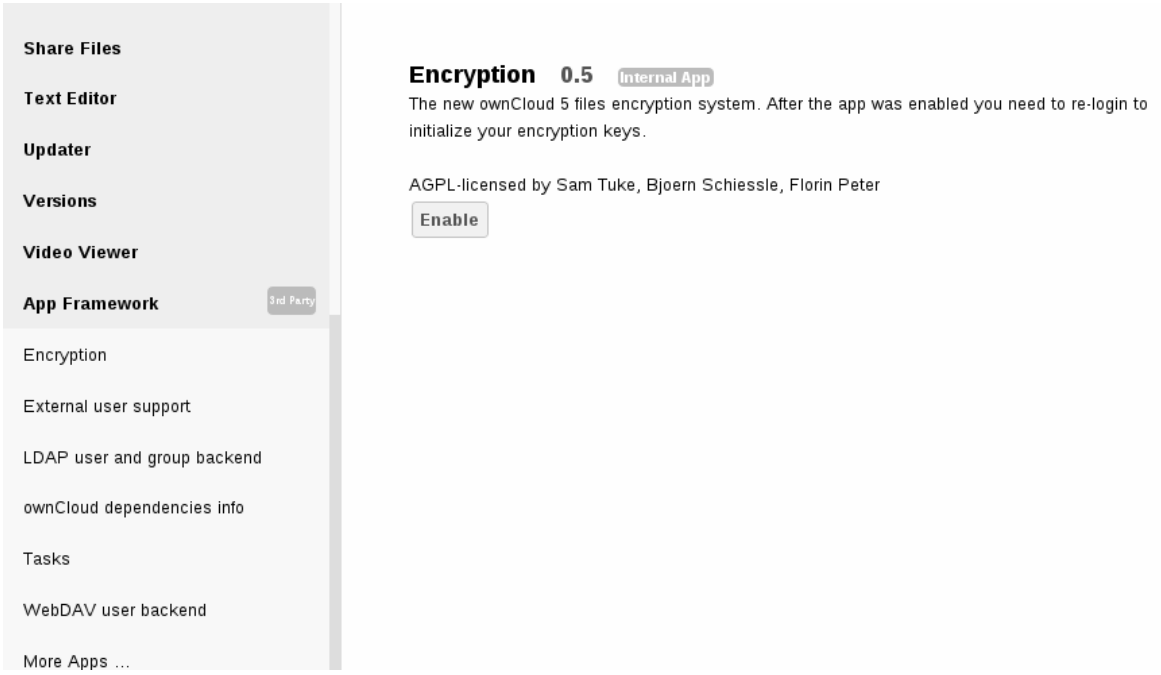
After installing ownCloud, you might want to provide added functionality on top of the core functionality that is installed by default. ownCloud enables you to enhance your experience and your user's experiences by installing various *apps*.

### 3.1.1 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.




**Administrator application page**

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

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

### 3.1.2 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.

### 3.1.3 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.

---

### 3.1.4 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,
```

## 3.2 User Management

ownCloud administrators can easily manage users via the web interface. To go into user management page, click your username on the web interface and select *Users*. A page similar to the image below will be shown:

A fictive use case will help you understand the concept of users, user groups and group admins.

Think of a small, 25-member staff company, named "Cloud Lovers", that is lead by its founder Richard. In this company Bob acts as IT operator and recently set up ownCloud. Being the installing user, Bob is member of the so

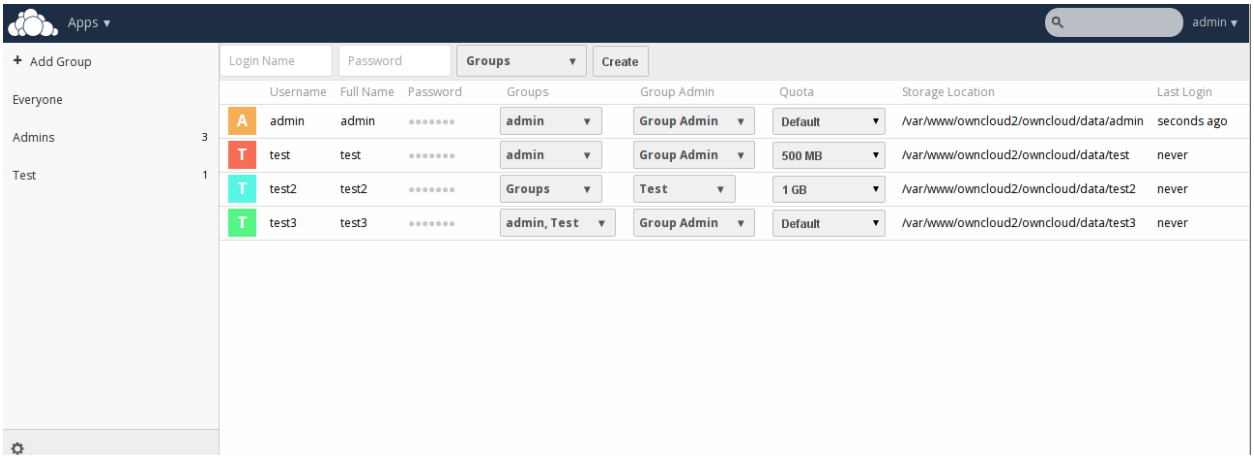



Figure 3.1: Users management page

called "admin" user group of ownCloud. His colleague Tom, who provides support if Bob is on holiday, is member of the "admin" user group as well. All employees, including Bob and Tom, are members of the user group "Internal", that is used to share data across the company. Mostly for operational data, that should not be accessible to all employees, Bob created the "Administration" user group having two members: Richard and his assistant Susan. Richard is group admin of this user group, so he can manage the members of the "Administration" user group on his own.

### 3.2.1 Users

A user represents an account of the ownCloud installation. In this section the core properties are listed.

***Login name (Username)*** This is the unique ID of a ownCloud user (e.g. test, jon.doe).

***Full Name*** This is the name that is used all over the user interface to identify the user i.e. when sharing data or sending mails. If no display name is set, it defaults to the login name.

***Password*** This is the password the user uses to login to ownCloud.

***Groups*** This is a list of security groups the user is assigned to. By default the user is not member of any user group.

***Group Admin*** This is a list of security groups the user has administration privileges for. By default the user is not registered as group admin for any user group.

***Quota*** This is the maximum disk space that may be used by the user. If the user reaches this limit he/she is not able to upload or sync further data. The storage quota is specified in the format *Number Unit* (e.g. 100 B (byte), 50 KB (kilobyte), 20 MB (megabyte), 5 GB (gigabyte)). If no unit is given, the number is interpreted as bytes.

Each user is able to change its display name and password.

#### Create a user

Before users can sign in and share data, they need ownCloud user accounts.

To create a user account:

1. Enter the new user's **Login Name** and its initial **Password** in the appropriate fields.
2. (Optional) Select the **Groups** to which you want to assign the new user.
3. Click **Create**.

4. (Optional) Edit additional user settings.

   To set other user settings, such as setting a display name or limiting the user's storage, see instructions as follows.

Created users will have the storage specified on *Default Storage* setting on the same page.

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

### Reset a user's password

To reset a user's password:

1. Hover the line of the user.
2. Click on the **pencil icon** next to the password field.
3. Enter the user's new password in the password field and then hit the **Enter** key of your keyboard.

Remember to provide the user with the new login information after you have reset the password.

### Rename a user

Each ownCloud user has two names: an unique *login name* used for authentication, and a *display name* (e.g. the user's first name and last name) used in the user interface. You can edit the display name of a user, but you cannot change the login name of any user.

To set a user's display name:

1. Hover the line of the user.
2. Click on the **pencil icon** next to the display name field.
3. Enter the user's new display name in the corresponding field and then hit the **Enter** key of your keyboard.

### Grant administrator privileges to a user

If a user has administrator privileges, the user has the right to manage other users. Within ownCloud there are two types of administrators: *Super Administrators* and *Group Administrators*.

Group administrators have the management rights to:

- Create new users and assign them to the group of the group administrator
- Edit and delete users that are assigned to the group of the group administrator

Group administrators cannot access system settings or modify installation-wide configuration like the default storage.

To assign the *super administrator* role to a user:

1. Use the drop-down list in *Groups* column of the user
2. Assign the user to the "admin" user group

To assign the *group administrator* role to a user:

Find the user and select the user groups from the **Group Admin** drop-down list you want the user become group administrator for.

#### Assign a user to a user group

To assign a user to a user group:

Find the user and select the user groups from the **Groups** drop-down list you want to assign the user to. You can use *add group* link to create a new group to assign the user to. You can assign the user more than one group by checking multiple groups.

---

**Note:** If a file/folder is shared with a group, newly created users will immediately have access to the share.

---

---

**Note:** If you assign a user to the *admin* user group, the user will become a *Super Administrator* with unlimited privileges.

---

#### Limit a user's storage

To limit a user's storage quota:

Find the user and select an item from the **Quota** drop-down list.

- If you select *Default*, the default storage limit, specified in the action bar at the top, is applied.
- If you select *Unlimited*, the user is not limited until the total disk space is consumed.
- If you want to enter a custom limit, select *Other...*, enter the storage quota of your choice and hit the **Enter** key of your keyboard.

If you edit the value of the **Default Quota** field by clicking on the **gear** icon, all users with storage *Default* are affected by this change, i.e. changing the default storage from *Unlimited* to *1 GB* will cause all users with *Default* storage being limited to 1 GB storage each.

#### Delete User

**Important considerations before deleting a user:**

- The user will no longer be able to sign in to your ownCloud installation.
- You cannot revert the deletion or restore a deleted account.

---

**Note:** If this user had a share with a group or user, the share also will be deleted permanently.

---

To delete a user account:

1. Hover the line of the user you want to delete.
2. Click the **cross icon** at the end of the line.

---

**Note:** If you accidentally delete a user, you can use undo button shown on notification bar at the top of the page.

---

### 3.2.2 User Groups

#### Create Group

To create a user group:

1. Click on **Add Group** button on the left side of the user management page.

2. Enter the name of the new group and then hit the **Enter** key of your keyboard.

You can *assign users* to the newly created user groups anytime by using users' group drop-down list.

#### Edit/Delete Group

Currently, groups cannot be renamed. This feature will be available in a future version of ownCloud. To delete a group, unassign all users of out of it and click on the trash icon next to the group name on the left pane.

---

**Note:** If you have direct access to the database, you can manually rename the group from database tables oc_groups and oc_group_user.

---

## 3.3 User Authentication with LDAP

ownCloud ships an LDAP backend, which allows full use of ownCloud for user logging in with LDAP credentials including:

- LDAP group support
- File sharing with users and groups
- Access via WebDAV and of course ownCloud Desktop Client
- Versioning, external Storages and all other ownCloud goodies

To connect to an LDAP server the configuration needs to be set up properly. Once the LDAP backend is activated (Apps Sidebar→Apps, choose **LDAP user and group backend**, click on **Enable**) the configuration can be found on Settings→Admin. Read on for a detailed description of the configuration fields.

### 3.3.1 Configuration

The LDAP backend follows a wizard-like approach, split into four tabs. A correctly completed first tab ("Server") is mandatory to access the other tabs. Also, the other tabs need to be reviewed by the admin, however the necessary settings are detected automatically. An indicator will show whether the configuration is incomplete, incorrect or OK.

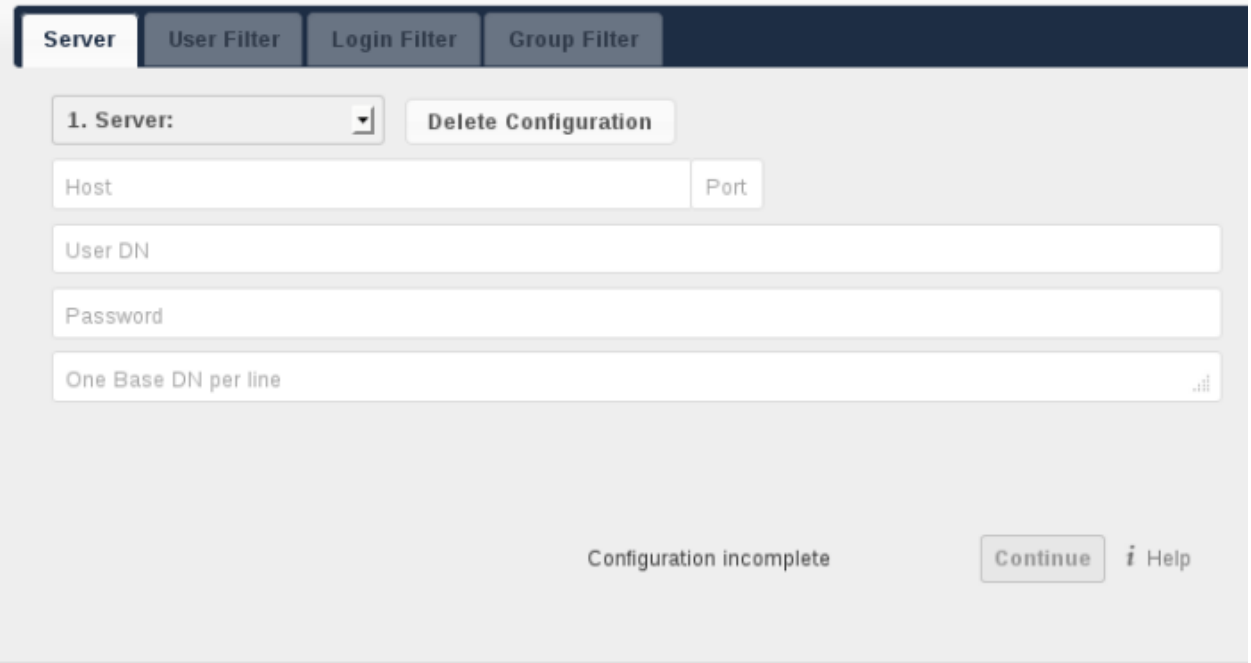
The settings are changed automatically, as soon as a input element loses the focus, i.e. the cursor is taken away by clicking somewhere else or pressing the tabulator key.

The other tabs can be navigated by clicking the tabs or by using the *Continue* and *Back* buttons. They are located on the lower right, next to the status indicator.

#### Server

The server tab contains the basic information on the LDAP server. They make sure that ownCloud will be able to connect to LDAP and be able to read data from there. The admin at least needs to provide a hostname. If anonymous access is not possible he will need to provide an account DN and a password, too. ownCloud attempts to auto-detect the port and the base DN.

**Server configuration:** ownCloud can be configured to connect to multiple LDAP servers. Using this control you can pick a configuration you want to edit or add a new one. The button **Delete Configuration** deletes the current configuration.




**Host:** The host name of the LDAP server. It can also be a **ldaps://** URI, for instance.

It is also possible to pass a port number, which speeds up port detection. It is especially useful, if a custom port is used. ownCloud will move the value to the port field subsequently.

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. After ownCloud attempted to determine the port, the field will be enabled for user input. A successfully found port will be inserted by ownCloud, of course.

Example:

- *389*

**User DN:** The name as DN of a user who is able to do searches in the LDAP directory. Leave it empty for anonymous access. It is recommended to have a special system user for ownCloud.

Example:

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

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

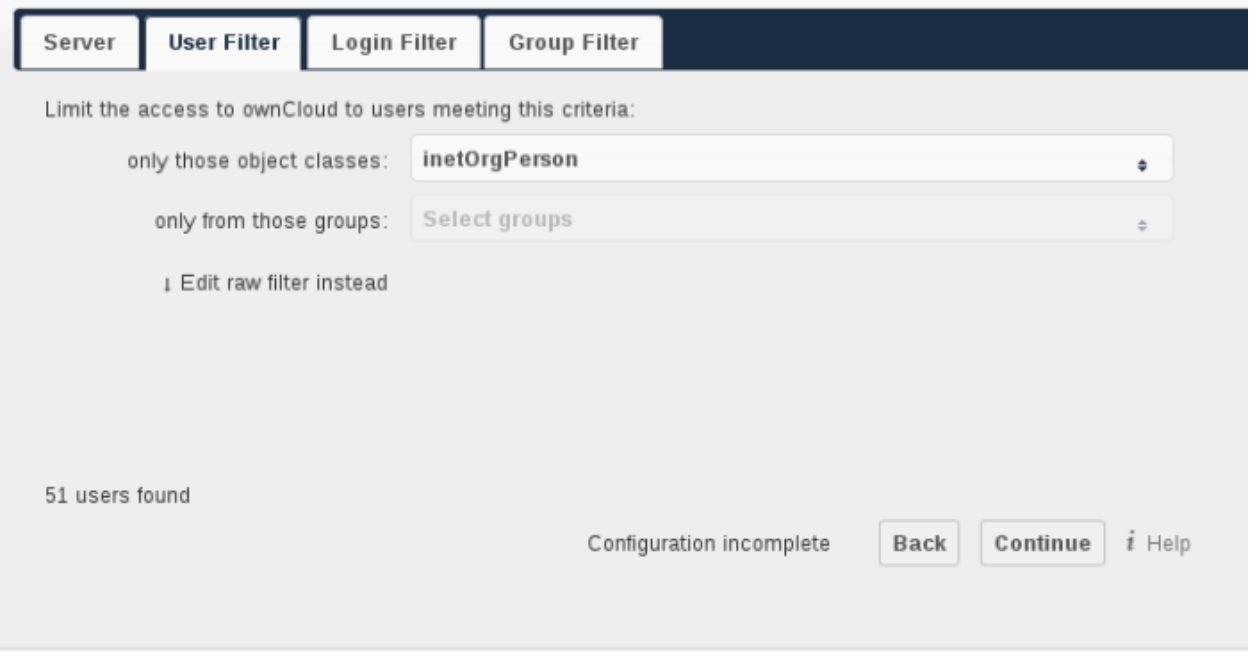
**Base DN:** The base DN of LDAP, from where all users and groups can be reached. Separated Base DNs for users and groups can be set in the Advanced tab. Nevertheless, this field is mandatory. ownCloud attempts to determine the Base DN according to the provided User DN or the provided Host.

Example:

- *dc=my-company,dc=com*

### User Filter

The settings in the user filter tab determine which LDAP users will appear and are allowed to log in into ownCloud. It is also possible to enter a raw LDAP filter.




**only those object classes:** ownCloud will determine the object classes that are typically available for (ideally only) user objects in your LDAP. ownCloud will automatically select the object class that returns the highest amount of users. You can 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 can 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 will toggle the filter mode. Instead of the assisted approach, you can enter the raw LDAP filter directly in the appearing field.

Example:

- *objectClass=inetOrgPerson*

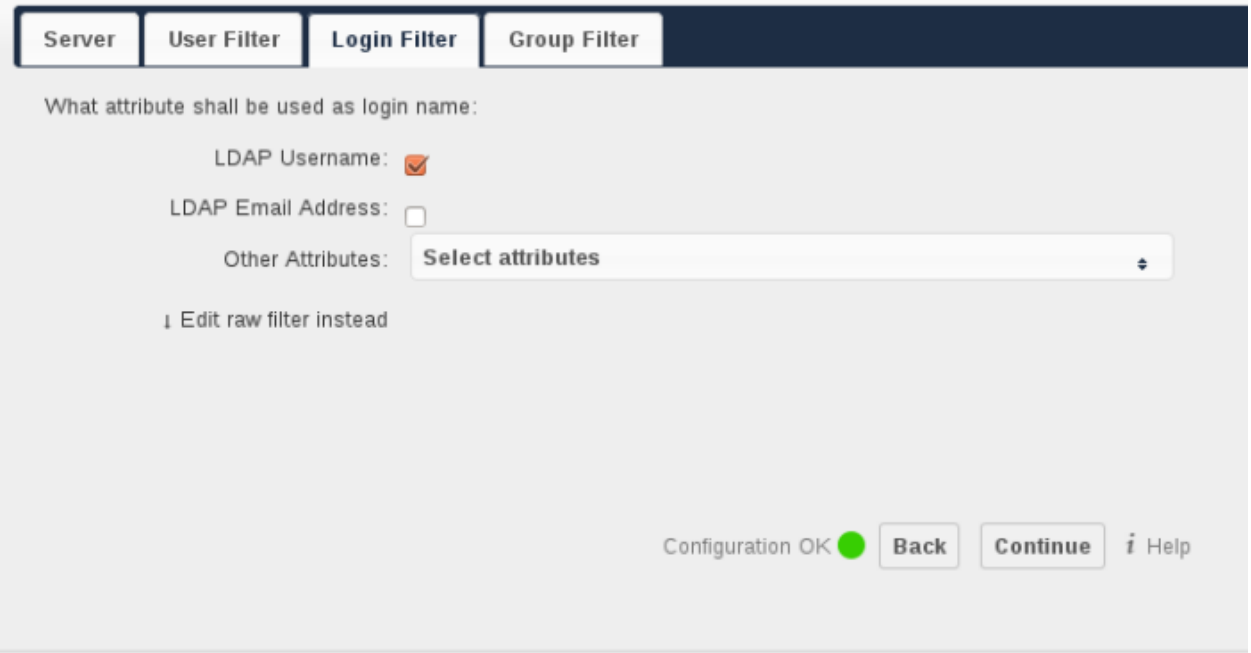
**x users found:** This is an indicator that tells you approximately how many users will be allowed to access ownCloud. The number will update after any change you do.

### Login Filter

The settings in the login filter tab determine which user detail will be compared to the login value entered by the user. It is possible to allow multiple user details. It is also possible to enter a raw LDAP filter.

The user limitation as set up in the previous tab is in effect, unless you manually configure the filter in raw mode.

**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. The email address will be looked for in the *mailPrimaryAddress* and *mail* attributes.

**Other Attributes:** This multiselect box allows you to select other attributes for the comparison. The list is generated automatically based on the attributes that a user object contains in your LDAP server.

**Edit raw filter instead:** Clicking on this text will toggle the filter mode. Instead of the assisted approach, you can enter the raw LDAP filter directly in the appearing field.

The **%uid** placeholder will be replaced with the login name entered by the user upon login. When you enter the filter manually.

Examples:

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

### Group Filter

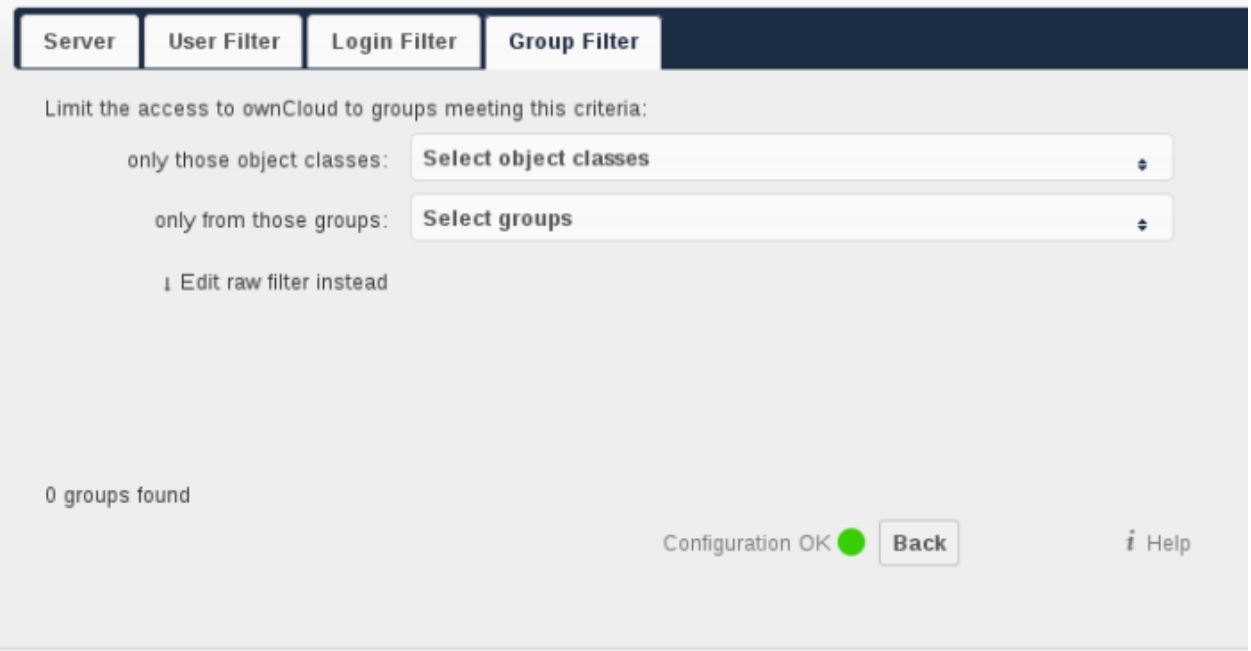
The settings in the group filter tab determine which groups will be available in ownCloud. It does not have any restrictions on logins, this has been dealt with in the prior tabs. It is also possible to enter a raw LDAP filter.

By default, no groups will be available in ownCloud. You actively need to enable groups.

**only those object classes:** ownCloud will determine the object classes that are typically available for (ideally only) group objects in your LDAP. 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:** This setting lets you pick certain groups that shall be available in ownCloud. This field follows a whitelist approach. ownCloud will generate a list of available groups found in your LDAP server. You can select multiple groups.

**Edit raw filter instead:** Clicking on this text will toggle the filter mode. Instead of the assisted approach, you can enter the raw LDAP filter directly in the appearing field.




Example:

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

**y groups found:** This is an indicator that tells you approximately how many groups will be available in ownCloud. The number will update after any change you do.

### 3.3.2 Advanced Settings

In the LDAP Advanced settings section you can define options, that are less common to set. They are not needed for a working connection. It can also have a positive effect on the performance to specify distinguished bases for user and group searches.

The Advanced Settings are structured into three parts:

- Connection Settings
- Directory Settings
- Special Attributes

#### Connection Settings

**Configuration Active:** Enables or Disables the current configuration. Disabled configuration will not connect to the LDAP server.

By default, it is turned off. It will be automatically turned on, when using the wizard and the configuration is OK and a test connection successful.

**Backup (Replica) Host:** A backup server can be defined here. ownCloud tries to connect to the backup server automatically, when the main host (as specified in basic settings) cannot be reached. It is import that the backup server is a replica of the main server, because the object UUIDs must match.

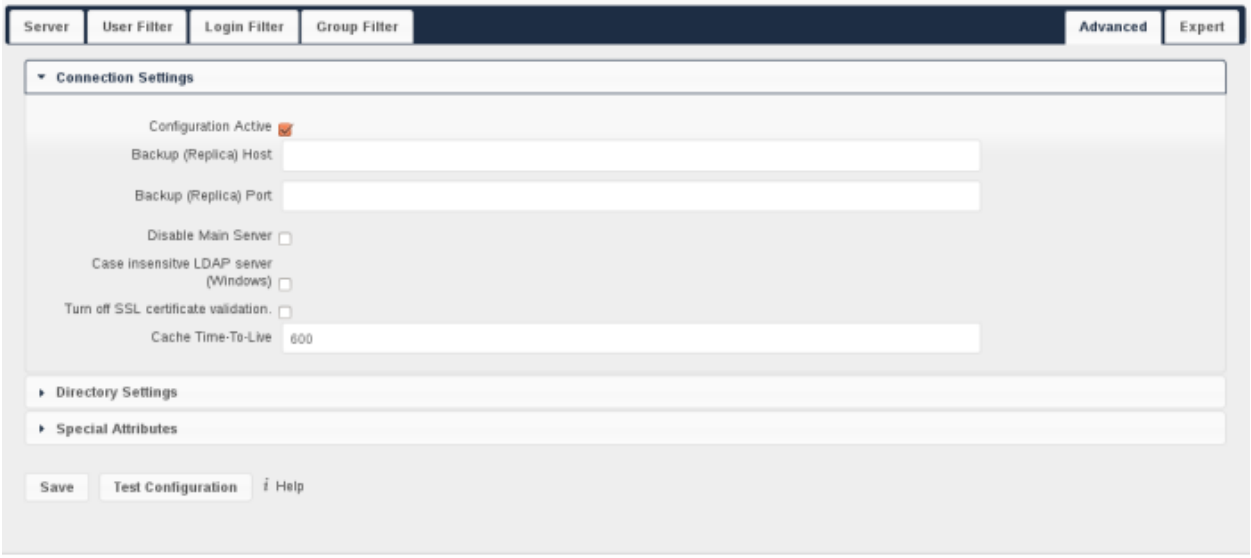



Figure 3.2: LDAP Advanced Settings, section Connection Settings

Example:

- *directory2.my-company.com*

**Backup (Replica) Port:** The port on which to connect to the backup LDAP server. If no port is given, but 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. It may be handy for planned downtimes.

**Case insensitive LDAP server (Windows):** Whether the LDAP server is running on a Windows Host. Usually, it is not necessary to check it, however.

**Turn off SSL certificate validation:** Turns of check of valid SSL certificates. Use it – if needed – for testing, only!

**Cache Time-To-Live:** A cache is introduced to avoid unnecessary LDAP traffic, for example lookups check whether the users exists on every page request or WebDAV interaction. It is also supposed to speed up the Admin → User page or list of users to share with, once it is populated. Saving the configuration empties the cache (changes are not necessary). The time is given in seconds.

Note that almost every PHP request would require to build up a new connection to the LDAP server. If you require a most up-to-dateness it is recommended not to totally switch off the cache, but define a minimum life time of 15s.

Examples:

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

### Directory Settings

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

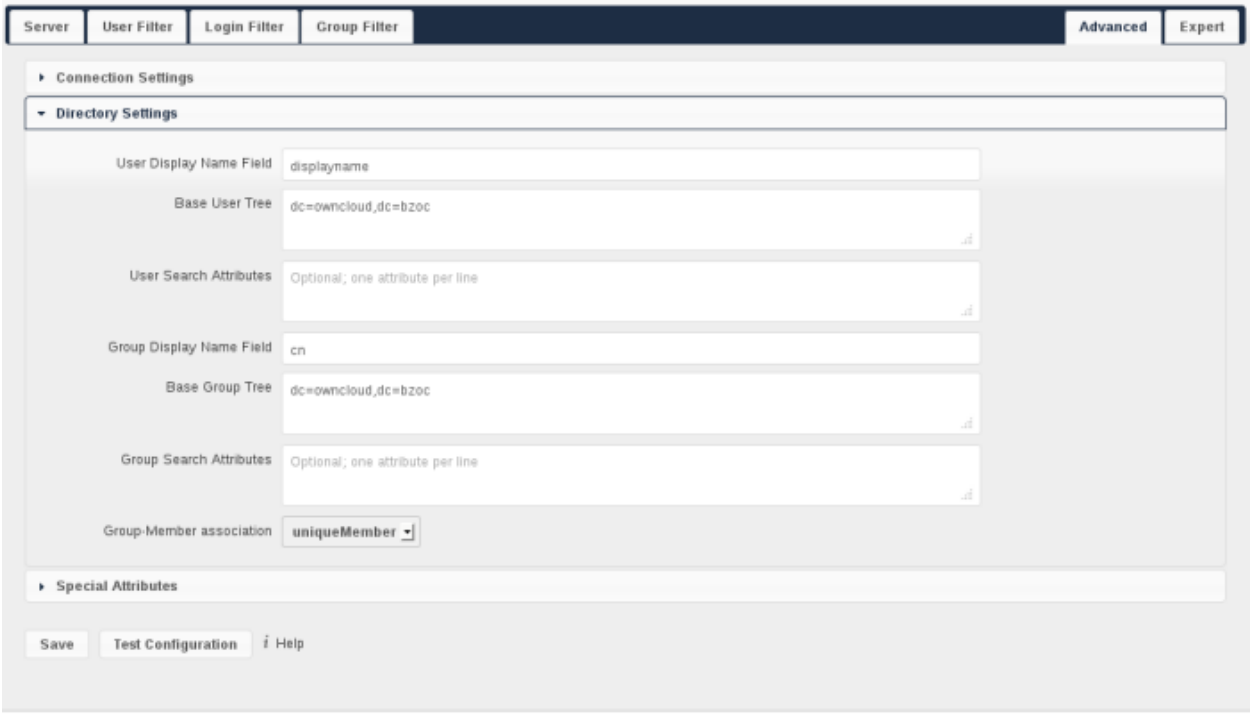



Figure 3.3: LDAP Advanced Settings, section Directory Settings

- Example: *displayName*

**Base User Tree:** The base DN of LDAP, from where all users can be reached. It needs to be given completely despite to the Base DN from the Basic settings. You can specify multiple base trees, one in 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 a search for users is done. This happens, for instance, in the share dialogue. By default the user display name attribute as specified above is being used. Multiple attributes can be given, one in each line.

Beware that if an attribute is not available on a user object, the user will neither be listed (e.g. in the share dialogue) nor be able to login. This also affects the display name attribute as specified above. If you override the default, the display name attribute will not be taken into account, unless you specify it as well.

- Example:

*displayName*
*mail*

**Group Display Name Field:** The attribute that should be used as ownCloud group name. ownCloud allows a limited set of characters (a-zA-Z0-9.-_@), every other character will be replaced in ownCloud. Once a group name is assigned, it will not be changed, i.e. changing this value will only have effect to new LDAP groups.

- Example: *cn*

**Base Group Tree:** The base DN of LDAP, from where all groups can be reached. It needs to be given completely despite to the Base DN from the Basic settings. 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. This happens, for instance, in the share dialogue. By default the group display name attribute as specified above is being used. Multiple attributes can be given, one in each line.

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

- Example:

*cn*
*description*

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

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

- Example: *uniquemember*

### Special Attributes

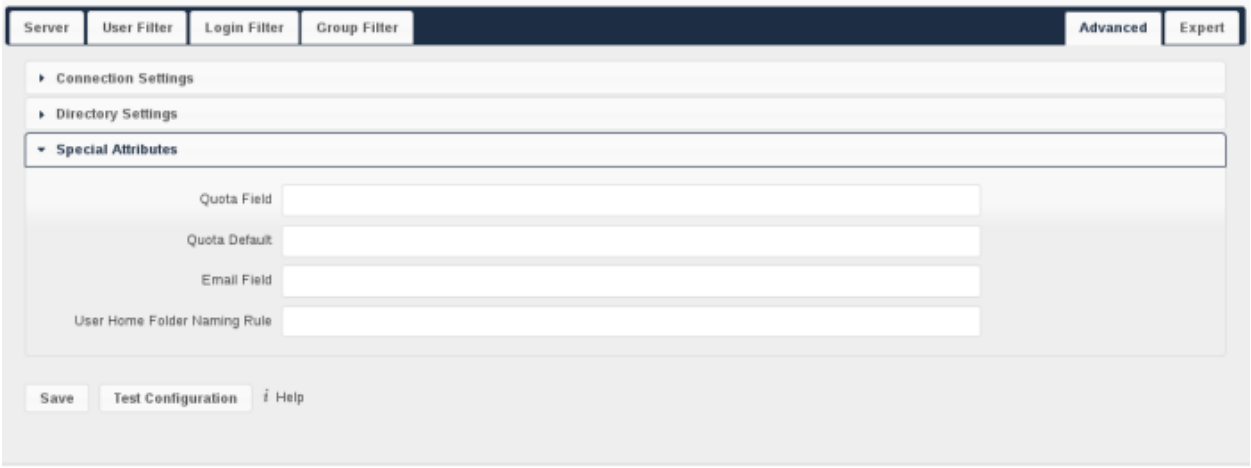



Figure 3.4: LDAP Advanced Settings, section Special Attributes

**Quota Field:** ownCloud can read an LDAP attribute and set the user quota according to its value. Specify the attribute here, otherwise keep it empty. The attribute shall 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 attribute given above.

- Example: *15 GB*

**Email Field:** ownCloud can read an LDAP attribute and set the user email there from. Specify the attribute here, otherwise keep it empty.

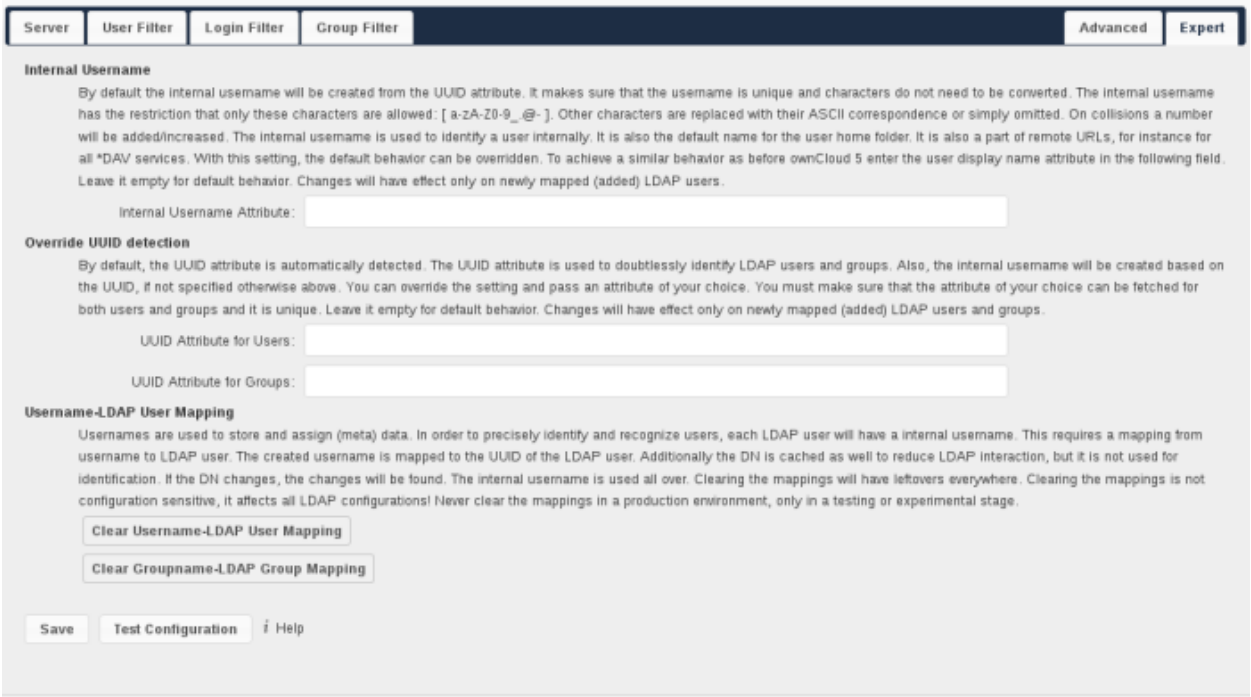
Although the wizard offers you to check login by email, the correct email attribute is not detected and you need to specify it manually.

- Example: *mail*

**User Home Folder Naming Rule:** By default, the ownCloud creates the user directory, where all files and meta data are kept, according to the ownCloud user name. 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*

### 3.3.3 Expert Settings




In the Expert Settings fundamental behavior can be adjusted to your needs. The configuration should be done before starting production use or when testing the installation.

**Internal Username:** The internal username is the identifier in ownCloud for LDAP users. By default it will be created from the UUID attribute. By using the UUID attribute it is made 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 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 also the default name for the user home folder in ownCloud. It is also a part of remote URLs, for instance for all *DAV services. With this setting the default behaviour can be overridden.

Leave it empty for default behaviour. Changes will have effect only on newly mapped (added) LDAP users.

- Example: *uid*

**Override UUID detection** By default, ownCloud auto-detects the UUID attribute. 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 behaviour. Changes will have effect only on newly mapped (added) LDAP users and groups. It also will have effect when a user's or group's DN changes and an old UUID was cached: It will result in a new user. Because of this, the setting should be applied before putting ownCloud in production use and cleaning the bindings (see below).

- Example: *cn*

**Username-LDAP User Mapping** ownCloud uses the usernames as key 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. Do never clear the mappings in a production environment. Only clear mappings in a testing or experimental stage.

**Clearing the Mappings is not configuration sensitive, it affects all LDAP configurations!**

### 3.3.4 Testing the configuration

In this version we introduced the **Test Configuration** button on the bottom of the LDAP settings section. It will always check 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 with the settings currently given in the input fields. The response will look like this:


Figure 3.5: 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**. Unfortunately it requires a reload – sorry for the inconvenience.

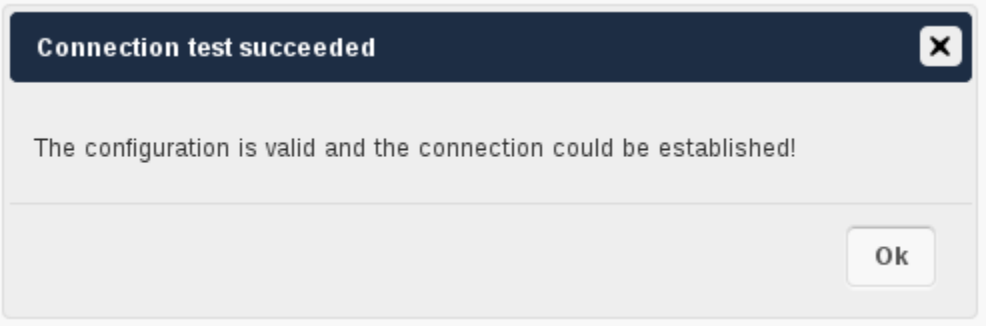



Figure 3.6: Success

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

### 3.3.5 ownCloud Avatar integration

ownCloud 6 incorporates a user profile picture feature, called Avatar. If a user has a photo stored in the *jpegPhoto* or, since 6.0.2, *thumbnailPhoto* attribute, it will be used as Avatar. The user then is not able to change his avatar in the personal settings. It must be done within LDAP. *jpegPhoto* is preferred over *thumbnailPhoto*.

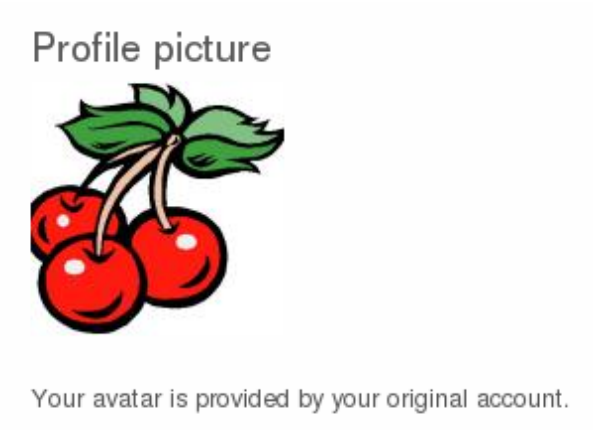



Figure 3.7: Profile picture fetched from LDAP, Personal Settings

If the *jpegPhoto* or *thumbnailPhoto* attribute is not set or empty, the default ownCloud behaviour is active, i.e. the user will be able to set and change his profile picture in the personal settings. If the user sets a profile picture within ownCloud it will _not_ be stored in LDAP.

The *jpegPhoto* or *thumbnailPhoto* attribute will be fetched once a day to make sure the current photo from LDAP is used in ownCloud. If a picture is added later, a possibly set profile picture will be overridden with the LDAP one. If a photo stored in the *jpegPhoto* and/or *thumbnailPhoto* attribute is deleted later, the last profile picture in ownCloud will still be used.

The photo taken from LDAP will be adjusted to the requirements of the ownCloud avatar automatically. I.e. it will be transformed into a square. If the photo needs to be cut, it will be done equally from both affected sides. The original photo stored in LDAP will stay the same, of course.

### 3.3.6 Troubleshooting, Tips and Tricks

### 3.3.7 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 686)

### 3.3.8 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 wizard-like set up process.

### 3.3.9 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 the configuration and enable it if you wish.

### 3.3.10 ownCloud LDAP Internals

Some parts of how the LDAP backend works are described here. May it be helpful.

### 3.3.11 Groups

At the moment, only secondary groups are read. That means that only the groups are retrieved, which are returned by the attribute auto-detected (or manually chosen) in Group-Member association. Primary groups are not being taken into account.

### 3.3.12 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, 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. group 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. If you want to rename a group, be very careful. Do not rename the user's internal name.

### 3.3.13 Caching

For performance reasons a cache has been introduced to ownCloud. He we store all users and groups, group memberships or internal userExists-requests. Since ownCloud is written in PHP and each and every page request (also done by Ajax) loads ownCloud and would execute one or more LDAP queries again, you do want to have some of those queries cached and save those requests and traffic. It is highly recommended to have the cache filled for a small amount of time, which comes also very handy when using the sync client, as it is yet another request for PHP.

### 3.3.14 Handling with Backup Server

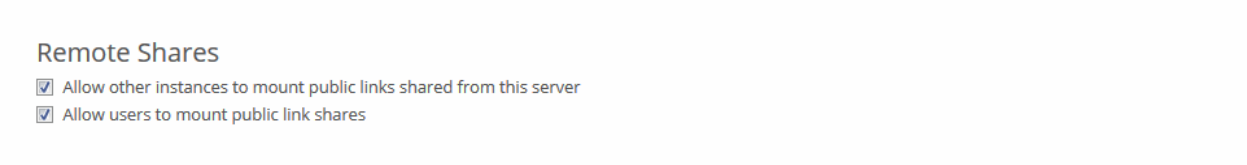
When ownCloud is not able to contact the main server, he will be treated as offline and no connection attempts will be done for the time specified in **Cache Time-To-Live**. If a backup server is configured, it will be connected instead. If you plan a maintained downtime, check **Disable Main Server** for the time being to avoid unnecessary connection attempts every now and then.

## 3.4 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. 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:

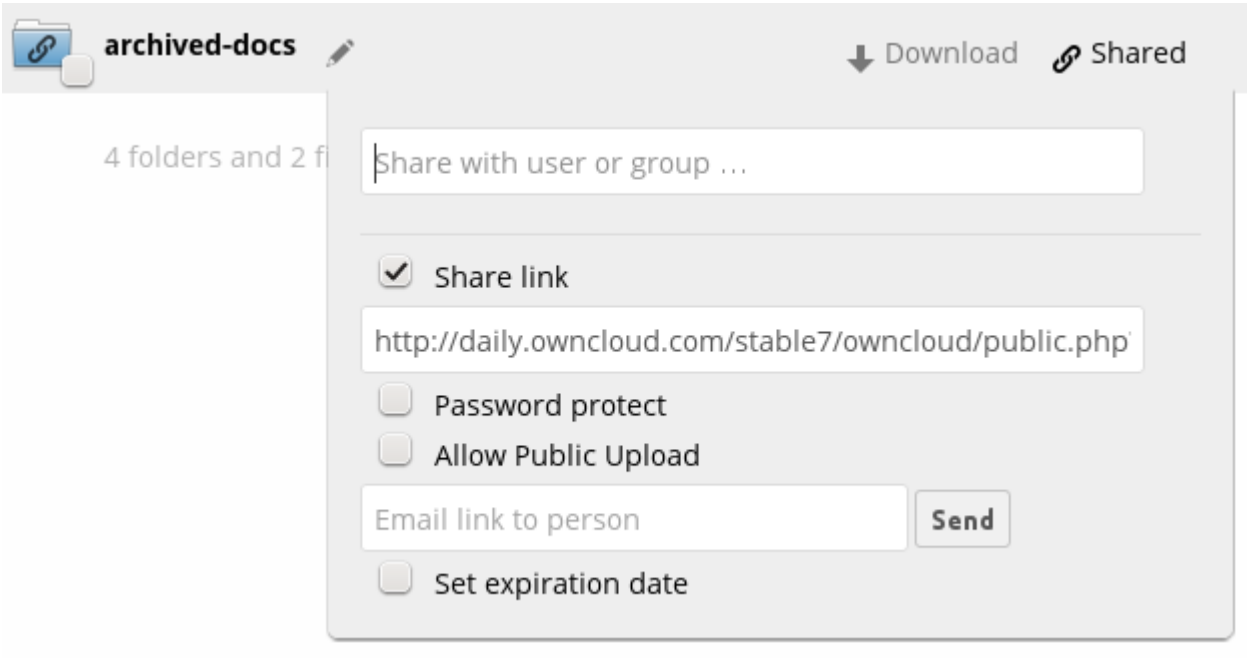
1. Go to `admin > Admin` in your ownCloud Web control panel, and scroll to the Remote Shares section.



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 create their own public shares by checking `Allow users to mount public link shares.` Leaving this unchecked disables user-created public shares.

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.

### 3.4.1 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`.

## 3.5 Defining Background Jobs

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

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

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

### 3.5.1 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

### 3.5.2 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 is does not require access to the system nor registration with a third party service. The disadvantage of this mechanism, when compared to the Webcron service, is that it requires regular visits to the page for it to be triggered.

#### Webcron

By registering your ownCloud `cron.php` script address at an external webcron service (for example, easyCron), you ensure that background jobs are executed regularly. To use this type of service, your server you must be able to access your server using the Internet. For example:

```
URL to call: http[s]://<domain-of-your-server>/owncloud/cron.php
```

#### Cron

Using the operating system cron feature is the preferred method for executing regular tasks. This method enables the execution of scheduled jobs without the inherent limitations the web server might have. 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
```

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.

## 3.6 Asset Management

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

# Original sentence: "In productive environments JavaScript and CSS files shall be delivered concatenated and minified."

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.

### 3.6.1 Parameters

```
<?php

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

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

## 3.7 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.

### 3.7.1 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",
```

## 3.8 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.

### 3.8.1 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 |

### 3.8.2 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*.

## 3.9 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

### 3.9.1 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",
```

## 3.10 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 installatins.

### 3.10.1 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.

### 3.10.2 Parameters

For setting up ownCloud to use any database, use the instructions in *Installation Wizard*. You should not have to edit the respective values in the `config/config.php`. However, in special cases (for example, if you want to connect your ownCloud instance to a database created by a previous installation of ownCloud), some modification might be required.

#### Configuring a MySQL or MariaDB Database

If you decide to use a MySQL or MariaDB database, ensure the following:

- That you have installed and enabled the MySQL extension in PHP
- That the **mysql.default_socket** points to the correct socket (if the database runs on same server as ownCloud).

---

**Note:** MariaDB is backwards compatible with MySQL. All instructions work for both. You will not need to replace mysql with anything.

---

The PHP configuration in `/etc/php5/conf.d/mysql.ini` could look like this:

```
# configuration for PHP MySQL module
extension=pdo_mysql.so
extension=mysql.so

[mysql]
mysql.allow_local_infile=On
mysql.allow_persistent=On
mysql.cache_size=2000
mysql.max_persistent=-1
mysql.max_links=-1
mysql.default_port=
mysql.default_socket=/var/lib/mysql/mysql.sock  # Debian squeeze: /var/run/mysqld/mysqld.sock
mysql.default_host=
mysql.default_user=
mysql.default_password=
mysql.connect_timeout=60
mysql.trace_mode=Off
```

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

To start the MySQL command line mode use:

```
mysql -uroot -p
```

Then a **mysql>** or **MariaDB [root]>** prompt will appear. Now enter the following lines and confirm them with the enter key:

```
CREATE USER 'username'@'localhost' IDENTIFIED BY 'password';
CREATE DATABASE IF NOT EXISTS owncloud;
GRANT ALL PRIVILEGES ON owncloud.* TO 'username'@'localhost' IDENTIFIED BY 'password';
```

You can quit the prompt by entering:

```
quit
```

An ownCloud instance configured with MySQL would contain the hostname on which the database is running, a valid username and password to access it, and the name of the database. The `config/config.php` as created by the *Installation Wizard* would therefore contain entries like this:

```
<?php

  "dbtype"        => "mysql",
  "dbname"        => "owncloud",
  "dbuser"        => "username",
  "dbpassword"    => "password",
  "dbhost"        => "localhost",
  "dbtableprefix" => "oc_",
```

### SQLite Database

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

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

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

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

```
<?php

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

### PostgreSQL Database

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

```
# configuration for PHP PostgreSQL module
extension=pdo_pgsql.so
extension=pgsql.so
```

```
[PostgresSQL]
pgsql.allow_persistent = On
pgsql.auto_reset_persistent = Off
pgsql.max_persistent = -1
pgsql.max_links = -1
pgsql.ignore_notice = 0
pgsql.log_notice = 0
```

The default configuration for PostgreSQL (at least in Ubuntu 14.04) is to use the peer authentication method. Check `/etc/postgresql/9.3/main/pg_hba.conf` to find out which authentication method is used in your setup. To start the postgres command line mode use:

```
sudo -u postgres psql -d template1
```

Then a **template1=#** prompt will appear. Now enter the following lines and confirm them with the enter key:

```
CREATE USER username CREATEDB;
CREATE DATABASE owncloud OWNER username;
```

You can quit the prompt by entering:

```
\q
```

An ownCloud instance configured with PostgreSQL would contain the path to the socket on which the database is running as the hostname, the system username the php process is using, and an empty password to access it, and the name of the database. The `config/config.php` as created by the *Installation Wizard* would therefore contain entries like this:

```
<?php

  "dbtype"        => "pgsql",
  "dbname"        => "owncloud",
  "dbuser"        => "username",
  "dbpassword"    => "",
  "dbhost"        => "/var/run/postgresql",
  "dbtableprefix" => "oc_",
```

**Note:** The host actually points to the socket that is used to connect to the database. Using localhost here will not work if postgreSQL is configured to use peer authentication. Also note, that no password is specified, because this authentication method doesn't use a password.

If you use another authentication method (not peer), you'll need to use the following steps to get the database setup: Now you need to create a database user and the database itself by using the PostgreSQL command line interface. The database tables will be created by ownCloud when you login for the first time.

To start the postgres command line mode use:

```
psql -hlocalhost -Upostgres
```

Then a **postgres=#** prompt will appear. Now enter the following lines and confirm them with the enter key:

```
CREATE USER username WITH PASSWORD 'password';
CREATE DATABASE owncloud TEMPLATE template0 ENCODING 'UNICODE';
ALTER DATABASE owncloud OWNER TO username;
GRANT ALL PRIVILEGES ON DATABASE owncloud TO username;
```

You can quit the prompt by entering:

```
\q
```

An ownCloud instance configured with PostgreSQL would contain the hostname on which the database is running, a valid username and password to access it, and the name of the database. The `config/config.php` as created by the *Installation Wizard* would therefore contain entries like this:

```
<?php

  "dbtype"        => "pgsql",
  "dbname"        => "owncloud",
  "dbuser"        => "username",
  "dbpassword"    => "password",
```

```
  "dbhost"        => "localhost",
  "dbtableprefix" => "oc_",
```

### Oracle Database

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

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

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

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

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

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

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

```
sqlplus system AS SYSDBA
```

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

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

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

You can quit the prompt by entering:

```
exit
```

An ownCloud instance configured with Oracle would contain the hostname on which the database is running, a valid username and password to access it, and the name of the database. The `config/config.php` as created by the *Installation Wizard* would therefore contain entries like this:

```
<?php

  "dbtype"        => "oci",
  "dbname"        => "XE",
  "dbuser"        => "owncloud",
  "dbpassword"    => "password",
  "dbhost"        => "localhost",
```

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

### 3.10.3 Troubleshooting

#### How can I find out if my MySQL/PostgreSQL server is reachable?

To check the server's network availability, use the ping command on the server's host name (db.server.com in this example):

```
ping db.server.dom

PING db.server.dom (ip-address) 56(84) bytes of data.
64 bytes from your-server.local.lan (192.168.1.10): icmp_req=1 ttl=64 time=3.64 ms
64 bytes from your-server.local.lan (192.168.1.10): icmp_req=2 ttl=64 time=0.055 ms
64 bytes from your-server.local.lan (192.168.1.10): icmp_req=3 ttl=64 time=0.062 ms
```

For a more detailed check whether the access to the database server software itself works correctly, see the next question.

#### How can I find out if a created user can access a database?

The easiest way to test if a database can be accessed is by starting the command line interface:

**SQLite**:

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

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

**MySQL**:

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

```
mysql -uUSERNAME -p
```

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

```
mysql -uUSERNAME -p -h HOSTNAME

mysql> SHOW VARIABLES LIKE "version";
+---------------+--------+
| Variable_name | Value  |
+---------------+--------+
| version       | 5.1.67 |
+---------------+--------+
```

```
1 row in set (0.00 sec)
mysql> quit
```

**PostgreSQL**:

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

```
psql -Uusername -downcloud
```

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

```
psql -Uusername -downcloud -h HOSTNAME

postgres=# SELECT version();
PostgreSQL 8.4.12 on i686-pc-linux-gnu, compiled by GCC gcc (GCC) 4.1.3 20080704 (prerelease), 32-bit
(1 row)
postgres=# \q
```

**Oracle**:

On the machine where your Oracle database is installed, type:

```
sqlplus username

SQL> select * from v$version;

BANNER
--------------------------------------------------------------------------------
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
PL/SQL Release 11.2.0.2.0 - Production
CORE  11.2.0.2.0      Production
TNS for Linux: Version 11.2.0.2.0 - Production
NLSRTL Version 11.2.0.2.0 - Production

SQL> exit
```

### Useful SQL commands

**Show Database Users**:

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

**Show available Databases**:

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

**Show ownCloud Tables in Database**:

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

**Quit Database**:

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

## 3.11 Use Server-Side Encryption

ownCloud ships En encryption app, which allows to encrypt all files stored in your ownCloud. Encryption and decryption always happen on the server-side. This enables the user to continue to use all the other apps to view and edit his data. The Encryption app is meant to protect user data on external storage.

The app uses the user's log-in password as encryption-password. This means that by default the user will lose access to his files if he loses his log-in password.

It might be a good idea to make regular backups of all encryption keys. 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 from all users)
- data/<user>/files_encryption (users' private keys and all other keys necessary to decrypt the users' files)

---

**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. This would require client side encryption, which this app does not provide. Read this blog post for more details.

---

### 3.11.1 Enable File Recovery Feature

The admin can offer the user some kind of protection against password loss. Therefore, you have to enable the recovery key in the admin settings and provide a strong recovery key password. The admin settings also enable you to change the recovery key password if you wish. But you should make sure to never lose this password because that's the only way to recover users' files.

Once the recovery key was enabled, every user can choose in his personal settings to enable this feature or not.

### 3.11.2 Recover User Files

If the recovery feature was enabled, the admin will see an additional input field at the top of the user management settings. After entering the recovery-key password the admin can change the user's log-in password which will automatically recover the user's file.

If you use a user back-end which doesn't allow you to change the log-in password directly within ownCloud, e.g. the LDAP back-end, than you can follow the same procedure to recover a user's files. The only difference is that you need to change the log-in password additionally at your back-end. In this case make sure to use both times the same password.

### 3.11.3 LDAP and other external user back-ends

If you configure an external user back-end you will be able to change the user's log-in password at the back-end. Since the encryption password must be the same as the user's log-in password this will result in a non-functional encryption

system. If the recovery feature was enabled, the administrator will be able to recover the user's files directly over the recovery feature. See the description above. Otherwise, the user will be informed that his log-in password and his encryption password no longer matches after his next log-in. In this case, the user will be able to adjust his encryption password in the personal settings by providing both, his old and his new log-in password.

## 3.12 Knowledge Base Configuration

The usage of ownCloud is more or less self explaining but nevertheless a user might run into a problem where he needs to consult the documentation or knowledge base. To ease access to the ownCloud documentation and knowledge base, a help menu item is shown in the settings menu by default.

### 3.12.1 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

## 3.13 Language Configuration

In normal cases ownCloud will automatically detect the language of the Web-GUI. If this does not work properly or you want to make sure that ownCloud always starts with a given language, you can use the **default_language** parameter.

Please keep in mind, that this will not effect a users language preference, which has been configured under "personal -> language" once he has logged in.

Please check `settings/languageCodes.php` for the list of supported language codes.

### 3.13.1 Parameters

```
<?php

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

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

## 3.14 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.

### 3.14.1 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",
```

## 3.15 Mail Configuration

ownCloud does not contain a full email program but contains some parameters to allow to send e.g. password reset email to the users. This function relies on the PHPMailer library. To take advantage of this function it needs to be configured properly.

### 3.15.1 Requirements

Different requirements need to be matched, depending on the environment which you are using and the way how you want to send email. You can choose between **SMTP**, **PHP mail**, **Sendmail** and **qmail**.

### 3.15.2 Parameters

All parameters need to be set in `config/config.php`

#### 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 server can be entered separated by semicolon:

```
<?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 it.

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

### 3.15.3 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.

### 3.15.4 Using Email Templates

As an added convenience to administrators, ownCloud provides several Email templates that you can use for sending messages to users.

#.. figure:: ../images/remote_shares.png

Found on the Admin page, you can choose from the following templates:

- Sharing email (http) – You can use this template to send emails to users about sharing links.
- Sharing email – You can use this template to send emails to users about sharing files.
- Lost password mail – When managing users, you can use this template to send emails to users about lost password recovery.
- Activity notification mail – You can use this template to send emails to users detailing their ownCloud activity.

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.

   **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.

5. Click `Save` to the file modifications.

   Once complete, the files are sent to users who choose to receive notifications through email.

   **Note:** ownCloud populates the variables with usernames and filenames prior to sending the email.

### 3.15.5 Troubleshooting

**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=3.64 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
```

**Question**: How can I find out if the SMTP server is listening on a specific TCP port?

**Answer**: SMTP servers usually listen on port **25/tcp** (smtp). In rare circumstances the SMTP server also listens on the outdated port **465/tcp** (smtps). You can use the `telnet` command to determining if a port is available:

```
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 22:28:14 +0100
```

**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/tcp**. See the previous answer to use the `telnet` command for checking the port availability.

**Question**: How can I determine if the SMTP server supports the TLS protocol?

**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
250-STARTTLS                                                  # <<< STARTTLS is supported!
250 HELP
QUIT                                                          # <<< enter this command
221 smtp.domain.dom closing connection
Connection closed by foreign host.
```

**Question**: How can I determine which authentication types or methods the SMTP server supports?

**Answer**: SMTP servers usually announce the available authentication types or methods immediately following the establishment of a connection. You can easily check this using the telnet command.

**Note:** You must enter the marked lines to obtrain 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                                 # <<< available Authentication
250-STARTTLS
250 HELP
QUIT                                                          # <<< enter this command
221 smtp.domain.dom closing connection
Connection closed by foreign host.
```

### 3.15.6 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:

```
.. code-block:: 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.

## 3.16 Preview Configuration

ownCloud 6 introduced the new thumbnail system. It is used to generate thumbnails from various file types. By default, it can generate previews for:

- Images
- Movies
- Cover from mp3 files
- various office files
- Pdf
- Svg
- Text

### 3.16.1 Soft dependencies:

#### imagick:

ownCloud needs the imagick PHP extension to generate previews from office, PDF and SVG files. For further information on how to install the imagick PHP extension on your system take a look at the PHP documentation. If imagick is not installed, ownCloud will show file type icons instead of previews.

#### LibreOffice / OpenOffice:

ownCloud comes with a php-only preview system for office files. But this preview system has limited capabilities and is only able to create previews from basic Microsoft Office files. If you need previews from advanced Microsoft Office files or OpenDocument files, you have to install LibreOffice or OpenOffice. To learn more about installing LibreOffice/OpenOffice consider your distribution's documentation.

#### avconv / ffmpeg:

ownCloud requires avconv of ffmpeg to generate previews from movies. To learn more about installing avconv or ffmpeg consider your distribution's documentation.

### 3.16.2 Parameters

#### Disabling previews:

Under certain circumstances like a big user base or limited resources you might want to consider disabling previews.

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

There is a config option called 'enable_previews'. By default it's set to true. You can disable previews by setting this option to false:

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

### Maximum preview size:

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

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

By default, both config options are set to null. 'Null' is equal to no limit. Numeric values represent the size in pixel. The following code limits previews to a maximum size of 100px by 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 you have a lot of small pictures and the preview system generates blurry previews, you might want to consider setting a maximum scale factor. By default, ownCloud scales pictures up 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,
```

### LibreOffice / OpenOffice:

You can set a custom path for the LibreOffice binary. If LibreOffice is not yet available on your system, you can also use OpenOffice instead.

```
<?php
  'preview_libreoffice_path' => '/usr/bin/libreoffice',
```

You can set custom LibreOffice / OpenOffice command line parameters by setting the preview_office_cl_parameters option.

```
<?php
  'preview_office_cl_parameters' => ' ',
```

## 3.17 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.

### 3.17.1 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.

### 3.17.2 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.

---

## 3.18 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:

### 3.18.1 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

### 3.18.2 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.

## 3.19 Enabling uploading big files

Note: The order of the following steps is important! If you swap steps described below, the settings may fail.

**Go to the admin section in the ownCloud Web Interface and do the following:**

- Under "File handling" set the Maximum upload size to the desired value (e.g. 16GB)
- Click the "save"-Button

**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** allows you to get performance benefits in some setups. Please make sure you know what you are doing before using it in production. As previously mentioned, add this option in your .htaccess file or edit your php.ini file:

- `output_buffering = 16384` (e.g., to stay consistent with the example value above)

As you can see, the "output_buffering" has to be given in MegaBytes but as a plain figure (without size-units as 'M' or 'G')

**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

## 3.20 Custom Mount Configuration (GUI)

ownCloud provides the ability to mount an external storage device. The external storage devices serves as a secondary storage device within ownCloud.

The ownCloud Admin has the ability to create such a mount. In addition, the ownCloud Admin may decide to provide the end user the ability to create the mount. The mounts may be created on a per-user, per group, or all user basis.

---

**Note:** Using `$user` in any option gets replaced by the current user. This can be used e.g to specify users' own paths easily by just writing `/home/$user/`.

---

### 3.20.1 Supported mounts

The following lists the supported storage types.

- Local
- Amazon S3
- Dropbox
- FTP
- Google Drive
- OpenStack Object Storage
- SMB/CIFS
- ownCloud/WebDAV
- SFTP
- iRODS

### 3.20.2 Configuration

### 3.20.3 Enable the app

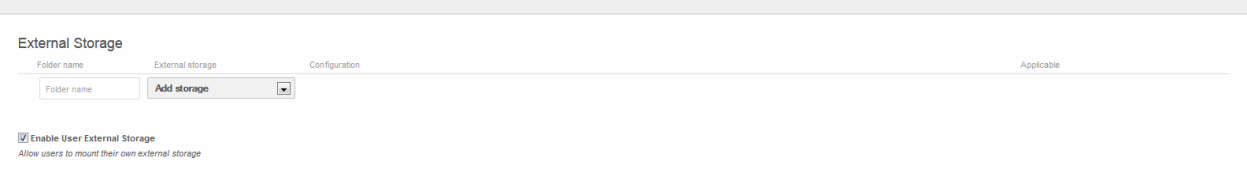
From the `APPs` Page within ownCloud, select `External Storage Support` and enable.


### 3.20.4 Configure mounts

As stated previously, the Admin has the ability to configure these mounts, as well as decide whether an end user can configure mounts for themselves. For the Admin, the configuration is performed in the `Admin` page. For end users,

the configuration is performed in the `Personal` Page. This document will discuss how the Admin configures the mounts, however, the configuration is the same for the end user.

On the `Admin` page, scroll to External Storage:

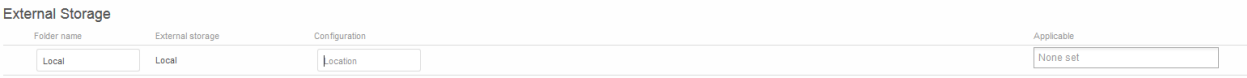



### Enable users to mount their own devices

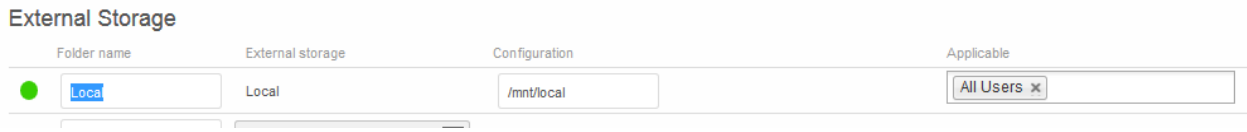
In order to allow end users to mount their own devices, select the radio button next to Enable User External Storage.

### Local Storage

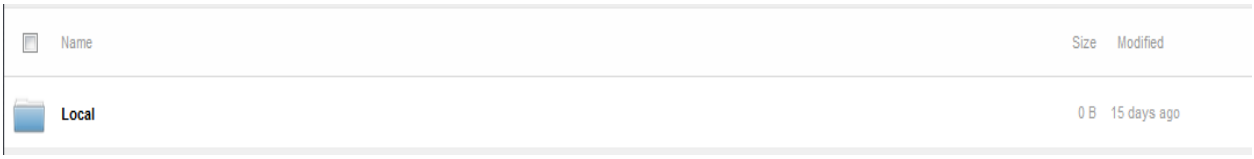
This is used to mount storage that is outside ownCloud's data directory




- Location – The directory to mount
- Applicable – A list users of who can see this mount

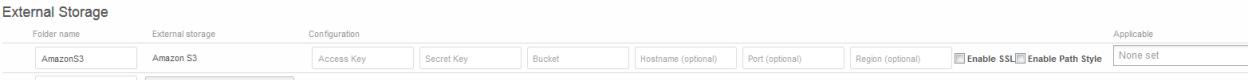



Note: When configured correctly, a *Green Light* will appear next to the Folder Name. If misconfigured, a *Red Light* will appear.



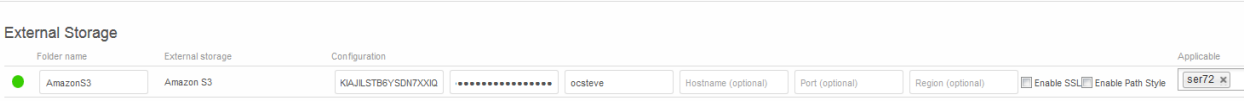

### Amazon S3

This is used to mount to an S3 server

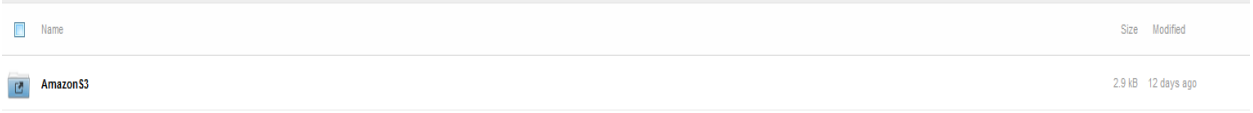



- Access Key – The access key provided by the S3 storage provider
- Secret Key – The secret key provided by the S3 storage provider
- Bucket – The bucket created within the S3 storage server
- Hostname (optional) – The host of the s3 storage server

- Port (optional) – The port to communicate to the host on
- Region (optional) – The region where the storage exists
- Applicable – A list of users who can see this mount




**Note:** When configured correctly, a Green Light will appear next to the Folder Name. If misconfigured, a Red Light will appear.



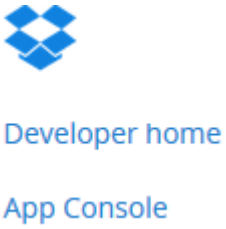

### Dropbox

Mounts a dropbox in the Dropbox cloud into the virtual file system.

#### Configure DropBox

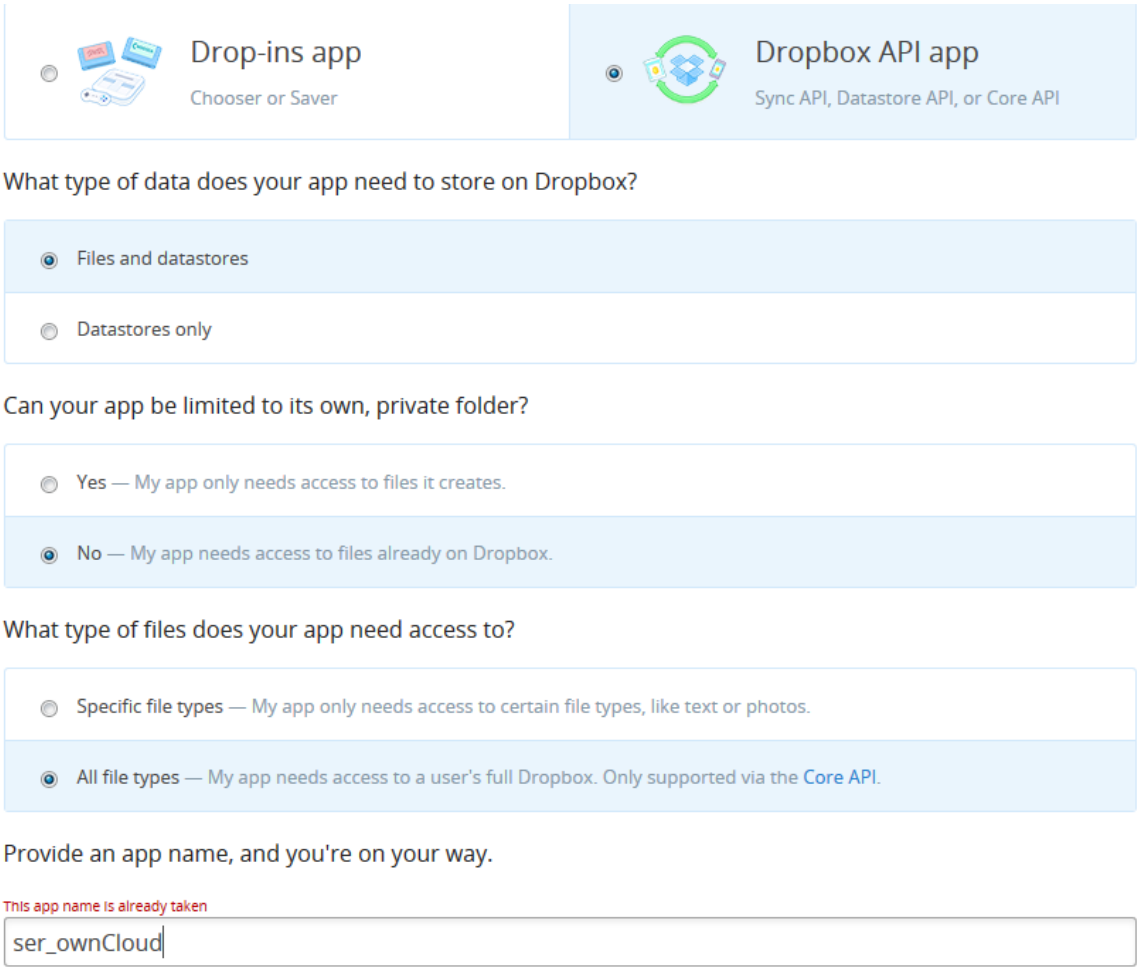
Log onto the Dropbox Developers page:

Select App Console:



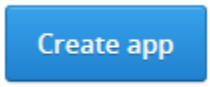

This will ask you to accept terms and conditions.

Select Dropbox API and configure down the page as follows:



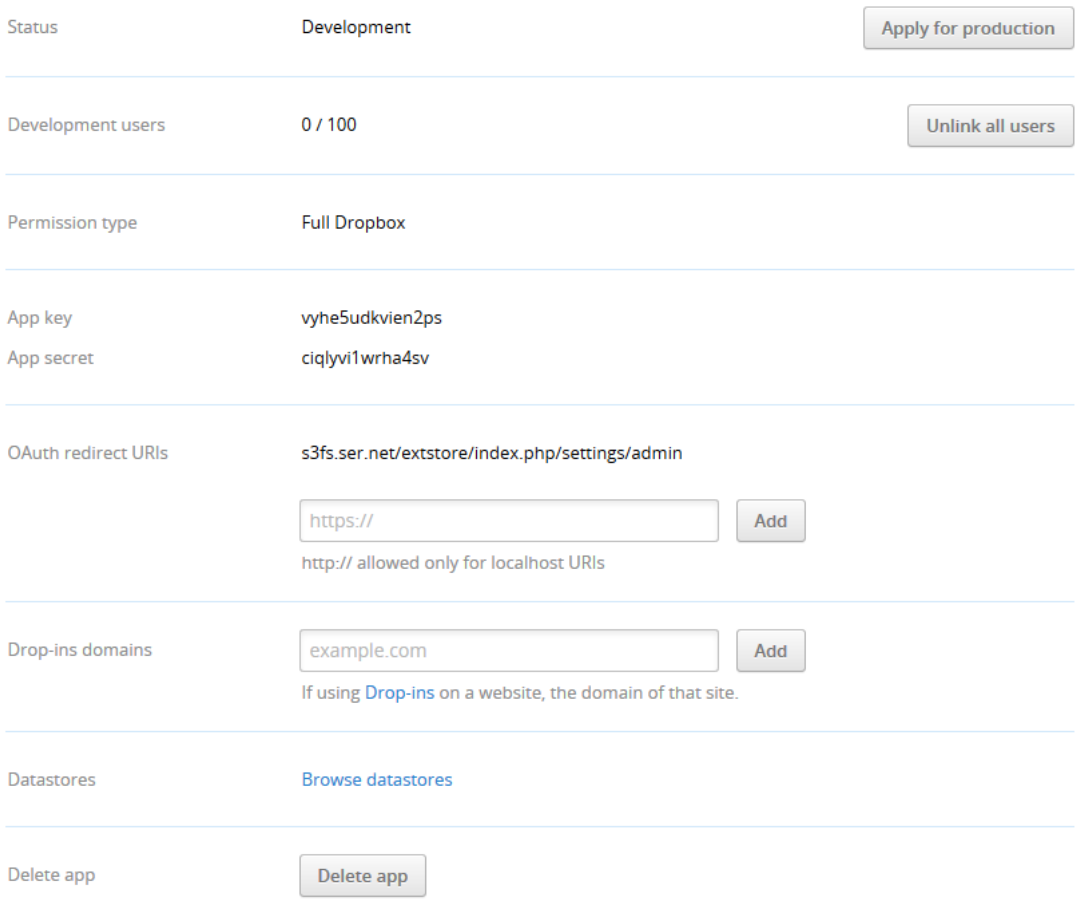

The name can be any unique name desired.

Select Create App



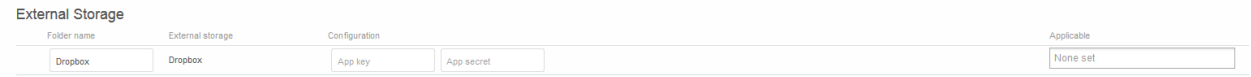

Enter the OAuth redirect URI as follows:

```
http://<ownCloud instance>/index.php/settings/personal
http://<ownCloud instance>/index.php/settings/admin
```

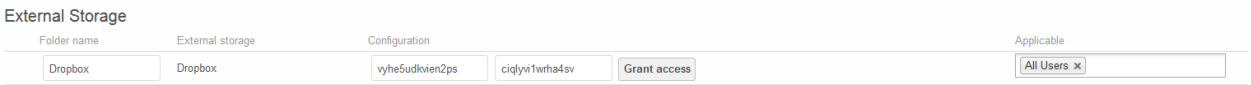



Take note of the App Key and App Secret and enter into ownCloud.

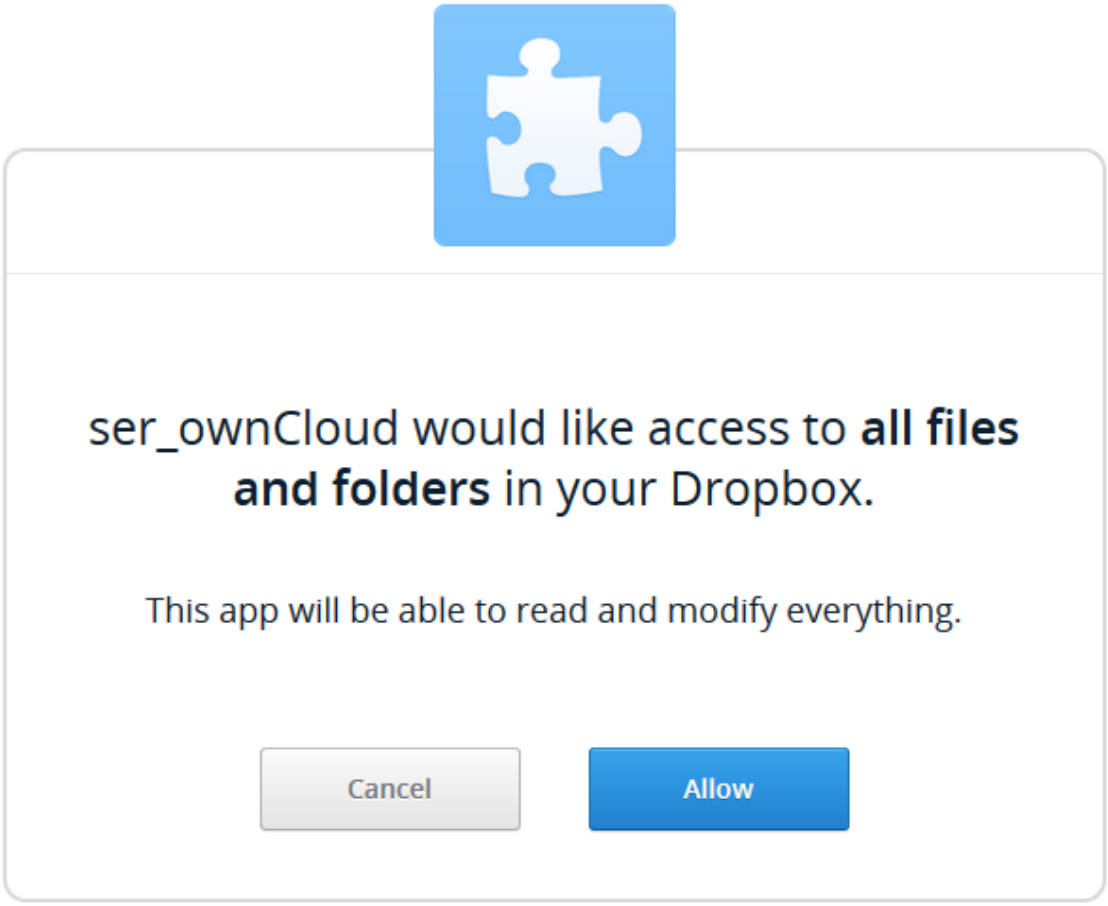
### ownCloud Configuration



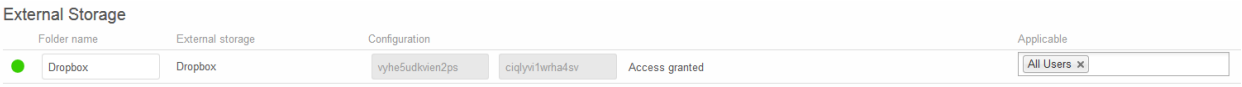

- App key – The app key to login to your Dropbox
- App secret – The app secret to login to your Dropbox
- Applicable – A list users of who can see this mount



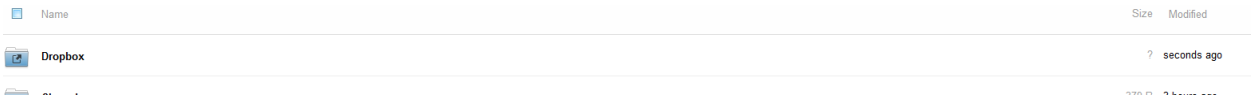

Select “Grant Access” and the following appears




Note if you are not logged into Dropbox, you will first be prompted to login. Select Allow.

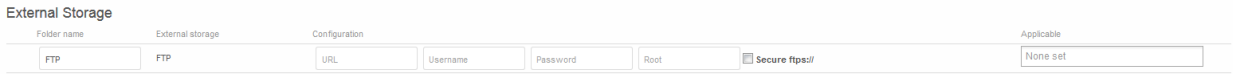



**Note:** When configured correctly, a Green Light will appear next to the Folder Name. If misconfigured, a Red Light will appear.



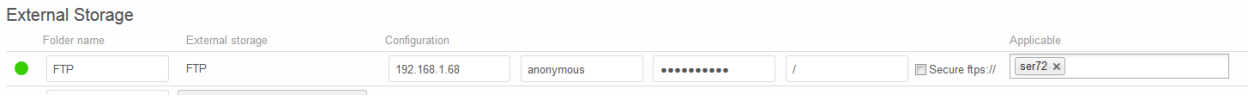

### FTP

Mounts a folder on a remote FTP or FTPS server

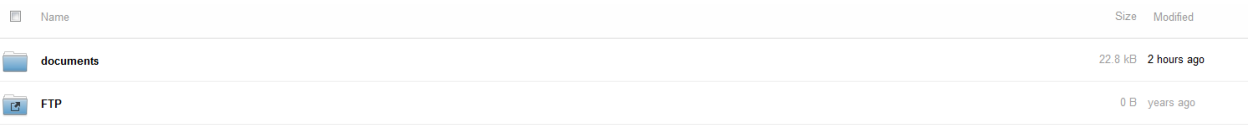



- URL – The hostname of the FTP/FTPS server

- Username – The username to login to the FTP/FTPS server
- Password – The password to login to the FTP/FTPS esrver
- Root – The folder inside the FTP/FTPS server to mount (optional – defaults to '/')
- Secure `ftps://` – Whether to use ftps:// to connect to the FTP server instead of `ftp://`
- Applicable – A list users of who can see this mount




**Note:** When configured correctly, a Green Light will appear next to the Folder Name. If misconfigured, a Red Light will appear.




### GoogleDrive

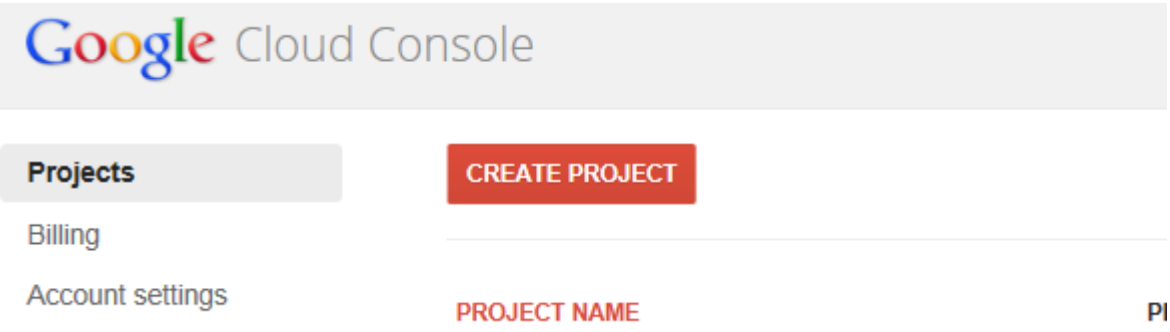
Mounts a share in the Google cloud.

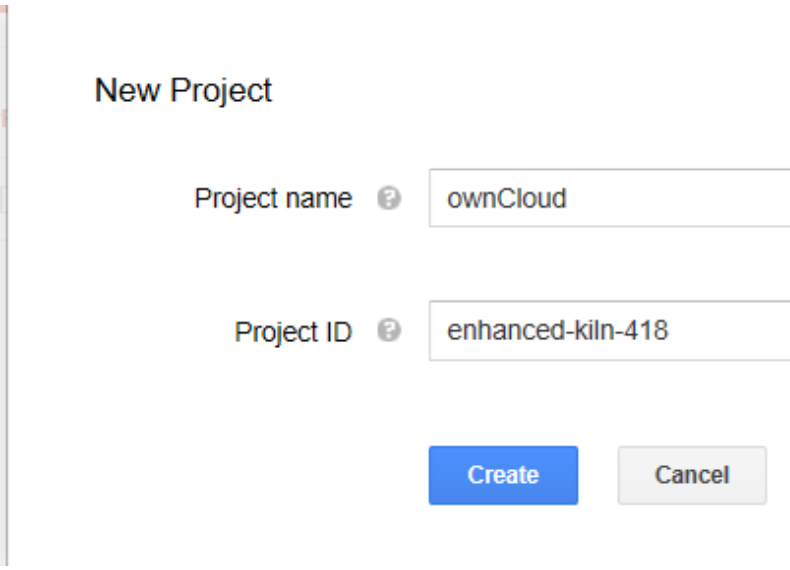
#### Configure GoogleDrive

All applications that access a Google API must be registered through the "Google Cloud Console". This can be accessed at the following URL:

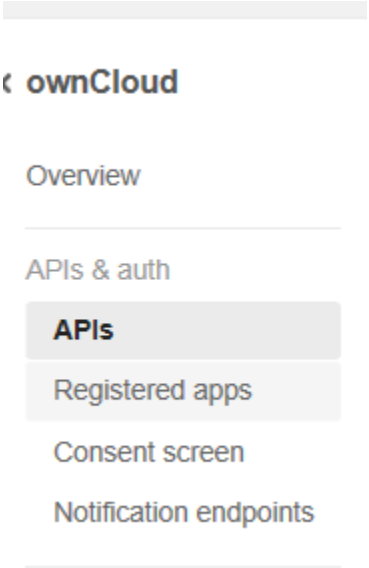
https://cloud.google.com

Once logged into Google, create a project by selecting Create Project






Enter a Project name and either keep or enter a new Project ID

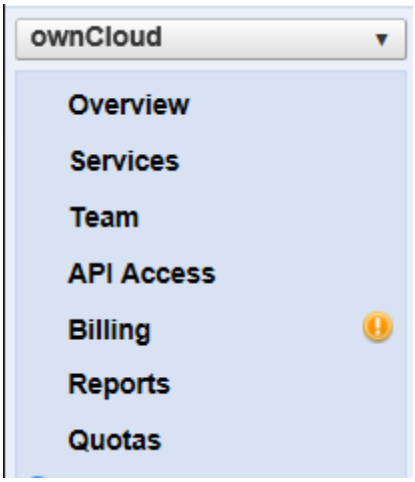



Select the project and choose the APIs & auth menu entry

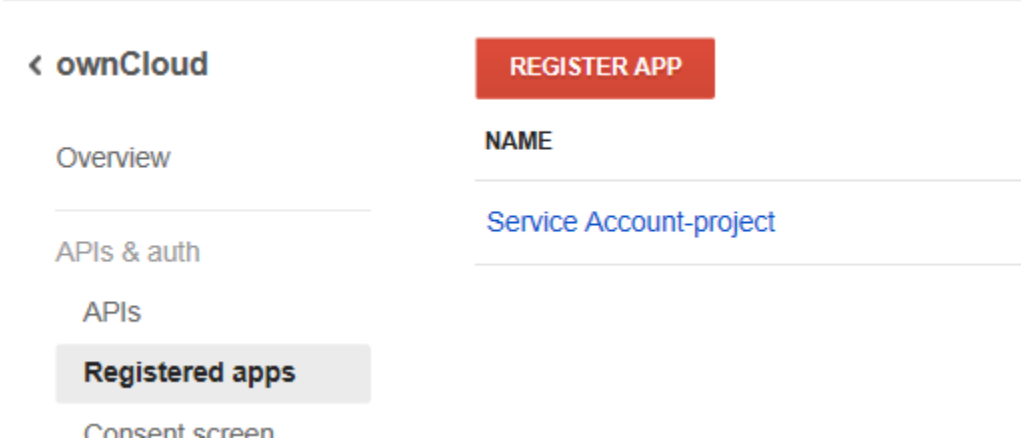
Enable `Drive API` and `Drive SDK` and then select the

next to either `Drive API` or `Drive SDK`

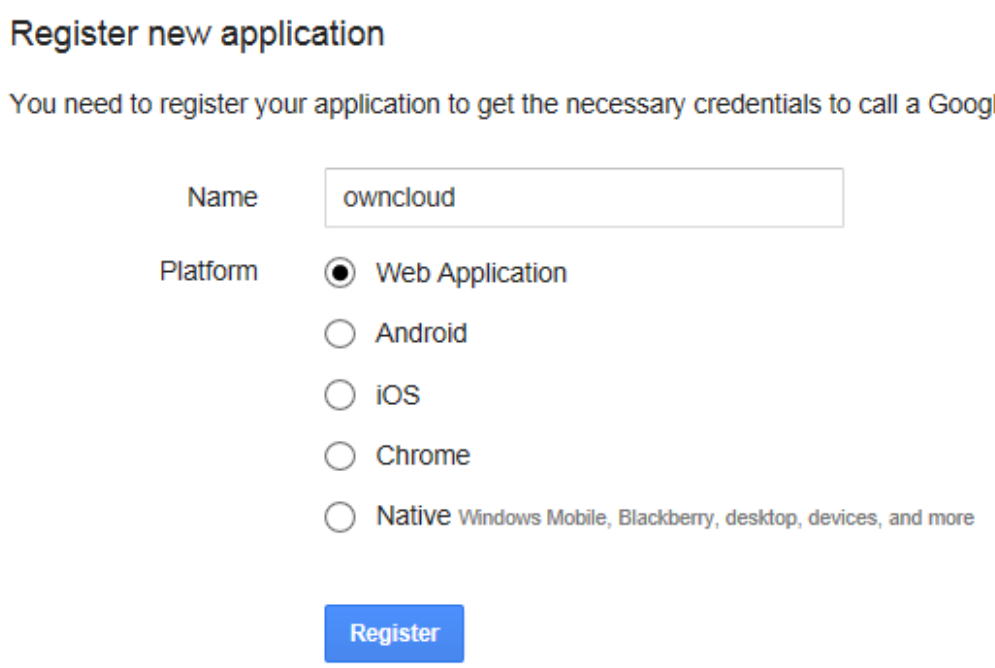

Select `API Access` on the menu




Select `REGISTER APP`




Enter a name and select `Web Application`

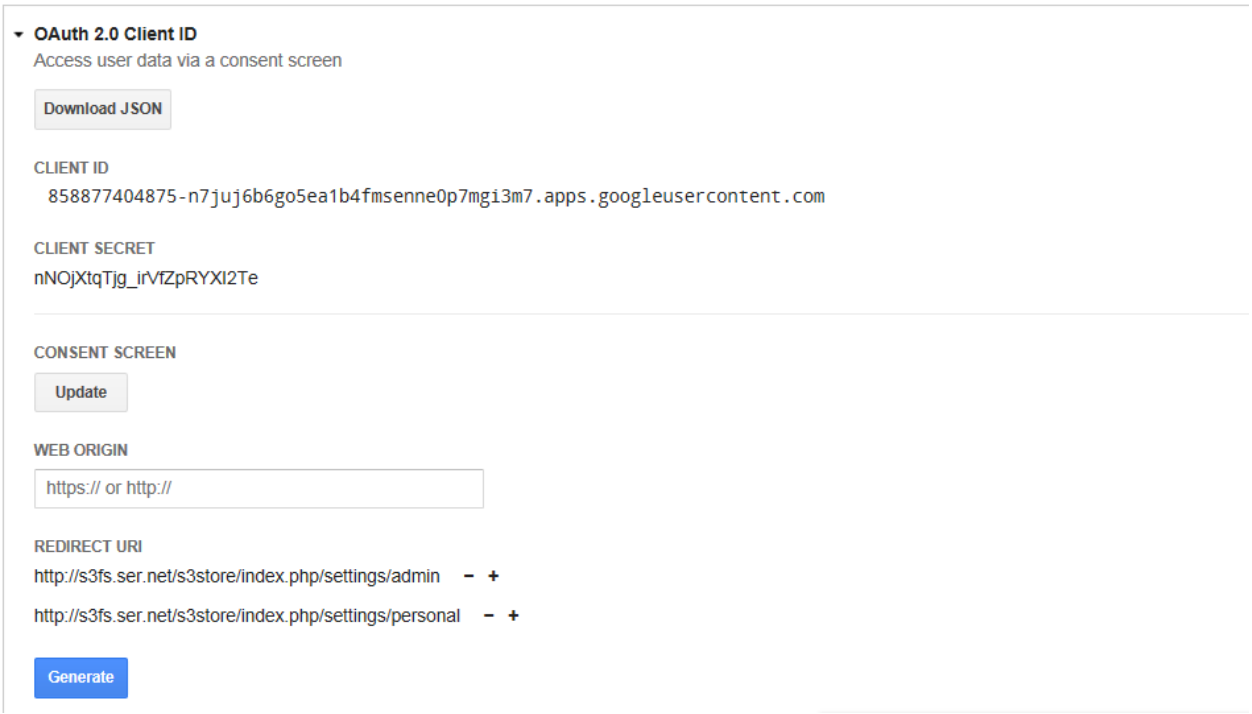



Expand `OAuth 2.0 Client ID` Enter the following in the `REDIRECT URI` field:

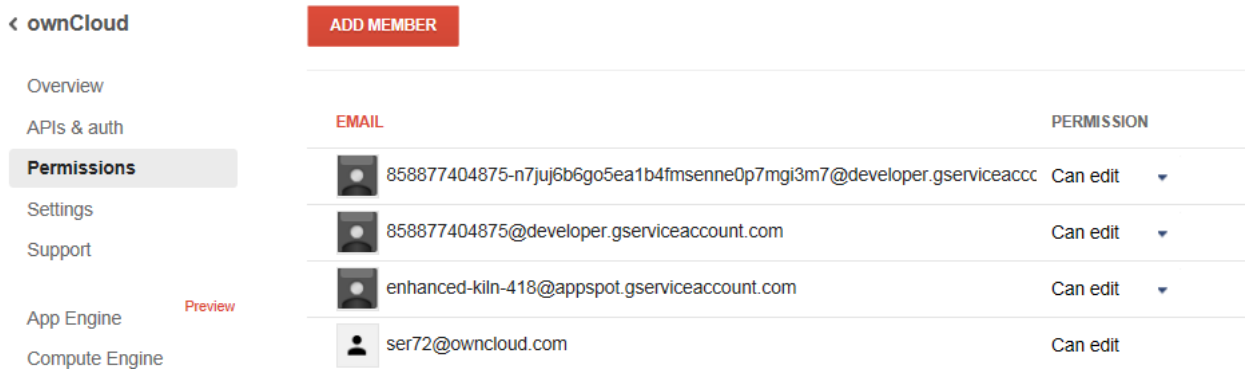
```
http://<ownCloud instance>/index.php/settings/personal
http://<ownCloud instance>/index.php/settings/admin
```

**Note:** The `<ownCloud instance>` must be a Fully Qualified Domain Name. It cannot be an IP address!

Select `Generate`

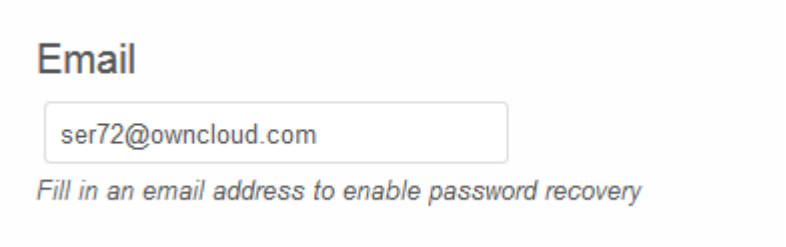


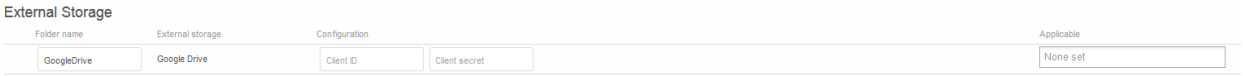
Verify that the required email addresses are in the Permissions tab




### Configure ownCloud

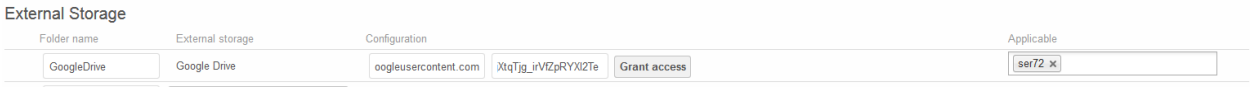
Prior to configuring the mount, an E-mail address needs to be configured in the `Personal` tab



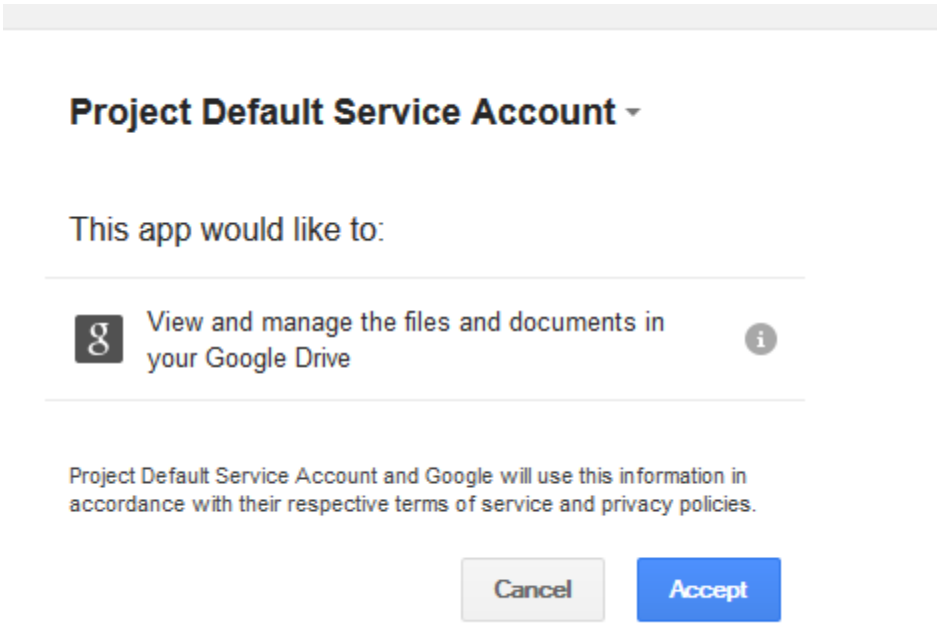



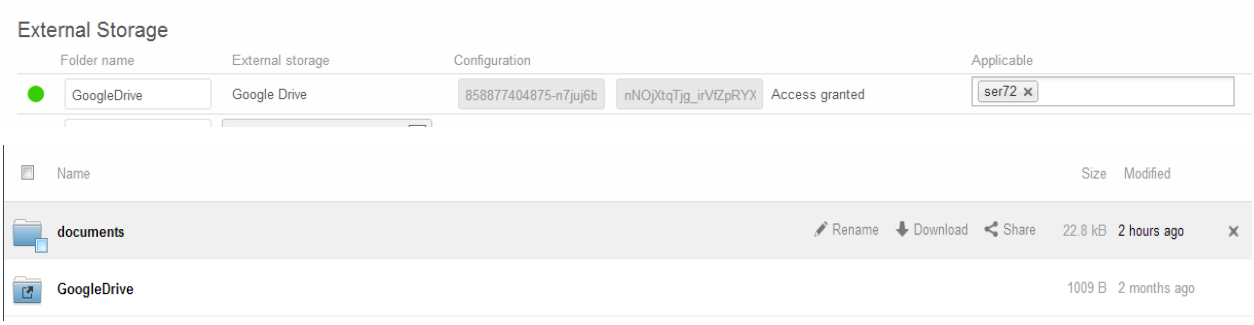

- Client ID – The client id to login to the Google Drive from `OAuth 2.0 Client ID` above
- Client secret – The client secret to login to the Google Drive from `OAuth 2.0 Client ID` above
- Applicable – A list users of who can see this mount

Once the required fields are filled in, a `Grant access` button appears. Select this button.




The following screen appears. Select `Accept`

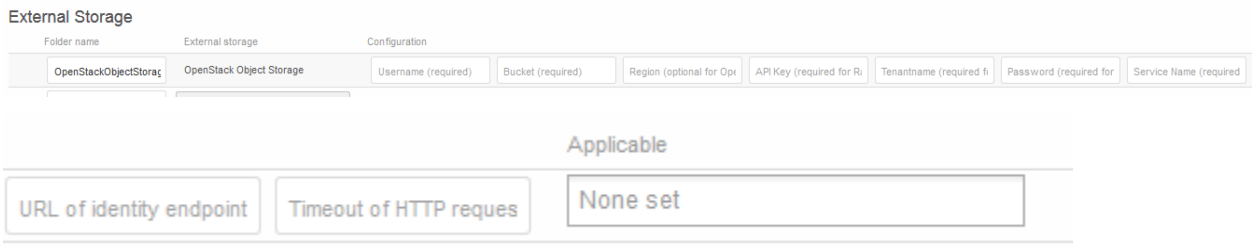



**Note:** When configured correctly, a Green Light will appear next to the Folder Name. If misconfigured, a Red Light will appear.

### OpenStack Object Storage

Mounts a container on an OpenStack Object Storage server.



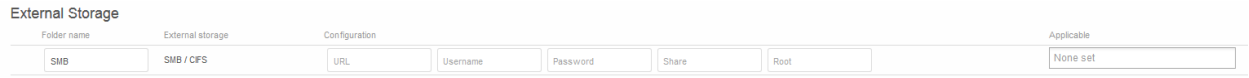

- Username
- Bucket
- Region

- API Key
- Tenantname
- Password
- Service Name
- URL of identity Endpoint
- Timeout of HTTP request
- Applicable – A list users of who can see this mount

**Note:** When configured correctly, a Green Light will appear next to the Folder Name. If misconfigured, a Red Light will appear.

### SMB/CIFS

Mounts a folder on a remote Samba server, NAS appliance, or Windows machine.



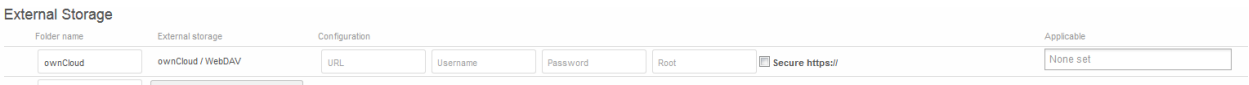

- URL – The host name of the Samba server.
- Username – The user name used to login to the Samba server.
- Password – The password to login to the Samba server.
- Share – The share on the Samba server to mount.
- Root – The folder inside the Samba share to mount (optional, defaults to '/')
- Applicable – A list users of who can see this mount


**Note:** When configured correctly, a Green Light will appear next to the Folder Name. If misconfigured, a Red Light will appear.

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


### ownCloud/WebDAV

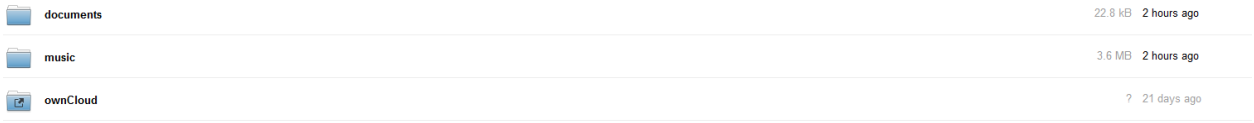
Mounts a folder on a WebDAV server (or another ownCloud instance via WebDAV).




- URL – The hostname of the WebDAV server.
- Username – The username used to login to the WebDAV server.

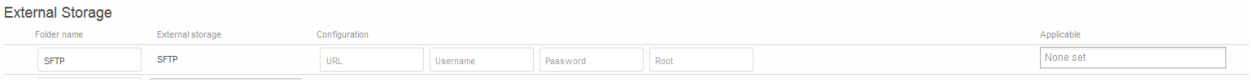
- Password – The password used to login to the WebDAV server.
- Root – The folder inside the WebDav server to mount (optional, defaults to '/')
- Secure `https://` - Whether to use `https://` to connect to the WebDav server instead of `http://`
- Applicable – A list users of who can see this mount


**Note:** When configured correctly, a Green Light will appear next to the Folder Name. If misconfigured, a Red Light will appear.

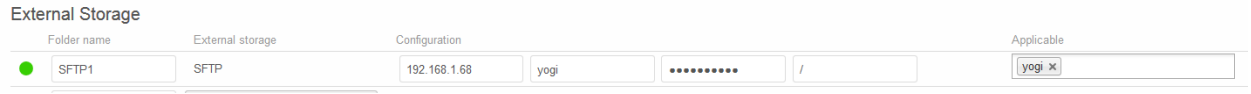



### SFTP

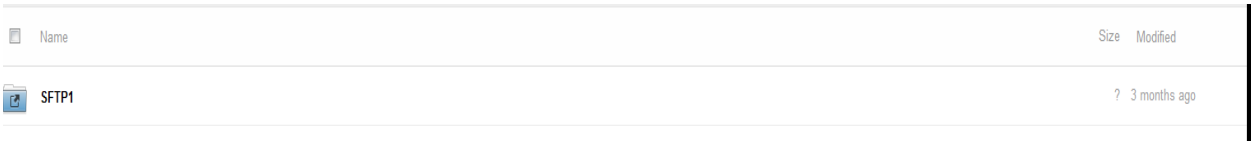
Mounts a folder on a remote SSH server.




- URL – The hostname of the SSH server.
- Username – The username used to login to the SSH server.
- Password – The password used to login to the SSH server.
- Root – The folder inside the SSH server to mount (optional, defaults to '/')
- Applicable – A list users of who can see this mount

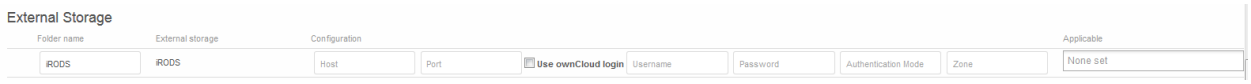



**Note:** When configured correctly, a Green Light will appear next to the Folder Name. If misconfigured, a Red Light will appear.




### iRODS

Mounts a folder on a iRODS server.




- Host

- Port
- Use ownCloud login
- Username
- Password
- Authentication Mode
- Zone
- Applicable – A list users of who can see this mount

---

**Note:** When configured correctly, a Green Light will appear next to the Folder Name. If misconfigured, a Red Light will appear.

---

### 3.20.5 Configuration File

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

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

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

Although configuration may be done by making modifications to the mount.json file, it is recommended to use the Web-GUI in the administrator panel (as described in the above section) to add, remove, or modify mount options in order to prevent any problems.

## 3.21 Custom Mount Configuration

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!

### 3.21.1 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
            }
        }
    }
}
```

### 3.21.2 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

### 3.21.3 Backends

#### Local Filesystem

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

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

##### Example

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

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

#### FTP (or FTPS)

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

- **host**: the hostname of the ftp server
- **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
}
```

## 3.22 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:

### 3.22.1 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}'
                               ),
    ),
),
```

### 3.22.2 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'
                               ),
    ),
),
```

## 3.23 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.

---

### 3.23.1 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

### 3.23.2 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

### 3.23.3 Nginx (X-Accel-Redirect)

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

#### Installation

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

#### Configuration

Configuration is similar to Apache:

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

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

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

```
#         alias /path/to/local-mountpoint;
#     }

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

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

}
```

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

### 3.23.4 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.

CHAPTER

FOUR

# MAINTENANCE

## 4.1 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.

### 4.1.1 Parameters

```
<?php

  "maintenance" => false,
```

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

## 4.2 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

### 4.2.1 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"`/
```

### 4.2.2 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
```

#### 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"`
```

## 4.3 Updating ownCloud

The Updater app provides a more automated method of updating ownCloud. To use the Updater app, it must be enabled in your ownCloud instance. The Updater is enabled in your ownCloud instance by default when you install.

To update ownCloud:

---

**Note:** To update ownCloud, the Updater app must be enabled in your ownCloud instance. The Updater app is enabled in your ownCloud instance by default when you install. However, to verify that it is enabled, or to enable the Updater app, see Enabling the Updater App.

---

1. Make a backup of the ownCloud folder and the database. See *Backing up ownCloud* for details.
2. Navigate to the 'Admin' page.
3. Click the 'Update' tab.
4. Refresh the page using Ctrl+F5.

If this procedure doesn't work (for example, ownCloud 5.0.10 doesn't show new any new version) you could try to perform a full upgrade to update to the latest point release (see below).

### 4.3.1 Verifying the Updater App is Enabled

However, to verify that the Updater is enabled in your ownCloud instance:

1. Select the "Admin" option from the "Personal Settings" dropdown menu.
2. Scroll down the resulting web page. If the Updater app appears in this window, the app is enabled. If not, then you must enable it. See Enabling the Updater App.

### 4.3.2 Enabling the Updater App

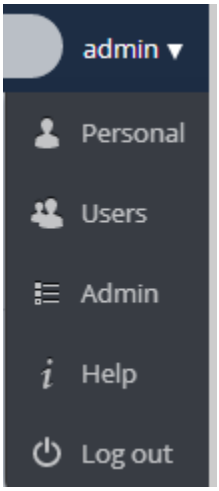
The Updater app is enabled in your ownCloud instance by default when you install. However, it is possible that it was disabled at some point. To enable the Updater app:

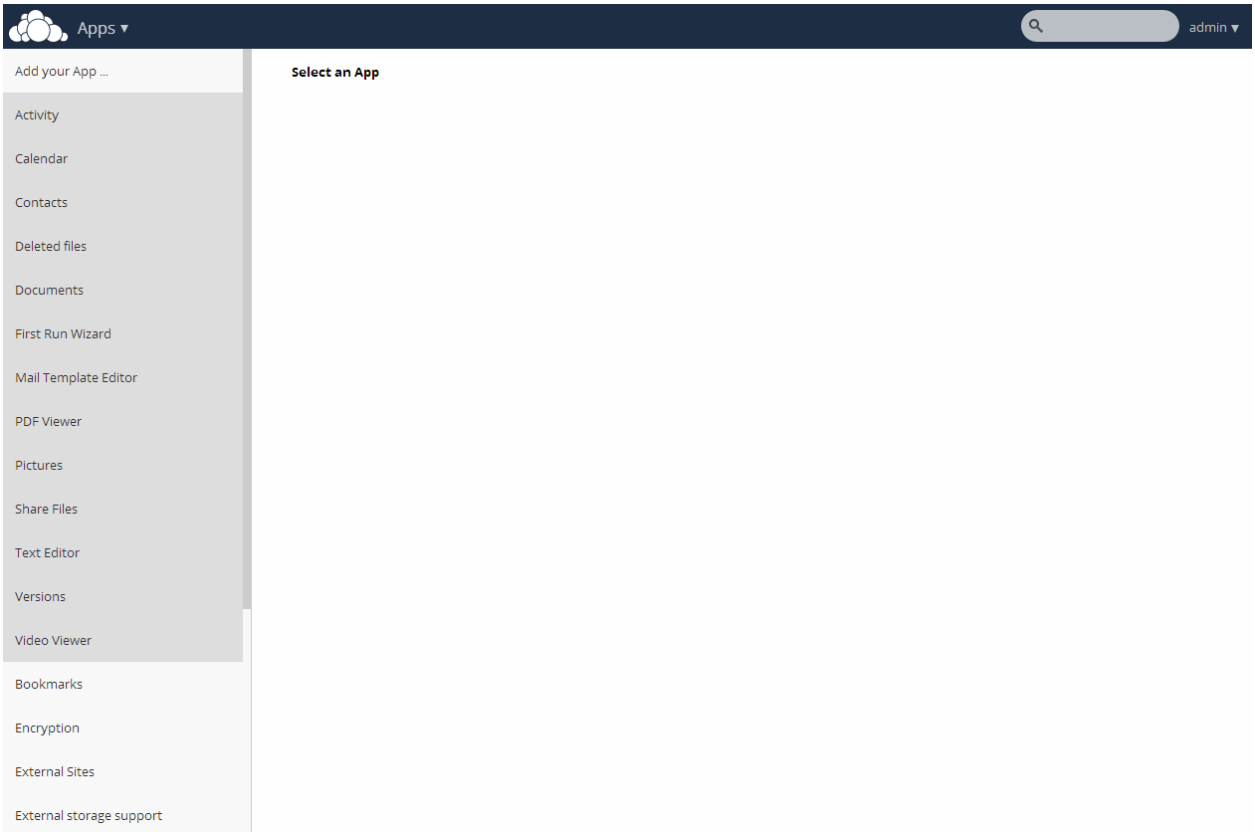
1. Click the "+ App" function in the Apps Selection Menu.

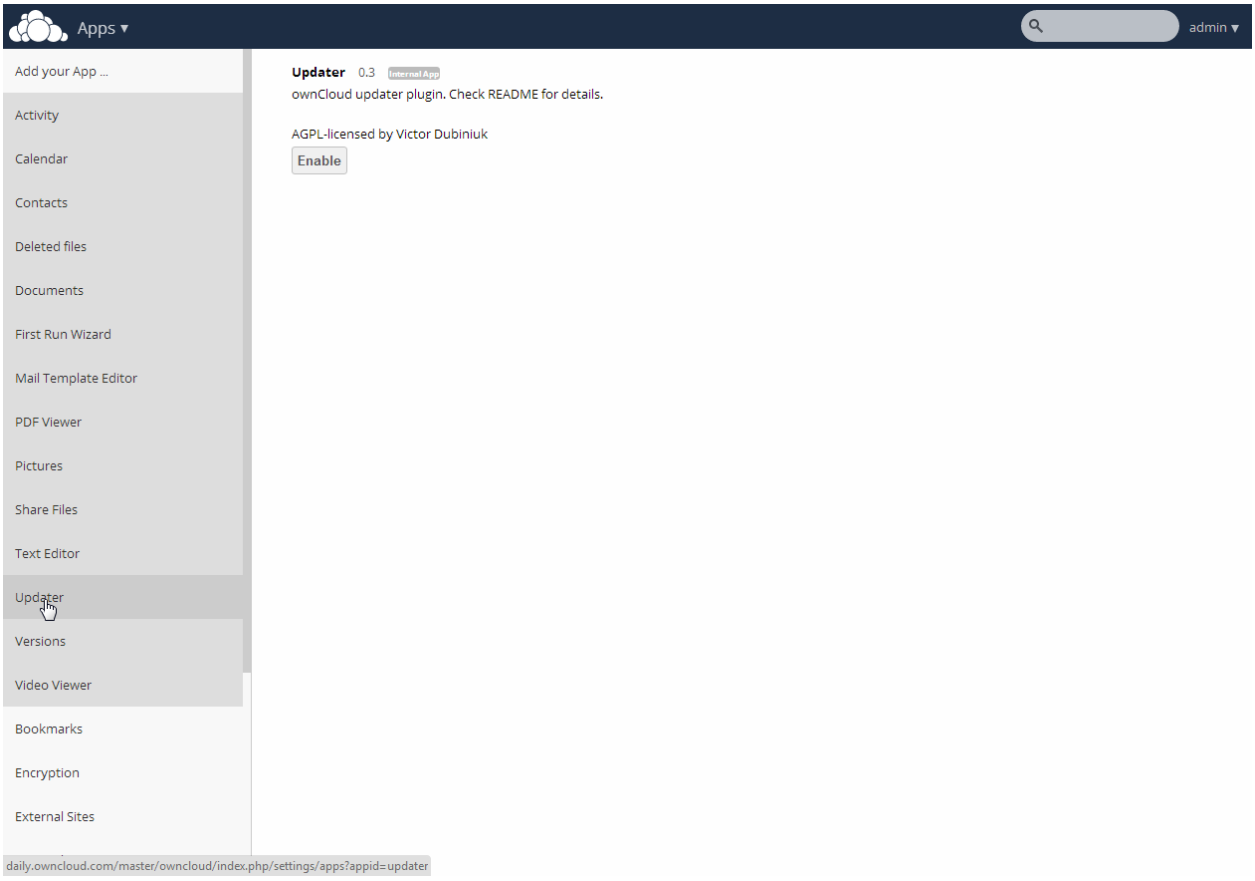
   The "Select an App" window opens.

   **Select an App window**

2. Scroll down the list of apps on the left side of the web page and select the Update app.

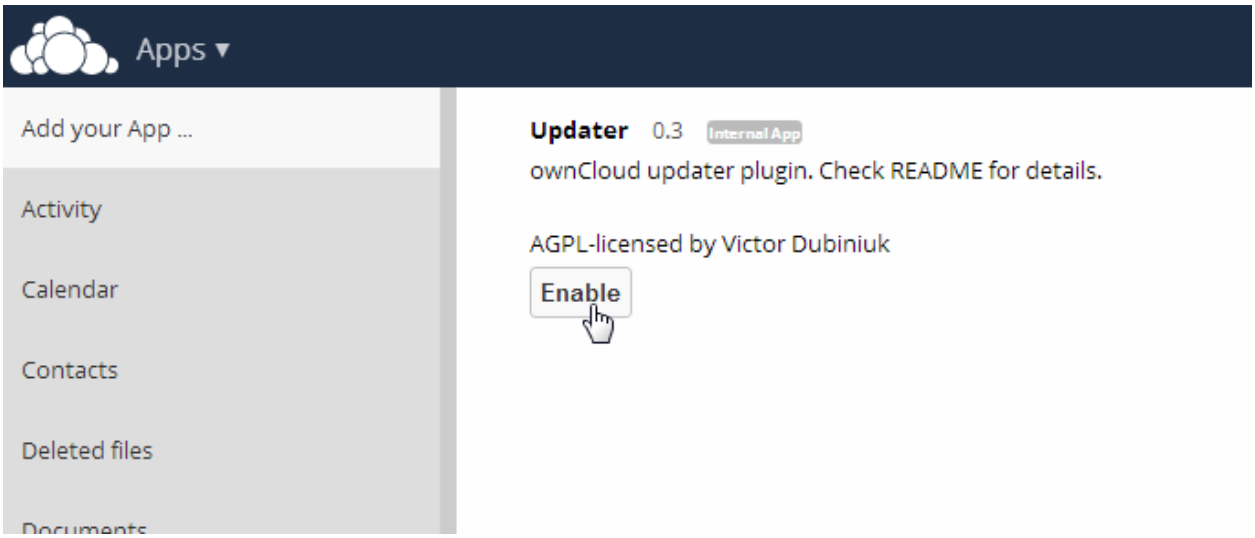

admin ▾
Personal
Users
Admin
Help
Log out


Apps ▾
admin ▾
Add your App ...
Activity
Calendar
Contacts
Deleted files
Documents
First Run Wizard
Mail Template Editor
PDF Viewer
Pictures
Share Files
Text Editor
Versions
Video Viewer
Bookmarks
Encryption
External Sites
External storage support
Select an App


Apps ▾
admin ▾
Add your App ...
Activity
Calendar
Contacts
Deleted files
Documents
First Run Wizard
Mail Template Editor
PDF Viewer
Pictures
Share Files
Text Editor
Updater
Versions
Video Viewer
Bookmarks
Encryption
External Sites
Updater 0.3 Internal App
ownCloud updater plugin. Check README for details.
AGPL-licensed by Victor Dubiniuk
Enable
daily.owncloud.com/master/owncloud/index.php/settings/apps?appid=updater

**Selecting the Updater app**

3. In the App View window, click "Enable."




**Enabling the Updater app**

ownCloud enables the Updater app.

## Upgrading the ownCloud Server

The process for upgrading the ownCloud Server is fairly straightforward but requires planning and proper file and folder management.

To upgrade your ownCloud Server:

1. Ensure that you are running the latest point release of your current major ownCloud version (for example, point release 5.0.14a in the version 5.0 series). To update to the latest point release see 'Updating ownCloud'_.
2. Deactivate all third party applications.

   ---

   **Note:** Not all third party applications are supported on all ownCloud Server versions. Make sure to check version compatibility prior to upgrading your ownCloud server.

   ---

3. Back up your existing ownCloud Server database. You can find these procedures in *Backing up ownCloud*.
4. Download the latest ownCloud Server version to your working directory.

   For Linux operating systems, use the following command:

   ```
   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. 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 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 (Redhat) | `apachectl stop` |
| Debian or Ubuntu | `/etc/init.d/apache2 stop` |
| openSUSE or SUSE (SLE) | `/usr/sbin/rcapache2 stop` |

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

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

**Note:** For specific instructions on how to stop, start, or manage your server, please refer to instructions for the server on your specific operating environment.

6. Move, or rename your current owncloud directory (named `/owncloud` if installed using defaults) to another location for use in your new version of ownCloud.

   **Note:** This step ensures that you have a version of ownCloud available for backup purposes.

7. Replace the old version of ownCloud Server with the new version of ownCloud Server:

   Assuming that your installation directory is called 'owncloud', and that it resides in your working directory, the command to unpack the release tarball into the directory would be as follows:

   ```
   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.

   **Note:** If you unpack into an existing installation, the autoloader might pick up classes twice because the files have been moved, resulting in a `Cannot redeclare class` error.

8. Copy and paste the `/config/config.php` file from the saved version of ownCloud to the `/config` directory of your new ownCloud version.

   **Note:** You must perform this step **before** restarting your web server.

9. If you chose to keep your /data directory in your `/owncloud` directory, copy and paste it from the old version of ownCloud to the `/owncloud` directory of your new ownCloud version.

**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 "Advanved Options" chapter in *Installation Wizard* for added details about changing the default database or data directory.

10. 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 restart an Apache server, refer to the following table for specific commands to use in different Linux operating systems:

| Operating System | Command (as root) |
|---|---|
| CentOS (Redhat) | `apachectl start` |
| Debian or Ubuntu | `/etc/init.d/apache2 start` |
| openSUSE or SUSE (SLE) | `/usr/sbin/rcapache2 start` |

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

| Method | Procedure |
|---|---|
| User Interface (UI) | 1. Open IIS Manager and navigate to the Web server node in the tree.<br>2. In the **Actions** pane, click **Start**. |
| Command Line | 1. Open an elevated command line window.<br>2. At the command prompt, type **net start W3SVC** and press **ENTER**. This command starts both WAS and W3SVC. |

    **Note:** For specific instructions on how to stop, start, or manage your server, please refer to instructions for the server on your specific operating environment.

11. Use a browser to your ownCloud server.

    This step is required. Accessing the server using a browser connection launches the server upgrade.

    **Note:** To avoid webserver timeouts on slow or overloaded systems you can also update your ownCloud on the command line if you have SSH access to your server. This also avoid other users accessing the server during the process. Before starting the server (step 10), run the command `php occ upgrade` in the ownCloud folder.

12. If third party applications were running on your system, ensure that they provide versions compatible with the new ownCloud release. If compatible, you can reinstall and enable these applications.

    **Note:** Update procedures should run when necessary.

## 4.4 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

### 4.4.1 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 a place outside of your ownCloud environment. You could use this command:

```
rsync -Aax owncloud-dirbkp/ owncloud/
```

### 4.4.2 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
```

## 4.5 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

CHAPTER

FIVE

# 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.

CHAPTER

SIX

# INDICES AND TABLES

- *genindex*